Chapter 12

Combining SAS Datasets

Overview

In this chapter you will learn how to combine observations from two or more datasets into a new dataset using the DATA step, including one-to-one combining or merging, concatenating, appending, interleaving, and match-merging.

One-to-One Merging (Combining)

Using multiple SET statements in a DATA step, you can read different datasets, or you can read the same dataset more than once, as if you were reading from separate datasets.

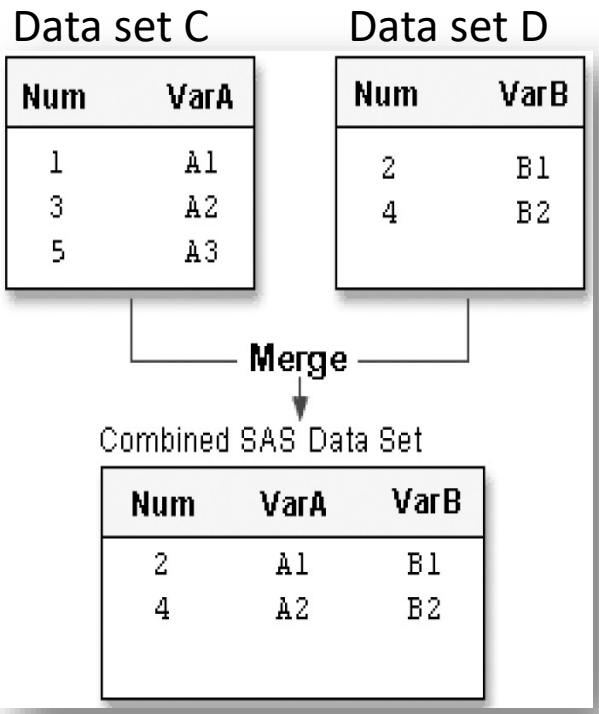
- The new dataset contains **all the variables from all the input datasets**. If the datasets contain variables that have the same names, the values that are read in from the last dataset overwrite the values that were read in from earlier datasets.

- The number of observations in the new dataset is **up to the number of observations in the smallest original dataset**.

- Observations are combined based on their relative positions in each dataset. That is, in its simplest form, the first observation in one dataset is joined with the first observation in the other, and so on.

One-to-One Merging (Combining)

```
DATA one2one;
      set c;
      set d;
run;
```



The same named variable *Num* must be of the same data type.

1. The 1st SET statement reads one observation from dataset C.

Num	VarA	VarB
1	A1	

2. The 2nd SET statement reads one observation from dataset D. The value of Num in dataset D overwrites the value of Num in dataset C.

Num	VarA	VarB
2	A1	B1

3. The 1st SET statement reads the second observation from dataset C.

Num	VarA	VarB
2	A1	B1
3	A2	

4. The 2nd SET statement reads the second observation from dataset D, overwriting the value of Num in dataset C.

Num	VarA	VarB
2	A1	B1
4	A2	B2

One-to-One Merging (Combining)

TABLE: Clinic.Patients

ID	Sex	Age
1129	F	48
1837	F	57
2304	F	16
2486	F	63
4759	F	60
5438	F	42
6488	F	59
9012	F	39
9125	F	56

LE: Clinic.Measure

ID	Height	Weight
1129	61	137
1387	64	142
2304	61	102
5438	62	168
6488	64	154
9012	63	157

```
data one2one;
   set clinic.patients;
   if age<60;
   set clinic.measure;
   if weight > 140 and height >=63;
run;
```

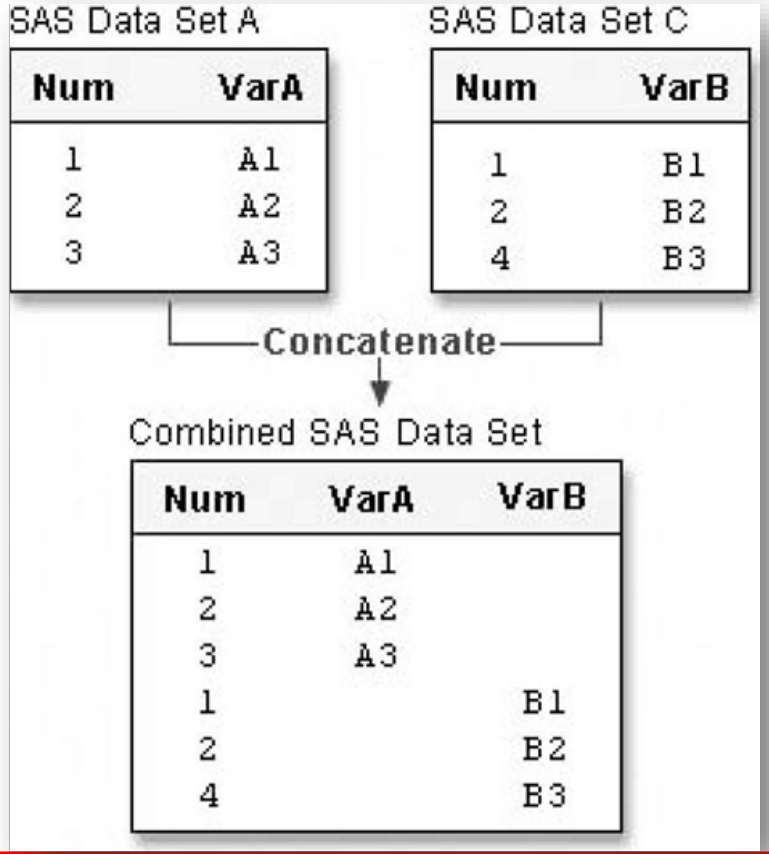
VIEWTABLE: Work.One2one

	ID	Sex	Age	Height	Weight
1	1387	F	57	64	142
2	6488	F	59	64	154
3	9012	F	39	63	157

Concatenating

Concatenating only uses a single SET statement in a data step. All of the observations are read from the first dataset, and then all of the observations are read from the second dataset, and so on, until all of the listed datasets have been read. The new dataset contains all of the variables and observations from all of the input datasets.

```
DATA concat;
     set a c;
run;
```



Note

- The same named variable *Num* must be of the same data type.
- The attributes of the 1st the same named variables of the 1st dataset determine those in the new dataset.

Concatenating

TABLE: Clinic.Therapy1999

Month	Year	AerClass	WalkJogRun	Swim
01	1999	26	78	14
02	1999	32	109	19
03	1999	15	106	22
04	1999	47	115	24
05	1999	95	121	31
06	1999	61	114	67
07	1999	67	102	72
08	1999	24	76	77
09	1999	78	77	54
10	1999	81	62	47
11	1999	84	31	52
12	1999	92	44	55

```
data concat;
        set clinic.therapy1999
            clinic.therapy2000;
run;
```

TABLE: Clinic.Therapy2000

Month	Year	AerClass	WalkJogRun	Swim
01	2000	37	91	83
02	2000	41	102	27
03	2000	52	98	19
04	2000	61	118	22
05	2000	49	88	29
06	2000	24	101	54
07	2000	45	91	69
08	2000	63	65	53
09	2000	60	49	68
10	2000	78	70	41
11	2000	82	44	58
12	2000	93	57	47

VIEWTABLE: Work.Concat

	Month	Year	AerClass	WalkJogRun	Swim
1	01	1999	26	78	14
2	02	1999	32	109	19
3	03	1999	15	106	22
4	04	1999	47	115	24
5	05	1999	95	121	31
6	06	1999	61	114	67
7	07	1999	67	102	72
8	08	1999	24	76	77
9	09	1999	78	77	54
10	10	1999	81	62	47
11	11	1999	84	31	52
12	12	1999	92	44	55
13	01	2000	37	91	83
14	02	2000	41	102	27
15	03	2000	52	98	19
16	04	2000	61	118	22
17	05	2000	49	88	29
18	06	2000	24	101	54
19	07	2000	45	91	69
20	08	2000	63	65	53
21	09	2000	60	49	68
22	10	2000	78	70	41
23	11	2000	82	44	58
24	12	2000	93	57	47

Interleaving

A combination of **Concatenation** and the **BY Statement** → interspersing observations from two or more datasets, based on one or more common variables.

```
DATA output-SAS-data-set;
   SET SAS-data-set-1 SAS-data-set-2;
   BY variable(s);
RUN;
```

where

output-SAS-data-set names the dataset to be created

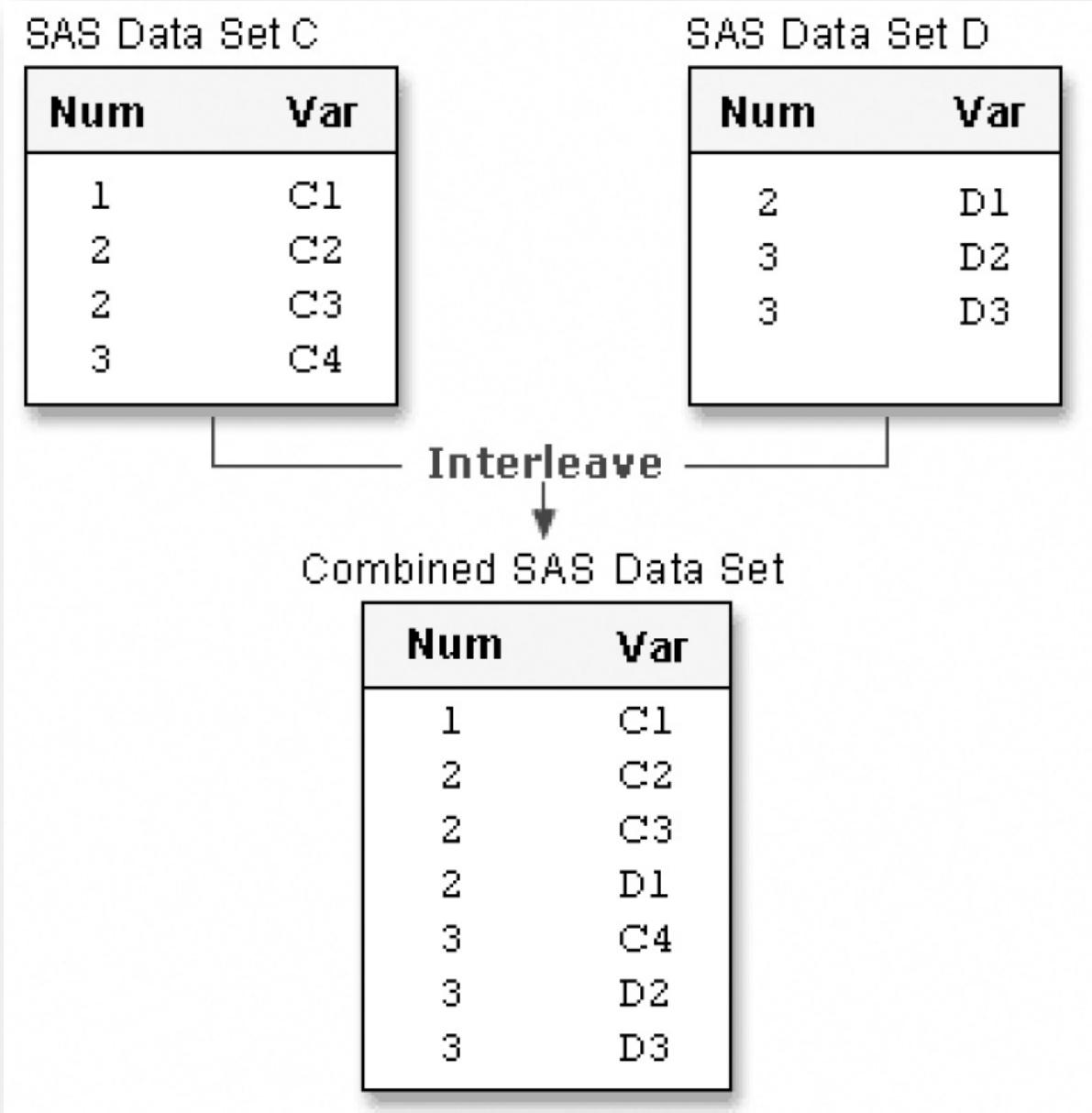
SAS-data-set-1 and SAS-data-set-2 specify the datasets to be read.

variable(s) specifies one or more variables that are used to interleave observations.

- You can specify any number of data sets in the SET statement. Each input dataset *must* be sorted or indexed based on the BY variable(s).
- Observations in each BY group in each dataset are read sequentially, in the order in which the datasets and BY variables are listed.
- The new dataset includes all the variables from all the input datasets, and it contains the total number of observations from all input datasets.

Interleaving

```
data interlv;
   set c d;
   by num;
run;
```



Interleaving

TABLE: Clinic.Therapy1999

Month	Year	AerClass	WalkJogRun	Swim
01	1999	26	78	14
02	1999	32	109	19
03	1999	15	106	22
04	1999	47	115	24
05	1999	95	121	31
06	1999	61	114	67
07	1999	67	102	72
08	1999	24	76	77
09	1999	78	77	54
10	1999	81	62	47
11	1999	84	31	52
12	1999	92	44	55

TABLE: Clinic.Therapy2000

Month	Year	AerClass	WalkJogRun	Swim
01	2000	37	91	83
02	2000	41	102	27
03	2000	52	98	19
04	2000	61	118	22
05	2000	49	88	29
06	2000	24	101	54
07	2000	45	91	69
08	2000	63	65	53
09	2000	60	49	68
10	2000	78	70	41
11	2000	82	44	58
12	2000	93	57	47

```
data interlv;
    set clinic.therapy1999
        clinic.therapy2000;
    by month;
Run;
```

VIEWTABLE: Work.Interlv

	Month	Year	AerClass	WalkJogRun	Swim
1	01	1999	26	78	14
2	01	2000	37	91	83
3	02	1999	32	109	19
4	02	2000	41	102	27
5	03	1999	15	106	22
6	03	2000	52	98	19
7	04	1999	47	115	24
8	04	2000	61	118	22
9	05	1999	95	121	31
10	05	2000	49	88	29
11	06	1999	61	114	67
12	06	2000	24	101	54
13	07	1999	67	102	72
14	07	2000	45	91	69
15	08	1999	24	76	77
16	08	2000	63	65	53
17	09	1999	78	77	54
18	09	2000	60	49	68
19	10	1999	81	62	47
20	10	2000	78	70	41
21	11	1999	84	31	52
22	11	2000	82	44	58
23	12	1999	92	44	55
24	12	2000	93	57	47

Appending

This is done through the **PROC APPEND** procedure, which simply adds the observations of one dataset to the end of a "master" (or BASE) dataset. SAS does not create a new dataset nor does it read the base dataset when executing the APPEND procedure.

```
PROC APPEND BASE=SAS-data-set
            DATA=SAS-data-set;
RUN;
```

where

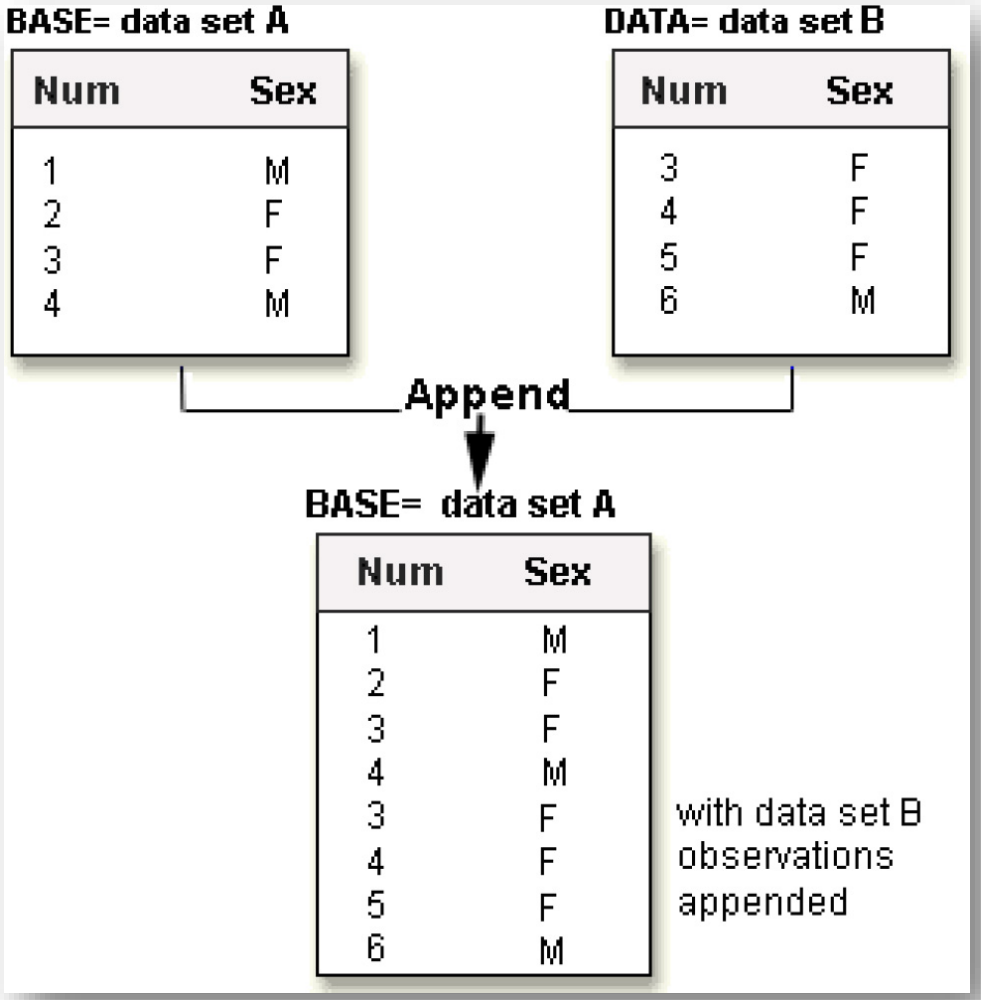
BASE= name the dataset to which observations are added

DATA= name the dataset containing observations that are added to the base dataset. This is the only data set that SAS actually reads.

Note that the final dataset is the original dataset (BASE) with appended observations and that no new dataset was created.

Appending

```
proc append base=A
   data=B;
run;
```



Note that the two datasets are like-structured datasets, that is, both datasets have the same variable information.

Appending

TABLE: Clinic.Therapy1999

Month	Year	AerClass	WalkJogRun	Swim
01	1999	26	78	14
02	1999	32	109	19
03	1999	15	106	22
04	1999	47	115	24
05	1999	95	121	31
06	1999	61	114	67
07	1999	67	102	72
08	1999	24	76	77
09	1999	78	77	54
10	1999	81	62	47
11	1999	84	31	52
12	1999	92	44	55

TABLE: Clinic.Therapy2000

Month	Year	AerClass	WalkJogRun	Swim
01	2000	37	91	83
02	2000	41	102	27
03	2000	52	98	19
04	2000	61	118	22
05	2000	49	88	29
06	2000	24	101	54
07	2000	45	91	69
08	2000	63	65	53
09	2000	60	49	68
10	2000	78	70	41
11	2000	82	44	58
12	2000	93	57	47

```
Data therapy1999
  set clinic.therapy1999;
proc append base=therapy1999
    data= clinic.therapy2000;
run;
```

	Month	Year	AerClass	WalkJogRun	Swim
1	01	1999	26	78	14
2	02	1999	32	109	19
3	03	1999	15	106	22
4	04	1999	47	115	24
5	05	1999	95	121	31
6	06	1999	61	114	67
7	07	1999	67	102	72
8	08	1999	24	76	77
9	09	1999	78	77	54
10	10	1999	81	62	47
11	11	1999	84	31	52
12	12	1999	92	44	55
13	01	2000	37	91	83
14	02	2000	41	102	27
15	03	2000	52	98	19
16	04	2000	61	118	22
17	05	2000	49	88	29
18	06	2000	24	101	54
19	07	2000	45	91	69
20	08	2000	63	65	53
21	09	2000	60	49	68
22	10	2000	78	70	41
23	11	2000	82	44	58
24	12	2000	93	57	47

Appending

Using the **FORCE** Option with Unlike-Structured Datasets

PROC APPEND *BASE=**SAS-data-set***
DATA=*SAS-data-set* ***FORCE***;
RUN;

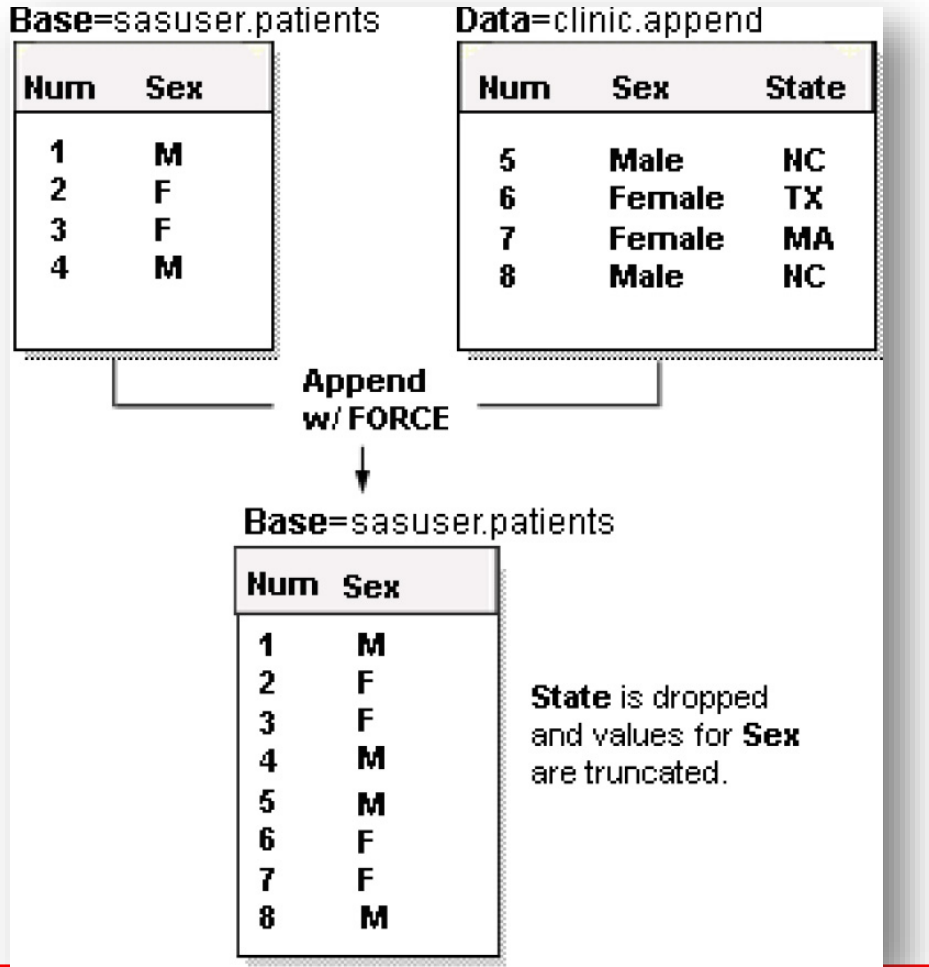
The FORCE option is needed when the DATA= dataset contains variables that meet any one of the following criteria: they are not in the BASE= dataset, variables of a different type, or of different lengths.

- If the length of a variable is longer in the DATA= dataset, SAS truncates values from the DATA= dataset to fit them into the length specified in the BASE= dataset.
- If the type of a variable in the DATA= dataset is different, SAS replaces all values for the variable in the DATA= dataset with missing values and keeps the variable type of the variable specified in the BASE= dataset.
- If the BASE= dataset contains a variable that is not in the DATA= dataset, the observations are appended, but the observations from the DATA= dataset have a missing value for the variable that was not present in the DATA= dataset.
- If the DATA= dataset contains a variable that is not in the BASE= dataset, the variable is dropped from the output.

Appending

Using the **FORCE** Option with Unlike-Structured Datasets

```
proc append base=sasuser.patients
    data=clinic.append force;
run;
```



Match-Merging

This method combines observations from two or more datasets into a single observation in a new dataset according to the values of a common variable.

```
DATA output-SAS-data-set;
   MERGE SAS-data-set-1 SAS-data-set-2;
   BY <DESCENDING> variable(s);
RUN;
```

where

output-SAS-data-set names the dataset to be created

SAS-data-set-1 and SAS-data-set-2 specify the datasets to be read.

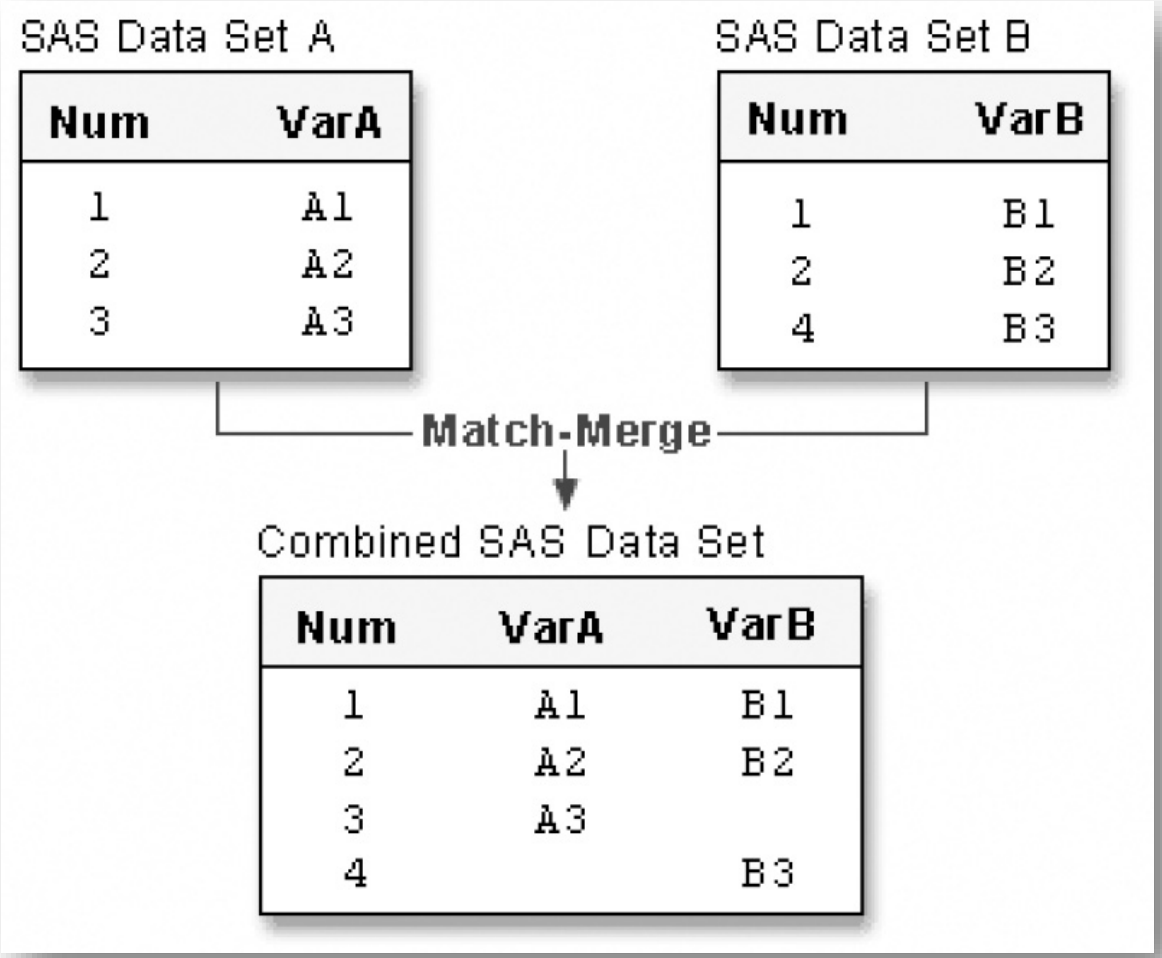
BY variable(s) specifies one or more variables that are used to match observations.

- Each input dataset in the MERGE statement *must* be sorted in the order of the values of the BY variable(s), or it must have an appropriate index. Each BY variable must have the same type in all the datasets to be merged.
- If you have more than one variable in the BY statement, DESCENDING applies only to the variable that immediately follows it. You cannot use the DESCENDING option with indexed datasets because indexes are always stored in ascending order.

Match-Merging

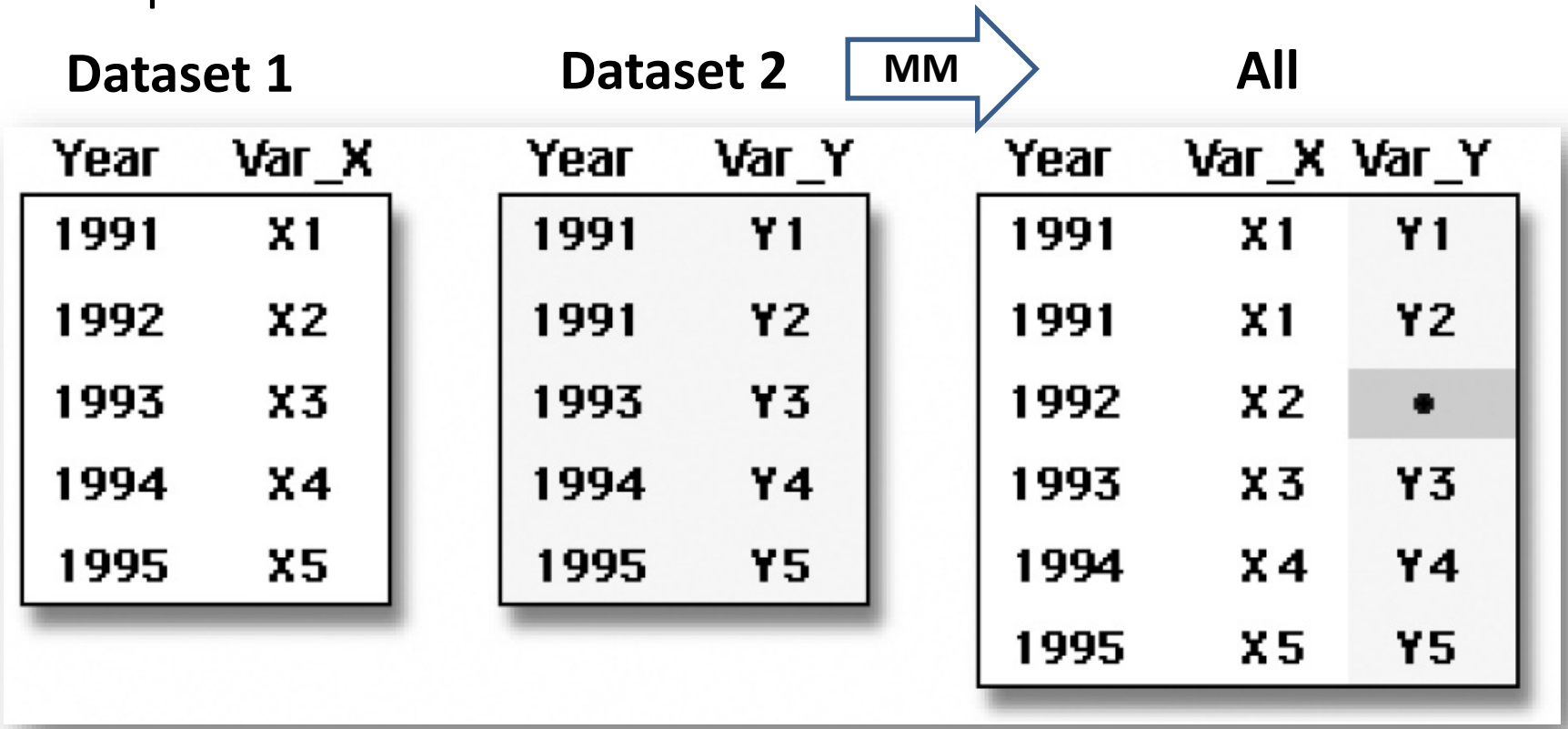
During match-merging, SAS sequentially checks each observation of each dataset to see whether the BY values match, and then writes the combined observation to the new dataset.

```
data merged;
    merge a b;
    by num;
run;
```



Match-Merging

If an input dataset doesn't have any observations for a particular value of the by-variable, then the observation in the output dataset contains missing values for the variables that are unique to that input dataset.



BY-variable: Year

Match-Merging

```
proc sort data=clinic.visit out=visit;
  by id;
run;
proc print data=visit;
run;

proc sort data=clinic.demog out=demog;
  by id;
run;
proc print data=visit;
run;

data merged;
  merge demog visit;
  by id;
run;

proc print data=merged;
run;
```

Match-Merging

Work.demog

Obs	ID	Age	Sex	Date
1	A001	21	m	05/22/07
2	A002	32	m	06/15/06
3	A003	24	f	08/17/07
4	A004	.		01/27/06
5	A005	44	f	02/24/05
6	A007	39	m	11/11/05

Work.visit

Obs	ID	Visit	SysBP	DiasBP	Weight	Date
1	A001	1	140	85	195	11/05/09
2	A001	2	138	90	198	10/13/09
3	A001	3	145	95	200	07/04/09
4	A002	1	121	75	168	04/14/09
5	A003	1	118	68	125	08/12/09
6	A003	2	112	65	123	08/21/09
7	A004	1	143	86	204	03/30/09
8	A005	1	132	76	174	02/27/09
9	A005	2	132	78	175	07/11/09
10	A005	3	134	78	176	04/16/09
11	A008	1	126	80	182	05/22/09

Work.merged

Obs	ID	Age	Sex	Date	Visit	SysBP	DiasBP	Weight
1	A001	21	m	11/05/09	1	140	85	195
2	A001	21	m	10/13/09	2	138	90	198
3	A001	21	m	07/04/09	3	145	95	200
4	A002	32	m	04/14/09	1	121	75	168
5	A003	24	f	08/12/09	1	118	68	125
6	A003	24	f	08/21/09	2	112	65	123
7	A004	.		03/30/09	1	143	86	204
8	A005	44	f	02/27/09	1	132	76	174
9	A005	44	f	07/11/09	2	132	78	175
10	A005	44	f	04/16/09	3	134	78	176
11	A007	39	m	11/11/05		.	.	.
12	A008	.		05/22/09	1	126	80	182

Match-Merging in Descending Order

```
proc sort data=clinic.visit out=visit;
  by descending id;
run;
proc print data=visit;
run;

proc sort data=clinic.demog out=demog;
  by descending id;
run;
proc print data=visit;
run;

data merged;
  merge demog visit;
  by descending id;
run;

proc print data=merged;
run;
```

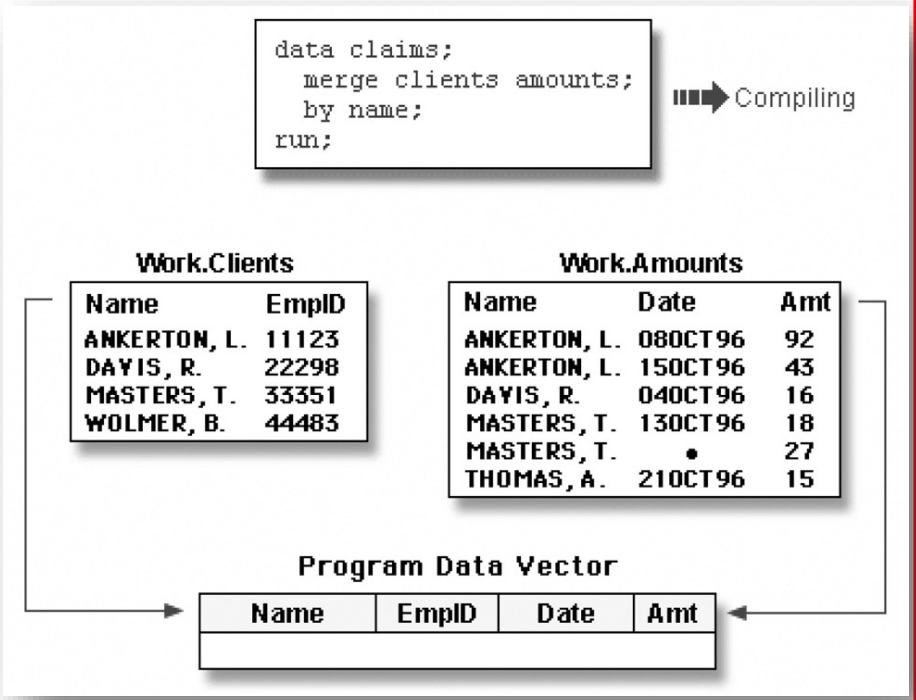
Work.merged

Obs	ID	Age	Sex	Date	Visit	SysBP	DiasBP	Weight
1	A008	.		05/22/09	1	126	80	182
2	A007	39	m	11/11/05		.	.	.
3	A005	44	f	02/27/09	1	132	76	174
4	A005	44	f	07/11/09	2	132	78	175
5	A005	44	f	04/16/09	3	134	78	176
6	A004	.		03/30/09	1	143	86	204
7	A003	24	f	08/12/09	1	118	68	125
8	A003	24	f	08/21/09	2	112	65	123
9	A002	32	m	04/14/09	1	121	75	168
10	A001	21	m	11/05/09	1	140	85	195
11	A001	21	m	10/13/09	2	138	90	198
12	A001	21	m	07/04/09	3	145	95	200

Match-Merge Processing: Compilation Phase

SAS checks the syntax of the SAS statements and compiles them (translates them into machine code). During this phase, SAS also sets up descriptor information for the output dataset and creates a program data vector (PDV), an area of memory where SAS builds your dataset, one observation at a time. To prepare to merge data sets, SAS

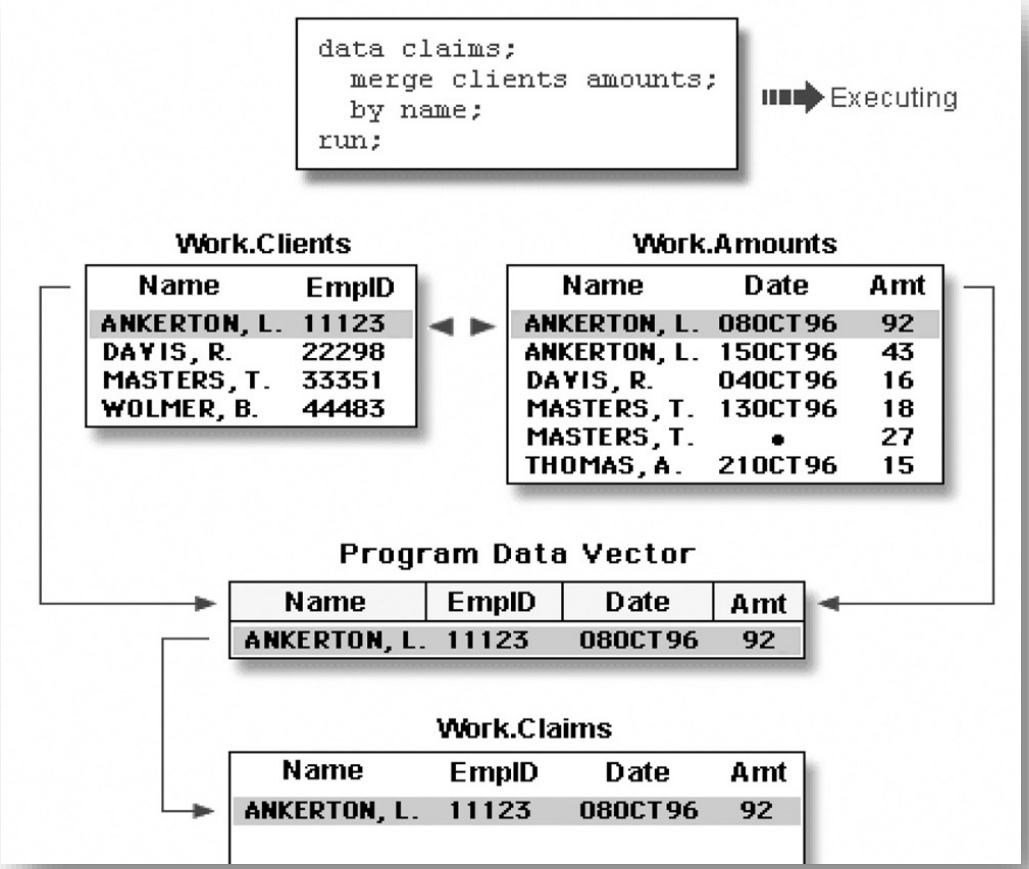
- reads the descriptor portions of the datasets that are listed in the MERGE statement
- reads the rest of the DATA step program
- creates a PDV for the merged dataset
- assigns a tracking pointer to each dataset listed in the MERGE statement.
- If variables having the same name appear in more than one dataset, the variable from the 1st dataset determines the length of the variable.



Match-Merge Processing: Execution Phase

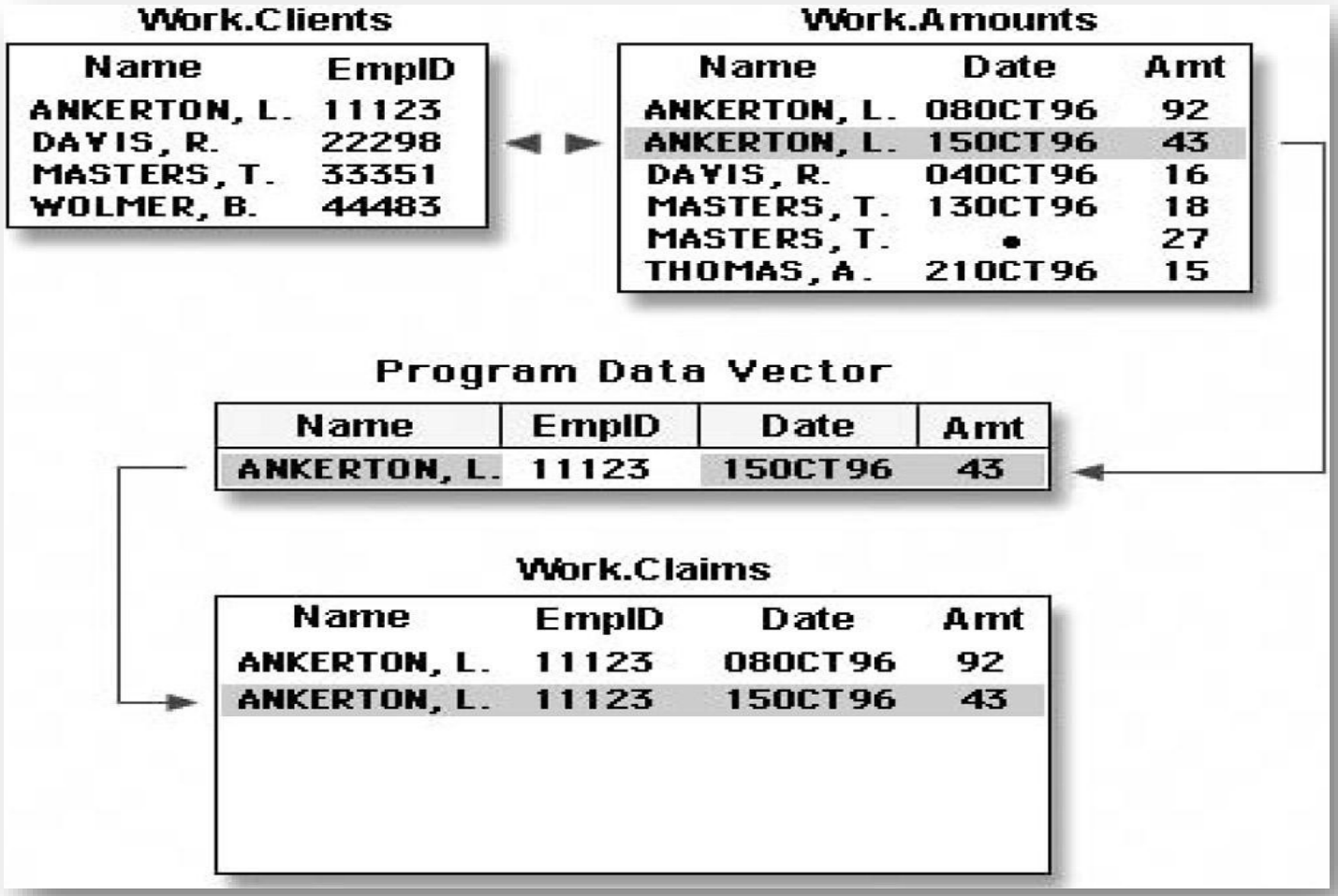
In this phase, data values are read into the appropriate variables in the program data vector. From here, the variables are written to the output dataset as a single observation. SAS sequentially match-merges observations by moving the pointers down each observation of each dataset and checking to see whether the BY variable values match.

If *Yes*, the observations are read into the PDV in the order in which the datasets appear in the MERGE statement. Values of any same-named variable are overwritten by values of the same-named variable in subsequent datasets. SAS writes the combined observation to the new dataset and retains the values in the PDV until the BY variable value changes in all the datasets



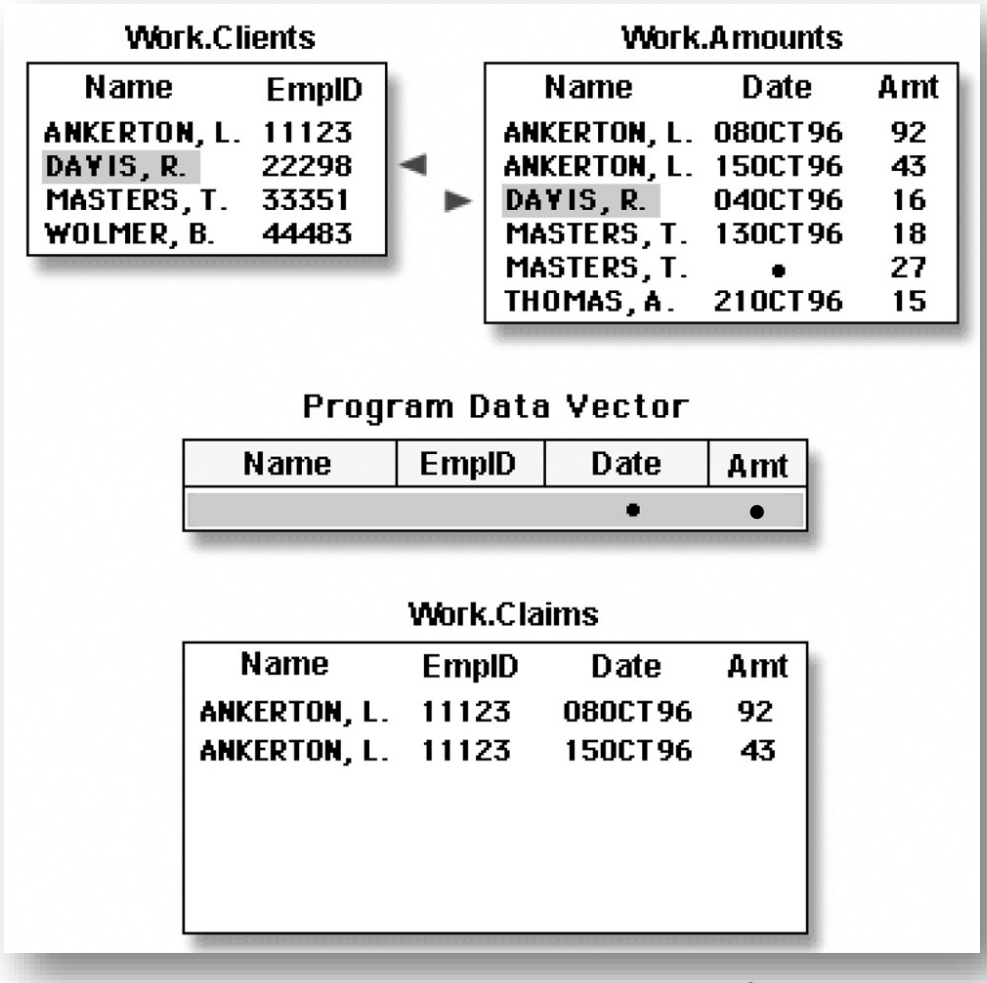
Match-Merge Processing: Execution Phase

If *No*, SAS determines which BY variable value comes first and reads the observation that contains this value into the PDV. Then the contents of the PDV are written.



Match-Merge Processing: Execution Phase

When the BY variable value changes in all the input datasets, the PDV is initialized to missing.

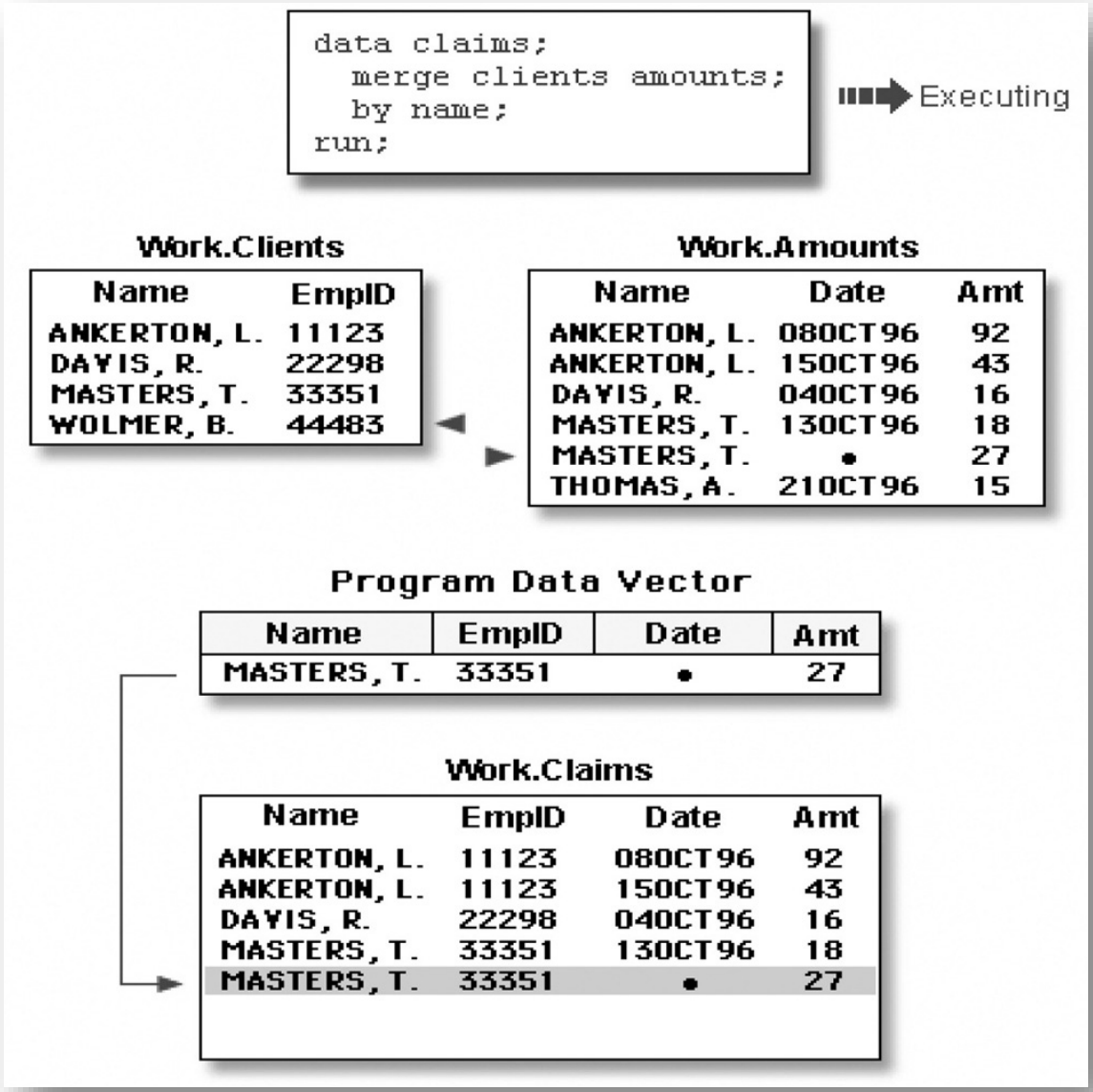


The DATA step merge continues to process every observation in each dataset until it has processed all observations in all datasets.

Match-Merge Processing: Execution Phase

Handling Unmatched Observations and Missing Values

If an observation contains missing values for a variable, then the observation in the output dataset contains the missing values as well.



Match-Merge Processing: Execution Phase

Handling Unmatched Observations and Missing Values

Work.Clients

Name	EmpID
ANKERTON, L.	11123
DAVIS, R.	22298
MASTERS, T.	33351
WOLMER, B.	44483

Work.Amounts

Name	Date	Amt
ANKERTON, L.	08OCT96	92
ANKERTON, L.	15OCT96	43
DAVIS, R.	04OCT96	16
MASTERS, T.	13OCT96	18
MASTERS, T.	.	27
THOMAS, A.	21OCT96	15

Program Data Vector

Name	EmpID	Date	Amt
THOMAS, A.		21OCT96	15

Work.Claims

Name	EmpID	Date	Amt
ANKERTON, L.	11123	08OCT96	92
ANKERTON, L.	11123	15OCT96	43
DAVIS, R.	22298	04OCT96	16
MASTERS, T.	33351	13OCT96	18
MASTERS, T.	33351	.	27
THOMAS, A.		21OCT96	15

The last observation for Wolmer would be added after the Thomas observation.

If an input dataset doesn't have a matching BY variable value, then the observation in the output data set contains missing values for the variables that are unique to that input dataset.

Results Viewer - SAS Output

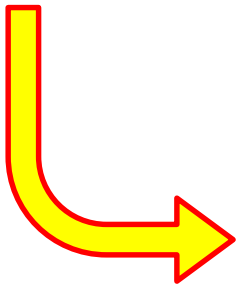
The SAS System

Obs	name	EmpID	Date	Amt
1	ANKERTON, L.	11123	13430	92
2	ANKERTON, L.	11123	13437	43
3	DAVIS, R.	22298	13426	16
4	MASTERS, T.	33351	13435	18
5	MASTERS, T.	33351	.	27
6	THOMAS, A.		13443	15
7	WOLMER, B.	44483	.	.

Renaming Variables

A problem:
The output of previous match-merging operation shows the effects of overwriting the values of a variable in the Clinic.Merged dataset. In most observations, the date is now the date of the clinic visit. In observation 11, the date is still the birth date because Clinic.Visit did not contain a matching ID value and did not contribute to the observation.

Obs	ID	Age	Sex	Date	Visit	SysBP	DiasBP	Weight
1	A001	21	m	11/05/09	1	140	85	195
2	A001	21	m	10/13/09	2	138	90	198
3	A001	21	m	07/04/09	3	145	95	200
4	A002	32	m	04/14/09	1	121	75	168
5	A003	24	f	08/12/09	1	118	68	125
6	A003	24	f	08/21/09	2	112	65	123
7	A004	.		03/30/09	1	143	86	204
8	A005	44	f	02/27/09	1	132	76	174
9	A005	44	f	07/11/09	2	132	78	175
10	A005	44	f	04/16/09	3	134	78	176
11	A007	39	m	11/11/05		.	.	.
12	A008	.		05/22/09	1	126	80	182



A dataset with values for the Date variable that mean two different things: date of birth and date of clinic visit.

Renaming Variables

Solution: using the **RENAME=** dataset option in the MERGE statement (**RENAME**=(*old-variable-name=new-variable-name*))

```
proc sort data=clinic.visit out=visit;
  by id;
run;
proc print data=visit;
run;
proc sort data=clinic.demog out=demog;
  by id;
run;
proc print data=visit;
run;
data merged_rename;
  merge demog (rename=(date=BirthDate))
        visit  (rename=(date=VisitDate));
  by id;
run;

proc print data=merged_rename;
run;
```

Match-Merging With Rename= Option

Work.demog

Obs	ID	Age	Sex	Date
1	A001	21	m	05/22/07
2	A002	32	m	06/15/06
3	A003	24	f	08/17/07
4	A004	.		01/27/06
5	A005	44	f	02/24/05
6	A007	39	m	11/11/05

Work.visit

Obs	ID	Visit	SysBP	DiasBP	Weight	Date
1	A001	1	140	85	195	11/05/09
2	A001	2	138	90	198	10/13/09
3	A001	3	145	95	200	07/04/09
4	A002	1	121	75	168	04/14/09
5	A003	1	118	68	125	08/12/09
6	A003	2	112	65	123	08/21/09
7	A004	1	143	86	204	03/30/09
8	A005	1	132	76	174	02/27/09
9	A005	2	132	78	175	07/11/09
10	A005	3	134	78	176	04/16/09
11	A008	1	126	80	182	05/22/09

Work.merged_rename

lts Viewer - SAS Output

The SAS System

ID	Age	Sex	BirthDate	Visit	SysBP	DiasBP	Weight	VisitDate
A001	21	m	05/22/07	1	140	85	195	11/05/09
A001	21	m	05/22/07	2	138	90	198	10/13/09
A001	21	m	05/22/07	3	145	95	200	07/04/09
A002	32	m	06/15/06	1	121	75	168	04/14/09
A003	24	f	08/17/07	1	118	68	125	08/12/09
A003	24	f	08/17/07	2	112	65	123	08/21/09
A004	.		01/27/06	1	143	86	204	03/30/09
A005	44	f	02/24/05	1	132	76	174	02/27/09
A005	44	f	02/24/05	2	132	78	175	07/11/09
A005	44	f	02/24/05	3	134	78	176	04/16/09
A007	39	m	11/11/05		.	.	.	.
A008	.		.	1	126	80	182	05/22/09

Excluding Unmatched Observations

By default, DATA step match-merging combines all observations in all input datasets. However, you may exclude the unmatched observations by using the **IN=** dataset option and the subsetting **IF** statement in your DATA step.

- The IN= dataset option creates and names a temporary variable that indicates whether the dataset contributed data to the current observation. The value of the variable is **1** if the dataset contributed data to the current observation. Otherwise, its value is **0.**
- The IN= variable is a variable available to program statements during the DATA step, but it is not included in the SAS dataset that is being created.
- The subsetting IF statement checks the IN= values and writes to the merged dataset only matching observations.

Excluding Unmatched Observations

```
data merged_excluded;
    merge demog (in=indemog
                 rename=(date=BirthDate))
          visit (in=invisit
                 rename=(date=VisitDate));
    by id;
    if indemog=1 and invisit=1;
run;
proc print data=merged_excluded;
run;
```

Obs	ID	Age	Sex	BirthDate	Visit	SysBP	DiasBP	Weight	VisitDate
1	A001	21	m	05/22/07	1	140	85	195	11/05/09
2	A001	21	m	05/22/07	2	138	90	198	10/13/09
3	A001	21	m	05/22/07	3	145	95	200	07/04/09
4	A002	32	m	06/15/06	1	121	75	168	04/14/09
5	A003	24	f	08/17/07	1	118	68	125	08/12/09
6	A003	24	f	08/17/07	2	112	65	123	08/21/09
7	A004	.		01/27/06	1	143	86	204	03/30/09
8	A005	44	f	02/24/05	1	132	76	174	02/27/09
9	A005	44	f	02/24/05	2	132	78	175	07/11/09
10	A005	44	f	02/24/05	3	134	78	176	04/16/09

Selecting Variables

As with reading raw data or reading SAS datasets, you can specify the variables you want to drop or keep by using the DROP= and KEEP= dataset options.

```
data merged_excluded_selected (drop=id);
    merge demog (in=indemog
                 rename=(date=BirthDate))
          visit (drop=weight in=invisit
                 rename=(date=VisitDate));
    by id;
    if indemog=1 and invisit=1;
run;
proc print data=merged_excluded_selected;
run;
```

The SAS System

Obs	Age	Sex	BirthDate	Visit	SysBP	DiasBP	VisitDate
1	21	m	05/22/07	1	140	85	11/05/09
2	21	m	05/22/07	2	138	90	10/13/09
3	21	m	05/22/07	3	145	95	07/04/09
4	32	m	06/15/06	1	121	75	04/14/09
5	24	f	08/17/07	1	118	68	08/12/09
6	24	f	08/17/07	2	112	65	08/21/09
7	.		01/27/06	1	143	86	03/30/09
8	44	f	02/24/05	1	132	76	02/27/09
9	44	f	02/24/05	2	132	78	07/11/09
10	44	f	02/24/05	3	134	78	04/16/09