# Chapter 6: Combining and Managing SAS Datasets

- Handling multiple input datasets with the SET and MERGE statements
- SAS special variables
- Transposing and aggregating datasets
- Reshaping SAS datasets
- Data library management

## Multiple Input Datasets

- Datasets can be combined in various ways using the following SAS statements to specify the input datasets:

| | |
|---|---|
| SET | Reads observations from one or more SAS datasets. |
| MERGE | Joins observations from multiple SAS datasets. |
| UPDATE | Modifies an existing "master file" SAS dataset with a transaction file" SAS dataset, creating a new "master file". |

- Use SET when combining datasets that contain the same information for different subjects. The result is that the dataset gets longer (has more observations).

- Use MERGE when combining datasets that contain different information on the same subjects. The result is that the dataset gets wider (has more variables)

- These statements can be controlled by using the BY statement in the DATA step.

- When the BY statement is used, all the input datasets must contain the BY variables and must be sorted by the BY variables. Observations that have the same values of all BY variables are said to belong to the same BY group.

## The SET Statement

- The SET statement names the SAS input datasets from which observations will be read.

- Syntax:

```
SET dataset1 dataset2 dataset3 etc....;
```

where:

  - `dataset1, dataset2`, etc., are names of the form `library.dataset` or '`path \ dataset`'.

- By default SAS will concatenate (that is, stack) all datasets in the order listed in the SET statement.
  - All observations will be read from the first dataset listed, then all observations will be read from the second dataset listed, and so on.

- When used with a BY statement, SAS will interleave observations from the datasets listed in the SET statement.
  - The observations are selected based on the sort order determined by the BY variables.
  - All input datasets must be sorted by the BY variables, and the resulting output dataset will be sorted by the BY variables as well.

## Example: Concatenating Datasets (Stacking)

Create dataset combining short and tall students.

**Dataset SHORT**

| NAME | SEX | AGE | HT | WT |
|---|---|---|---|---|
| PIGGY M | F | . | 48 | . |
| FROG K | M | 3 | 12 | 1 |
| GONZO | | 14 | 25 | 45 |

**Dataset TALL**

| NAME | SEX | AGE | HT | WT |
|---|---|---|---|---|
| CHRISTIANSEN | M | 37 | 71 | 195 |
| HOSKING J | M | 31 | 70 | 160 |
| HELMS R | M | 41 | 74 | 195 |

```
data work.both;
   set work.short work.tall;
run;
```

**Concatenated Dataset BOTH**

| NAME | SEX | AGE | HT | WT |
|---|---|---|---|---|
| PIGGY M | F | . | 48 | . |
| FROG K | M | 3 | 12 | 1 |
| GONZO | | 14 | 25 | 45 |
| CHRISTIANSEN | M | 37 | 71 | 195 |
| HOSKING J | M | 31 | 70 | 160 |
| HELMS R | M | 41 | 74 | 195 |

## Example: Interleaving Datasets

**Dataset SHORT**

| NAME | SEX | AGE | HT | WT |
|---|---|---|---|---|
| FROG K | M | 3 | 12 | 1 |
| GONZO | | 14 | 25 | 45 |
| PIGGY M | F | . | 48 | . |

**Dataset TALL**

| NAME | SEX | AGE | HT | WT |
|---|---|---|---|---|
| CHRISTIANSEN | M | 37 | 71 | 195 |
| HELMS R | M | 41 | 74 | 195 |
| HOSKING J | M | 31 | 70 | 160 |

```
proc sort data=work.short; by name; run;

proc sort data=work.tall; by name; run;

data work.bothsort;
   set short tall;
   by name;
run;
```

**Interleaved Dataset BOTHSORT**

| NAME | SEX | AGE | HT | WT |
|---|---|---|---|---|
| CHRISTIANSEN | M | 37 | 71 | 195 |
| FROG K | M | 3 | 12 | 1 |
| GONZO | | 14 | 25 | 45 |
| HELMS R | M | 41 | 74 | 195 |
| HOSKING J | M | 31 | 70 | 160 |
| PIGGY M | F | . | 48 | . |

## The MERGE Statement

- The MERGE statement joins observations from the input datasets into a single observation in the output dataset.

- Syntax:

```
MERGE dataset1 dataset2 dataset3 etc....;
```

where:

- `dataset1, dataset2`, etc., are names of the form `library.dataset` or '`path \ dataset`'.

- One-to-one merging is performed when no BY statement is used. The first observation in each dataset is joined to all the other first observations, the second to the second, etc. One-to-one merging should be performed with caution, especially when the datasets contain different numbers of observations.

- Match merging is performed when a BY statement is used. This joins the observations only when the values of the BY variables match.

- One-to-many match merging allows variable values to be spread or "splayed" across all observations with the same BY values.

- Many-to-many match merging joins observations one to one within the BY group until one dataset reaches the last observation within the current BY group. This observation is then splayed across the remaining observations in the BY group.

## Example: One-to-One Merge with Equal Numbers of Observations

Combine the LIST and DEMO datasets to produce the NEWCLASS dataset.

**Dataset LIST**

| NAME | AGE | HT | WT |
|---|---|---|---|
| CHRISTIANSEN | 37 | 71 | 195 |
| FROG K | 3 | 12 | 1 |
| GONZO | 14 | 25 | 45 |
| HELMS R | 41 | 74 | 195 |
| HOSKING J | 31 | 70 | 160 |
| PIGGY M | . | 48 | . |

**Dataset DEMO**

| NAME | SEX |
|---|---|
| CHRISTIANSEN | M |
| FROG K | M |
| GONZO | |
| HELMS R | M |
| HOSKING J | M |
| PIGGY M | F |

```
data newclass;
   merge list demo;
run;
```

**Merged Dataset NEWCLASS**

| NAME | AGE | HT | WT | SEX |
|---|---|---|---|---|
| CHRISTIANSEN | 37 | 71 | 195 | M |
| FROG K | 3 | 12 | 1 | M |
| GONZO | 14 | 25 | 45 | |
| HELMS R | 41 | 74 | 195 | M |
| HOSKING J | 31 | 70 | 160 | M |
| PIGGY M | . | 48 | . | F |

## Example: One-to-One Merge with Unequal Numbers of Observations

Join two datasets with unequal numbers of observations.

**Dataset LIST2**

| NAME | SEX |
|---|---|
| CHRISTIANSEN | M |
| FROG K | M |
| GONZO | |
| HOSKING J | M |
| PIGGY M | F |

**Dataset DEMO2**

| NAME | AGE | HT | WT |
|---|---|---|---|
| CHRISTIANSEN | 37 | 71 | 195 |
| FROG K | 3 | 12 | 1 |
| HELMS R | 41 | 74 | 195 |
| PIGGY M | . | 48 | . |

```
data newclass;
   merge list2 demo2;
run;
```

**Merged Dataset NEWCLASS**

| NAME | SEX | AGE | HT | WT |
|---|---|---|---|---|
| CHRISTIANSEN | M | 37 | 71 | 195 |
| FROG K | M | 3 | 12 | 1 |
| HELMS R | | 41 | 74 | 195 |
| PIGGY M | M | . | 48 | . |
| PIGGY M | F | . | . | . |

- Note: You will not find many occasions where one-to-one merging (with no BY variable) is really what you want to do.
- The MERGENOBY=WARN option used in the standard SAS OPTIONS statement for this course is intended to protect you from allowing this to happen accidentally.

## Match Merging

Join the datasets from the previous example, but this time join only those observations with the same values of the BY variables. Note that both input datasets are sorted by the BY variable.

**Dataset LIST2**

| NAME | SEX |
|---|---|
| CHRISTIANSEN | M |
| FROG K | M |
| GONZO | |
| HOSKING J | M |
| PIGGY M | F |

**Dataset DEMO2**

| NAME | AGE | HT | WT |
|---|---|---|---|
| CHRISTIANSEN | 37 | 71 | 195 |
| FROG K | 3 | 12 | 1 |
| HELMS R | 41 | 74 | 195 |
| PIGGY M | . | 48 | . |

```
data newclass;
   merge list2 demo2;
   by name;
run;
```

**Merged Dataset NEWCLASS**

| NAME | SEX | AGE | HT | WT |
|---|---|---|---|---|
| CHRISTIANSEN | M | 37 | 71 | 195 |
| FROG K | M | 3 | 12 | 1 |
| GONZO | | . | . | . |
| HELMS R | | 41 | 74 | 195 |
| HOSKING J | M | . | . | . |
| PIGGY M | F | . | 48 | . |

Match merging can be one-to-one, one-to-many, or many-to-many (a situation to avoid), as shown in the following examples.

## One-to-One Match Merge Example 1

Combine blood pressure and lipid data into one dataset.

**Dataset SBP**

| id | sbp |
|---|---|
| 101 | 83 |
| 102 | 100 |
| 103 | 120 |
| 104 | 118 |

**Dataset LIPIDS**

| id | hdl | ldl |
|---|---|---|
| 101 | 45 | 180 |
| 102 | 50 | 200 |
| 103 | 60 | 210 |
| 105 | 65 | 220 |

```
data all;
   merge sbp lipids;
   by id;
run;
```

**Merged Dataset ALL**

| id | sbp | hdl | ldl |
|---|---|---|---|
| 101 | 83 | 45 | 180 |
| 102 | 100 | 50 | 200 |
| 103 | 120 | 60 | 210 |
| 104 | 118 | . | . |
| 105 | . | 65 | 220 |

## One-to-One Match Merge Example 2

Merge a pre-randomization dataset with a post-randomization dataset. Both datasets contain common variables other than the BY variables.

Notice the resulting dataset. When two merged datasets have common variables other than the BY variables, the new dataset contains the values value of the common variables from the right-most dataset in the MERGE statement.

**Dataset PRERAND**

| id | hdl | ldl |
|---|---|---|
| 101 | 40 | 200 |
| 102 | 45 | 210 |
| 103 | 55 | 215 |
| 104 | 60 | 220 |

**Dataset POSTRAND**

| id | hdl | ldl |
|---|---|---|
| 101 | 45 | 180 |
| 102 | 50 | 200 |
| 103 | 60 | 210 |
| 105 | 65 | 230 |

```
data all;
   merge prerand postrand;
   by id;
run;
```

**Merged Dataset ALL**

| id | hdl | ldl |
|---|---|---|
| 101 | 45 | 180 |
| 102 | 50 | 200 |
| 103 | 60 | 210 |
| 104 | 60 | 220 |
| 105 | 65 | 230 |

## One-to-One Match Merge Example 3

Merge a pre-randomization dataset with a post-randomization dataset. Both datasets contain common variables other than the BY variables, and you don't want to lose any variable values. Use the RENAME dataset option to rename the common variables.

**Dataset PRERAND**

| id | hdl | ldl |
|---|---|---|
| 101 | 40 | 200 |
| 102 | 45 | 210 |
| 103 | 55 | 215 |
| 104 | 60 | 220 |

**Dataset POSTRAND**

| id | hdl | ldl |
|---|---|---|
| 101 | 45 | 180 |
| 102 | 50 | 200 |
| 103 | 60 | 210 |
| 105 | 65 | 230 |

```
data all;
   merge prerand(rename=(hdl=hdl0 ldl=ldl0))
         postrand(rename=(hdl=hdl1 ldl=ldl1));
   by id;
   hdldiff = hdl1 - hdl0;
   ldldiff = ldl1 - ldl0;
run;
```

**Merged Dataset ALL**

| id | hdl0 | ldl0 | hdl1 | ldl1 | hdldiff | ldldiff |
|---|---|---|---|---|---|---|
| 101 | 40 | 200 | 45 | 180 | 5 | -20 |
| 102 | 45 | 210 | 50 | 200 | 5 | -10 |
| 103 | 55 | 215 | 60 | 210 | 5 | -5 |
| 104 | 60 | 220 | . | . | . | . |
| 105 | . | . | 65 | 230 | . | . |

- When you use a RENAME statement, use the old variable names in the DATA step.

- When you use RENAME dataset option, use the new variable names in the DATA step.

## One-to-One Match Merge Example 4
## Two BY Variables

```
data all;
    merge sbp ldl;
    by id visit;
run;
```

**Dataset LDL**

| ldl | id | visit |
|---|---|---|
| 130 | 1 | 1 |
| 135 | 1 | 2 |
| 140 | 1 | 3 |
| 145 | 2 | 1 |
| 150 | 2 | 2 |
| 155 | 2 | 3 |
| 160 | 3 | 1 |
| 165 | 3 | 3 |
| 170 | 4 | 1 |
| 175 | 4 | 2 |
| 180 | 5 | 1 |
| 185 | 5 | 2 |

**Dataset SBP**

| sbp | id | visit |
|---|---|---|
| 100 | 1 | 1 |
| 105 | 1 | 2 |
| 110 | 1 | 3 |
| 115 | 2 | 1 |
| 120 | 2 | 3 |
| 122 | 3 | 1 |
| 125 | 3 | 2 |
| 130 | 3 | 3 |
| 135 | 4 | 3 |

**Merged Dataset ALL**

| Obs | sbp | id | visit | ldl |
|---|---|---|---|---|
| 1 | 100 | 1 | 1 | 130 |
| 2 | 105 | 1 | 2 | 135 |
| 3 | 110 | 1 | 3 | 140 |
| 4 | 115 | 2 | 1 | 145 |
| 5 | . | 2 | 2 | 150 |
| 6 | 120 | 2 | 3 | 155 |
| 7 | 122 | 3 | 1 | 160 |
| 8 | 125 | 3 | 2 | . |
| 9 | 130 | 3 | 3 | 165 |
| 10 | . | 4 | 1 | 170 |
| 11 | . | 4 | 2 | 175 |
| 12 | 135 | 4 | 3 | . |
| 13 | . | 5 | 1 | 180 |
| 14 | . | 5 | 2 | 185 |

## One-to-Many Match Merging

- One dataset in a MERGE statement may contain multiple observations with the same values of the BY variables within each BY group.
    - MERGE joins the observation from the "one" dataset with each observation in the "many" dataset.

- In match merging, the program data vector is initialized to missing only at the beginning of a new BY group.
    - This allows information from the "one" dataset to be splayed across the corresponding BY group observations in the "many" dataset.

## One-to-Many Match Merge Example 1

Dataset LIST has one observation per value of NAME.
Dataset HW has multiple observations per value of NAME.

**Dataset LIST**

| NAME | SEX |
|---|---|
| CHRISTIANSEN | M |
| FROG K | M |
| GONZO | |
| HELMS R | M |
| HOSKING J | M |
| PIGGY M | F |

**Dataset HW**

| name | hwnum | score |
|---|---|---|
| CHRISTIANSEN | 1 | 10 |
| CHRISTIANSEN | 2 | 3 |
| CHRISTIANSEN | 4 | 9 |
| CHRISTIANSEN | 5 | 10 |
| FROG K | 1 | 10 |
| FROG K | 2 | 10 |
| FROG K | 3 | 8 |
| GONZO | 1 | . |
| HOSKING J | 1 | 10 |
| PIGGY M | 2 | 10 |
| PIGGY M | 3 | 10 |

```
data grades;
   merge list hw;
   by name;
run;
```

## Merged Dataset GRADES

| NAME | SEX | hwnum | score |
|---|---|---|---|
| CHRISTIANSEN | M | 1 | 10 |
| CHRISTIANSEN | M | 2 | 3 |
| CHRISTIANSEN | M | 4 | 9 |
| CHRISTIANSEN | M | 5 | 10 |
| FROG K | M | 1 | 10 |
| FROG K | M | 2 | 10 |
| FROG K | M | 3 | 8 |
| GONZO | | 1 | . |
| HELMS R | M | . | . |
| HOSKING J | M | 1 | 10 |
| PIGGY M | F | 2 | 10 |
| PIGGY M | F | 3 | 10 |

## One-to-Many Match Merge Example 2

When the same variables (other than BY variables) are in more than one dataset, the value of the variable from the last dataset mentioned in the MERGE statement is used. However, this does not always work as people expect, so this is not recommended.

**Dataset GRADES**

| NAME | SEX | hwnum | score |
|---|---|---|---|
| CHRISTIANSEN | M | 1 | 10 |
| CHRISTIANSEN | M | 2 | 3 |
| CHRISTIANSEN | M | 4 | 9 |
| CHRISTIANSEN | M | 5 | 10 |
| FROG K | M | 1 | 10 |
| FROG K | M | 2 | 10 |
| FROG K | M | 3 | 8 |
| GONZO | | 1 | . |
| HELMS R | M | . | . |
| HOSKING J | M | 1 | 10 |
| PIGGY M | F | 2 | 10 |
| PIGGY M | F | 3 | 10 |

**Dataset FIXHW**

| name | hwnum | score |
|---|---|---|
| CHRISTIANSEN | 1 | 9 |
| CHRISTIANSEN | 2 | 10 |
| CHRISTIANSEN | 2 | 2 |
| CHRISTIANSEN | 4 | . |
| FROG K | 2 | . |
| HOSKING J | 1 | . |

```
data grades2;
   merge grades fixhw;
   by name hwnum;
run;
```

**Merged Dataset GRADES2**

| NAME | SEX | hwnum | score |
|---|---|---|---|
| CHRISTIANSEN | M | 1 | 9 |
| CHRISTIANSEN | M | 2 | 10 |
| CHRISTIANSEN | M | 2 | 2 |
| CHRISTIANSEN | M | 4 | . |
| CHRISTIANSEN | M | 5 | 10 |
| FROG K | M | 1 | 10 |
| FROG K | M | 2 | . |
| FROG K | M | 3 | 8 |
| GONZO | | 1 | . |
| HELMS R | M | . | . |
| HOSKING J | M | 1 | . |
| PIGGY M | F | 2 | 10 |
| PIGGY M | F | 3 | 10 |

Notice that missing values in FIXHW overwrite values in GRADES.

# One-to-Many Match Merge Example 3

**Dataset SBP**

| ID | VISIT | SBP |
|---|---|---|
| 1 | 1 | 100 |
| 1 | 2 | 105 |
| 1 | 3 | 110 |
| 2 | 1 | 115 |
| 2 | 3 | 120 |
| 3 | 1 | 122 |
| 3 | 2 | 125 |
| 3 | 3 | 130 |
| 4 | 3 | 135 |

**Dataset DEMOGRAPHICS**

| ID | SEX | AGE |
|---|---|---|
| 1 | F | 30 |
| 2 | M | 35 |
| 3 | F | 40 |
| 4 | F | 45 |
| 5 | M | 50 |

```
data all;
   merge sbp demographics;
   by id;
run;
```

**Merged Dataset ALL**

| ID | VISIT | SBP | SEX | AGE |
|---|---|---|---|---|
| 1 | 1 | 100 | F | 30 |
| 1 | 2 | 105 | F | 30 |
| 1 | 3 | 110 | F | 30 |
| 2 | 1 | 115 | M | 35 |
| 2 | 3 | 120 | M | 35 |
| 3 | 1 | 122 | F | 40 |
| 3 | 2 | 125 | F | 40 |
| 3 | 3 | 130 | F | 40 |
| 4 | 3 | 135 | F | 45 |
| 5 | . | . | M | 50 |

# One-to-Many Match Merge Example 4

**Dataset SBP**

| ID | VISIT | SBP |
|---|---|---|
| 1 | 1 | 100 |
| 1 | 2 | 105 |
| 1 | 3 | 110 |
| 2 | 1 | 115 |
| 2 | 3 | 120 |
| 3 | 1 | 122 |
| 3 | 2 | 125 |
| 3 | 3 | 130 |
| 4 | 3 | 135 |

**Dataset DEMOGRAPHICS**

| ID | SEX | AGE |
|---|---|---|
| 1 | F | 30 |
| 2 | M | 35 |
| 3 | F | 40 |
| 4 | F | 45 |
| 5 | M | 50 |

```
data all;
   merge sbp
         demographics;
   by id;
   age = age*10;
run;
```

What's wrong with AGE??

**Merged Dataset ALL**

| ID | VISIT | SBP | SEX | AGE |
|---|---|---|---|---|
| 1 | 1 | 100 | F | 300 |
| 1 | 2 | 105 | F | 3000 |
| 1 | 3 | 110 | F | 30000 |
| 2 | 1 | 115 | M | 350 |
| 2 | 3 | 120 | M | 3500 |
| 3 | 1 | 122 | F | 400 |
| 3 | 2 | 125 | F | 4000 |
| 3 | 3 | 130 | F | 40000 |
| 4 | 3 | 135 | F | 450 |
| 5 | . | . | M | 500 |

## One-to-Many Match Merge Example 5

Objective: Merge the departmental means onto each record.

```
proc means data=bios511.sales nway;
    class dept;
    var cost;
    output out=out1 mean=meancost;
run;

proc print data=out1;
    title1 "Mean of Cost By Department";
    title2 "Output Dataset From PROC MEANS";
run;

proc sort data=bios511.sales out=sales;
    by dept;
run;

data sales2;
    merge sales out1;
    by dept;
run;

proc print data=sales2;
    title1 "Dataset SALES2";
    title2 "Departmental Means Merged Onto Original
              Data";
run;
```

## Mean of Cost By Department
## Output Dataset From PROC MEANS

| Obs | DEPT | _TYPE_ | _FREQ_ | meancost |
|---|---|---|---|---|
| 1 | FURS | 1 | 6 | 203.670 |
| 2 | SHOES | 1 | 44 | 35.036 |

## Dataset SALES2
## Departmental Means Merged Onto Original Data

| DEPT | CLERK | PRICE | COST | WEEKDAY | DAY | _TYPE_ | _FREQ_ | meancost |
|---|---|---|---|---|---|---|---|---|
| FURS | BURLEY | 599.95 | 180.01 | THR | 5 | 1 | 6 | 203.670 |
| FURS | BURLEY | 800.00 | 240.00 | MON | 9 | 1 | 6 | 203.670 |
| FURS | AGILE | 590.00 | 182.00 | SAT | 14 | 1 | 6 | 203.670 |
| FURS | BURLEY | 499.95 | 200.01 | SAT | 14 | 1 | 6 | 203.670 |
| FURS | BURLEY | 700.00 | 210.00 | THR | 19 | 1 | 6 | 203.670 |
| FURS | AGILE | 700.00 | 210.00 | WED | 25 | 1 | 6 | 203.670 |
| SHOES | CLEVER | 99.95 | 41.21 | TUE | 3 | 1 | 44 | 35.036 |
| SHOES | AGILE | 95.00 | 40.49 | WED | 4 | 1 | 44 | 35.036 |
| SHOES | CLEVER | 65.00 | 33.44 | WED | 4 | 1 | 44 | 35.036 |
| SHOES | CLEVER | 65.00 | 33.44 | WED | 4 | 1 | 44 | 35.036 |
| SHOES | AGILE | 49.95 | 28.07 | THR | 5 | 1 | 44 | 35.036 |
| SHOES | AGILE | 69.95 | 34.93 | THR | 5 | 1 | 44 | 35.036 |
| SHOES | BURLEY | 69.95 | 34.93 | THR | 5 | 1 | 44 | 35.036 |
| SHOES | CLEVER | 84.95 | 38.65 | SAT | 7 | 1 | 44 | 35.036 |
| SHOES | CLEVER | 54.95 | 30.00 | SAT | 7 | 1 | 44 | 35.036 |

Etc.. Remaining observations omitted for brevity.

## One-to-Many Match Merge Example 6

Objective: Merge the overall mean onto each record.

```
proc means data=bios511.class noprint;
    var wt;
    output out=out1 mean=meanwt;
run;

proc print data=out1;
    title1 "Overall Mean of Cost";
run;

data new;
    set bios511.class;
    if _n_=1 then set out1;

    wtdiff = wt-meanwt;
    drop _type_ _freq_;
run;

proc print data=new noobs;
    title1 "Merged Dataset NEW";
run;
```

**Overall Mean of Cost**

| Obs | _TYPE_ | _FREQ_ | meanwt |
|---|---|---|---|
| 1 | 0 | 6 | 119.2 |

**Merged Dataset NEW**

| NAME | SEX | AGE | HT | WT | meanwt | wtdiff |
|---|---|---|---|---|---|---|
| CHRISTIANSEN | M | 37 | 71 | 195 | 119.2 | 75.8 |
| HOSKING J | M | 31 | 70 | 160 | 119.2 | 40.8 |
| HELMS R | M | 41 | 74 | 195 | 119.2 | 75.8 |
| PIGGY M | F | . | 48 | . | 119.2 | . |
| FROG K | M | 3 | 12 | 1 | 119.2 | -118.2 |
| GONZO | | 14 | 25 | 45 | 119.2 | -74.2 |

Why does this work??? Because variables read with a SET (or MERGE or UPDATE) statement are automatically retained from one iteration of the DATA step loop to the next.

## Many-to-Many Merge Example

The datasets SBP and LIPIDS both contain multiple observations with the same level of ID. What happens when you merge by ID?

**Dataset SBP**

| id | sbp |
|---|---|
| 101 | 80 |
| 101 | 85 |
| 102 | 90 |
| 102 | 95 |
| 102 | 100 |

**Dataset LIPIDS**

| id | hdl | ldl |
|---|---|---|
| 101 | 45 | 180 |
| 101 | 50 | 200 |
| 102 | 60 | 210 |
| 102 | 65 | 220 |

```
data all;
    merge sbp lipids;
    by id;
run;
```

**Merged Dataset ALL**

| id | sbp | hdl | ldl |
|---|---|---|---|
| 101 | 80 | 45 | 180 |
| 101 | 85 | 50 | 200 |
| 102 | 90 | 60 | 210 |
| 102 | 95 | 65 | 220 |
| 102 | 100 | 65 | 220 |

```
520
521  data all;
522      merge sbp lipids;
523      by id;
524  run;

NOTE: MERGE statement has more than one data set with repeats of BY
values.
NOTE: There were 5 observations read from the data set WORK.SBP.
NOTE: There were 4 observations read from the data set WORK.LIPIDS.
NOTE: The data set WORK.ALL has 5 observations and 4 variables.
NOTE: DATA statement used (Total process time):
      real time           0.04 seconds
      cpu time            0.03 seconds
```

This is usually a situation you want to avoid!

Possible strategies if you encounter it:

- Have you omitted a BY variable that would result in unique matches? For example, see page 14 where both ID and VISIT were needed as BY variables.

- Does at least one of the datasets have duplicate records that you should get rid of?

## The UPDATE Statement

- The UPDATE statement joins observations from two SAS datasets, an "old master file" dataset and a "transaction file" dataset. The resulting observations form a "new master file" dataset. These observations contain the values of the variables in the old master file, except for those modifications specified in the transaction file.

- Syntax:

```
UPDATE SASDS1 SASDS2;
BY Variables;
```

  where:

  - SASDS1 is the old master file. It must be sorted by the BY variables and must contain only one observation for each BY group.

  - SASDS2 is the transaction file. It must be sorted by the BY variables, but may have multiple observations per BY group.

- Variables with non-missing values in the transaction file will replace those values in the PDV. Missing values will not replace values in the PDV.

- The PDV is initialized to missing only at the beginning of a new BY group.

- The PDV is output to the new master file only at the end of a BY group.

## Update Example 1

```
data grades2;
   update grades fixhw;
   by name hwnum;
run;
```

Objective: Update the scores in the GRADES dataset with the scores in the FIXHW dataset.

**Master file**

**Dataset GRADES**

| NAME | SEX | hwnum | score |
|---|---|---|---|
| CHRISTIANSEN | M | 1 | 10 |
| CHRISTIANSEN | M | 2 | 3 |
| CHRISTIANSEN | M | 4 | 9 |
| CHRISTIANSEN | M | 5 | 10 |
| FROG K | M | 1 | 10 |
| FROG K | M | 2 | 10 |
| FROG K | M | 3 | 8 |
| GONZO | | 1 | . |
| HELMS R | M | . | . |
| HOSKING J | M | 1 | 10 |
| PIGGY M | F | 2 | 10 |
| PIGGY M | F | 3 | 10 |

**Transaction file**

**Dataset FIXHW**

| name | hwnum | score |
|---|---|---|
| CHRISTIANSEN | 1 | 9 |
| CHRISTIANSEN | 2 | 10 |
| CHRISTIANSEN | 2 | 2 |
| CHRISTIANSEN | 4 | . |
| FROG K | 2 | . |
| HOSKING J | 1 | . |

**Updated Dataset GRADES2**

| NAME | SEX | hwnum | score |
|---|---|---|---|
| CHRISTIANSEN | M | 1 | 9 |
| CHRISTIANSEN | M | 2 | 2 |
| CHRISTIANSEN | M | 4 | 9 |
| CHRISTIANSEN | M | 5 | 10 |
| FROG K | M | 1 | 10 |
| FROG K | M | 2 | 10 |
| FROG K | M | 3 | 8 |
| GONZO | | 1 | . |
| HELMS R | M | . | . |
| HOSKING J | M | 1 | 10 |
| PIGGY M | F | 2 | 10 |
| PIGGY M | F | 3 | 10 |

Notice that observations in the FIXHW dataset with a missing value for SCORE were NOT updated in the GRADES2 dataset. In other words, these observations retain their original value for SCORE as given in the GRADES dataset.

## Update Example 2: Regular Missing Values

Sometimes you DO want to update a value to a missing value. In order to set the value of a variable to a regular missing value using an UPDATE statement, the variable must be set to a single underscore ( ._ ) as a special missing value in the transaction file. The resulting value in the master dataset will be a period ( . ) for missing numeric values and a blank for missing character values.

```
data grades2;
   update grades fixhw;
   by name hwnum;
run;
```

**Master file**

**Dataset GRADES**

| NAME | SEX | hwnum | score |
|---|---|---|---|
| CHRISTIANSEN | M | 1 | 10 |
| CHRISTIANSEN | M | 2 | 3 |
| CHRISTIANSEN | M | 4 | 9 |
| CHRISTIANSEN | M | 5 | 10 |
| FROG K | M | 1 | 10 |
| FROG K | M | 2 | 10 |
| FROG K | M | 3 | 8 |
| GONZO | | 1 | . |
| HELMS R | M | . | . |
| HOSKING J | M | 1 | 10 |
| PIGGY M | F | 2 | 10 |
| PIGGY M | F | 3 | 10 |

**Transaction file**

**Dataset FIXHW**

| name | hwnum | score |
|---|---|---|
| CHRISTIANSEN | 1 | 9 |
| CHRISTIANSEN | 2 | 10 |
| CHRISTIANSEN | 2 | 2 |
| CHRISTIANSEN | 4 | _ |
| FROG K | 2 | . |
| HOSKING J | 1 | _ |

Notice the missing values for ._ here

**Updated Dataset GRADES2**

| NAME | SEX | hwnum | score |
|---|---|---|---|
| CHRISTIANSEN | M | 1 | 9 |
| CHRISTIANSEN | M | 2 | 2 |
| CHRISTIANSEN | M | 4 | . |
| CHRISTIANSEN | M | 5 | 10 |
| FROG K | M | 1 | 10 |
| FROG K | M | 2 | 10 |
| FROG K | M | 3 | 8 |
| GONZO | | 1 | . |
| HELMS R | M | . | . |
| HOSKING J | M | 1 | . |
| PIGGY M | F | 2 | 10 |
| PIGGY M | F | 3 | 10 |

Notice that observations in the FIXHW dataset with a ._ missing value for SCORE were updated to a missing value of . in the GRADES2 dataset.

## Update Example 3: Special Missing Values

If special missing values A through Z appear in the transaction dataset, numeric variables in the master dataset are updated to those values.

```
data all;
   update master trans;
   by id;
run;
```

**Dataset MASTER**

| id | x |
|---|---|
| 1 | 10 |
| 2 | 20 |
| 3 | 30 |
| 4 | 40 |

**Dataset TRANS**

| id | x |
|---|---|
| 1 | A |
| 1 | . |
| 2 | C |
| 3 | 10 |
| 3 | . |
| 4 | _ |

**Dataset ALL**

| id | x |
|---|---|
| 1 | A |
| 2 | C |
| 3 | 10 |
| 4 | . |

## SAS Special Variables

- The following special variables can be created in DATA steps using SET, MERGE, or UPDATE statements.

    1. The IN variable indicates whether the dataset contributed to the current observation.

    2. The END variable indicates that the current observation is the last to be processed.

    3. FIRST.byvariable and LAST.byvariable indicate if the current observation is the first or last observation in the BY group.

- Syntax: On a SET, MERGE, or UPDATE statement, use the following options when specifying the dataset:

```
dataset(IN=invariable1) END=endvariable;
```

- You control creation of the IN and END variables. The FIRST. and LAST. variables are created whenever your DATA step includes a BY statement, as follows.

  If a DATA step includes the BY statement: `BY var1 var2 ...;`

  Then variables FIRST.var1, LAST.var1, FIRST.var2, LAST.var2, and so on, are automatically created.

## SAS Special Variables

- The special variables exist in the PDV, but not in the output datasets.

- The special variables are indicator variables with values of 0 (False) and 1 (True).

- An IN variable can be created for each dataset in a SET, MERGE, or UPDATE statement.

- An END variable can be created for each SET, MERGE, or UPDATE statement.

- At any point in a DATA step, you can use a `PUT _ALL_;` statement to see the current state of the PDV.

## Special Variables Example 1

```
data carsales;
   set jan(in=inj) feb(in=inf) end=eof;
   by make;
   put _all_;
run;
```

**Dataset JAN**

| MAKE | YEAR | PRICE |
|---|---|---|
| CHEVY | 76 | 3550 |
| CHEVY | 72 | 1350 |
| CHEVY | 69 | 1165 |
| FORD | 69 | 700 |
| FORD | 76 | 3245 |

**Dataset FEB**

| MAKE | YEAR | PRICE |
|---|---|---|
| CHEVY | 76 | 2525 |
| FORD | 75 | 2985 |
| FORD | 63 | 695 |
| OLD | 72 | 2383 |

Written to the log:

```
663
664  data carsales;
665     set jan(in=inj) feb(in=inf) end=eof;
666     by make;
667     put _all_;
668  run;

inj=1 inf=0 eof=0 MAKE=CHEVY YEAR=76 PRICE=3550 FIRST.MAKE=1 LAST.MAKE=0 _ERROR_=0 _N_=1
inj=1 inf=0 eof=0 MAKE=CHEVY YEAR=72 PRICE=1350 FIRST.MAKE=0 LAST.MAKE=0 _ERROR_=0 _N_=2
inj=1 inf=0 eof=0 MAKE=CHEVY YEAR=69 PRICE=1165 FIRST.MAKE=0 LAST.MAKE=0 _ERROR_=0 _N_=3
inj=0 inf=1 eof=0 MAKE=CHEVY YEAR=76 PRICE=2525 FIRST.MAKE=0 LAST.MAKE=1 _ERROR_=0 _N_=4
inj=1 inf=0 eof=0 MAKE=FORD YEAR=69 PRICE=700 FIRST.MAKE=1 LAST.MAKE=0 _ERROR_=0 _N_=5
inj=1 inf=0 eof=0 MAKE=FORD YEAR=76 PRICE=3245 FIRST.MAKE=0 LAST.MAKE=0 _ERROR_=0 _N_=6
inj=0 inf=1 eof=0 MAKE=FORD YEAR=75 PRICE=2985 FIRST.MAKE=0 LAST.MAKE=0 _ERROR_=0 _N_=7
inj=0 inf=1 eof=0 MAKE=FORD YEAR=63 PRICE=695 FIRST.MAKE=0 LAST.MAKE=1 _ERROR_=0 _N_=8
inj=0 inf=1 eof=1 MAKE=OLD YEAR=72 PRICE=2383 FIRST.MAKE=1 LAST.MAKE=1 _ERROR_=0 _N_=9
NOTE: There were 5 observations read from the data set WORK.JAN.
NOTE: There were 4 observations read from the data set WORK.FEB.
NOTE: The data set WORK.CARSALES has 9 observations and 3 variables.
```

Contents of the Program Data Vector:

| MAKE | YEAR | PRICE | INJ | INF | EOF | FIRST. MAKE | LAST. MAKE |
|---|---|---|---|---|---|---|---|
| CHEVY | 76 | 3550 | 1 | 0 | 0 | 1 | 0 |
| CHEVY | 72 | 1350 | 1 | 0 | 0 | 0 | 0 |
| CHEVY | 69 | 1165 | 1 | 0 | 0 | 0 | 0 |
| CHEVY | 76 | 2525 | 0 | 1 | 0 | 0 | 1 |
| FORD | 69 | 700 | 1 | 0 | 0 | 1 | 0 |
| FORD | 76 | 3245 | 1 | 0 | 0 | 0 | 0 |
| FORD | 75 | 2985 | 0 | 1 | 0 | 0 | 0 |
| FORD | 63 | 695 | 0 | 1 | 0 | 0 | 1 |
| OLDS | 72 | 2383 | 0 | 1 | 1 | 1 | 1 |

## Special Variables Example 2

Create a dataset containing only observations where a match occurred on the BY variable.

**Dataset LIST2**

| NAME | SEX |
|---|---|
| CHRISTIANSEN | M |
| FROG K | M |
| GONZO | |
| HOSKING J | M |
| PIGGY M | F |

**Dataset DEMO2**

| NAME | AGE | HT | WT |
|---|---|---|---|
| CHRISTIANSEN | 37 | 71 | 195 |
| FROG K | 3 | 12 | 1 |
| HELMS R | 41 | 74 | 195 |
| PIGGY M | . | 48 | . |

```
data matches;
   merge list2(in=inlist) demo2(in=indemo);
   by name;
   put 'Before IF: ' _all_;
   if inlist and indemo;
   put 'After IF: ' _all_;
run;
```

**Merged Dataset MATCHES**

| NAME | SEX | AGE | HT | WT |
|---|---|---|---|---|
| CHRISTIANSEN | M | 37 | 71 | 195 |
| FROG K | M | 3 | 12 | 1 |
| PIGGY M | F | . | 48 | . |

SAS Log:

```
703  data matches;
704     merge list2(in=inlist) demo2(in=indemo);
705     by name;
706     put 'Before IF: ' _all_;
707     if inlist and indemo;
708     put 'After IF: ' _all_;
709  run;

Before IF: inlist=1 indemo=1 NAME=CHRISTIANSEN SEX=M AGE=37 HT=71 WT=195 FIRST.NAME=1 LAST.NAME=1
_ERROR_=0 _N_=1
After IF: inlist=1 indemo=1 NAME=CHRISTIANSEN SEX=M AGE=37 HT=71 WT=195 FIRST.NAME=1 LAST.NAME=1
_ERROR_=0 _N_=1
Before IF: inlist=1 indemo=1 NAME=FROG K SEX=M AGE=3 HT=12 WT=1 FIRST.NAME=1 LAST.NAME=1 _ERROR_=0
_N_=2
After IF: inlist=1 indemo=1 NAME=FROG K SEX=M AGE=3 HT=12 WT=1 FIRST.NAME=1 LAST.NAME=1 _ERROR_=0
_N_=2
Before IF: inlist=1 indemo=0 NAME=GONZO SEX=  AGE=. HT=. WT=. FIRST.NAME=1 LAST.NAME=1 _ERROR_=0
_N_=3
Before IF: inlist=0 indemo=1 NAME=HELMS R SEX=  AGE=41 HT=74 WT=195 FIRST.NAME=1 LAST.NAME=1
_ERROR_=0 _N_=4
Before IF: inlist=1 indemo=0 NAME=HOSKING J SEX=M AGE=. HT=. WT=. FIRST.NAME=1 LAST.NAME=1
_ERROR_=0 _N_=5
Before IF: inlist=1 indemo=1 NAME=PIGGY M SEX=F AGE=. HT=48 WT=. FIRST.NAME=1 LAST.NAME=1
_ERROR_=0 _N_=6
After IF: inlist=1 indemo=1 NAME=PIGGY M SEX=F AGE=. HT=48 WT=. FIRST.NAME=1 LAST.NAME=1 _ERROR_=0
_N_=6
NOTE: There were 5 observations read from the data set WORK.LIST2.
NOTE: There were 4 observations read from the data set WORK.DEMO2.
NOTE: The data set WORK.MATCHES has 3 observations and 5 variables.
NOTE: DATA statement used (Total process time):
      real time           0.06 seconds
      cpu time            0.04 seconds
```

Other possibilities:

```
if inlist;
if indemo;
if inlist and ^indemo;
if indemo and ^inlist;
```

By default:

```
if inlist or indemo;
```

## One-to-One Match Merge Example Using IN Variables

Objectives:

- Merge pre-rand and post-rand datasets by ID.
- Only keep subjects who have a record in both files.
- Create variables for the difference between pre-randomization and post-randomization values.

```
data all;
   merge prerand(in=inpre rename=(hdl=hdl0 ldl=ldl0))
       postrand(in=inpost rename=(hdl=hdl1 ldl=ldl1));
   by id;

   if (inpre=1) & (inpost=1);
     /* or if inpre & inpost; */

   hdldiff = hdl1 - hdl0;
   ldldiff = ldl1 - ldl0;
run;
```

**Dataset PRERAND**

| id | hdl | ldl |
|---|---|---|
| 101 | 40 | 200 |
| 102 | 45 | 210 |
| 103 | 55 | 215 |
| 104 | 60 | 220 |

**Dataset POSTRAND**

| id | hdl | ldl |
|---|---|---|
| 101 | 45 | 180 |
| 102 | 50 | 200 |
| 103 | 60 | 210 |
| 105 | 65 | 230 |

**Merged Dataset ALL**

| id | hdl0 | ldl0 | hdl1 | ldl1 | hdldiff | ldldiff |
|---|---|---|---|---|---|---|
| 101 | 40 | 200 | 45 | 180 | 5 | -20 |
| 102 | 45 | 210 | 50 | 200 | 5 | -10 |
| 103 | 55 | 215 | 60 | 210 | 5 | -5 |

## Another Match Merge Example Using IN Variables

Merge SBP and LIPID datasets by ID and VISIT. Only keep combinations of ID and VISIT with a record in both files.

**Dataset SBP**

| id | visit | sbp |
|---|---|---|
| 101 | 1 | 80 |
| 101 | 2 | 85 |
| 102 | 1 | 90 |
| 102 | 2 | 95 |
| 102 | 4 | 100 |
| 103 | 1 | 110 |

**Dataset LIPIDS**

| id | visit | hdl | ldl |
|---|---|---|---|
| 101 | 1 | 45 | 180 |
| 101 | 2 | 50 | 200 |
| 102 | 1 | 60 | 210 |
| 102 | 2 | 65 | 220 |
| 102 | 3 | 70 | 230 |

```
data all;
   merge sbp(in=ins)
         lipids(in=inl);
   by id visit;
   if ins & inl;    /* include if in both files */
run;
```

**Merged Dataset ALL**

| id | visit | sbp | hdl | ldl |
|---|---|---|---|---|
| 101 | 1 | 80 | 45 | 180 |
| 101 | 2 | 85 | 50 | 200 |
| 102 | 1 | 90 | 60 | 210 |
| 102 | 2 | 95 | 65 | 220 |

Contents of the Program Data Vector

| ID | VISIT | SBP | HDL | LDL | INS | INL |
|---|---|---|---|---|---|---|
| 101 | 1 | 80 | 45 | 180 | 1 | 1 |
| 101 | 2 | 85 | 50 | 200 | 1 | 1 |
| 102 | 1 | 90 | 60 | 210 | 1 | 1 |
| 102 | 2 | 95 | 65 | 220 | 1 | 1 |
| 102 | 3 | . | 70 | 230 | 0 | 1 |
| 102 | 4 | 100 | . | . | 1 | 0 |
| 103 | 1 | 110 | . | . | 1 | 0 |

## Taking a Closer Look at FIRST. and LAST.

Say you have a dataset of addresses sorted by State, City, and ZipCode, in that order. The following are all of the observations in four BY groups, along with their corresponding FIRST. and LAST. values.

| State | City | Zip Code | Street | FIRST. State | LAST. State | FIRST. City | LAST. City | FIRST. ZipCode | LAST. ZipCode |
|---|---|---|---|---|---|---|---|---|---|
| AZ | Tucson | 85730 | Gleeson Pl | 1 | 1 | 1 | 1 | 1 | 1 |
| FL | Miami | 33133 | Calumet St | 1 | 0 | 1 | 0 | 1 | 0 |
| FL | Miami | 33133 | Thomas Ave | 0 | 0 | 0 | 0 | 0 | 0 |
| FL | Miami | 33133 | Surrey Dr | 0 | 0 | 0 | 0 | 0 | 1 |
| FL | Miami | 33146 | Nervia St | 0 | 0 | 0 | 0 | 1 | 0 |
| FL | Miami | 33146 | Corsica St | 0 | 1 | 0 | 1 | 0 | 1 |
| OH | Miami | 45056 | Myrtle St | 1 | 1 | 1 | 1 | 1 | 1 |

With the FIRST. and LAST. variables, you can detect four conditions:

1. Whether an observation is the first observation in a BY group.

2. Whether an observation is the last observation in a BY group.

3. Whether an observation is neither the first nor the last observation in a BY group.

4. Whether an observation is both the first and last (the only) observation in a BY group.

## Match Merge Example Using IN and FIRST.BY / LAST.BY Variables

Objectives:

- Merge SBP and LIPID datasets by ID and VISIT.
- Only keep subjects who have a record in both files.
- only keep the first record for each subject.

**Dataset SBP**

| id | visit | sbp |
|---|---|---|
| 101 | 1 | 80 |
| 101 | 2 | 85 |
| 102 | 1 | 90 |
| 102 | 2 | 95 |
| 102 | 4 | 100 |
| 103 | 1 | 110 |

**Dataset LIPIDS**

| id | visit | hdl | ldl |
|---|---|---|---|
| 101 | 1 | 45 | 180 |
| 101 | 2 | 50 | 200 |
| 102 | 1 | 60 | 210 |
| 102 | 2 | 65 | 220 |
| 102 | 3 | 70 | 230 |

```
data all;
   merge sbp(in=ins)
         lipids(in=inl);
   by id visit;  /*must sort by id and visit*/
   if (ins & inl) & first.id;
run;
```

**Merged Dataset ALL**

| id | visit | sbp | hdl | ldl |
|---|---|---|---|---|
| 101 | 1 | 80 | 45 | 180 |
| 102 | 1 | 90 | 60 | 210 |

Contents of the Program Data Vector:

| ID | VISIT | INS | INL | FIRST. ID | LAST. ID | FIRST. VISIT | LAST. VISIT |
|---|---|---|---|---|---|---|---|
| 101 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 101 | 2 | 1 | 1 | 0 | 1 | 1 | 1 |
| 102 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 102 | 2 | 1 | 1 | 0 | 0 | 1 | 1 |
| 102 | 3 | 0 | 1 | 0 | 0 | 1 | 1 |
| 102 | 4 | 1 | 0 | 0 | 1 | 1 | 1 |
| 103 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |

## END Variable Example

```
DATA one;
    SET bios511.class END=eof;

    KEEP n f;

    n + 1;
        /* or RETAIN n 0; n = n + 1; */

    f + (sex='F');
        /* or RETAIN f 0; f = f + (sex='F'); */

    IF (eof=1) THEN OUTPUT;
        /* Output if last record */

RUN;
```

**Dataset CLASS**

| NAME | SEX | AGE | HT | WT |
|---|---|---|---|---|
| CHRISTIANSEN | M | 37 | 71 | 195 |
| HOSKING J | M | 31 | 70 | 160 |
| HELMS R | M | 41 | 74 | 195 |
| PIGGY M | F | . | 48 | . |
| FROG K | M | 3 | 12 | 1 |
| GONZO | | 14 | 25 | 45 |

**Dataset ONE**

| n | f |
|---|---|
| 6 | 1 |

**Dataset ONE Without IF (EOF=1)**

| n | f |
|---|---|
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 1 |
| 5 | 1 |
| 6 | 1 |

## Transposing and Aggregating

- Transposing means going from several observations to one while preserving all variable values of interest, or vice versa.
    - Transposing from several records to one is also called "flattening" or going from a "long" to a "wide" dataset.
    - Transposing from one record to several records is also called going from a "wide" to a "long" dataset.
    - If transposing is done correctly, you can go from "wide" to "long" to "wide" to "long" ad infinitum without losing information since all values of interest are preserved.
    - Depending on the variables you want to create, the reports you want to produce, or the analyses you need to do, either a "wide" or a "long" dataset might be more appropriate.
- Aggregating means going from several observations to one without preserving all variable values, instead transforming the values into characteristics of interest.
- PROC TRANSPOSE is a SAS procedure specifically for transposing data. We do not cover it in this course because it is somewhat mysterious, and because I think it is important for you to understand how to transpose in a DATA step. However, there are some situations where PROC TRANSPOSE can be useful, and I encourage you to read about the procedure in the SAS documentation if you are interested.

## Example: Transpose Dataset LIPIDS From Multiple Records Per Subject to One Record Per Subject, or Going From "Long" to "Wide"

(General solution to problem on p. 77 of Chapter 5)

```
proc sort data=lipids;
    by id visit;
run;

data two;
    set lipids;
    by id;
    retain ldl1 ldl2 hdl1 hdl2;

    array vars{4} ldl1 ldl2 hdl1 hdl2;
    if first.id then do i = 1 to 4;
        vars{i} = .;
        end;

    if visit=1 then do;
        ldl1 = ldl;
        hdl1 = hdl;
        end;
    if visit=2 then do;
        ldl2 = ldl;
        hdl2 = hdl;
        end;

    keep id ldl1 ldl2 hdl1 hdl2;

    if last.id then output;
run;
```

## Dataset LIPIDS
## Long Format

| id | visit | ldl | hdl |
|---|---|---|---|
| 101 | 1 | 180 | 45 |
| 101 | 2 | 185 | 50 |
| 102 | 1 | 160 | 40 |
| 102 | 2 | 170 | . |
| 103 | 1 | 190 | 60 |

## Transposed Dataset TWO
## Wide Format

| id | ldl1 | ldl2 | hdl1 | hdl2 |
|---|---|---|---|---|
| 101 | 180 | 185 | 45 | 50 |
| 102 | 160 | 170 | 40 | . |
| 103 | 190 | . | 60 | . |

## Aggregation Techniques

- Sometimes you will need to collapse a group of records to one record that represents the group and reflects some group quality: perhaps the number of records in the group, or whether a certain condition ever occurred in the group, or the minimum, maximum, sum, or mean of some variable in the group. This is also called aggregating or aggregation.

- In the following examples, ID is the grouping variable. In each initial dataset, there are >=1 records per ID, but in each final dataset, there is only one record per ID. A sample initial dataset (code to create is in sample programs):

**Dataset INITIAL**

| id | x |
|---|---|
| 1 | 4 |
| 1 | 5 |
| 1 | 1 |
| 2 | 1 |
| 3 | 2 |
| 3 | 5 |

## Aggregation Example: Counting the Number of Records in a Group

```
proc sort data=initial;
    by id;
run;

data idcounts1;
    set initial;
    by id;
    retain count;
    if first.id then count=0;
    count = count + 1;
    if last.id then output;
    keep id count;
run;
```

Or:

```
proc freq data=initial;
    tables id / out=idcounts2;
run;
```

**Dataset IDCOUNTS1**
**Counts from DATA Step**

| id | count |
|---|---|
| 1 | 3 |
| 2 | 1 |
| 3 | 2 |

**Dataset IDCOUNTS2**
**Counts from PROC FREQ**

| id | COUNT | PERCENT |
|---|---|---|
| 1 | 3 | 50.0000 |
| 2 | 1 | 16.6667 |
| 3 | 2 | 33.3333 |

## Aggregation Example: Flagging Whether a Certain Condition Ever Occurred in a Group

```
proc sort data=initial;
    by id;
run;

data idflags;
    set initial;
    by id;
    retain flag;
    if first.id then flag=0;
    if x>1 then flag=1;
    if last.id then output;
    keep id flag;
run;
```

**Dataset IDFLAGS**
**Flag Obs with X>1**

| id | flag |
|---|---|
| 1 | 1 |
| 2 | 0 |
| 3 | 1 |

## Aggregation Example: Finding the Sum/ Minimum/ Maximum/ Mean of a Variable Within a Group

```
proc means data=initial nway noprint;
    class id;
    var x;
    output out=idsums1 sum=sum;
run;
```

Or:

```
proc sort data=initial;
    by id;
run;

data idsums2;
    set initial;
    by id;
    retain sum;
    if first.id then sum=.;
    sum = sum(sum,x);
    if last.id then output;
    keep id sum;
run;
```

**Dataset IDSUMS1**
**Sum of X by ID, Using PROC MEANS**

| id | _TYPE_ | _FREQ_ | sum |
|---|---|---|---|
| 1 | 1 | 3 | 10 |
| 2 | 1 | 1 | 1 |
| 3 | 1 | 2 | 7 |

**Dataset IDSUMS2**
**Sum of X by ID, Using DATA Step**

| id | sum |
|---|---|
| 1 | 10 |
| 2 | 1 |
| 3 | 7 |

## Multiple Output Datasets

- A single DATA step can create more than one SAS dataset using the more general forms of the DATA and OUTPUT statements:

- Example:

```
data sasds1  sasds2 . . . sasdsn;
    output sasdsx . . .;
run;
```

  where:

  - SASDS1 . . . SASDSN are the names of the SAS datasets to be created in this step, and

  - SASDSX is a one of the N dataset names listed in the DATA statement.

- A DATA step may contain multiple OUTPUT statements.

- Each OUTPUT statement may specify any combination of the SAS datasets listed in the DATA statement.

- The current contents of the PDV are written to these datasets each time the OUTPUT statement is executed.

- The statement OUTPUT; (without a list of dataset names) writes the PDV to all datasets in the DATA statement.

- A DATA step without an OUTPUT statement defaults to writing to all output datasets each time the bottom of the step is reached.

## Example: Create SHORT and TALL datasets From CLASS in one DATA step

```
data short tall;
    set class;
    if ht lt 60 then output short;
    else if ht ge 60 then output tall;
run;
```

**Dataset CLASS**

| NAME | SEX | AGE | HT | WT |
|---|---|---|---|---|
| CHRISTIANSEN | M | 37 | 71 | 195 |
| HOSKING J | M | 31 | 70 | 160 |
| HELMS R | M | 41 | 74 | 195 |
| PIGGY M | F | . | 48 | . |
| FROG K | M | 3 | 12 | 1 |
| GONZO | | 14 | 25 | 45 |

**Dataset SHORT**

| NAME | SEX | AGE | HT | WT |
|---|---|---|---|---|
| PIGGY M | F | . | 48 | . |
| FROG K | M | 3 | 12 | 1 |
| GONZO | | 14 | 25 | 45 |

**Dataset TALL**

| NAME | SEX | AGE | HT | WT |
|---|---|---|---|---|
| CHRISTIANSEN | M | 37 | 71 | 195 |
| HOSKING J | M | 31 | 70 | 160 |
| HELMS R | M | 41 | 74 | 195 |

## Example: Combining Multiple Input and Output Datasets

```
data tall boys all;
    merge list demo;
    by name;
    if ht >= 60 then output tall;
    if sex = 'M' then output boys;
    output all;
run;
```

**Dataset LIST**

| NAME | SEX |
|---|---|
| CHRISTIANSEN | M |
| FROG K | M |
| GONZO | |
| HELMS R | M |
| HOSKING J | M |
| PIGGY M | F |

**Dataset DEMO**

| NAME | AGE | HT | WT |
|---|---|---|---|
| CHRISTIANSEN | 37 | 71 | 195 |
| FROG K | 3 | 12 | 1 |
| GONZO | 14 | 25 | 45 |
| HELMS R | 41 | 74 | 195 |
| HOSKING J | 31 | 70 | 160 |
| PIGGY M | . | 48 | . |

## Dataset TALL

| NAME | SEX | AGE | HT | WT |
|---|---|---|---|---|
| CHRISTIANSEN | M | 37 | 71 | 195 |
| HOSKING J | M | 31 | 70 | 160 |
| HELMS R | M | 41 | 74 | 195 |

## Dataset BOYS

| NAME | SEX | AGE | HT | WT |
|---|---|---|---|---|
| CHRISTIANSEN | M | 37 | 71 | 195 |
| FROG K | M | 3 | 12 | 1 |
| HELMS R | M | 41 | 74 | 195 |
| HOSKING J | M | 31 | 70 | 160 |

v

## Dataset ALL

| NAME | SEX | AGE | HT | WT |
|---|---|---|---|---|
| CHRISTIANSEN | M | 37 | 71 | 195 |
| FROG K | M | 3 | 12 | 1 |
| GONZO | | 14 | 25 | 45 |
| HELMS R | M | 41 | 74 | 195 |
| HOSKING J | M | 31 | 70 | 160 |
| PIGGY M | F | . | 48 | . |

## Example: Multiple Output Datasets Containing Different Types of Information

```
data one(drop=n f) counts(keep=n f);
   set bios511.class end=eof;

   retain n f 0;

   n = n + 1;
   f = f + (sex='F');

   if (eof) then output counts;
   output one;
run;
```

**Dataset CLASS**

| NAME | SEX | AGE | HT | WT |
|---|---|---|---|---|
| CHRISTIANSEN | M | 37 | 71 | 195 |
| HOSKING J | M | 31 | 70 | 160 |
| HELMS R | M | 41 | 74 | 195 |
| PIGGY M | F | . | 48 | . |
| FROG K | M | 3 | 12 | 1 |
| GONZO | | 14 | 25 | 45 |

**Dataset ONE**

| NAME | SEX | AGE | HT | WT |
|---|---|---|---|---|
| CHRISTIANSEN | M | 37 | 71 | 195 |
| HOSKING J | M | 31 | 70 | 160 |
| HELMS R | M | 41 | 74 | 195 |
| PIGGY M | F | . | 48 | . |
| FROG K | M | 3 | 12 | 1 |
| GONZO | | 14 | 25 | 45 |

**Dataset COUNTS**

| n | f |
|---|---|
| 6 | 1 |

## Reshaping SAS Datasets

- Multiple output observations can be created from a single input observation by using several OUTPUT statements. This can be called transposing from "wide" to "long".

- The DROP and KEEP dataset options can be used to control which variables are included or excluded from the PDV and the output datasets.

- The RENAME dataset option can change the names of variables in the PDV and output datasets.

- The WHERE dataset option can control which observations are read into the PDV from an input dataset.

## Example: Multiple Output Observations From a Single Input Observation (Transposing: Going From "Wide" to "Long")

Reshape the dataset from one observation per pupil to one observation per test.

See p. 75 of Chapter 5 for a similar example.

```
data test;
   set work.students;
   exam=1; grade=test1; output;
   exam=2; grade=test2; output;
   exam=3; grade=test3; output;
run;
```

**Dataset STUDENTS**

| name | test1 | test2 | test3 |
|---|---|---|---|
| John | 90 | 70 | 50 |
| Albert | 100 | 33 | 89 |
| Sally | 95 | 80 | 90 |
| Mary | 50 | 100 | 92 |

**Dataset TEST**

| name | test1 | test2 | test3 | exam | grade |
|---|---|---|---|---|---|
| John | 90 | 70 | 50 | 1 | 90 |
| John | 90 | 70 | 50 | 2 | 70 |
| John | 90 | 70 | 50 | 3 | 50 |
| Albert | 100 | 33 | 89 | 1 | 100 |
| Albert | 100 | 33 | 89 | 2 | 33 |
| Albert | 100 | 33 | 89 | 3 | 89 |
| Sally | 95 | 80 | 90 | 1 | 95 |
| Sally | 95 | 80 | 90 | 2 | 80 |
| Sally | 95 | 80 | 90 | 3 | 90 |
| Mary | 50 | 100 | 92 | 1 | 50 |
| Mary | 50 | 100 | 92 | 2 | 100 |
| Mary | 50 | 100 | 92 | 3 | 92 |

## Example: Multiple Output Observations From a Single Input Observation (Transposing: Going From "Wide" to "Long") Using an Array and DO Loop

This example also shows dropping unwanted variables with the DROP statement. See p. 75 of Chapter 5 for a similar example.

```
data test;
    set work.students;
    drop test1-test3 i;
    array test{3} test1-test3;
    do i = 1 to 3;
        exam = i;
        grade = test{i};
        output;
    end;
run;
```

**Dataset TEST**

| name | exam | grade |
|---|---|---|
| John | 1 | 90 |
| John | 2 | 70 |
| John | 3 | 50 |
| Albert | 1 | 100 |
| Albert | 2 | 33 |
| Albert | 3 | 89 |
| Sally | 1 | 95 |
| Sally | 2 | 80 |
| Sally | 3 | 90 |
| Mary | 1 | 50 |
| Mary | 2 | 100 |
| Mary | 3 | 92 |

## Example: Expanding a Dataset for Counting Purposes

In dataset HOBBIES, how can we count how many people had each hobby?

**HOBBIES Dataset**

| NAME | SEX | AGE | HT | WT | Hobbies |
|---|---|---|---|---|---|
| CHRISTIANSEN | M | 37 | 71 | 195 | read, swim, baseball |
| HOSKING J | M | 31 | 70 | 160 | eat, travel |
| HELMS R | M | 41 | 74 | 195 | travel |
| PIGGY M | F | . | 48 | . | read, primp, eat |
| FROG K | M | 3 | 12 | 1 | swim |
| GONZO | | 14 | 25 | 45 | eat, baseball |

```
data hobbies;
    set bios511.hobbies;
    keep name hobby;

    i=1;
    do until (scan(hobbies,i)=' ');
        hobby=scan(hobbies,i);
        output;
        i=i+1;
    end;
run;

proc print data=hobbies;
    title1 'Expanded HOBBIES dataset';
run;
```

```
proc freq data=hobbies;
    tables hobby;
    title1 'How many people had each hobby?';
run;
```

## Expanded HOBBIES Dataset

| NAME | hobby |
|---|---|
| CHRISTIANSEN | read |
| CHRISTIANSEN | swim |
| CHRISTIANSEN | baseball |
| HOSKING J | eat |
| HOSKING J | travel |
| HELMS R | travel |
| PIGGY M | read |
| PIGGY M | primp |
| PIGGY M | eat |
| FROG K | swim |
| GONZO | eat |
| GONZO | baseball |

## How many people had each hobby?

### The FREQ Procedure

| hobby | Frequency | Percent | Cumulative Frequency | Cumulative Percent |
|---|---|---|---|---|
| baseball | 2 | 16.67 | 2 | 16.67 |
| eat | 3 | 25.00 | 5 | 41.67 |
| primp | 1 | 8.33 | 6 | 50.00 |
| read | 2 | 16.67 | 8 | 66.67 |
| swim | 2 | 16.67 | 10 | 83.33 |
| travel | 2 | 16.67 | 12 | 100.00 |

## The DROP and KEEP Dataset Options

The DROP and KEEP dataset options can be used on each output dataset named in the DATA statement to control which of the variables in the PDV get written to which output datasets.

```
data list(keep=name sex) demo(drop=sex age);
    set bios511.class;
run;
```

**Dataset CLASS**

| NAME | SEX | AGE | HT | WT |
|---|---|---|---|---|
| CHRISTIANSEN | M | 37 | 71 | 195 |
| HOSKING J | M | 31 | 70 | 160 |
| HELMS R | M | 41 | 74 | 195 |
| PIGGY M | F | . | 48 | . |
| FROG K | M | 3 | 12 | 1 |
| GONZO | | 14 | 25 | 45 |

Dataset CLASS contains variables: `NAME, SEX, AGE, HT, WT`

The PDV contains variables: `NAME, SEX, AGE, HT, WT`

Dataset LIST contains variables: `NAME, SEX`

Dataset DEMO contains variables: `NAME, HT, WT`

## The DROP and KEEP Dataset Options

The DROP and KEEP options can be used on the input datasets listed in SET, MERGE, and UPDATE statements to control which variables get included in the PDV. This can lead to great improvements in program efficiency.

```
data both;
    merge class1(drop=age) class2(drop=sex);
    by name;
run;
```

Suppose dataset CLASS1 has variables NAME, SEX and AGE and dataset CLASS2 has variables NAME, SEX, AGE, and HT.

| | |
|---|---|
| Variables included in CLASS1: | NAME, SEX, AGE |
| Variables read from CLASS1: | NAME, SEX |
| Variables included in CLASS2: | NAME, SEX, AGE, HT |
| Variables read from CLASS2: | NAME, AGE, HT |
| The PDV contains variables: | NAME, SEX, AGE, HT |
| Dataset BOTH contains variables: | NAME, SEX, AGE, HT |

Note that variable SEX in dataset BOTH comes from CLASS1 (not from CLASS2), whereas variable AGE in dataset BOTH comes from CLASS2 (not from CLASS1).

## The RENAME Dataset Option

- The RENAME option can be used on SET, MERGE and UPDATE statements to change the names of SAS variables between the input datasets and the PVD, and on the DATA statement to change names between the PDV and the output datasets.

- Syntax:

```
sasds(rename=(oldname=newname . . .))
```

- Example:

Suppose both datasets FALL and SPRING contain variables for NAME and a test grade, except that the grade variable is named EXAM in the FALL dataset and is named TEST in the SPRING dataset. In order to properly concatenate these datasets, we need to rename one of the exam variables, so that both variables have the same name.

```
data combo;
    set fall spring(rename=(test=exam));
run;
```

| | |
|---|---|
| Variables included in FALL: | NAME, EXAM |
| Variables read from FALL: | NAME, EXAM |
| | |
| Variables included in SPRING: | NAME, TEST |
| Variables read from SPRING: | NAME, EXAM |
| | |
| The PDV contains variables: | NAME, EXAM |
| Dataset COMBO contains variables: | NAME, EXAM |

## The WHERE Dataset Option

- The WHERE option can be used on an input dataset in a SET, MERGE, or UPDATE statement to select the observations to be brought into the PDV from the dataset.

- Syntax:

```
    sasds(where=(where_expression))
```

- If you are combining datasets, it is recommended that you use a WHERE dataset option on the appropriate incoming dataset rather than using a WHERE statement in the DATA step in which you merge.

    - Using the WHERE dataset option instead of a WHERE statement makes it totally clear to SAS how to apply the selection criteria.

    - If a WHERE statement is used, all datasets being combined must contain any variables used in the WHERE condition.

```
data short;
    merge list(where=(ht<30) in=in1 keep=name ht wt)
          demo;
    by name;
    if in1;
run;
```

**Dataset LIST**

| NAME | AGE | HT | WT |
|---|---|---|---|
| CHRISTIANSEN | 37 | 71 | 195 |
| FROG K | 3 | 12 | 1 |
| GONZO | 14 | 25 | 45 |
| HELMS R | 41 | 74 | 195 |
| HOSKING J | 31 | 70 | 160 |
| PIGGY M | . | 48 | . |

**Dataset DEMO**

| NAME | SEX |
|---|---|
| CHRISTIANSEN | M |
| FROG K | M |
| GONZO | |
| HELMS R | M |
| HOSKING J | M |
| PIGGY M | F |

**Merged Dataset SHORT**

| NAME | HT | WT | SEX |
|---|---|---|---|
| FROG K | 12 | 1 | M |
| GONZO | 25 | 45 | |

This example also shows that it is fine to use several dataset options at the same time.

## Other Dataset Option Examples

Dataset options can be used just about anywhere that you reference a dataset, and they can give your code great flexibility.

```
proc means data=bios511.prostate(where=(tumorsize^=-9999));
    var tumorsize;
run;


proc print data=bios511.cars2008(obs=10
                                 drop=fueltype enginesize);
run;


proc freq data=bios511.sales noprint;
        tables dept*clerk / out=freqout(drop=percent);
run;
```

## Example: Changing a Variable's Type Without Changing its Name Using "Swap and Drop"

- What if you wanted to change the type of a variable from numeric to character or character to numeric without changing the variable name? A useful technique is informally called "swap and drop".

- For example, say that you wanted to make MemberNum in the Weight_club dataset a character variable rather than a numeric variable. Recall that this variable has values such as 1023.

- Variables in original Weight_club dataset:

**Alphabetic List of Variables and Attributes**

| # | Variable | Type | Len |
|---|---|---|---|
| 5 | EndWeight | Num | 8 |
| 6 | Loss | Num | 8 |
| 1 | MemberNum | Num | 8 |
| 2 | Name | Char | 19 |
| 4 | StartWeight | Num | 8 |
| 3 | Team | Char | 8 |

```
data weight_club;
    set bios511.weight_club(rename=(membernum=nmembernum));

    length MemberNum $ 4;

    membernum = put(nmembernum,best4.);

    drop nmembernum;

run;
```

- Variables in modified Weight_club dataset:

| Alphabetic List of Variables and Attributes | | | |
|---|---|---|---|
| # | Variable | Type | Len |
| 4 | EndWeight | Num | 8 |
| 5 | Loss | Num | 8 |
| 6 | MemberNum | Char | 4 |
| 1 | Name | Char | 19 |
| 3 | StartWeight | Num | 8 |
| 2 | Team | Char | 8 |

## SAS Data Library Management

- A SAS data library may be permanent or temporary. Permanent libraries have a library name specified by the user, which must correspond to a LIBNAME statement. Temporary libraries have a library name of WORK.

- There are several SAS procedures that are useful in the management of SAS data libraries. PROC COPY can be used to copy all or some of the SAS datasets from one SAS data library to another. PROC DATASETS can be used to delete or modify datasets in a SAS data library.

- A single dataset can be copied from one data library to another with a DATA step of the form:

```
data newlib.sasds;
set oldlib.sasds;
run;
```

- Either of the above methods of copying datasets can be used to "back-up" datasets. "Backing-up" of a dataset means making one or more copies of the dataset and storing the copies in a safe place. One then has:

    - Backup copies stored safely away, and
    - A working copy used for calculations or other processing.

- You can also back up SAS datasets by copying and pasting to different locations on your PC or workstation.

## The COPY Procedure

- The COPY procedure can be used to copy some or all of the SAS datasets from one library to another.

- Syntax:

```
proc copy in=libraryname out=libraryname;
```

- Statements used with COPY:

```
  SELECT list of dataset names;
  EXCLUDE list of dataset names;
```

- The SELECT statement causes only the datasets listed to be copied. The EXCLUDE statement causes all datasets except those listed to be copied.

- If neither a SELECT nor an EXCLUDE statement is used, all datasets in the input library are copied.

- If the output library contains a dataset with the same name as a selected dataset, it will be replaced.

- Operating system commands (like DOS's COPY) and operations (like copy and paste) can also be used to copy SAS datasets.

# PROC COPY Example

```
49   PROC COPY IN=v8sem OUT=work;
50   RUN;

NOTE: Copying V8SEM.COLOR to WORK.COLOR (memtype=DATA).
NOTE: There were 27 observations read from the dataset V8SEM.COLOR.
NOTE: The dataset WORK.COLOR has 27 observations and 4 variables.
NOTE: Copying V8SEM.SAT_SCORES to WORK.SAT_SCORES (memtype=DATA).
NOTE: There were 108 observations read from the dataset
V8SEM.SAT_SCORES.
NOTE: The dataset WORK.SAT_SCORES has 108 observations and 4
variables.
NOTE: PROCEDURE COPY used:
      real time           0.29 seconds
      cpu time            0.09 seconds


51
52   PROC COPY IN=bios511 OUT=work;
53       SELECT sales;
54   RUN;

NOTE: Copying BIOS511.SALES to WORK.SALES (memtype=DATA).
NOTE: There were 50 observations read from the dataset BIOS511.SALES.
NOTE: The dataset WORK.SALES has 50 observations and 6 variables.
NOTE: PROCEDURE COPY used:
      real time           0.09 seconds
      cpu time            0.03 seconds
```

## The DATASETS Procedure

- The DATASETS procedure is used to manage datasets within a SAS library.

- With PROC DATASETS you can list, copy, rename, and delete SAS datasets in a SAS data library.

- PROC DATASETS can alter the descriptor portion of SAS datasets. It can be used to rename variables and to change formats, informats, or labels.

- Example Syntax:

```
proc datasets library=libref ;

* Delete files specified in mem_list;
delete mem_list ;

* rename sas dataset from old name to new ;
change old_name=new_name ;

* Copy to another sas library
copy out=libref2; select mem_list;

* Get contents of a dataset;
contents data=sas_data_set ;

* Get contents of a sas library;
contents data=libref._all_ nods;
```

```
* Modify descriptor part of a dataset;
modify sas_data_set_ ;

    * Rename a variable;
    rename old_name=new_name ;

    * Add or change variable label;
    label var='Label' ;

    * Add, change, or delete a format;
    format var format.;

* PROC DATASETS is an interactive procedure, so must
end with a QUIT statement not a RUN statement;
quit ;
```

- Syntax to delete all datasets in a library:

```
proc datasets lib=libraryname kill memtype=data;
run; quit;
```

## PROC DATASETS Example 1

```
proc datasets library=work;

    *Copy all datasets from BIOS 511 library
        into the WORK library;
    copy in=bios511 out=work;

    *Delete WORK.HW4;
    delete hw4;

    *Change the name of WORK.SCORES
        to have the name TEST;
    change scores=test;

    *Print contents of WORK.CLASS;
    contents data=class;

    *Change the descriptor portion of WORK.CLASS;
    modify class;
        label wt='wt in lbs.'; *Change variable label;
        format ht 5.0;         *Change variable format;
        rename sex=gender;     *Rename variable;

    *Print contents of WORK.CLASS after modifications;
    contents data=class;

quit; *Must end with QUIT instead of RUN;
```

## SAS Log:

```
199  proc datasets library=work;
200
201      *Copy all datasets from BIOS 511 library
202          into the WORK library;
203      copy in=bios511 out=work;
204
205      *Delete WORK.HW4;
NOTE: Copying BIOS511.BASEBALL to WORK.BASEBALL (memtype=DATA).
NOTE: There were 322 observations read from the data set BIOS511.BASEBALL.
NOTE: The data set WORK.BASEBALL has 322 observations and 22 variables.
NOTE: Copying BIOS511.BIRTHS to WORK.BIRTHS (memtype=DATA).
NOTE: There were 33 observations read from the data set BIOS511.BIRTHS.
NOTE: The data set WORK.BIRTHS has 33 observations and 2 variables.
NOTE: Copying BIOS511.BP_CUTPOINTS to WORK.BP_CUTPOINTS (memtype=DATA).
```

## Etc.. Lines omitted for brevity.

```
NOTE: There were 6 observations read from the data set BIOS511.BP_CUTPOINTS.
NOTE: Copying BIOS511.WEIGHT_CLUB to WORK.WEIGHT_CLUB (memtype=DATA).
NOTE: There were 5 observations read from the data set BIOS511.WEIGHT_CLUB.
NOTE: The data set WORK.WEIGHT_CLUB has 5 observations and 6 variables.
NOTE: Copying BIOS511.WELL1 to WORK.WELL1 (memtype=DATA).
NOTE: There were 33 observations read from the data set BIOS511.WELL1.
NOTE: The data set WORK.WELL1 has 33 observations and 10 variables.
206      delete hw4;
207
208      *Change the name of WORK.SCORES
209          to have the name TEST;
210      change scores=test;
211
212      *Print contents of WORK.CLASS;
NOTE: Deleting WORK.HW4 (memtype=DATA).
NOTE: Changing the name WORK.SCORES to WORK.TEST (memtype=DATA).
213      contents data=class;
214
215      *Change the descriptor portion of WORK.CLASS;
216      modify class;
217          label wt='wt in lbs.'; *Change variable label;
218          format ht 5.0;         *Change variable format;
219          rename sex=gender;
NOTE: Renaming variable sex to gender.
NOTE: MODIFY was successful for WORK.CLASS.DATA.
219!                               *Rename variable;
```

```
220
221       *Print contents of WORK.CLASS after modifications;
222       contents data=class;
223
224   quit;

NOTE: PROCEDURE DATASETS used (Total process time):
      real time           2.74 seconds
      cpu time            1.01 seconds


224!        *Must end with QUIT instead of RUN;
```

Contents of WORK.CLASS before modifications:

| Alphabetic List of Variables and Attributes | | | |
|---|---|---|---|
| # | Variable | Type | Len |
| 3 | AGE | Num | 8 |
| 4 | HT | Num | 8 |
| 1 | NAME | Char | 12 |
| 2 | SEX | Char | 1 |
| 5 | WT | Num | 8 |

Contents of WORK.CLASS after modifications:

| Alphabetic List of Variables and Attributes | | | | | |
|---|---|---|---|---|---|
| # | Variable | Type | Len | Format | Label |
| 3 | AGE | Num | 8 | | |
| 4 | HT | Num | 8 | F5. | |
| 1 | NAME | Char | 12 | | |
| 5 | WT | Num | 8 | | wt in lbs. |
| 2 | gender | Char | 1 | | |

## PROC DATASETS Example 2

Remove all formats and labels from all variables in a dataset.

```
title1 "Labels and Formats of BIOS511.CARS2011";
proc contents data=bios511.cars2011;
run;

proc datasets lib=work memtype=data;

    * Copy the CARS2011 dataset into the work library;
    copy in=bios511 out=work;
    select cars2011;

    * Modify labels and formats of WORK.CARS2011;
    modify cars2011;
        attrib _all_ label=' ';
        attrib _all_ format=;
quit;

title1 "Labels and Formats of Modified WORK.CARS2011";
proc contents data=work.cars2011;
run;
```

# Labels and Formats of BIOS511.CARS2011

## The CONTENTS Procedure

| Alphabetic List of Variables and Attributes | | | | | | |
|---|---|---|---|---|---|---|
| # | Variable | Type | Len | Format | Informat | Label |
| 4 | BaseMSRP | Num | 8 | | | Base Manufacturer's Suggested Retail Price |
| 5 | CityMPG | Num | 8 | | | Estimated miles per gallon in city driving |
| 7 | Country | Char | 13 | $13. | $13. | Car make country of origin |
| 9 | CurbWeight | Num | 8 | | | Curb weight in pounds |
| 6 | HwyMPG | Num | 8 | | | Estimated miles per gallon in highway driving |
| 1 | Make | Char | 13 | $13. | $13. | Make |
| 2 | Model | Char | 17 | $17. | $17. | Model |
| 12 | OwnerCost5Years | Num | 8 | | | Five-year projected cost to own vehicle, combining depreciation, interest, insurance, sales tax, fuel, and maintenance/repair |
| 10 | Reliability | Num | 8 | | | How well the model should hold up over time, based on Consumer Reports Annual Auto Survey: 1=more reliable, 5=less reliable |
| 11 | Satisfaction | Num | 8 | | | 1=80% or more would buy or lease again, 5=less than 50% would |
| 8 | Seating | Num | 8 | | | Maximum number of belted seating positions |
| 3 | Type | Char | 18 | $18. | $18. | Type of car |

## Labels and Formats of Modified WORK.CARS2011

### The CONTENTS Procedure

| Alphabetic List of Variables and Attributes | | | | |
|---|---|---|---|---|
| # | Variable | Type | Len | Informat |
| 4 | BaseMSRP | Num | 8 | |
| 5 | CityMPG | Num | 8 | |
| 7 | Country | Char | 13 | $13. |
| 9 | CurbWeight | Num | 8 | |
| 6 | HwyMPG | Num | 8 | |
| 1 | Make | Char | 13 | $13. |
| 2 | Model | Char | 17 | $17. |
| 12 | OwnerCost5Years | Num | 8 | |
| 10 | Reliability | Num | 8 | |
| 11 | Satisfaction | Num | 8 | |
| 8 | Seating | Num | 8 | |
| 3 | Type | Char | 18 | $18. |