Combining SAS Datasets

Neelesh Singh

Module Content

- Set and Append
- Merge using data Step
- Merge using Proc SQL

- ✓ What does the Set statement do ?
  - ✓ Reads an observation from one or more SAS datasets
- ✓ Set statement -- Syntax
  - ✓ SET <SAS Dataset(s) <*(dataset options)*>> *<options>;*
    - ✓ SAS Dataset(s)
      - ✓ specifies the SAS dataset name(s)
    - ✓ (dataset options)
      - ✓ specifies actions SAS is to take when it reads variables or observations for processing
    - ✓ options
      - ✓ end = *variable*
        - ✓ creates & names a temporary variable that contains an end of file indicator
        - ✓ the variable, which is initialized to zero, is set to 1 when SET reads the last observation of the last dataset listed
        - ✓ this variable is not added to any new dataset

- ✓ Options….
  - ✓ nobs = *variable*
    - ✓ creates & names a temporary variable whose value is usually the total no. of observations in the input dataset(s)
    - ✓ if more than one dataset is listed in the SET statement, nobs = total no. of observations in the listed dataset(s)

✓ Example :

✓ SAS Code --

```
DATA T3;
SET T1 T2;
RUN;
```

Sample Codes for SET Statement

T1

A	B
1	44
2	34
3	56

T2

A	B
4	23
5	12
6	78

T3

A	B
1	44
2	34
3	56
4	23
5	12
6	78

T1

T2

APPEND PROCEDURE

- ✓ What does the Append Procedure do ?
  - ✓ Adds the observations from one SAS dataset to the end of another SAS dataset
- ✓ Append Procedure -- Syntax
  - ✓ PROC APPEND BASE = <libref.>SAS Dataset

    <DATA = <libref.>SAS Dataset > <FORCE>;
    - ✓ BASE = <libref.>SAS Dataset
      - ✓ names the dataset to which you want to add observations
      - ✓ *libref* specifies the library that contains the SAS dataset
        - ✓ If the libref is omitted, the default is WORK
      - ✓ *SAS Dataset* names a SAS dataset.
        - ✓ The BASE = dataset is the current SAS data set after all append operations regardless of whether one is creating a new dataset or appending to an existing dataset.
      - ✓ Alias : OUT

✓Append Procedure – Syntax……

- ✓ DATA = <libref.>SAS Dataset
  - ✓ names the dataset containing observations that one wants to append to the end of the SAS dataset specified in the BASE = argument
  - ✓ *libref* specifies the library that contains the SAS dataset
    - ✓ If the libref is omitted, the default is WORK
  - ✓ *SAS Dataset* names a SAS dataset.
    - ✓ If the APPEND procedure cannot find an existing data set with this name, it stops processing.
  - ✓ Alias : NEW

✓ Append Procedure – Syntax......

- ✓ FORCE
  - ✓ forces the APPEND procedure to concatenate datasets when the DATA = dataset contains variables that are either
    - ✓ not in BASE = dataset
    - ✓ do not have the same type as the variables in the BASE = dataset
    - ✓ longer than the variables in the BASE = dataset

✓Appending to datasets with different variables

- ✓ If the DATA = dataset contains variables that are not in the BASE = dataset we need to use the FORCE option to concatenate the two datasets.
  - ✓ The APPEND procedure drops the extra variables and issues a warning message
- ✓ If the BASE = dataset contain a variable that is not in the DATA = dataset, the APPEND procedure concatenates the two datasets, but the observations from the DATA = dataset have a missing value for the variable that was not present in the DATA = dataset.
  - ✓ The FORCE option is not necessary in this case

✓ Appending datasets that contains variables with different attributes

- ✓ If the variable has different attributes in the BASE = dataset than it does in the DATA = dataset, <u>the attributes in the BASE = dataset prevail</u>.
- ✓ If the formats in the DATA = dataset are different from those in the BASE = dataset, <u>then the formats in the BASE = dataset prevail</u>.
- ✓ If the length of a variable is longer in the DATA = dataset than in the BASE = dataset, or if the same variable is defined as a character in one dataset & numeric in the other, use the FORCE option. Using the FORCE option has the following consequences
  - ✓ The <u>length</u> of the variables in the BASE = dataset <u>takes precedence</u>.
    - ✓ SAS <u>truncates</u> values from the DATA = dataset to fit them into the length that is specified in the BASE = dataset

- ✓ Choosing between the SET statement & the APPEND procedure
  - ✓ If the SET statement is used to concatenate two datasets, then SAS must process all the observations in both the datasets to create a new one.
  - ✓ The APPEND procedure bypasses the processing of data & directly adds observations to the end of the original dataset.
  - ✓ Using the APPEND procedure can be more efficient if
    - ✓ the BASE = dataset is large
    - ✓ all variables in the BASE = dataset have the same length & type as the variables in the DATA = dataset and if all variables exist in both datasets

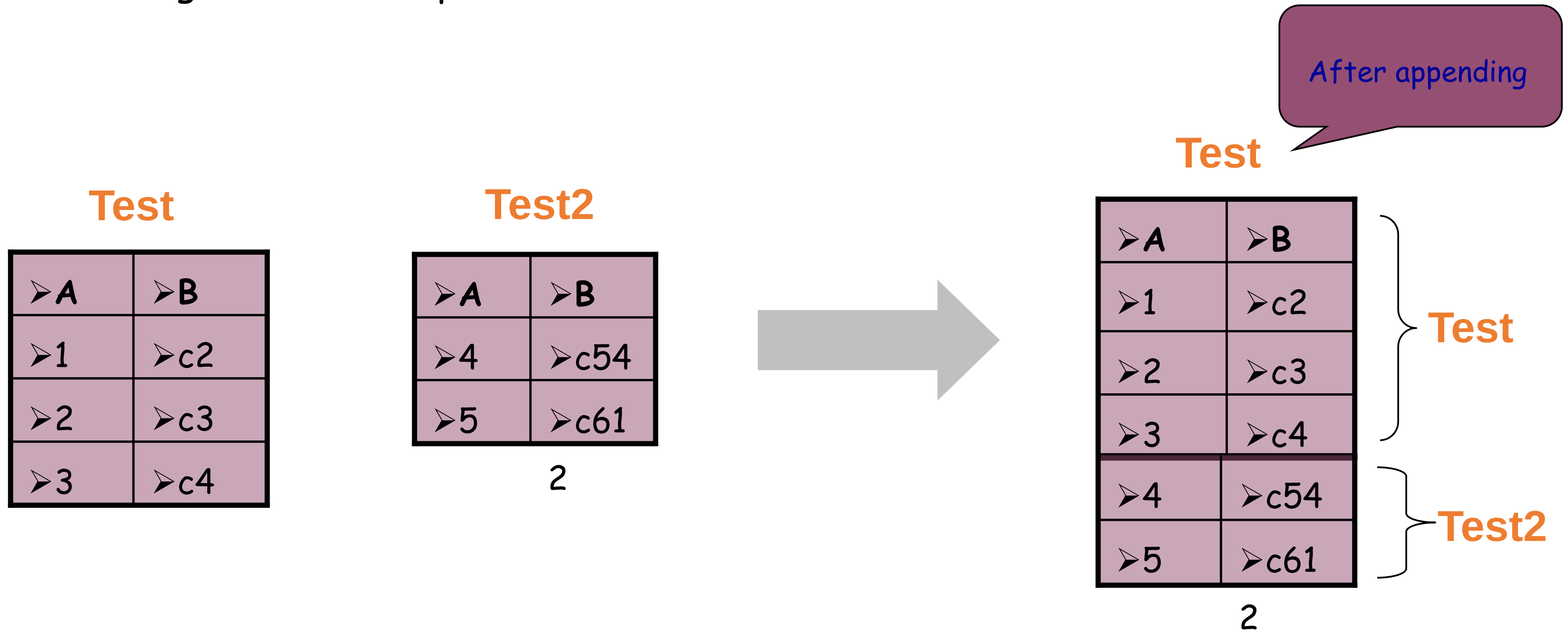
      N.B. : One can use the CONTENTS procedure to see the variable lengths & types
  - ✓ The APPEND procedure is especially useful if observations are frequently added to a SAS dataset
    - ✓ E.g. Operation of production programs

- ✓ Examples :
  - ✓ Using the FORCE option –
    - ✓ An example where the FORCE option is a must for appending
      - ✓ Run the following piece of code --

```
data test;
input a b $6.;
cards;
1 c2
2 c3
3 c4
 ;
data test2;
input a b $8.;
cards;
4 c545
5 c612
 ;
 proc append base = test  data = test2  force; run;
```

Sample Codes for APPEND Procedure

✓ Using the FORCE option....



Test

➢A	➢B
➢1	➢c2
➢2	➢c3
➢3	➢c4

Test2

➢A	➢B
➢4	➢c54
➢5	➢c61

Test

➢A	➢B
➢1	➢c2
➢2	➢c3
➢3	➢c4
➢4	➢c54
➢5	➢c61

- ✓ What happens if one does not use the FORCE option for appending (in the above example)

- ✓ The appending does not happen & the SAS log displays a <u>warning</u> & an <u>error</u> message as follows

```
NOTE:  Appending WORK.TEST2 to WORK.TEST.
WARNING:  Variable b has different lengths on BASE and DATA files (BASE 6 DATA 8).
ERROR:  No appending done because of anomalies listed above. Use FORCE option to append
        these files
NOTE: 0 observations added.
NOTE: The data set WORK.TEST has 3 observations and 2 variables.
NOTE: Statements not processed because of errors noted above.
```

ORDERING DATASETS
a. SORT Procedure

SORT PROCEDURE

- ✓ What does the Sort Procedure do ?
  - ✓ Orders SAS dataset observations by the values of one or more character or numeric variables
    - ✓ It either replaces the original dataset or creates a new dataset
    - ✓ It produces only an output dataset
- ✓ Sort Procedure -- Syntax
  - ✓ PROC SORT <other options>;

    BY <DESCENDING> variable-1 ……… <DESCENDING> variable-n;
    - ✓ <other options>
      - ✓ DATA = SAS Dataset
        - ✓ identifies the input SAS Dataset
      - ✓ OUT = SAS Dataset
        - ✓ names the output data set
        - ✓ if SAS dataset does not exist, then PROC SORT creates it

- ✓Sort Procedure Syntax – Options….
  - ✓ <other options>….
    - ✓ NODUPKEY
      - ✓ checks for & eliminates observations with duplicate BY values
      - ✓ if this option is specified, PROC SORT compares all BY values for each observation to those for the previous observation that is written to the output dataset
      - ✓ if an exact match is found, then the observation is not written to the output dataset
    - ✓ NODUPRECS (ALIAS : NODUP)
      - ✓ checks for & eliminates duplicate observations
      - ✓ if this option is specified, PROC SORT compares all variable values for each observation to those for the previous observation that is written to the output dataset
      - ✓ if an exact match is found, then the observation is not written to the output dataset
    - ✓ DUPOUT = SAS Dataset
      - ✓ specifies the output dataset to which duplicate observations are written

✓Sort Procedure Syntax – Options....

- ✓ BY STATEMENT - Options
  - ✓ DESCENDING
    - ✓ reverse the sort order for the variable that immediately follows in the statement so that observations are sorted from the largest value to the smallest value

Sample Code for SORT Procedure

- ✓Examples
  - ✓ Running PROC SORT with an OUT = option

```
proc sort data =test  out=test1  ;
by a ;
run;
```

N.B. If the OUT = option is not mentioned, then SAS <u>overwrites</u> the existing dataset

Test

A	B
12	23
45	24
12	23
34	25
25	56
45	98
12	22
78	11
23	18
12	24

Test1

A	B
12	23
12	23
12	22
12	24
23	18
25	56
34	25
45	24
45	98
78	11

✓Examples

✓ Using the NODUPKEY option

```
 proc sort data
=test  out=test1
nodupkey;

by a ;

run;
```

Test

A	B
12	23
45	24
12	23
34	25
25	56
45	98
12	22
78	11
23	18
12	24

Deleted

Test1

A	B
12	23
23	18
25	56
34	25
45	24
78	11

N.B. SAS log will have a note containing the no. of duplicate key observations deleted

Tip : Always use an OUT= option along with NODUPKEY

✓Examples

✓ Using the NODUPRECS option

```
proc sort data =test
out=test1  noduprecs;
by _all_ ;
run;
```

Deleted

N.B. SAS log will have a note containing the no. of duplicate observations deleted

Test

A	B
12	23
45	24
12	23
34	25
25	56
45	98
12	22
78	11
23	18
12	24

Tip : Always use an OUT= option along with NODUPRECS

Test1

A	B
12	23
12	22
12	24
23	18
25	56
34	25
45	24
45	98
78	11

✓Examples

- ✓ Using the DESCENDING option

```
proc sort data =test out=test1;
by descending a ;
run;
```

Test

A	B
12	23
45	24
12	23
34	25
25	56
45	98
12	22
78	11
23	18
12	24

Test1

A	B
78	11
45	24
45	98
34	25
25	56
23	18
12	23
12	23
12	22
12	24

MERGING DATASETS

- MERGE Statement

- ✓ What does the Merge Statement do ?
  - ✓ Joins observations from two or more SAS datasets into single observations
- ✓ Merge Statement – Syntax
  - ✓ MERGE SAS Dataset-1 <*dataset options*> ......

    SAS Dataset-n <*dataset options*>;

    < BY <DESCENDING> variable-1 .... <DESCENDING> variable-n >;
    - ✓ SAS Dataset(s)
      - ✓ names at least two existing SAS datasets from which observations are read
        - ✓ Optionally, one can specify additional datasets

✓ Merge Statement – Syntax....

- ✓ *<dataset options>*
  - ✓ specifies one or more SAS dataset options in parentheses after a SAS dataset name
  - ✓ IN = variable
    - ✓ names the new variable whose value indicates whether that input dataset contributed to the current observation
    - ✓ within the DATA step, the value of the variable is 1 if the dataset contributed to the current observation and 0 otherwise

Merge is defined as the activity of combining columns from two or more datasets into a single row of a new dataset.

There are two types of Merging techniques :

- **One – to – One Merge**

  One-to-one merging combines observations from two or more SAS data sets into a single observation in a new data set. To perform a one-to-one merge, use either the **SET** statement or the **MERGE** statement **without** a **BY statement.**

- **Match Merge**

  Match-merging combines observations from two or more SAS data sets into a single observation in a new data set according to the values of a common variable.

****Sorting / Index creation is necessary before Match Merging.**

✓ Types of Merge

✓ One to One Merging

- ✓ combines observations from two or more SAS datasets into a single observation in a new dataset
- ✓ to perform such a merge, use the MERGE statement <u>without</u> a BY statement
- ✓ SAS combines the first observation from all datasets that are named in the MERGE statement into the first observation in the new dataset, the second observation from all datasets into the second observation in the new dataset and so on
- ✓ the no. of observations in the new dataset is equal to the no. of observations in the largest dataset named in the MERGE statement

Tip : Use care when performing a one to one merging

- ✓ Types of Merge....
  - ✓ Match Merging
    - ✓ combines observations from two or more SAS datasets into a single observation in a new dataset acc. to the values of a common variable
    - ✓ to perform such a merge, use a BY statement immediately after the MERGE statement
      - ✓ the variables in the BY statement must be common to all datasets
      - ✓ only one BY statement can accompany each MERGE statement in a DATA step
      - ✓ the datasets that are listed in the MERGE statement must be sorted in the order of the values of the variables that are listed in the BY statement
    - ✓ the no. of observations in the new dataset is the sum of the largest no. of observations in each BY group in all datasets

✓ Examples

✓ One to One Merging

```
data tt;
merge t1 t2;
run;
```

T1

A	B
1	44
2	34
3	56

T2

A	C
2	23
3	12
6	78

→

TT

A	B	C
2	44	23
3	34	12
6	56	78

The values of the variable, A, will be the ones from the last dataset used for merging

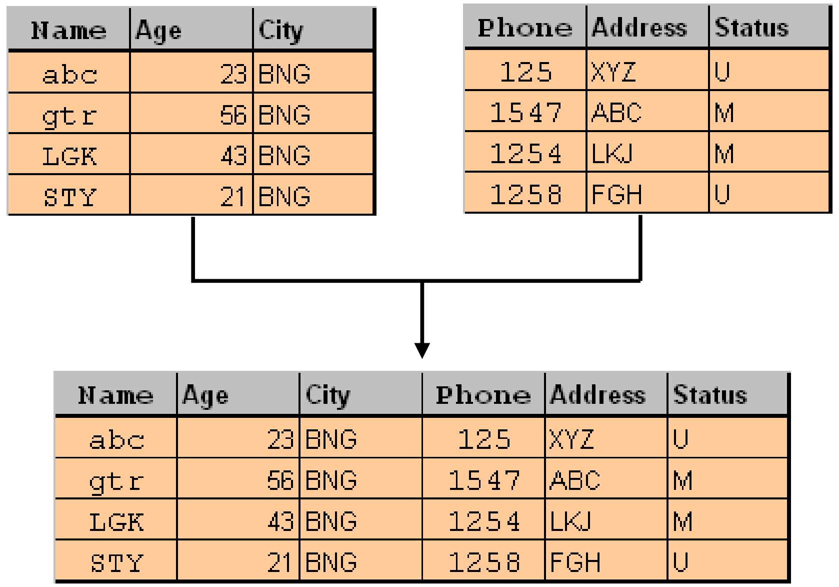
One – to – One Merge

EXAMPLE 1 (Using SET Keyword)

```
Data MergedData;
        Set Data1;
        Set Data2;
run;
```

EXAMPLE 2 (Using MERGE Keyword)

```
Data MergedData;
        MERGE Data1 Data2;
run;
```



Name	Age	City
abc	23	BNG
gtr	56	BNG
LGK	43	BNG
STY	21	BNG

Phone	Address	Status
125	XYZ	U
1547	ABC	M
1254	LKJ	M
1258	FGH	U

Name	Age	City	Phone	Address	Status
abc	23	BNG	125	XYZ	U
gtr	56	BNG	1547	ABC	M
LGK	43	BNG	1254	LKJ	M
STY	21	BNG	1258	FGH	U

✓ Examples....

✓ Match Merging

```
proc sort data = t1;by a; run;

proc sort data = t2;by a; run;

data tt;
merge t1 t2;
by a;
run;
```

T1

A	B
1	44
2	34
3	56

T2

A	C
2	23
3	12
6	78

→

TT

A	B	C
1	44	.
2	34	23
3	56	12
6	.	78

Match Merge

EXAMPLE

```
PROC SORT Data = Data1 Nodupkey;
By Name;
Run;

PROC SORT Data = Data2 Nodupkey;
By Name;
Run;

Data MergedData3;
                MERGE Data1(In = a) Data2(In = b);
                By Name;
                IF a AND b;  /* OR / NOT / AND */
run;
```

name	age	city
abc	23	BNG
gtr	56	DEL
LGK	43	DEL
STY	21	BNG

name	phone	address	status
LGK	1258	FGH	U
STY	1254	LJK	M
abc	125	XYZ	U
gtr	1547	ABC	M

name	age	city	phone	address	status
LGK	43	DEL	1258	FGH	U
STY	21	BNG	1254	LJK	M
abc	23	BNG	125	XYZ	U
gtr	56	DEL	1547	ABC	M

✓ Examples....

✓ Match Merging using the IF statement

```
proc sort data = t1;by a; run;

proc sort data = t2;by a; run;

data tt;
merge t1 (in = aa) t2 (in = bb);
by a; if aa;
run;
```

will keep only those records (w.r.t. the BY variable) in the merged dataset which come from t1

T1

A	B
1	44
2	34
3	56

T2

A	C
2	23
3	12
6	78

TT

A	B	C
1	44	.
2	34	23
3	56	12

- ✓ Examples....
  - ✓ Match Merging using the IF statement

```
proc sort data = t1;by a; run;

proc sort data = t2;by a; run;

data tt;
merge t1 (in = aa) t2 (in = bb);
by a; if aa and bb;
run;
```

will keep only those records (w.r.t. the BY variable) in the merged dataset which are common to both the datasets

T1

A	B
1	44
2	34
3	56

T2

A	C
2	23
3	12
6	78

TT

A	B	C
2	34	23
3	56	12

- ✓ Examples....
  - ✓ Match Merging using the IF statement

```
proc sort data = t1;by a; run;

proc sort data = t2;by a; run;

data tt;
merge t1 (in = aa) t2 (in = bb);
by a; if aa and not bb;
run;
```

will keep only those records (w.r.t. the BY variable) in the merged dataset which belong to t1 but <u>not</u> to t2

T1

A	B
1	44
2	34
3	56

T2

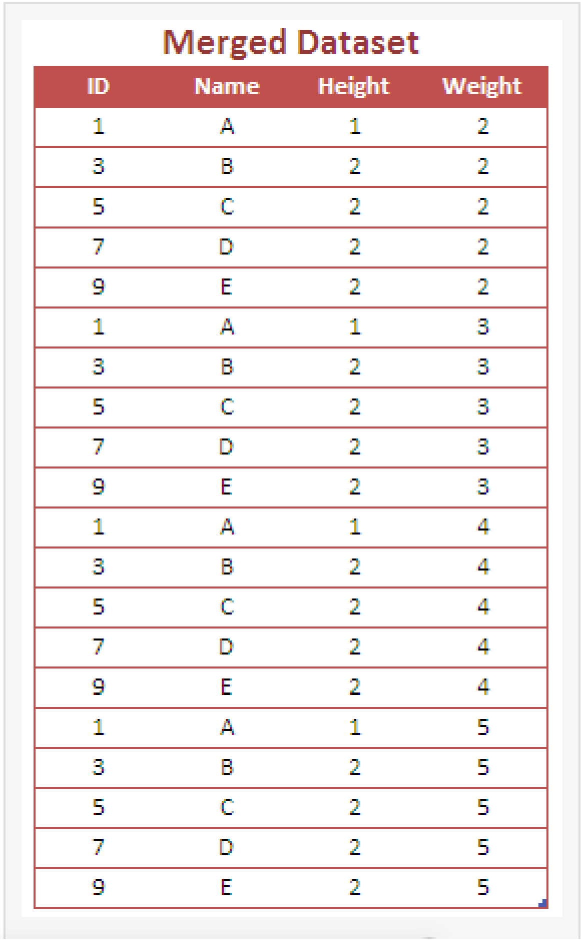
A	C
2	23
3	12
6	78

TT

A	B	C
1	44	.

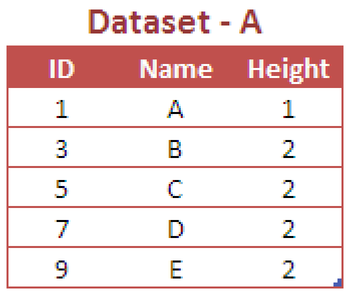
Sample Code for Merge Statement

Cartesian



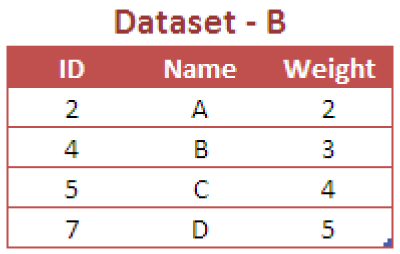
Merged Dataset

ID	Name	Height	Weight
1	A	1	2
3	B	2	2
5	C	2	2
7	D	2	2
9	E	2	2
1	A	1	3
3	B	2	3
5	C	2	3
7	D	2	3
9	E	2	3
1	A	1	4
3	B	2	4
5	C	2	4
7	D	2	4
9	E	2	4
1	A	1	5
3	B	2	5
5	C	2	5
7	D	2	5
9	E	2	5



Dataset - A

ID	Name	Height
1	A	1
3	B	2
5	C	2
7	D	2
9	E	2

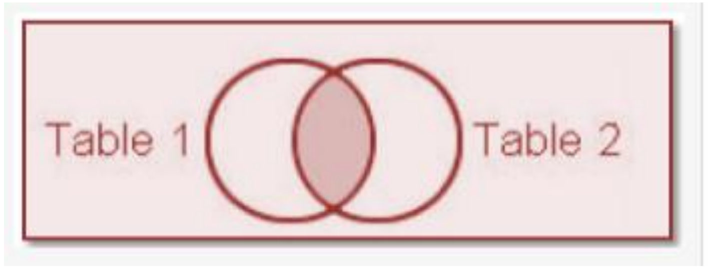


Dataset - B

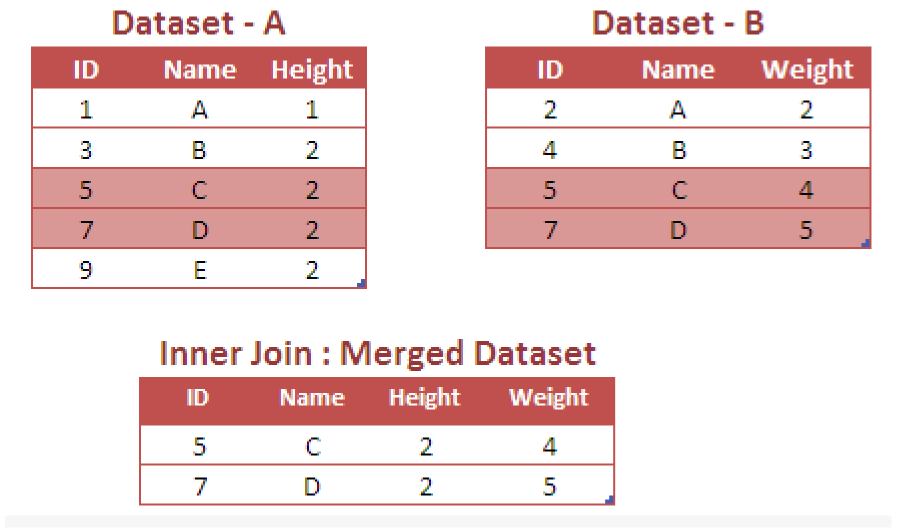
ID	Name	Weight
2	A	2
4	B	3
5	C	4
7	D	5

```
PROC SQL;

Create table dummy as

Select * from A as x cross join B as y;

Quit;
```

Inner Join



```
PROC SQL;

Create table dummy as

Select * from A as x inner join B as y

On x.ID = y.ID;

Quit;
```



Dataset - A

ID	Name	Height
1	A	1
3	B	2
5	C	2
7	D	2
9	E	2

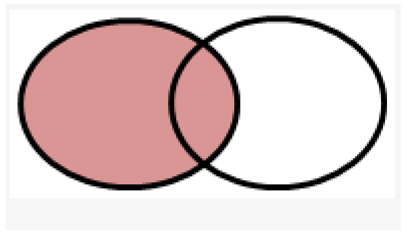
Dataset - B

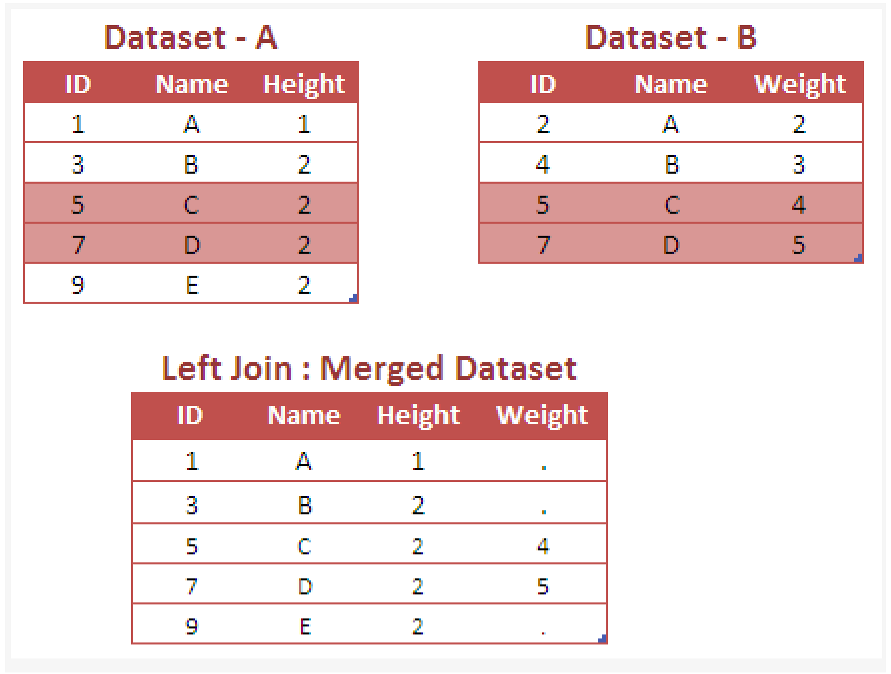
ID	Name	Weight
2	A	2
4	B	3
5	C	4
7	D	5

Inner Join : Merged Dataset

ID	Name	Height	Weight
5	C	2	4
7	D	2	5

Left Join





Dataset - A

ID	Name	Height
1	A	1
3	B	2
5	C	2
7	D	2
9	E	2

Dataset - B

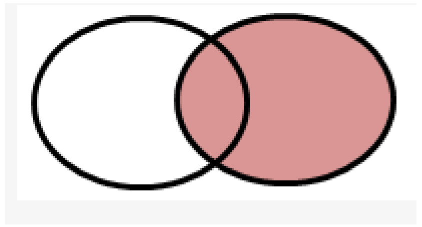
ID	Name	Weight
2	A	2
4	B	3
5	C	4
7	D	5

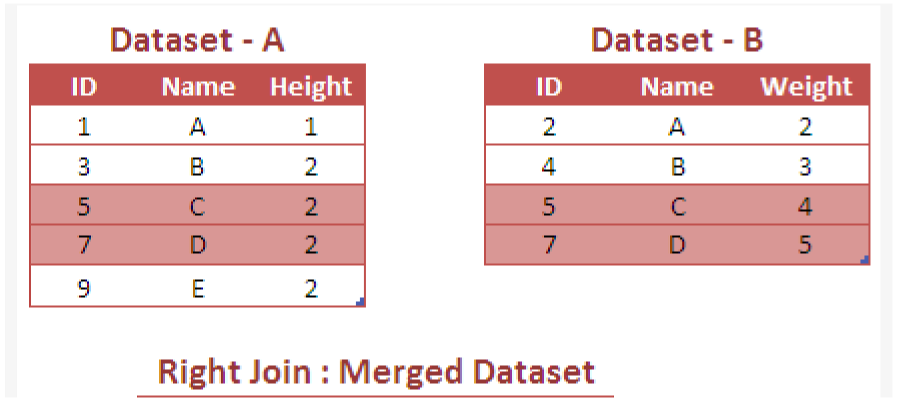
Left Join : Merged Dataset

ID	Name	Height	Weight
1	A	1	.
3	B	2	.
5	C	2	4
7	D	2	5
9	E	2	.

```
PROC SQL;

Create table dummy as

Select * from A as x left join B as y

On x.ID = y.ID;

Quit;
```

Right Join





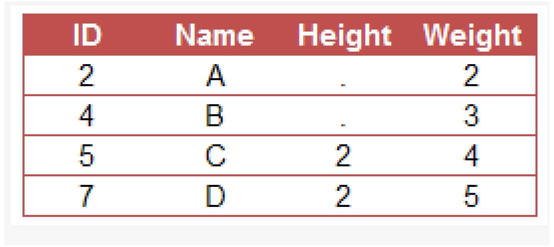
Dataset - A

ID	Name	Height
1	A	1
3	B	2
5	C	2
7	D	2
9	E	2

Dataset - B

ID	Name	Weight
2	A	2
4	B	3
5	C	4
7	D	5

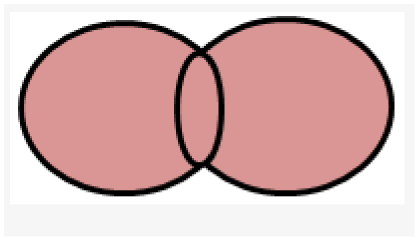
Right Join : Merged Dataset



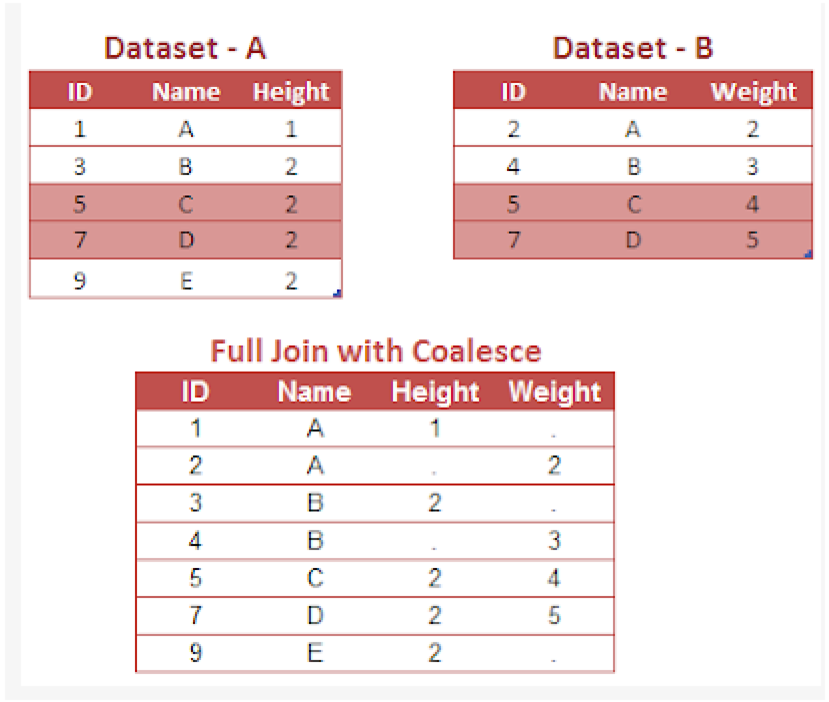
ID	Name	Height	Weight
2	A	.	2
4	B	.	3
5	C	2	4
7	D	2	5

```
PROC SQL;

Create table dummy as

Select * from A as x right join B as y

On x.ID = y.ID;

Quit;
```

Full join



```
proc sql;

create table dummy as

select coalesce (x.ID,y.ID) as ID, coalesce (x.name,y.name) as
name,height,weight

from a as x full join b as y

on x.id = y.id;

quit;
```



Dataset - A

ID	Name	Height
1	A	1
3	B	2
5	C	2
7	D	2
9	E	2

Dataset - B

ID	Name	Weight
2	A	2
4	B	3
5	C	4
7	D	5

Full Join with Coalesce

ID	Name	Height	Weight
1	A	1	.
2	A	.	2
3	B	2	.
4	B	.	3
5	C	2	4
7	D	2	5
9	E	2	.

greatlearning
Learning for Life

Thank you