The purpose of this presentation is to provide an overview of the SET, MERGE, and UPDATE statements. It is not my intention to provide readers of this paper with any new or innovative uses of the SET, MERGE, and UPDATE, but I hope that novice users will find this to be a quick guide to the uses and differences of each statement and its options. These statements only work with SAS datasets, so all datasets in this presentation and paper are SAS datasets.

The SET has four basic purposes: it can be used for copying datasets, interleaving datasets, subsetting datasets, and concatenating datasets. A SET statement should always be followed by the dataset name it references. Otherwise, the SET stands alone and the last dataset created is what is referenced. SET can be used with either permanent or temporary datasets, or a combination of the two. A maximum of 50 datasets can appear with one SET statement.

To copy a dataset, consider the following example:

```sas
data one; input @1 name $10. @11 height $4. @16 weight 3. @20 eyes $5. @26 hair $6. @33 age;
cards;
   Richard  6'0"  140 brown brown  27
   Theresa  5'8"  130 brown brown  31
   Beth     5'7"  130 blue blonde  39
   Steve    6'0"  150 blue brown  28;

data two; set one;
   proc print;
```

In this case, the data for dataset one is created by the input statement in data one. In data two what we would like to do is make a copy of this dataset, which is done by using the SET statement. The result is a copy of data one.

If a similar dataset existed for data two as for data one, then these two datasets could be concatenated in data three:

```sas
data three; set one two;
   proc print;
```

Concatenation would cause data three to contain all inputs from all the datasets listed. If there are some variables in one dataset that are not in the other, then missing values are used for the variables that are defined in one of the datasets.

The SET statement has various options that can be used with it. These options are used for interleaving and subsetting datasets.

Interleaving is the process of combining datasets in a sorted order. This is accomplished using the BY statement along with the SET statement. Two conditions must exist for this to take place: 1) The datasets must be previously sorted. If not, use PROC SORT to accomplish this. 2) The variables listed with the BY statement must be present in the datasets that are being interleaved. Note that the PROC SORT must be used if the dataset is not pre-sorted. Also note that the datasets must be sorted in the same order on the BY variables. If there is one dataset sorted in ascending order and one in descending order, they cannot be interleaved without re-sorting one of them into the other's order.

In this example, we want to create a dataset with all the people in data one and data two in alphabetical order. Since they are not already sorted in alphabetical order, we must use a PROC SORT in each case to do so. Then in data three, we use the SET with the BY to create our sorted list:

```sas
data one; input @1 name $10. @11 age 3.; cards;
   Bob      40
   Carole   39
   Frances  50
   David    48
   proc sort; by name;
```
data two; input @1 name $10.
@11 age 3.;
cards;
Paul 47
Clarice 48
Susan 29
Kathy 37
Pat 42
;
proc sort; by name;
data three; set one two; by name;
proc print;
Data three would contain all nine observations in alphabetical order. It is also possible to sort both these datasets by age and have a dataset with all nine observations in order by age.

To use the SET to subset variables in a dataset, use the KEEP and/or DROP statements as a dataset option. In this example, we wish to keep the variables name, height, and weight. Using the KEEP as an option:

data two;
set one(keep=name height weight);
As might be expected, the DROP accomplishes the opposite effect. Because the KEEP was used as an option with the SET, only the variables name, height, and weight are available for processing. That is a difference between using the KEEP and/or DROP as dataset option rather than as a dataset statement. If we had used the KEEP in data two as follows:

data two;
set one(keep=name height weight);
then ALL the variables would have been processed, but only name, height, and weight would have been sent to the output data set. This is similar to what would happen had we used the KEEP as an option in the data statement as follows:

data two (keep=name height weight)
three (keep=age);
set one;
Here all the variables in data one are present in the data step for processing. However, for data two, only name, height, and weight are available for output. For data three, only age is available for output.

Using the KEEP and DROP can control the number of variables; using the OBS option can control the number of observations to be processed. For example, if data one had five thousand observations to be processed and it would really help if only the first 100 observations were processed, then using the OBS as an option:

data two; set one(obs=100);
This would allow only the first 100 observations to be processed. A variation of this is the FIRSTOBS= option, which tells SAS at which observation to begin processing. Using the above example, with 5000 observations in data one, let us now assume we wanted to begin processing at observation 2500:

data two; set one(fristobs=2500);
This would start at observation 2500 and continue to the end of the file.

In discussing the OBS= and FIRSTOBS= options, it should be noted that it is possible to also pick out certain observations for processing. This can be done using the SET inside a DO loop with the POINT= option. I only wish to mention it here; it is described in the SAS manual. Another option which I would like to mention is the END= which can be used as an end of file marker. This is also described in the manual.

One more option available to SET is the IN=. This option creates a special variable during compilation but it is not added to the data set being created. (The variable is available during the processing but is not written to the output). One reason for using it would be to determine how many observations are being contributed from which dataset. This will be shown with the MERGE.
Now on to the MERGE. As with the SET, it is possible to merge up to 50 datasets. There are two types of merging possible: 1) observation by observation (also called one to one merging) and 2) match-merge, which uses the BY statement. MERGE is primarily used when there exists two or more datasets which need to be combined into one dataset for further processing. Consider the following example, where there are two datasets. Each has three observations but the variables are different.

```plaintext
data one; input @1 name $10. @11 height $4. @16 weight 3.; cards;
   Jody 5'8" 130
   Paul 6'3" 160
   Errica 5'6" 130
; 
data two;
   input @1 eyes $5. @7 hair $6. @14 age 2.; cards;
   blue brown 29
   brown brown 30
   blue brown 25
; 
data three; merge one two;
proc print;
```

Data three would consist of three observations, with the variables name, height, weight, eyes, hair, and age all present in the dataset. This is also an example of an observation by observation merge; the first observation of data two was merged with the first observation of data one; the second with the second, and so on. This is fine if it is already known that the information for each observation in data two corresponds to each observation in data one in the same order. There is no control over this type of merging. However, there is an alternative available, and that is match-merge. Suppose it was necessary to make a distinction as to how the merge should take place. This can be done using the BY statement with the MERGE. The BY allows for the merging based on the listing of the BY variable(s). There are some constraints that must be kept in mind when merging with the BY. First, the datasets must be sorted by the same variable that appears in the BY (just like the SET). Also, the BY can only be used if each dataset to be merged has that BY variable as part of the dataset. More than one variable can be listed with the BY, but each variable must be a part of each dataset to be merged.

Using the BY statement can also create some interesting results. For example, assume that the names in data one are not in the same order as in data two:

```plaintext
data one; input @1 name $10. @11 height $4. @16 weight 3.; cards;
   Mike 6'1" 195
   Gerry 5'2" 110
   Luan 6'6" 180
; 
data two;
   input @1 eyes $5. @7 hair $6. @14 age 2. @17 name $10.; cards;
   hazel brown 39 Gerry
   brown brown 34 Luan
   brown brown 29 Mike
; 
data three; merge one two;
proc print;
```

This will produce a dataset with three observations but the merge will create a dataset with "mixed" results: Gerry will be listed with height 6'1", weight 195, hazel eyes and brown hair, and age 39. Luan and Mike would follow in order.

The IN= option can also be used with the MERGE. Suppose there existed a big dataset (data one) which had 1000 observations and five variables (name, age, gender, height, weight). Now assume there is a
second dataset (data two) which consisted of only a few observations and only two of the variables (name and age). These two datasets have to be merged and it is important to know which observations were contributed from data two. Assuming these two datasets are already sorted by name and age:

```
data three;
merge one two (in=a); by name age;
if a=1;
```

The variable created by IN= (named 'a') will take on a value of 1 if the observation is contributed from data two, and will be 0 otherwise. So in the above example the statements following the IF statement are executed only if the observation came from data two.

The UPDATE statement is similar to the MERGE, except that the UPDATE is limited to two datasets, in which one is treated as a "master" dataset and the other a "transaction" dataset. A BY statement must always be used with an UPDATE statement, so both datasets must be sorted by the variable(s) appearing with the BY. The master dataset cannot have duplicate values of this BY variable. It is also possible to UPDATE to missing values.

In comparing the terms "master dataset" and "transaction dataset", it might be helpful to keep in mind that the UPDATE is a right to left operation. For example, a statement such as:

```
data three; update one two;
```

would mean that dataset one is the master dataset and dataset two is the transaction dataset. The transaction dataset contains the updated values; it could be considered to be a temporary dataset, though it does not necessarily have to be. The master dataset is the dataset to be updated.

For example, suppose data one contains four observations with variables name, height, and weight. Data two contains four observations with variables eyes, hair, age, and name. These four observations in data two have the same values for name that data one has; only the values for height are different.

```
data one; input @1 name $10. @11 height $5. @16 weight 3.; cards;
Ann 5'3" 115 Cindy 5'2" 110 Scott 5'9" 140 Lee 6'0" 150
;
proc sort; by name;
data two;
input @1 eyes $5. @7 hair $6. @14 age 2. @17 name $10. @28 height $5.;
cards;
blue red 32 Ann 5'4" blue blonde 28 Cindy 5'4"
blue blonde 28 Scott 5'10"
blue brown 28 Lee 5'11"
;
proc sort; by name;
data three; update one two; by name;
proc print;
```

This UPDATE statement would replace the values of height in data one with the ones in data two. Data two in this case is the transaction dataset, and data one is the master.

To update to missing values, the MISSING statement is used. Using the above example as a model, suppose missing values for height were appropriate for data one. The dataset already has values for height, so in data two a MISSING statement would be used:

```
data two; missing _;
```

Here, missing values are to be indicated by the underscore. If there are missing values for height, the underscore should appear in the input statement in
data two to represent these missing values. Using the UPDATE as in the above example would then update the values for height in data one with the underscore to represent missing values.

It is my hope that this helps serve as an introduction to the SET, MERGE, and UPDATE. For further reference, consult the SAS manuals.

References:

*SAS is the registered trademark of SAS Institute, Inc., Cary, NC


The author may be contacted at:

Richard Alonso
Department of Family Practice
Medical College of Virginia
MCV Station, Box 251
Richmond, VA 23298-0251