Chapter 7
Combining Data Sets

Arthur Li

Vertically Combining Data Sets
Concatenating Data Sets

- Concatenating: combining multiple data sets, one after the other, into a single data set.

```
SET data-set(s);
```

Record1

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

Record2

	Name	Course	Grade
1	Joe	ENGLISH	96
2	John	ENGLISH	89
3	Mary	ENGLISH	78
4	Tom	ENGLISH	.
5	Dave	ENGLISH	98

	Name	Course	Score	Grade
1	John	MATH	90	.
2	John	MATH	85	.
3	Mary	MATH	.	.
4	Tom	MATH	92	.
5	Joe	ENGLISH	.	96
6	John	ENGLISH	.	89
7	Mary	ENGLISH	.	78
8	Tom	ENGLISH	.	.
9	Dave	ENGLISH	.	98

Concatenating Data Sets

Program 7.1:

```
data ex7_1;
    set record1 record2;
run;
title 'Concatenating record1 and record2';
proc print data=ex7_1;
run;
```

```
                Concatenating record1 and record2

      Obs      Name      Course      Score      Grade

       1       John       MATH         90           .
       2       John       MATH         85           .
       3       Mary       MATH          .           .
       4       Tom        MATH         92           .
       5       Joe        ENGL          .          96
       6       John       ENGL          .          89
       7       Mary       ENGL          .          78
       8       Tom        ENGL          .           .
       9       Dave       ENGL          .          98
```

Concatenating Data Sets

Log from Program 7.1:

```
629  data ex7_1;
630      set record1 record2;
631  run;

WARNING: Multiple lengths were specified for the variable Course
         by input data set(s). This may cause truncation of data.
NOTE: There were 4 observations read from the data set
      WORK.RECORD1.
NOTE: There were 5 observations read from the data set
      WORK.RECORD2.
NOTE: The data set WORK.EX7_1 has 9 observations and 4 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.00 seconds
```

Concatenating Data Sets

- If the common variables have different…
  - *type* attributes → an error message
  - *length*, *label*, *format*, or *informat* attributes → SAS uses the attribute from the first data set

Concatenating Data Sets

- The RENAME= data set option:

```
RENAME=(old-name-1=new-name-1
        <...old-name-n=new-name-n>)
```

Concatenating Data Sets

Program 7.2:

```
data ex7_2;
    length Course $ 7;
    set record1 record2(rename=(grade=score));
run;
title 'Renaming the GRADE variable before concatenating the
data';
proc print data=ex7_2;
run;
```

```
 Renaming the GRADE variable before concatenating the data


             Obs     Course     Name     Score
              1      MATH       John       90
              2      MATH       John       85
              3      MATH       Mary        .
              4      MATH       Tom        92
              5      ENGLISH    Joe        96
              6      ENGLISH    John       89
              7      ENGLISH    Mary       78
              8      ENGLISH    Tom         .
              9      ENGLISH    Dave       98
```

Interleaving Data Sets

- Interleaving: utilizing BY-group processing with the SET statement to combine two or more data sets vertically

```
SET data-set(s);
BY variable(s);
```

BY variable: NAME

Record1

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

Record2

	Name	Course	Grade
1	Joe	ENGLISH	96
2	John	ENGLISH	89
3	Mary	ENGLISH	78
4	Tom	ENGLISH	.
5	Dave	ENGLISH	98

	Name	Course	Score	Grade
1	Dave	ENGLISH	.	98
2	Joe	ENGLISH	.	96
3	John	MATH	90	.
4	John	MATH	85	.
5	John	ENGLISH	.	89
6	Mary	MATH	.	.
7	Mary	ENGLISH	.	78
8	Tom	MATH	92	.
9	Tom	ENGLISH	.	.

Interleaving Data Sets

Program 7.3:

```
proc sort data=record1 out=record1_sort;
    by Name;
run;

proc sort data=record2 out=record2_sort;
    by Name;
run;

data ex7_3;
    length Course $ 7;
    set record1_sort record2_sort;
    by Name;
run;
```

Interleaving Data Sets

```
            Interleaving record1 and record2

      Obs     Course      Name      Score     Grade

       1      ENGLISH     Dave         .        98
       2      ENGLISH     Joe          .        96
       3      MATH        John        90         .
       4      MATH        John        85         .
       5      ENGLISH     John         .        89
       6      MATH        Mary         .         .
       7      ENGLISH     Mary         .        78
       8      MATH        Tom         92         .
       9      ENGLISH     Tom          .         .
```

Horizontally Combining Data Sets
One-to-One Reading

- One-to-one reading utilizes multiple SET statements to combine observations from two or more input data sets independently, forming one observation that contains all of the variables from each contributing data set.

- Observations are combined based on their relative position in each data set.

```
SET data-set-1;
SET data-set-2;
```

One-to-One Reading

Record1:

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

Record2:

	Name	Course	Grade
1	Joe	ENGLISH	96
2	John	ENGLISH	89
3	Mary	ENGLISH	78
4	Tom	ENGLISH	.
5	Dave	ENGLISH	98

Ex7_4:

	Name	Course	Score
	John	MATH	90

Program 7.4:

```
data ex7_4;
    set record1;
    set record2;
run;
```

One-to-One Reading

Record1:

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

Record2:

	Name	Course	Grade
1	Joe	ENGLISH	96
2	John	ENGLISH	89
3	Mary	ENGLISH	78
4	Tom	ENGLISH	.
5	Dave	ENGLISH	98

Ex7_4:

	Name	Course	Score	Grade
1	Joe	ENGL	90	96

Program 7.4:

```
data ex7_4;
    set record1;
    set record2;
run;
```

One-to-One Reading

Record1:

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

Record2:

	Name	Course	Grade
1	Joe	ENGLISH	96
2	John	ENGLISH	89
3	Mary	ENGLISH	78
4	Tom	ENGLISH	.
5	Dave	ENGLISH	98

Ex7_4:

	Name	Course	Score	Grade
1	Joe	ENGL	90	96
2	John	MATH	85	.

Program 7.4:

```
data ex7_4;
  set record1;
  set record2;
run;
```

One-to-One Reading

Record1:

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

Record2:

	Name	Course	Grade
1	Joe	ENGLISH	96
2	John	ENGLISH	89
3	Mary	ENGLISH	78
4	Tom	ENGLISH	.
5	Dave	ENGLISH	98

Ex7_4:

	Name	Course	Score	Grade
1	Joe	ENGL	90	96
2	John	ENGL	85	89

Program 7.4:

```
data ex7_4;
    set record1;
    set record2;
run;
```

One-to-One Reading

Record1:

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

Record2:

	Name	Course	Grade
1	Joe	ENGLISH	96
2	John	ENGLISH	89
3	Mary	ENGLISH	78
4	Tom	ENGLISH	.
5	Dave	ENGLISH	98

Ex7_4:

	Name	Course	Score	Grade
1	Joe	ENGL	90	96
2	John	ENGL	85	89
3	Mary	MATH	.	.

Program 7.4:

```
data ex7_4;
  set record1;
  set record2;
run;
```

One-to-One Reading

Record1:

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

Record2:

	Name	Course	Grade
1	Joe	ENGLISH	96
2	John	ENGLISH	89
3	Mary	ENGLISH	78
4	Tom	ENGLISH	.
5	Dave	ENGLISH	98

Ex7_4:

	Name	Course	Score	Grade
1	Joe	ENGL	90	96
2	John	ENGL	85	89
3	Mary	ENGL	.	78

Program 7.4:

```
data ex7_4;
    set record1;
    set record2;
run;
```

One-to-One Reading

Record1:

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

Record2:

	Name	Course	Grade
1	Joe	ENGLISH	96
2	John	ENGLISH	89
3	Mary	ENGLISH	78
4	Tom	ENGLISH	.
5	Dave	ENGLISH	98

Ex7_4:

	Name	Course	Score	Grade
1	Joe	ENGL	90	96
2	John	ENGL	85	89
3	Mary	ENGL	.	78
4	Tom	MATH	92	.

Program 7.4:

```
data ex7_4;
   set record1;
   set record2;
run;
```

One-to-One Reading

Record1:

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

Record2:

	Name	Course	Grade
1	Joe	ENGLISH	96
2	John	ENGLISH	89
3	Mary	ENGLISH	78
4	Tom	ENGLISH	.
5	Dave	ENGLISH	98

Ex7_4:

	Name	Course	Score	Grade
1	Joe	ENGL	90	96
2	John	ENGL	85	89
3	Mary	ENGL	.	78
4	Tom	ENGL	92	.

Program 7.4:

```
data ex7_4;
    set record1;
    set record2;
run;
```

One-to-One Reading

Record1:

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

The end-of-file marker

Record2:

	Name	Course	Grade
1	Joe	ENGLISH	96
2	John	ENGLISH	89
3	Mary	ENGLISH	78
4	Tom	ENGLISH	.
5	Dave	ENGLISH	98

Ex7_4:

	Name	Course	Score	Grade
1	Joe	ENGL	90	96
2	John	ENGL	85	89
3	Mary	ENGL	.	78
4	Tom	ENGL	92	.

Program 7.4:

```
data ex7_4;
     set record1;
     set record2;
run;
```

One-to-One Reading

- Using the IF/THEN statement with a SET statement to merge one single value with a data set.

Record1:

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

	Name	Course	Score	Mean_Score
1	John	MATH	90	89
2	John	MATH	85	89
3	Mary	MATH	.	89
4	Tom	MATH	92	89

One-to-One Reading

Program 7.5:

```
proc means data=record1 noprint;
    var score;
    output out=record1_mean mean=mean_score;
run;

proc print data=record1_mean;
run;
```

```
                                         mean_
     Obs      _TYPE_       _FREQ_        score

      1          0            4            89
```

One-to-One Reading

Program 7.5 (continue):

```
data ex7_5;
    set record1;
    if _n_=1 then set record1_mean(keep=mean_score);
run;

title 'Use One-to-one reading to merge the mean score
with record1';
proc print data=ex7_5;
run;
```

```
 Use One-to-one reading to merge the mean score with record1

                                                        mean_
           Obs      Name      Course      Score      score

             1      John       MATH         90         89
             2      John       MATH         85         89
             3      Mary       MATH          .         89
             4      Tom        MATH         92         89
```

One-to-One Reading

Program 7.5 (continue):

```
data ex7_5;
    set record1;
    if _n_=1 then set record1_mean(keep=mean_score);
run;

title 'Use One-to-one reading to merge th
with record1';
proc print data=ex7_5;
run;
```

Using SET and IF statements together ensures that SAS will not encounter an end-of-file marker that would abruptly terminate the data step.

```
 Use One-to-one reading to merge the mear

                                                    _
          Obs     Name     Course     Score     score

           1      John      MATH        90        89
           2      John      MATH        85        89
           3      Mary      MATH         .        89
           4      Tom       MATH        92        89
```

One-to-One Merging

- One-to-one merging: similar to the results obtained from one-to-one reading, except that one-to-one merging continues processing all observations in all data sets that were named in the MERGE statement

MERGE data-set(s);

One-to-One Merging

Record1:

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

Record2:

	Name	Course	Grade
1	Joe	ENGLISH	96
2	John	ENGLISH	89
3	Mary	ENGLISH	78
4	Tom	ENGLISH	.
5	Dave	ENGLISH	98

Ex7_4:

	Name	Course	Score	Grade
1	Joe	ENGL	90	96
2	John	ENGL	85	89
3	Mary	ENGL	.	78
4	Tom	ENGL	92	.
5	Dave	ENGL	.	98

Program 7.6:

```
data ex7_6;
    merge record1 record2;
run;
```

Match-merging

- Match-merging: combines observations from two or more SAS data sets into a single observation according to the values of one or more common variables

```
MERGE data-set(s);
BY variable(s);
```

Match-merging

- ❖One-to-one matching: a single observation in one data set relating to a single observation from another based on the value of one or more common variables.

- ❖One-to-many matching: a single observation in one data set is associated with multiple observations from another data set.

- ❖Many-to-many matching: multiple observations from each input data set can be related based on values of one or more common variables.

Match-merging

Record1:

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

Record1_sort:

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

Record2:

	Name	Course	Grade
1	Joe	ENGLISH	96
2	John	ENGLISH	89
3	Mary	ENGLISH	78
4	Tom	ENGLISH	.
5	Dave	ENGLISH	98

Record2_sort:

	Name	Course	Grade
1	Dave	ENGLISH	98
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	78
5	Tom	ENGLISH	.

```
proc sort data=record1 out=record1_sort;
    by Name;
run;

proc sort data=record2 out=record2_sort;
    by Name;
run;
```

Match-merging

	Name	Course	Score	Grade
1	Dave	ENGL	.	98
2	Joe	ENGL	.	96
3	John	ENGL	90	89
4	John	MATH	85	89
5	Mary	ENGL	.	78
6	Tom	ENGL	92	.

Record1_sort:

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

Record2_sort:

	Name	Course	Grade
1	Dave	ENGLISH	98
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	78
5	Tom	ENGLISH	.

```
data ex7_7;
    merge record1_sort
          record2_sort;
    by Name;
run;
```

Match-merging

```
data ex7_7;
    merge record1_sort
          record2_sort;
    by Name;
run;
```

_N_ (D)	FIRST.NAME (D)	LAST.NAME (D)
1	1	1

NAME (K)	COURSE (K)	SCORE (K)	GRADE (K)
		.	.

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

	Name	Course	Grade
1	Dave	ENGLISH	98
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	78
5	Tom	ENGLISH	.

1st iteration:

- During the execution phase, SAS determine which BY group should appear first in the output data set.

Match-merging

```
data ex7_7;
  merge record1_sort
        record2_sort;
    by Name;
run;
```

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

_N_ D	FIRST.NAME D	LAST.NAME D
1	1	1

NAME K	COURSE K	SCORE K	GRADE K
Dave	ENGL	.	98

	Name	Course	Grade
1	Dave	ENGLISH	98
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	78
5	Tom	ENGLISH	.

1st iteration:

- The MERGE statement executes.
- FIRST.NAME ←1
- LAST.NAME ← 1

Match-merging

```
data ex7_7;
  merge record1_sort
        record2_sort;
    by Name;
run;
```

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

_N_	D	FIRST.NAME	D	LAST.NAME	D
1		1		1	

	Name	Course	Grade
1	Dave	ENGLISH	98
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	78
5	Tom	ENGLISH	.

NAME	K	COURSE	K	SCORE	K	GRADE	K
Dave		ENGL		.		98	

1st iteration:

❖The MERGE statement executes.

❖RECORD2_SORT→PDV

Match-merging

```
data ex7_7;
    merge record1_sort
          record2_sort;
    by Name;
run;
```

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

	Name	Course	Grade
1	Dave	ENGLISH	98
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	78
5	Tom	ENGLISH	.

_N_	D	FIRST.NAME	D	LAST.NAME	D
1		1		1	

NAME	K	COURSE	K	SCORE	K	GRADE	K
Dave		ENGL		.		98	

	Name	Course	Score	Grade
1	Dave	ENGL	.	98

1st iteration:

❖The implicit OUTPUT executes.

Match-merging

```
data ex7_7;
    merge record1_sort
          record2_sort;
    by Name;
run;
```

_N_ D	FIRST.NAME D	LAST.NAME D
2	1	1

NAME K	COURSE K	SCORE K	GRADE K
		.	.

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

	Name	Course	Grade
1	Dave	ENGLISH	98
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	78
5	Tom	ENGLISH	.

	Name	Course	Score	Grade
1	Dave	ENGL	.	98

2nd iteration:

❖When SAS has read all observations in the current BY group from all data sets, it sets all the non-automatic variables to missing in the PDV

Match-merging

```
data ex7_7;
  merge record1_sort
        record2_sort;
    by Name;
run;
```

_N_	D	FIRST.NAME	D	LAST.NAME	D
2		1		1	

NAME	K	COURSE	K	SCORE	K	GRADE	K
Joe		ENGL		.		96	

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

	Name	Course	Grade
1	Dave	ENGLISH	98
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	78
5	Tom	ENGLISH	.

	Name	Course	Score	Grade
1	Dave	ENGL	.	98

2nd iteration:

- The MERGE statement executes.
- FIRST.NAME ←1
- LAST.NAME ← 1

Match-merging

```
data ex7_7;
  merge record1_sort
        record2_sort;
    by Name;
run;
```

_N_ D	FIRST.NAME D	LAST.NAME D
2	1	1

NAME K	COURSE K	SCORE K	GRADE K
Joe	ENGL	.	96

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

	Name	Course	Grade
1	Dave	ENGLISH	98
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	78
5	Tom	ENGLISH	.

	Name	Course	Score	Grade
1	Dave	ENGL	.	98

2nd iteration:

- The MERGE statement executes.
- RECORD2_SORT→PDV

Match-merging

```
data ex7_7;
    merge record1_sort
          record2_sort;
    by Name;
run;
```

_N_	D	FIRST.NAME	D	LAST.NAME	D
2		1		1	

NAME	K	COURSE	K	SCORE	K	GRADE	K
Joe		ENGL		.		96	

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

	Name	Course	Grade
1	Dave	ENGLISH	98
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	78
5	Tom	ENGLISH	.

	Name	Course	Score	Grade
1	Dave	ENGL	.	98
2	Joe	ENGL	.	96

2nd iteration:

❖The implicit OUTPUT executes.

Match-merging

```
data ex7_7;
    merge record1_sort
          record2_sort;
    by Name;
run;
```

_N_ (D)	FIRST.NAME (D)	LAST.NAME (D)
3	1	1

NAME (K)	COURSE (K)	SCORE (K)	GRADE (K)
		.	.

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

	Name	Course	Grade
1	Dave	ENGLISH	98
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	78
5	Tom	ENGLISH	.

	Name	Course	Score	Grade
1	Dave	ENGL	.	98
2	Joe	ENGL	.	96

3rd iteration:

- When SAS has read all observations in the current BY group from all data sets, it sets all the non-automatic variables to missing in the PDV

Match-merging

```
data ex7_7;
  merge record1_sort
        record2_sort;
    by Name;
run;
```

_N_ D	FIRST.NAME D	LAST.NAME D
3	1	0

NAME K	COURSE K	SCORE K	GRADE K
John	MATH	90	.

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

	Name	Course	Grade
1	Dave	ENGLISH	98
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	78
5	Tom	ENGLISH	.

	Name	Course	Score	Grade
1	Dave	ENGL	.	98
2	Joe	ENGL	.	96

3rd iteration:

- The MERGE statement executes.
- FIRST.NAME ← 1
- LAST.NAME ← 0

Match-merging

```
data ex7_7;
  merge record1_sort
        record2_sort;
    by Name;
run;
```

_N_	D	FIRST.NAME	D	LAST.NAME	D
3		1		0	

NAME	K	COURSE	K	SCORE	K	GRADE	K
John		MATH		90		.	

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

	Name	Course	Grade
1	Dave	ENGLISH	98
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	78
5	Tom	ENGLISH	.

	Name	Course	Score	Grade
1	Dave	ENGL	.	98
2	Joe	ENGL	.	96

3rd iteration:

- The MERGE statement executes.
- RECORD1_SORT→PDV

Match-merging

```
data ex7_7;
  merge record1_sort
        record2_sort;
    by Name;
run;
```

_N_	D	FIRST.NAME	D	LAST.NAME	D
3		1		0	

NAME	K	COURSE	K	SCORE	K	GRADE	K
John		ENGL		90		89	

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

	Name	Course	Grade
1	Dave	ENGLISH	98
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	78
5	Tom	ENGLISH	.

	Name	Course	Score	Grade
1	Dave	ENGL	.	98
2	Joe	ENGL	.	96

3rd iteration:

- The MERGE statement executes.
- RECORD2_SORT→PDV

Match-merging

```
data ex7_7;
    merge record1_sort
          record2_sort;
    by Name;
run;
```

_N_	FIRST.NAME	LAST.NAME
3	1	0

NAME	COURSE	SCORE	GRADE
John	ENGL	90	89

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

	Name	Course	Grade
1	Dave	ENGLISH	98
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	78
5	Tom	ENGLISH	.

	Name	Course	Score	Grade
1	Dave	ENGL	.	98
2	Joe	ENGL	.	96
3	John	ENGL	90	89

3rd iteration:

❖The implicit OUTPUT executes.

Match-merging

```
data ex7_7;
    merge record1_sort
          record2_sort;
    by Name;
run;
```

_N_	FIRST.NAME	LAST.NAME
4	1	0

NAME	COURSE	SCORE	GRADE
John	ENGL	90	89

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

	Name	Course	Grade
1	Dave	ENGLISH	98
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	78
5	Tom	ENGLISH	.

	Name	Course	Score	Grade
1	Dave	ENGL	.	98
2	Joe	ENGL	.	96
3	John	ENGL	90	89

4th iteration:

- The variables in the PDV are retained because we didn’t finish reading all the observations within this BY group.

Match-merging

```
data ex7_7;
   merge record1_sort
         record2_sort;
   by Name;
run;
```

_N_	FIRST.NAME	LAST.NAME
4	0	1

NAME	COURSE	SCORE	GRADE
John	MATH	85	89

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

	Name	Course	Grade
1	Dave	ENGLISH	98
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	78
5	Tom	ENGLISH	.

	Name	Course	Score	Grade
1	Dave	ENGL	.	98
2	Joe	ENGL	.	96
3	John	ENGL	90	89

4th iteration:

- The MERGE statement executes.
- FIRST.NAME ←0
- LAST.NAME ← 1

Match-merging

```
data ex7_7;
  merge record1_sort
        record2_sort;
    by Name;
run;
```

_N_	D	FIRST.NAME	D	LAST.NAME	D
4		0		1	

NAME	K	COURSE	K	SCORE	K	GRADE	K
John		MATH		85		89	

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

	Name	Course	Grade
1	Dave	ENGLISH	98
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	78
5	Tom	ENGLISH	.

	Name	Course	Score	Grade
1	Dave	ENGL	.	98
2	Joe	ENGL	.	96
3	John	ENGL	90	89

4th iteration:

- The MERGE statement executes.
- RECORD1_SORT→PDV
- There no record to read in RECORD2_SORT

Match-merging

```
data ex7_7;
    merge record1_sort
          record2_sort;
    by Name;
run;
```

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

_N_ D	FIRST.NAME D	LAST.NAME D
4	0	1

	Name	Course	Grade
1	Dave	ENGLISH	98
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	78
5	Tom	ENGLISH	.

NAME K	COURSE K	SCORE K	GRADE K
John	MATH	85	89

	Name	Course	Score	Grade
1	Dave	ENGL	.	98
2	Joe	ENGL	.	96
3	John	ENGL	90	89
4	John	MATH	85	89

4th iteration:

❖The implicit OUTPUT executes.

Match-merging

```
data ex7_7;
    merge record1_sort
          record2_sort;
    by Name;
run;
```

_N_	FIRST.NAME	LAST.NAME
D	D	D
5	0	1

NAME	COURSE	SCORE	GRADE
K	K	K	K
		.	.

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

	Name	Course	Grade
1	Dave	ENGLISH	98
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	78
5	Tom	ENGLISH	.

	Name	Course	Score	Grade
1	Dave	ENGL	.	98
2	Joe	ENGL	.	96
3	John	ENGL	90	89
4	John	MATH	85	89

5th iteration:

❖ When SAS has read all observations in the current BY group from all data sets, it sets all the non-automatic variables to missing in the PDV

Match-merging

- The number of observations in the combined data set equals the sum of the largest number of observations in each BY group among all the input data sets.

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

	Name	Course	Grade
1	Dave	ENGLISH	98
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	78
5	Tom	ENGLISH	.

	Name	Course	Score	Grade
1	Dave	ENGL	.	98
2	Joe	ENGL	.	96
3	John	ENGL	90	89
4	John	MATH	85	89
5	Mary	ENGL	.	78
6	Tom	ENGL	92	.

Match-merging

Program 7.8

```
data ex7_8;
   merge record1_sort(drop=course
             rename=(score=Math_score))
         record2_sort(drop=course
         rename=(grade=English_score));
    by Name;
run;
title 'An improved approach to merge
record1 and record2';
proc print data=ex7_8;
run;
```

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

	Name	Course	Grade
1	Dave	ENGLISH	98
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	78
5	Tom	ENGLISH	.

```
     An improved approach to merge record1 and record2
                                 Math_     English_
           Obs     Name          score       score

             1     Dave            .          98
             2     Joe             .          96
             3     John           90          89
             4     John           85          89
             5     Mary            .          78
             6     Tom            92           .
```

Match-merging

- ❖Using the IN= data set option to include/exclude observations:

IN=variable

- ❑ The VARIABLE a temporary variable
- ❑ The VARIABLE equals 1 if the input data set contributes to the current observation in the PDV; otherwise, its value equals 0.
- ❑ The IN= data set option can also be used with the MERGE, SET, MODIFY, and UPDATE statements.

Match-merging

Program 7.9

```
data ex7_9;
   merge record1_sort(drop=course
           rename=(score=Math_score)
           in=in_record1)
         record2_sort(drop=course
           rename=(grade=English_score)
           in=in_record2);
   by Name;
   if in_record1 and in_record2;
run;
title 'Excluding unmatched observations';
proc print data=ex7_9;
run;
```

	Name	Course	Score
1	John	MATH	90
2	John	MATH	85
3	Mary	MATH	.
4	Tom	MATH	92

	Name	Course	Grade
1	Dave	ENGLISH	98
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	78
5	Tom	ENGLISH	.

```
                  Excluding unmatched observations

                               Math_     English_
                  Obs    Name  score      score

                   1     John    90        89
                   2     John    85        89
                   3     Mary     .        78
                   4     Tom     92         .
```

Updating Data Sets

- Use the UPDATE statement to update a master data set with a transaction data set.

```
UPDATE master-data-set
            transaction-data-set;
BY variable(s);
```

- The master-data-set contains the original information.
- The transaction-data-set contains new information.
- # of obs. in the resulting data set =
  # of obs. in the master data set +
  # unmatched obs. in the transaction data set.

Updating Data Sets

- Use the UPDATE statement to update a master data set with a transaction data set.

```
UPDATE master-data-set
                transaction-data-set;
BY variable(s);
```

  - When the transaction data set contains duplicate values of the BY variable, only the last values that are copied to the PDV are written to the output data set.
  - If the master data set contains duplicate values of the BY variable, only the first observation in the master data set is updated.

Updating Data Sets

- Use the UPDATE statement to update a master data set with a transaction data set.

```
UPDATE master-data-set
              transaction-data-set;
BY variable(s);
```

- Updating data sets is similar to match-merging with the MERGE statement.
- Missing values in the transaction data set do not replace the existing values in the master data set.

Updating Data Sets

Record2_sort:

	Name	Course	Grade
1	Dave	ENGLISH	98
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	78
5	Tom	ENGLISH	.

Record3:

	Name	Grade
1	Joe	.
2	John	.
3	Mary	82
4	Tom	90
5	Dave	97

Record3_sort:

	Name	Grade
1	Dave	97
2	Joe	.
3	John	.
4	Mary	82
5	Tom	90

```
proc sort data=record3
          out=record3_sort;
    by Name;
run;
```

Updating Data Sets

Record2_sort:

	Name	Course	Grade
1	Dave	ENGLISH	98
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	78
5	Tom	ENGLISH	.

Record3_sort:

	Name	Grade
1	Dave	97
2	Joe	.
3	John	.
4	Mary	82
5	Tom	90

Ex7_10:

	Name	Course	Grade
1	Dave	ENGLISH	97
2	Joe	ENGLISH	96
3	John	ENGLISH	89
4	Mary	ENGLISH	82
5	Tom	ENGLISH	90

```
data ex7_10;
    update record2_sort
           record3_sort;
    by name;
run;
```