Combining SAS Data Sets

STAT2005

Chapter 12

Introduction

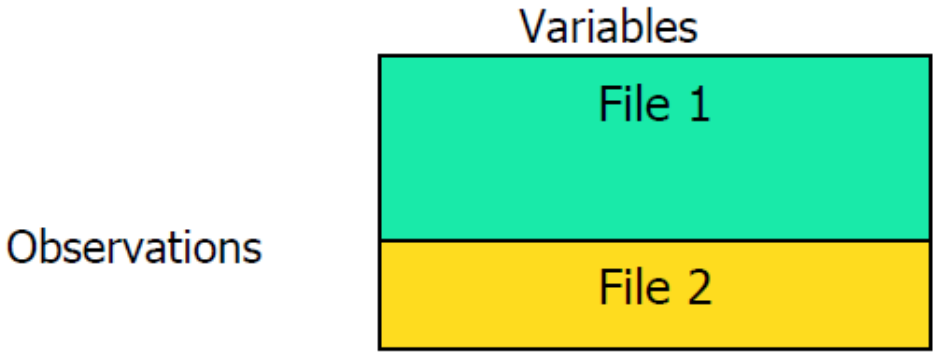
Sometimes we store data in separate SAS data sets and may want to combine them.

There are five ways to combine SAS data sets designed for different purposes:

- Concatenation
- Interleaving
- One-to-One Merge
- Matched Merge
- Updating

Method 1: Concatenation

This is applicable if we want to stack all the records of two or more data sets into one data set. For example, we want to stack the scores of all students from two classes.



The SET statement can be used in a DATA step to read observations from one or more SAS data sets. SAS treats all the data sets listed in the SET statement as one aggregate SAS data set.

Concatenation requires an extended form of SET statement:

```
SET {[fileref.]dset[(IN=var other_opts)]}…;
```

`fileref` : optional reference to a permanent SAS data set library.

`dset` : name of a SAS data set

`IN = var`: create a new variable `var` which is equal to `1` if the current observation comes from `dset`, and `0` otherwise. `var` would not be kept in the output data set.

`other_opts`: other data set options, such as KEEP, DROP, RENAME, etc.

Example 1

```
DATA D1; INPUT X $ Y @@;
CARDS;
A 10 C 11 H 13
RUN;
DATA D2; INPUT X $ Y @@;
CARDS;
B 12 D 14
RUN;
DATA COMB; SET D1 D2; RUN;
```

D1

X	Y
A	10
C	11
H	13

+

D2

X	Y
B	12
D	14

=

COMB

X	Y
A	10
C	11
H	13
B	12
D	14

Example 2

```
DATA D1; INPUT X $ Y @@;
CARDS;
A 10 C 11 H 13
RUN;
DATA D2; INPUT X $ Z @@;
CARDS;
B 12 D 14
RUN;
DATA COMB; SET D1 D2; RUN;
```

D1

X	Y
A	10
C	11
H	13

+

D2

X	Z
B	12
D	14

=

COMB

X	Y	Z
A	10	.
C	11	.
H	13	.
B	.	12
D	.	14

Question: If the variable Z in D2 is in fact the same variable as Y in D1, how can we get the result of Example 1?

Answer: Use the RENAME option by changing the SET statement:

```
DATA COMB; SET D1 D2(RENAME=(Z=Y));
RUN;
```

Example 3

Suppose we have two data sets, JANUARY and JUNE. We want to combine these two data sets into a new data set, called NEW.

In NEW, we want to have a variable, MONTH. It takes value either 'Jan' or 'Jun' depending on which file the record comes from.

```
DATA NEW; SET JANUARY(IN=INJAN) JUNE;
IF INJAN THEN MONTH = 'Jan'; ELSE MONTH =
'Jun';
RUN;
```

Example 4

A college has a SAS file, called STUD, storing students' records.

We want to find out whether there are any clerical errors in STUD, and make the necessary correction.

For simplicity, assume that the only type of error is that there is numerical digit in the variable NAME. We can run the following SAS program to find out the problematic records.

```
DATA STUD PROB;
SET STUD;
IF INDEXC(NAME,'0123456789') THEN OUTPUT PROB;
ELSE OUTPUT STUD;
RUN;
```

The data set is split into two parts, the problematic cases allocated to PROB, and the correct cases allocated to STUD.

We then check whether the data set PROB is empty or not.

If it is not empty, we make the correction using the Viewtable window. Then use the following program to put the corrected records back to STUD.

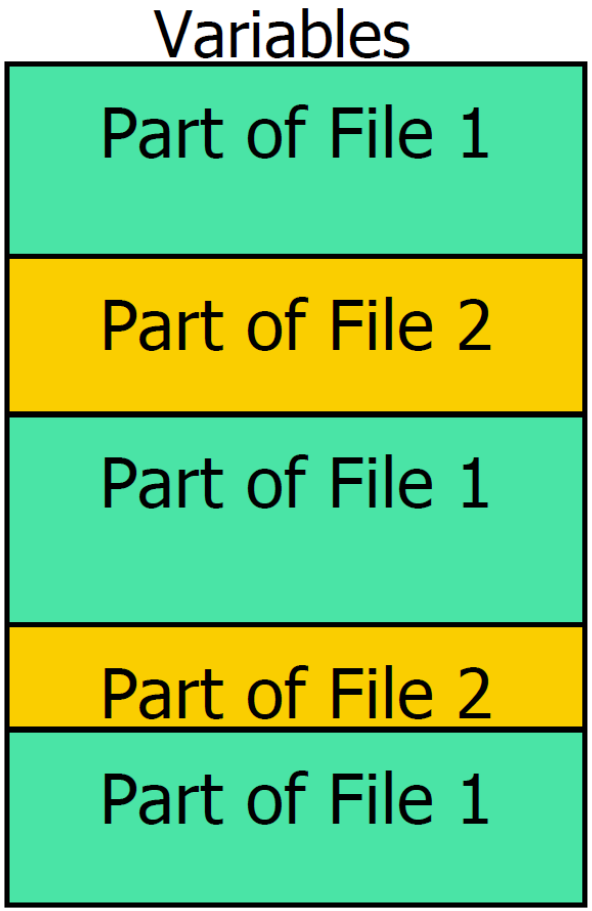
```
DATA STUD; SET STUD PROB; RUN;
```

Method 2: Interleaving

Interleaving is an operation that combines data sets in sorted order. We use SET and BY statement to interleave two or more data sets. The data sets listed in the SET statement must be sorted by the variables in the BY statement (otherwise the step will stop with an error).

The new data set contains all the observations in the input data sets in sorted order by the variable(s) listed in the BY statement. The syntax is

```
DATA [dsname];
SET {[fileref.]dset[(IN=var other_opts)]}
. . .;
BY [DESCENDING] var;
RUN;
```



The position of a record in the combined file is determined by sort key(s).

Therefore, observations with the same value of the sort variables are physically adjacent in the output data set.

Example 5

```
DATA D1; INPUT X $ Y @@;
CARDS;
A 10 C 11 D 13
RUN;
DATA D2; INPUT X $ Y @@;
CARDS;
B 12 D 14
RUN;
DATA COMB; SET D1 D2; BY X; RUN;
```

D1

X	Y
A	10
C	11
D	13

+

D2

X	Y
B	12
D	14

=

COMB

X	Y
A	10
B	12
C	11
D	13
D	14

Example 6

```
DATA southentrance;
INFILE 'D:\SAS\South.dat';
INPUT Entrance $ PassNumber PartySize Age;
PROC PRINT DATA = southentrance;
TITLE 'South Entrance Data'; RUN;
DATA northentrance;
INFILE 'D:\SAS\North.dat';
INPUT Entrance $ PassNumber PartySize Age
Lot;
PROC SORT DATA = northentrance;
BY PassNumber;
PROC PRINT DATA = northentrance;
TITLE 'North Entrance Data'; RUN;
```

South Entrance Data

Obs	Entrance	PassNumber	PartySize	Age
1	S	43	3	27
2	S	44	3	24
3	S	45	3	2

North Entrance Data

Obs	Entrance	PassNumber	PartySize	Age	Lot
1	N	21	5	41	1
2	N	65	2	67	1
3	N	66	2	7	1
4	N	87	4	33	3

```
* Interleave observations by PassNumber;
DATA interleave;
SET northentrance southentrance;
BY PassNumber; RUN;
PROC PRINT DATA = interleave;
TITLE 'Both Entrances, By Pass Number';
RUN;
```

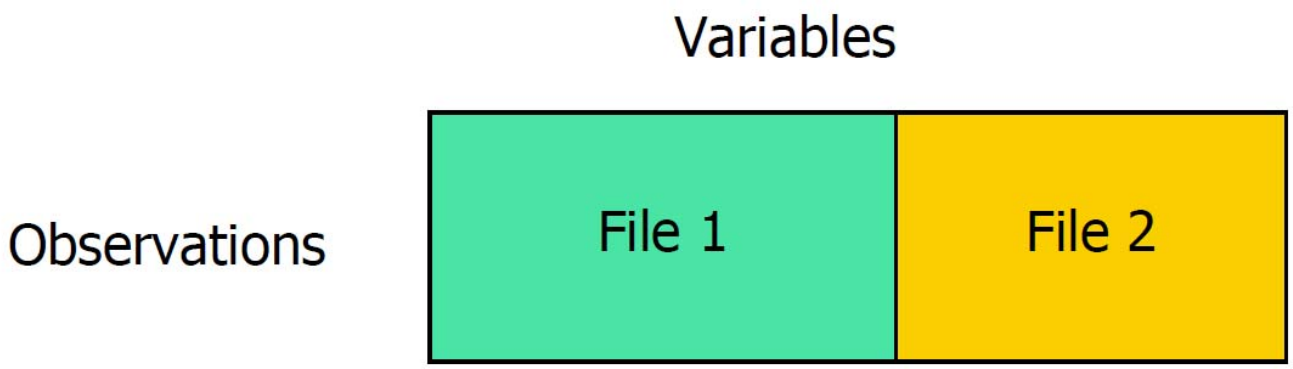
Both Entrances, By Pass Number

Obs	Entrance	PassNumber	PartySize	Age	Lot
1	N	21	5	41	1
2	S	43	3	27	.
3	S	44	3	24	.
4	S	45	3	2	.
5	N	65	2	67	1
6	N	66	2	7	1
7	N	87	4	33	3

Method 3: One-to-One Merge

The MERGE statement joins variables from two or more SAS data sets into single observations in a new SAS data set.

Merging without a BY statement corresponds to a one-to-one merge. One-to-one merging places data sets side by side. The *i*-th observation in the first data set is paired with the *i*-th observation in the second data set, and so on.



Its syntax is

```
MERGE
[fileref.]dset1 [(IN=var1 other_opts)]
[fileref.]dset2 [(IN=var2 other_opts)]
... ;
```

The variable `var1` will have value `1` if the observation form `dset1` contributes to the current observation for output and `0` otherwise.
`var1` will not be added to any SAS data set being created.
The same is true for `var2` and so on.
`other_opts` specifies any other SAS data set options.

Example 7

```
DATA D1; INPUT X $ Y @@;
CARDS;
A 10 C 11 D 13
RUN;
DATA D2; INPUT Z $ Y @@;
CARDS;
B . D 14
RUN;
DATA COMB; MERGE D1 D2; RUN;
```

Note that the Y-values in D2 overwrite those in D1.
So the ordering of the data sets is of importance if the data sets have common variable(s).

D1

X	Y
A	10
C	11
D	13

+

D2

Z	Y
B	.
D	14

=

COMB

X	Y	Z
A	.	B
C	14	D
D	13	.

Example 8

What is the difference between the following two programs?

```
DATA COMB1;
MERGE D1 D2; RUN;
DATA COMB2;
SET D1; SET D2; RUN;
```

For the first program, SAS stops when all records in D1 and D2 are read.

For the second program, SAS stops when it finishes reading either D1 or D2.

For example, if D1 has three records and D2 has two records, then COMB1 has three records and COMB2 has two records.

The records in COMB2 are just the first two records in COMB1.

```
DATA D1; INPUT X $ Y @@;
CARDS;
A 10 C 11 D 13
RUN;
DATA D2; INPUT X $ Y @@;
CARDS;
B . D 14
RUN;
DATA COMB1; MERGE D1 D2; RUN;
PROC PRINT; RUN;
DATA COMB2; SET D1; SET D2; RUN;
PROC PRINT; RUN;
```

COMB1

Obs	X	Y
1	B	.
2	D	14
3	D	13

COMB2

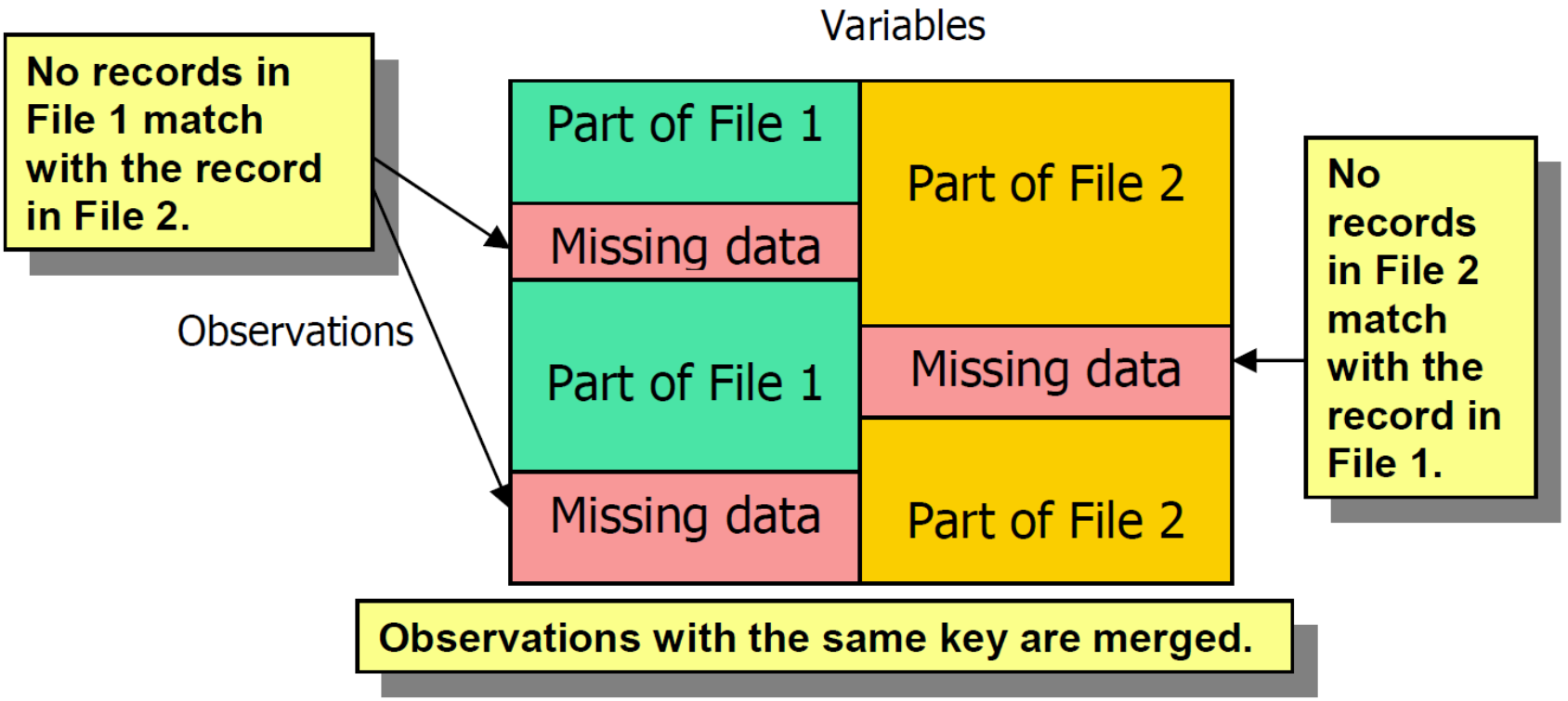
Obs	X	Y
1	B	.
2	D	14

Method 4: Matched Merge

- Merging with a BY statement is called match-merging, in which records are joined by matching values of the variable(s) listed in the BY statement.
- The BY variable(s) must be common to all data sets, and each data set must be sorted by these variable(s).
- The BY variable(s) are used for matching. When two files have the same variable, the value of the variable in the later file will overwrite the value in the former file.
- We can use the RENAME option to avoid undesired overwriting.

Syntax:

```
MERGE
[fileref.]dset1 [(IN=var1 other_opts)]
[fileref.]dset2 [(IN=var2 other_opts)]
... ;
BY {[DESCENDING] var} ... ;
```



Example 9

```
DATA D1; INPUT X $ Y @@;
CARDS;
A 10 C 11 H 13
RUN;
DATA D2; INPUT X $ Z S @@;
CARDS;
C 12 10 C 17 15 D 14 9
RUN;
DATA COMB(KEEP = X Y S W);
MERGE D1 D2; BY X; W = Y+Z; RUN;
```

D1

X	Y
A	10
C	11
H	13

+

D2

X	Z	S
C	12	10
C	17	15
D	14	9

=

COMB

X	Y	S	W
A	10	.	.
C	11	10	23
C	11	15	28
D	.	9	.
H	13	.	.

Suppose we only want COMB to contain records where there was a contribution from both of the data sets. Replace the DATA step by the following.

```
DATA COMB(KEEP= X Y S W);
MERGE D1(IN=A) D2(IN=B);
BY X; W = Y+Z;
IF A AND B THEN OUTPUT;
RUN;
```

Obs	X	Y	S	W
1	C	11	10	23
2	C	11	15	28

Example 10

```
DATA DATA1; INPUT YEAR VARX $ @@;
CARDS;
1991 X1 1992 X2 1993 X3 1994 X4 1994 X5 1995 X6 1995
X7
RUN;
DATA DATA2; INPUT YEAR VARY $ @@;
CARDS;
1991 Y1 1991 Y2 1993 Y3 1994 Y4 1994 Y5 1995 Y6
RUN;
DATA ALL;
MERGE DATA1 DATA2; BY YEAR;
RUN;
```

DATA1

YEAR	VARX
1991	X1
1992	X2
1993	X3
1994	X4
1994	X5
1995	X6
1995	X7

+

DATA2

YEAR	VARY
1991	Y1
1991	Y2
1993	Y3
1994	Y4
1994	Y5
1995	Y6

=

ALL

YEAR	VARX	VARY
1991	X1	Y1
1991	X1	Y2
1992	X2	.
1993	X3	Y3
1994	X4	Y4
1994	X5	Y5
1995	X6	Y6
1995	X7	Y6

If the merge statement is changed to be

```
MERGE DATA2 DATA1;
```

the data set ALL will be the same except that the order of the variables becomes YEAR, VARY, and VARX.

To understand more about the many-to-one merge, let us consider the following modification of the example.

```
DATA ALL;
MERGE DATA1 DATA2; BY YEAR;
OUTPUT; IF VARX = 'X1' THEN VARX='***';
RUN;
```

The only difference is that the second record of ALL has VARX = '***'.

It means that the value of VARX is carried forward to the second record.

ALL

YEAR	VARX	VARY
1991	X1	Y1
1991	***	Y2
1992	X2	.
1993	X3	Y3
1994	X4	Y4
1994	X5	Y5
1995	X6	Y6
1995	X7	Y6

Example 11

There are two SAS data sets, NATLPARK and PRES.

Variables in NATLPARK:

PARK : Park name (character)

ST : Principal state (character)

COAST : East/West of Mississippi River ('E' or 'W') (character)

YRESTAB : Year established (numeric)

ACRES : Areas in park (numeric)

Variables in PRES:

NAME : President's name (character)

INAUG : Year inaugurated (numeric)

Create a data set, PARK storing the same information in NATLPARK together with one more variable, PRESID storing the name of president when the park is established.

Variable PRESID can be constructed using variables NAME and INAUG in the data set PRES.

If YRESTAB = INAUG, we are not sure whether the president should be the one who just inaugurated or the one preceding it. In this case, PRESID stores both names with a "?" added behind.

In the following program, P stores the president's name in the preceding record. we need to update P when B = 1, before going back to handle a new record.

```
LIBNAME IN 'D:\SAS';
PROC SORT
     DATA=IN.NATLPARK OUT=TEMP1;
     BY YRESTAB;
RUN;
PROC SORT
     DATA=IN.PRES OUT=TEMP2;
     BY INAUG;
RUN;
DATA IN.PARK;
LENGTH P $ 20 PRESID $ 20;
RETAIN P;
MERGE
     TEMP1(IN=A)
     TEMP2(IN=B RENAME=(INAUG=YRESTAB NAME=PRESID));
     BY YRESTAB;
```

```
IF A AND B THEN DO;
    TEMP=PRESID;
    PRESID= TRIM(P)||', '||TRIM(PRESID)||' ?';
    OUTPUT; P=TEMP;
END;
IF B = 0 THEN DO;
    PRESID=P; OUTPUT;
END;
    ELSE P=PRESID;
KEEP PARK -- ACRES PRESID;
RUN;
```

We use matched merge to combine the two sorted data sets using YRESTAB as the key. We have renamed `INAUG` to YRESTAB in `TEMP2` .

We have three possible cases:

Case 1: `A = B = 1`,

In this case, YRESTAB `= INAUG`. We need to put both the president's name in the preceding record stored in P and the president's name in the current record stored in `PRESID` and add a `"?"`.

Case 2: `A = 1` and `B = 0`,

The president's name should be the president's name in the preceding record stored in P. As `A = 1`, the record corresponds to a new national park, and we need to output this record to Park.

Case 3: `A = 0` and `B = 1`,

We need to update P. As `A = 0`, the record does not correspond to a national park, and we do not need to output the record.

Method 5: Updating

We have two files. One is called the master data set and the other is called the transaction data set.

Only non-missing transaction data set values can replace values for matched observations in the master data set.

Updating is a restricted form of the matched merge. The restriction is placed on the missing values.

In a matched merge, a missing value for name matched variables will overwrite a non-missing value.

This is usually an undesirable result and is not allowed in an update.

The syntax of UPDATE is as follows:

```
UPDATE
[fileref.]master[(IN=var1 other_opts)]
[fileref.]transac[(IN=var2 other_opts)];
BY {[DESCENDING] var} ... ;
```

Only two data sets may be combined, and the sort (BY statement) variable's occurrence must be unique in the first, or "master," data set.

Non-missing values in the second, or "transaction" data set, are used to replace, or update, those in the master data set.

If any transaction observations were not matched with master observations, these become new observations in the new data set as well.

Example 12

```
DATA NEW;
UPDATE PAYROLL CHANGE; BY STAFF_ID;
RUN;
```

payroll

staff_id	salary
1635	8000
1640	9345
1747	11200
1831	22310

+

change

staff_id	salary
1635	8500
1747	.
1810	8700

=

new

staff_id	salary
1635	8500
1640	9345
1747	11200
1810	8700
1831	22310

Example 13

```
DATA MASTER;
UPDATE MASTER TRANS; BY YEAR;
RUN;
```

MASTER

YEAR	VARX	VARY
1990	X1	Y1
1991	X2	Y2
1992	X3	Y3
1993	X4	Y4
1994	X5	Y5

+

TRANS

YEAR	VARX	VARY
1991	P2	.
1992	P3	Q2
1993	P4	.
1993	.	Q3
1995	P5	Q4

=

MASTER

YEAR	VARX	VARY
1990	X1	Y1
1991	P2	Y2
1992	P3	Q2
1993	P4	Q3
1994	X5	Y5
1995	P5	Q4

Example 14

A hospital maintains a master database with information about patients. A sample appears below. Each record contains the patient's account number, last name, address, date of birth, sex, insurance code, and the date that patient's information was last updated.

```
620135 Smith    234 Aspen St.      12-21-1975 m CBC 02-16-1998
645722 Miyamoto 65 3rd Ave.        04-03-1936 f MCR 05-30-1999
645739 Jensvold 505 Glendale Ave.  06-15-1960 f HLT 09-23-2006
874329 Kazoyan  76-C La Vista      .          . MCD 01-15-2011
```

Whenever a patient is admitted to the hospital, the admissions staff check the data for that patient. They create a transaction record for every new patient and for any returning patients whose status has changed.

Here are three transactions:

```
620135 .       .                .          . HLT 06-15-2012
874329 .       .                04-24-1954 m .   06-15-2012
235777 Harman 5656 Land Way     01-18-2000 f MCD 06-15-2012
```

The first transaction is for a returning patient whose insurance has changed.

The second transaction fills in missing information for a returning patient.

The last transaction is for a new patient who must be added to the database.

The following program puts the master data into a permanent data set named PATIENTMASTER. It reads the transaction data and sorts them with PROC SORT. Then it adds the transactions to PATIENTMASTER with an UPDATE statement. The master data set is already sorted by Account and, therefore, doesn't need to be sorted again:

```
LIBNAME IN 'D:\SAS';

DATA IN.PATIENTMASTER;
     INFILE 'D:\SAS\Admit.dat';
     INPUT
          ACCOUNT
          LASTNAME $ 8-16
          ADDRESS $ 17-34
          BIRTHDATE MMDDYY10.
          SEX $
          INSCODE $ 48-50
          @52 LASTUPDATE MMDDYY10.;
RUN;
```

```
DATA TRANSACTIONS;
     INFILE 'D:\SAS\NewAdmit.dat';
     INPUT
          ACCOUNT
          LASTNAME $ 8-16
          ADDRESS $ 17-34
          BIRTHDATE MMDDYY10.
          SEX $
          INSCODE $ 48-50
          @52 LASTUPDATE MMDDYY10.;
RUN;

PROC SORT
     DATA = TRANSACTIONS;
     BY ACCOUNT;
RUN;
```

```
* Update patient data with transactions;
DATA IN.PATIENTMASTER;
    UPDATE IN.PATIENTMASTER TRANSACTIONS;
    BY ACCOUNT;
RUN;

PROC PRINT
    DATA = IN.PATIENTMASTER;
    FORMAT BIRTHDATE LASTUPDATE MMDDYY10.;
    TITLE 'Admissions Data';
RUN;
```

The results of the PROC PRINT look like this:

Admissions Data

Obs	ACCOUNT	LASTNAME	ADDRESS	BIRTHDATE	SEX	INSCODE	LASTUPDATE
1	235777	Harman	5656 Land Way	01/18/2000	f	MCD	06/15/2008
2	620135	Smith	234 Aspen St.	12/21/1975	m	HLT	06/15/2008
3	645722	Miyamoto	65 3rd Ave.	04/03/1936	f	MCR	05/30/1999
4	645739	Jensvold	505 Glendale Ave.	06/15/1960	f	HLT	09/23/2006
5	874329	Kazoyan	76-C La Vista	04/24/1954	m	MCD	06/15/2008

General remarks

- When ***interleaving***, ***match merging***, or ***updating*** data sets, the data sets must be sorted by one or more common variables.
- For ***one-to-one merge***, ***matched merge***, and ***update***, SAS reads variables names from the leftmost data set in the data set list, and then reads the variable names from the next data set in the list. If this data set has names not already in the name list being built, SAS adds the names to the list.

- If two data sets have variables with the same name, the data set on the right will overwrite, or replace, the value of the variable in the observation for output.
- ***Update*** will replace a value only if the right-hand side data set (the transaction data set) has a non-missing value.
- Observations are added during ***matched merging*** or ***updating*** if an observation has a unique combination of the BY variable(s) used to link the data sets.