8

Actions on SAS Datasets

Usually, in a SAS program, you work with a SAS dataset one observation at a time. Sometimes, though, it makes sense to think of working with the SAS dataset as a whole. This is especially the case in database-style applications, which are likely to keep SAS datasets for an extended period of time and use them both interactively and in SAS programs.

Creating

A SAS dataset can be created in a data step or a proc step. By default, a data step creates a SAS dataset that contains all the variables of the data step, with just a few kinds of variables excluded. The data step writes one observation to the output SAS dataset each time it reads input data; a data step that does not read input data writes just one observation. Proc steps also create SAS datasets; the proc determines what observations and variables it creates.

Usually, a SAS dataset is created together with the data that it contains. However, it is also possible to create an empty SAS dataset. The idea is that you will add data to the SAS dataset in a later process. There are several ways to create an empty SAS dataset.

◎ To create a new, empty SAS data file in a data step, write a LENGTH statement to define the variables. If necessary, use other statements, such as LABEL and FORMAT statements, to define other attributes of the variables. This example creates a SAS data file CORP.MEMBER with two variables, NAME and DOB, and no observations.

```
DATA CORP.MEMBER;
  LENGTH NAME $ 28 DOB 4;
  FORMAT DOB DATE9.;
  INFORMAT DOB DATE9.;
  STOP;
RUN;
```

```
NOTE: Variable NAME is uninitialized.
NOTE: Variable DOB is uninitialized.
```

```
NOTE: The data set CORP.MEMBER has 0 observations and 2 variables.
```

◎ An empty SAS data file can also be created in SQL using a CREATE TABLE statement. In the CREATE TABLE statement, the name of the SAS data file (or table) is followed by a list, in parentheses, of variable definitions.

A variable definition includes the variable name and a data type, which can be one of the following:

```
    CHARACTER(width)
    NUMERIC
```

The data types can be abbreviated as CHAR and NUM. SQL syntax allows several other data type names, but SAS implements all the SQL data types as either the numeric or character data type. The width argument for the character data type determines the length of the character variable.

The column attributes INFORMAT=, FORMAT=, and LABEL= can follow the data type to set other attributes of the variable. Unfortunately, the LENGTH= column attribute is not recognized in this kind of CREATE TABLE statement, and there is no way to set the length of a numeric variable.

This example creates the same SAS data file that was created in the earlier data step example, except that the numeric variable DOB has the default length of 8.

```
PROC SQL;
CREATE TABLE CORP.MEMBER
  (NAME CHARACTER(28),
  DOB NUMERIC INFORMAT=DATE9. FORMAT=DATE9.);
```

```
NOTE: Table CORP.MEMBER created, with 0 rows and 2 columns.
```

◎ To create a new, empty SAS data file with the same variables and attributes as an existing SAS dataset, use a data step with a SET statement and a STOP statement. This data step creates a new SAS data file CORP.NX based on CORP.MEMBER.

```
DATA CORP.NX;
  SET CORP.MEMBER;
  STOP;
RUN;
```

Alternatively, use the `OBS=0` dataset option in the SET statement. With the `OBS=0` dataset option, the STOP statement is not necessary.

◎ To do the same thing in SQL, use a CREATE TABLE statement with a LIKE clause, as in this example.

```
PROC SQL;
```

```
CREATE TABLE CORP.NX LIKE CORP.MEMBER;
```

◎ To create a SAS data file that has the variables of several existing SAS datasets, list those SAS datasets in the SET statement of a data step, followed by a STOP statement. The new SAS data file contains every variable that is in any of the SAS datasets listed.

◎ Use the KEEP= or DROP= dataset option to create a new SAS dataset that contains only some of the variables of an existing SAS dataset.

Describing

◎ The CONTENTS proc creates a report that shows the details of the form of a SAS dataset. Identify the SAS dataset in the `PROC CONTENTS` statement:

```
PROC CONTENTS DATA=SAS dataset;
RUN;
```

The information that is available about the SAS dataset depends on the environment and engine. The general information about the SAS dataset is followed by a table of all the attributes of the variables. It lists variables in alphabetical order.

Use the OUT= option to create a SAS dataset of most of this information. The output SAS dataset contains one observation for each variable.

◎ In SQL, the DESCRIBE TABLE statement provides information about attributes of variables in a very different format. It writes a log message that includes a CREATE TABLE statement that could be used to create the SAS data file.

```
PROC SQL;
DESCRIBE TABLE CORP.MEMBER;
```

```
NOTE: SQL table CORP.MEMBER was created like:

create table CORP.MEMBER( bufsize=4096 )
  (
   NAME char(28),
   DOB num format=DATE9. informat=DATE9.
  );
```

From a data step view or SQL view, you can obtain the program that is contained in the view. See chapter 60, "Data Step Views," and chapter 42, "SQL," for details.

Changing

◎ There are many ways to change the data contained in a SAS data file, including these:

- Use a data step with a SET statement to replace the SAS data file.
- Use a data step with a MODIFY statement to modify the SAS data file in place.
- Edit the SAS data file interactively in the Viewtable application, the FSEDIT or FSVIEW proc, or an AF entry that uses an extended table to allow editing of the SAS data file.
- Add a set of observations from another SAS dataset with the APPEND proc, as described later in this chapter.
- Add, delete, or change observations with SQL statements.
- Use the SORT proc to change the order of observations.

◎ Use the CHANGE statement of the DATASETS proc to change the name of a SAS dataset or other SAS file. The DATASETS proc requires you to provide the libref and member name separately. Indicate the libref in the LIBRARY= option in the `PROC DATASETS` statement. Indicate member names in the specific action statements of the proc. In the CHANGE statement, write the old name, an equals sign, and the new name for the member. This example changes the name of WORK.OFFICE to WORK.PLACE:

```
PROC DATASETS LIBRARY=WORK NOLIST;
  CHANGE OFFICE=PLACE;
RUN;
```

```
NOTE: Changing the name WORK.OFFICE to WORK.PLACE (memtype=DATA).
```

☛ Ordinarily, the `PROC DATASETS` statement writes a member list in the log. Use the NOLIST option whenever you will not be using the member list.

◎ To swap the names of two SAS files, use the EXCHANGE statement in the DATASETS proc. This example changes the name of WORK.SPACE to WORK.PLACE and changes the name of WORK.PLACE to WORK.SPACE:

```
PROC DATASETS LIBRARY=WORK NOLIST;
  EXCHANGE SPACE=PLACE;
RUN;
```

```
NOTE: Exchanging the names WORK.SPACE and WORK.PLACE (memtype=DATA).
```

◎ The AGE statement of the DATASETS proc renames a sequence of SAS files and deletes a SAS file in a single action. Each member listed in the AGE statement has its name changed to the next name in the list. The last member in the list is deleted. This example deletes BE.BEFORE, changes the name of

BE.NOW to BE.BEFORE, and changes the name of BE.NEXT to BE.NOW.

```
PROC DATASETS LIBRARY=BE NOLIST;
  AGE NEXT NOW BEFORE;
RUN;
```

```
NOTE: Deleting BE.BEFORE (memtype=DATA).
NOTE: Aging the name BE.NOW to BE.BEFORE (memtype=DATA).
NOTE: Aging the name BE.NEXT to BE.NOW (memtype=DATA).
```

◎ To change the dataset label or dataset type of a SAS dataset, use the MODIFY statement of the DATASETS proc. In the MODIFY statement, write the member name, followed by dataset options: the LABEL= option or the TYPE= option, respectively. This example changes the dataset label of CORP.NEWS to "The latest news stories."

```
PROC DATASETS LIBRARY=CORP NOLIST;
  MODIFY NEWS (LABEL='The latest news stories');
RUN;
```

◎ To change the length of a variable in a SAS dataset, replace the SAS dataset with a new SAS dataset in which you set the length of the variable. Use a data step with a LENGTH statement followed by a SET statement. This example changes the length of the variable NAME in the SAS dataset CORP.MEMBER to 32.

```
DATA CORP.MEMBER;
  LENGTH NAME $ 32;
  SET CORP.MEMBER;
RUN;
```

◆ There is no direct way to change the type of a variable in a SAS dataset. Instead, it is necessary to replace the SAS dataset while creating a new variable of the new type and removing the old variable. See chapter 50, "Type Conversion," for details.

◎ To change the names and other attributes of variables in a SAS data file, use secondary statements after the MODIFY statement in the DATASETS proc. Use the INFORMAT, FORMAT, LABEL, and RENAME statements. The code model below shows the use of these statements.

```
PROC DATASETS LIBRARY=libref NOLIST;
  MODIFY SAS dataset;
    INFORMAT variable . . . informat . . . ;
    FORMAT variable . . . format . . . ;
    LABEL variable='label' . . . ;
    RENAME old name=new name . . . ;
RUN;
```

Copying

There are several ways to copy a SAS dataset in a SAS program, each with its own advantages and disadvantages.

- The COPY proc is the most direct way to copy a SAS file. It is the only way to copy indexes along with a SAS dataset. It cannot change the member name of a SAS dataset that it copies or copy SAS datasets within the same library.
- Using a data step to copy a SAS dataset has several advantages. It is the most efficient way to make multiple copies at the same time, and the data step makes it possible to make changes in the data. It can create a standard SAS data file, regardless of the form of the SAS dataset being copied.
- The CREATE TABLE statement of SQL offers another way to copy data while making changes, such as computing additional variables.
- The SORT proc is the way to copy a SAS dataset if you want the copy to have a different sort order.
- The APPEND proc provides an efficient way to add observations from one SAS dataset to another.

To copy all the SAS files of one library to another, use the COPY proc. Use the IN= and OUT= options in the `PROC COPY` statement to identify the libraries. No other statements are required.

```
PROC COPY IN=libref OUT=libref;
RUN;
```

To restrict the actions of the COPY proc to specific member types, write the MTYPE= option in the `PROC COPY` statement. To copy only SAS datasets, indicate the member types DATA and VIEW, as shown here:

```
PROC COPY IN=libref OUT=libref MTYPE=(DATA VIEW);
RUN;
```

To copy only selected members of a library, add the SELECT statement to the COPY proc: This example copies WORK.STATUS to MAIN.STATUS.

```
PROC COPY IN=WORK OUT=MAIN;
  SELECT STATUS;
RUN;
```

Alternative, use the EXCLUDE statement to copy all members except specific ones you mention.

Use the `INDEX=YES` option in the `PROC COPY` statement to copy indexes along with SAS data files. Also use the `CONSTRAINT=YES` option to copy the integrity constraints of SAS data files. Use the INDEX=NO option to copy SAS data files without copying their indexes or integrity constraints.

◎ To copy a SAS file and change its name, first make the copy using the COPY proc. Then use the CHANGE statement of the DATASETS proc to change the name. This example copies WORK.STATUS to MAIN.STATE.

```
PROC COPY IN=WORK OUT=MAIN;
  SELECT STATUS;
RUN;
PROC DATASETS LIBRARY=MAIN NOLIST;
  CHANGE STATUS=STATE;
RUN;
```

```
NOTE: Copying WORK.STATUS to MAIN.STATUS (memtype=DATA).
NOTE: There were 141 observations read from the data set WORK.STATUS.
NOTE: The data set TEST.STATUS has 141 observations and 2 variables.

NOTE: Changing the name MAIN.STATUS to MAIN.STATE (memtype=DATA).
```

◆ The system options OBS= and FIRSTOBS= affect the COPY proc. To copy all the observations of SAS data files, use the default values FIRSTOBS=1 and OBS=MAX.

◎ To copy a SAS dataset in SQL, use the form of the CREATE TABLE statement shown in this code model.

```
PROC SQL;
CREATE TABLE SAS dataset AS SELECT * FROM SAS dataset;
```

◎ To copy a SAS dataset in a data step, use a SET statement, as shown in this code model.

```
DATA SAS dataset;
  SET SAS dataset;
RUN;
```

To make multiple copies, list several output SAS datasets in the DATA statement. To make changes in the data, add additional statements to the data step.

◎ To copy observations from one SAS dataset to the end of another SAS dataset, use the APPEND proc. This example copies the observations of ADD to the end of HERE.

```
PROC APPEND DATA=ADD OUT=HERE;
RUN;
```

◆ The APPEND proc works only if the two SAS datasets are reasonably similar to each other. Use the FORCE option in the PROC APPEND statement if the input SAS dataset has variables that are longer than those in the output SAS dataset or are not in the output SAS dataset. If the variable names do not match, use the RENAME= dataset option on the input SAS dataset to make them match.

◎ To make a copy of a SAS dataset, use the APPEND proc with an output SAS dataset that does not already exist. If necessary, you can delete the output SAS dataset beforehand with the DATASETS proc, as shown in this example.

```
PROC DATASETS LIBRARY=WORK NOLIST NOWARN;
  DELETE TO (MTYPE=DATA);
RUN;
PROC APPEND DATA=WORK.FROM OUT=WORK.TO;
RUN;
```

◎ To create a sorted copy of a SAS dataset, use the OUT= option in the SORT proc. This example creates WORK.USSORT as a sorted copy of WORK.US.

```
PROC SORT DATA=WORK.US OUT=WORK.USSORT;
   BY KEY1 KEY2;
RUN;
```

For more information about sorting, see chapter 43, “Sorting.”

Deleting

Use the DATASETS proc to delete SAS datasets and other SAS files.

◆ Before deleting a file, make sure that you no longer need the file and that you have correctly identified the file to delete.

◎ To delete all the SAS files in a library, use the KILL option in the PROC DATASETS statement. This example deletes all SAS files in the WORK library.

```
PROC DATASETS LIBRARY=WORK NOLIST KILL;
RUN;
```

◎ To delete specific SAS files from a library, list them in a DELETE statement. This example deletes the SAS data files WORK.TEMP and WORK.TEST.

```
PROC DATASETS LIBRARY=WORK MTYPE=DATA NOLIST NOWARN;
  DELETE TEMP TEST;
RUN;
```

☛ Ordinarily, the DELETE statement writes a log note if the member to delete does not exist. Use the NOWARN option in the `PROC DATASETS` statement to suppress this log note.

☛ The MTYPE= option can also be written in the DELETE statement, for example:

```
PROC DATASETS LIBRARY=WORK NOLIST NOWARN;
  DELETE TEMP TEST / MTYPE=DATA;
RUN;
```

Writing the option this way makes it possible to delete different member types in separate DELETE statements in the same step.

◎ To delete all but a few SAS files from a library, write a SAVE statement listing the members to keep. This example deletes everything from the WORK library except WORK.RESULT and WORK.FORMATS.

```
PROC DATASETS LIBRARY=WORK NOLIST NOWARN;
  SAVE RESULT FORMATS;
RUN;
```

◎ In SQL, use the DROP TABLE and DROP VIEW statements to delete SAS data files and SQL views. This example deletes the SAS data file WORK.OBSOLETE.

```
PROC SQL;
DROP TABLE WORK.OBSOLETE;
```

◎ To delete all the observations from a SAS data file while keeping the variables, replace the SAS data file in a data step with a SET statement and a STOP statement, as shown in this code model.

```
DATA SAS dataset;
  SET SAS dataset;
  STOP;
RUN;
```

Alternatively, in SQL, use the DELETE FROM statement to remove observations from a SAS data file, as shown in this code model.

```
PROC SQL;
DELETE FROM SAS dataset;
```

◆ A SAS dataset is automatically deleted when you replace it with a new SAS dataset of the same name. This deletion occurs as soon as the new SAS dataset is successfully written. To avoid accidentally deleting SAS datasets by replacing them, give each SAS dataset its own name.

◆ A SAS dataset can only replace another SAS dataset of the same kind. An error condition occurs if a new SAS dataset has the same name as a SAS dataset of a different kind. It is an error if you attempt to:

- replace a SAS data file with a view
- replace a view with a SAS data file
- replace a view with a different kind of view

To avoid the error, delete or rename the old SAS dataset before you create the new SAS dataset. Alternatively, use a new name for the new SAS dataset.