Paper #303

The Power of PROC DATASETS

Lisa M. Davis, Bank of America, Jacksonville, Florida

ABSTRACT

The DATASETS procedure can be used to do many functions that are normally done within a SAS data step more efficiently.

For example:

- Labeling and renaming variables
- Concatenating and indexing datasets

This paper will demonstrate the power of the PROC DATASETS procedure and the enhancements made in V8. This paper is intended for beginning to intermediate SAS users. PROC DATASETS is a powerful procedure that everyone needs to know.

INTRODUCTION

PROC DATASETS is a SAS utility used to manage more than one SAS file at a time. This procedure allows you to append, copy, delete, label, rename, index, and collect information about the dataset that has been modified, all in one step. The DATASETS procedure executes in order. The first statement executes first, then the second, and so on. This allows you to concatenate two data sets, then rename the variables, change the labels and create an index all in the same procedure. The ability to do so improves processing time, programming length, and data steps needed. The following is a list of statements used in the DATASETS procedure:

```
PROC DATASETS;
   AGE;
   AUDIT;
   CHANGE;
   CONTENTS;
   COPY;
      EXCLUDE;
      SELECT;
   DELETE;
   EXCHANGE;
   MODIFY;
       FORMAT;
       IC CREATE;
       IC DELETE;
       IC REACTIVB;
       INDEX CREATE;
       INDEX DELETE;
       INFORMAT;
       LABEL;
       RENAME;
   REPAIR;
   SAVE;
RUN;
QUIT;
```

This tutorial will show you why and how to use PROC DATASETS. We will not cover all statements and options with the DATASETS procedure, but you will walk away knowing how powerful this procedure is.

THE DATASETS STATEMENT

The DATASETS procedure is an interactive procedure that executes immediately and does not stop processing until QUIT or RUN CANCEL command is issued. The DATASETS statement executes a list of all of the members in a SAS library in the log of your program. The list can contain members with a member type of: data, view, access, catalog, any and program.

The general form of the DATASETS statement is:

```
PROC DATASETS LIBRARY=LIBREF
              MEMTYPE=MEM-LIST <OPTIONS>;
```

The LIBRARY= and MEMTYPE= are options, but I always stress to specify both a library reference and a member type so you know what library and member type you are working with. This is crucial when you have several member types in the same library that could be named the same.

If you haven't worked with a library over a period of time, this is a good way to find out what is in that library. Here is the code to do so:

```
libname mylib 'c:\temp';

proc datasets library=mylib memtype=data;
run; quit;
```

Refer to OUTPUT 1.1 in the appendix

The output generated in you r log, gives you the libref pointing to the library, engine the data set was created with, physical name, file name, name of the datasets that are located in that library, the memtype (in this case memtype = data), file size, and the day and time the data set was last modified. As you can see there are three data sets in the library: MORTGAGES, PLUS, SECUREDLOANS. These are the data sets we will be working with throughout this tutorial.

One other option I want to cover with the DATASETS statement is KILL. KILL deletes all data sets within a library automatically. The general form is:

```
proc datasets library=mylib memtype=data
kill;
```

Caution: KILL executes immediately before the DATASETS procedure completes processing.

THE CONTENTS STATEMENT

The CONTENTS statement acts the same as the CONTENTS procedure. This statement gives you information about the variables within a SAS library. How do you know which one to use? The CONTENTS statement is very useful when you are combining other DATASETS statements to manipulate a SAS library. Otherwise PROC CONTENTS is recommended to use.

The general form of the CONTENTS statement is:

```
CONTENTS DATA=LIBREF.MEMBER <OPTIONS>
```

DATA = is an option that is very useful to always use. This specifies which library and member you want contents on. The libref is not always needed in the case that the libref is specified in the DATASETS statement. _ALL_ is also an option that may be used when you want contents on all of the data sets that reside in that library.

To find out what variables reside in the data set SECUREDLOANS submit the following code:

```
libname mylib 'c:\temp';
```

```
proc datasets library=mylib memtype=data;
contents data=securedloans;
run; quit;
```

Refer to OUTPUT 1.2 in the appendix.

The CONTENTS statement is added after the DATASETS statement after all the changes have been done to that dataset. The CONTENTS statement gives you number of observations, engine created with, last date modified, and engine host information. It also gives you a list of variables within the data set, type of variable, length, format, position, informat, and labels. Examples of how the CONTENTS statement is used with other statements will be covered later in the tutorial.

THE APPEND STATEMENT

The APPEND statement is used to concatenate two SAS data sets together. SAS takes one data set and appends the second data set to the bottom of the first. Being able to do this in one step saves processing time and space allocation. Only SAS data sets can be concatenated together.

The general form of an APPEND statement is:

```
APPEND BASE=SAS DATASET DATA=SAS DATASET <FORCE>
```

The BASE= is the SAS dataset that you want the observations added to. The DATA= is the SAS dataset that you want added. FORCE is an option that is used when you want to force a concatenation when the two data sets have different variable names. The following code is an example of concatenating two data sets and viewing the contents after the two are appended.

```
libname mylib 'c:\temp';

proc datasets library=mylib memtype=data;
append base=securedloans data=mortgages;
contents data=securedloans;
run; quit;
```

Refer to OUTPUT 1.3 in the appendix

This example appends the MORTGAGES data set to the bottom of the SEURED LOANS data set. The contents is ran on the SECUREDLOANS data set showing that the number of observations in MORTGAGES have been added to SECUREDLOANS. The MORTGAGES data set still exists in the library. This was done in one procedure versus a data step, creating a third data set, and the CONTENTS procedure. This may not be noticeably faster with small data sets, but as the data sets exist of million of rows, the processing time surpasses by hours and space allocation is greatly reduced. Of course this means major savings, these days.

THE DELETE STATEMENT

The DELETE statement deletes specified data sets within a library. Multiple data sets or all of the data sets can be deleted at the same time in a library. The deletion occurs immediately, and does not wait for the DATASETS procedure to complete. For example: If you delete a member in the first line of the DATASETS procedure you cannot run a CONTENTS statement referring to that member. You will receive a 'this file does not exist' error. The advantage of using the DELETE statement is during a long process you can delete data sets that are no longer being used. This frees up space and reallocates this space to be used in your same process.

The general form of the DELETE statement is:

```
DELETE MEMBER-LIST
```

For an example:

```
libname mylib 'c:\temp';

proc datasets library=mylib memtype=data;
append base=securedloans data=mortgages;
delete mortgages;
contents data=securedloans;
run; quit;
```

The DELETE statement here deletes the data set MORTGAGES because this data set has been appended to the SECUREDLOANS data set and is no longer needed. The space MORTGAGES was occupying is now free to be used to store another data set.

THE MODIFY STATEMENT

The MODIFY statement; in my opinion is the most powerful and useful statement in the DATASETS procedure. Within the MODIFY statement you can label, rename, create and delete indexes, create integrity constraints, delete integrity constraints, reactivate integrity constraints, format, and informat variables within a library. These actions can only occur after a MODIFY statement. We will discuss several of these actions that are most used in this procedure. The structure of the MODIFY statement is:

```
MODIFY DATA SET <OPTIONS>;
    FORMAT;
    IC CREATE;
    IC DELETE;
    IC REACTIVB;
    INDEX CREATE;
    INDEX DELETE;
    INFORMAT;
    LABEL;
    RENAME;
```

The MODIFY statement alone points to the data set that you want to change. The LABEL option allows creating or deleting a label on the data set specified. For example:

```
libname mylib 'c:\temp';

proc datasets library=mylib memtype=data;
modify securedloans(label='Secured Loans');
contents data=securedloans;
run; quit;
```

Refer to OUTPUT 1.4 in the appendix

As you can see in the contents output you can see that the SECREDLOANS data set has been labeled 'SECURED LOANS'. If you wanted to delete this label you would use the label option and leave a blank ' '. Also with multiple MODIFY statements you can modify more than one dataset at a time. The reason you may want to do this is to prepare two data sets to be merged without all of the preparation data steps.

```
libname mylib 'c:\temp';

proc datasets library=mylib memtype=data;
modify securedloans(label='Secured Loans');
modify mortgages (label='Mortgage Loans');
contents data=securedloans;
contents data=mortgages; run;quit;
```

THE INDEX STATEMENT

The INDEX statement allows you to create or delete an index on a SAS data set. Creating an index on a SAS data set allows for more efficient processing of observations. If you wanted to do BY processing on two data sets with an index created, sorting is not needed. By eliminating sorting, again processing time and space is saved. The advantage of

creating an index instead of sorting is within the DATASETS procedure you can combine several statements to manipulate the data set in one procedure instead of multiple. If you wanted to merge two data sets you could do so without sorting. Once an index is created, you can rename, copy, label, etc… and the index will be transferred.
The general form of an INDEX statement is:

```
INDEX CREATE  VARIABLE-LIST  or
INDEX DELETE INDEX-LIST
```

If you want to merge two very large data sets by two variables, you can first create an index on these two variables on both data sets at the same time. For example:

```
libname mylib 'c:\temp';

proc datasets library=mylib memtype=data;
     modify securedloans(label='Secured Loans');
            index create accno cct_no;
     modify plus (label='Plus Customers');
            index create account companycost;
     contents data=securedloans;
run;quit;
```

Refer to OUTPUT 1.5 in the appendix
By creating an index on the SECUREDLOANS data set and PLUS data set, you have avoided two SORT procedures, saving time and space again. As you look in the indexes portion of the contents output, you can see that the indexes create two extensions of the data sets. These extensions are treated as the data set; so all indexes transfer through all modifications.

THE LABEL STATEMENT
The LABEL statement allows you to label variable within a data set. Multiple variables can be labeled within one MODIFY statement. Multiple variables from different data set can also be labeled within several MODIFY statements.
The general form of a LABEL statement is:

```
LABEL VARIABLE='LABEL';
```

For example:

```
libname mylib 'c:\temp';

proc datasets library=mylib memtype=data;
     modify securedloans(label='Secured Loans');
            label accno='Account Number'
                  cct_no='Cost Center';
     modify plus (label='Plus Customers');
            label acct='Account Number'
                  cost='Cost Center';
     contents data=securedloans;

run;quit;
```

Refer to OUPUT 1.6 in the appendix
As you can see in the variable list portion of the contents output you can see that ACCNO and CCT_NO have been labeled within the SECUREDLOANS data set. Along with ACCT and COST within the PLUS data set. The advantage of using the LABEL statement with the DATASETS procedure is that the labels are stored permanently in the data set. If you execute a LABEL statement within other procedures such as: PROC FREQ, PROC PRINT, etc, the label is only active for that procedure. With the labels being stored permanently, you do not have to worry about label consistency throughout the reports you produce.

THE RENAME STATEMENT
The RENAME statement allows you to rename variables within a data set. Multiple variables can be renamed at one time.
The general form of a RENAME statement is:

```
RENAME VARIALBLE=NEW VARIABLE
```

Once you rename a variable, the new name overwrites the old name. As you can tell we are building step-by-step of the DATASETS procedure to allow you to get the most benefit and power of this procedure. So if you wanted to merge two data sets by two variables that were named different in both data sets, you would rename the variables so they matched for merging. First you would want to create an index, so you could avoid sorting, second rename the variables so they match each other, avoiding one possibly two data steps. For example:

```
libname mylib 'c:\temp';

proc datasets library=mylib memtype=data;
modify securedloans(label='Secured Loans');
     index create accno cct_no;
     rename accno=accountnumber
            cct_no=costcenter;
     label accountnumber='Account Number'
           costcenter='Cost Center';

modify plus (label='Plus Customers');
     index create account companycost;
     rename account=accountnumber
            companycost=costcenter;
     label accountnumber='Account Number'
           costcenter='Cost Center';
contents data=securedloans;
run;quit;
```

Refer to OUPUT 1.7 in the appendix
In this example we are preparing our data sets to be able to merge. We have labeled our data sets in the MODIFY statements. We created indexes on the two data sets. Now we renamed ACCNO to ACCOUNTNUMBER and CCT_NO to COSTCENTER in the SECUREDLOANS data set. Then we did the same in the PLUS data set. We renamed ACCOUNT to ACCOUNTNUMBER and COMPANYCOST to COSTCENTER. After renaming the variables, the index is transferred to the new names of the variables. This is why it was crucial to create the index before renaming or modifying the data set any further. In the contents output of SECUREDLOANS you can see that the variables have been renamed and the indexes have been transferred to the new names.

THE FORMAT STATEMENT
The FORMAT statement is used to modify, change, or add a format onto a variable. You can also use the INFORMAT statement to change how the variable is read in. The FORMAT statement changes how the variable is put out.

The general form of the FORMAT and INFORMAT statements are:

```
FORMAT VARIABLE-LIST format    or
INFORMAT VARIABLE-LIST format
```

The following is an example of how the FORMAT statement is used:

```
libname mylib 'c:\temp';

proc datasets library=mylib memtype=data;
     modify securedloans(label='Secured
Loans');
```

```
            index create accno cct_no;
            rename accno=accountnumber
                   cct_no=costcenter;
            label accountnumber='Account Number'
                     costcenter='Cost Center';
            format accountnumber $12.
                   costcenter 8.;
     modify plus (label='Plus Customers');
            index create account companycost;
            rename account=accountnumber
               companycost=costcenter;
            label accountnumber='Account Number'
                   costcenter='Cost Center';
            format accountnumber $12.
                   costcenter 8.;
     contents data=securedloans;

run;quit;
```

In this example we have made ACCOUNTNUMBER to be outputted as a character with a length of 12, COSTCENTER a numeric with a length of 8. Notice that both the LABEL and FORMAT statements were done on the new variable names. This is allowed because the DATASETS procedure executes in order and automatically.

TYING EVERYTHING TOGETHER

Now that we have learned the basics of the DATASETS procedure, I want to give a complete example of everything we have learned and compare it to what you would have to do if you did not use the DATASETS procedure.
Example:

```
libname mylib 'c:\temp';

proc datasets library=mylib memtype=data;
     append base=securedloans data=mortgages;
     delete mortgages;
     modify securedloans(label='Secured Loans');
            index create accno cct_no;
            rename accno=accountnumber
                   cct_no=costcenter;
            label accountnumber='Account Number'
                   costcenter='Cost Center';
            format accountnumber $12.
                   costcenter 8.;
     modify plus (label='Plus Customers');
            index create account companycost;
            rename account=accountnumber
               companycost=costcenter;
            label accountnumber='Account Number'
                   costcenter='Cost Center';
            format accountnumber $12.
                   costcenter 8.;
     contents data=securedloans;

run;quit;
```

Refer to OUTPUT 1.8 in the appendix
This example gives us a complete look at the statements we have covered. This example is good for the following scenario:
You have three SAS data sets. Two of the data sets have the same data but about different products. You want to combine the two product data sets and merge it with the third data set to get demographic information on those customers with these precuts. So the steps would be:

1. Concatenate the two product data sets together
2. Delete the data set that was concatenated so you can save of space
3. Create an index so when you merge the two data sets you do not have to sort them
4. Rename the variables on the two data sets so they will be able to merge
5. Label the data set and variables to have consistency on reports
6. Format how you want the variables outputted on your reports
7. Get information about the two data sets to make sure everything is correct

All seven steps can be done in one procedure. Here is an example of what would have to been done if the DATASETS procedure was not used:

```
libname mylib 'c:\temp';

data mylib.three;
     format accountnumber $12.
                costcenter 8.;
  set mylib.one(rename=(accno=accountnumber
                     cct_no=costcenter))
    mylib.two(rename=(accno=accountnumber
                     cct_no=costcenter));
     label accountnumber='Account Number'
              costcenter='Cost Center';
     run;

proc datasets library=mylib memtype=data;
     delete one tow;
     run;quit;

proc contents data=mylib.three;
     run;

data mylib.plus2;
     format accountnumber $12.
                costcenter 8.;
  set mylib.plus(rename=(acct=accountnumber
                companycost=costcenter));
     label accountnumber='Account Number'
              costcenter='Cost Center';
     run;

proc datasets library=mylib memtype=data;
     delete plus;
     run;quit;

proc contents data=mylib.plus2;
     run;

proc sort data=mylib.three;
     by accountnumber costcenter;
     run;

proc sort data=mylib.plus2;
     by accountnumber costcenter; run;
```

In this example you can see that the program is much longer (code wise) , multiple data steps and procedures were used. Processing time was always faster using the DATASETS procedure, but the time was greatly reduced when using large amounts of data. By using the second example, it requires you to know more syntax, procedures and data steps within SAS. If you know the DATASETS procedure you can do all of this with only knowing one procedure.

CONCLUSION

The DATASETS procedure is a powerful procedure to know. I only touched on the basis of this procedure; how to know what data sets exist in your library, information on the variable

within you data set, append two data sets together, indexing data sets, renaming variables, labeling data sets and variables, modifying data sets, deleting data sets, all within one procedure. There are a lot more things this procedure can do and I challenge you learn all you can about this procedure. Saving time, space and work is what our goal is as programmers. PROC DATASETS does all three for us with little effort.

REFERENCES

SAS Institute Inc. (1990), *SAS Procedures Guide, Version 6, Third Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1990), *SAS Language Reference, Version 6, Third Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1999)*, "SAS Procedures", SAS Version 8 Online Documentation*, Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

Special thanks to Christine Grande and Lee Robson for allowing the time to create this tutorial.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Lisa M. Davis
Bank of America
9000 Southside Blvd.
FL9-400-05-04
Jacksonville, Florida 32256
Work Phone: 904-987-3753
Fax: 904-987-3747
Email: Lisa.M.Davis@bankofamerica.com

APPENDIX

OUTPUT 1.1

```
                Libref:        MYLIB
                Engine:        V8
                Physical Name: c:\temp
                File Name:     c:\temp


                  File
   #  Name          Memtype     Size  Last Modified
ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ
1  MORTGAGES     DATA      648192   19JUL2000:21:53:54
2  PLUS          DATA      123904   19JUL2000:21:52:56
3  SECUREDLOANS  DATA      656384   19JUL2000:16:02:02
```

OUTPUT 1.2

```
                           The DATASETS Procedure

Data Set Name: MYLIB.SECUREDLOANS      Observations: 10000
Member Type: DATA                      Variables:    8
Engine:        V8                      Indexes:      0
Created:15:10 Tuesday, July 11, 2000   Observation Length: 64
Last Modified: 16:02 Wednesday, July 19, 2000
Deleted Observations: 0
Protection:                            Compressed:   NO
Data Set Type:                         Sorted:       YES
Label:

                  -----Engine/Host Dependent Information-----

                   Data Set Page Size:         8192
                   Number of Data Set Pages:   80
                   First Data Page:            1
                   Max Obs per Page:           127
                   Obs in First Data Page:     96
                   Number of Data Set Repairs: 0
                   File Name: c:\temp\securedloans.sas7bdat
                   Release Created:            8.0000M0
                   Host Created:               WIN_NT

#    Variable         Type    Len    Pos    Format Informat  Label

ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ
 1  ACCNO             Char     21     32    $21.    $21.     ACCNO
 3  ACC_OPN_DT        Num       8      8    DATE9.  DATE9.   ACC_OPN_DT
 6  ACC_PD_CTGY_CD    Char      3     53    $3.     $3.      ACC_PD_CTGY_CD
 2  CCT_NO            Num       8      0    11.     11.      CCT_NO
 4  CLS_DT            Num       8     16    DATE9.  DATE9.   CLS_DT
 7  PRD_PRMRY_TYPE_CD Char      3     56    $3.     $3.      PRD_PRMRY_TYPE_CD
 8  PRD_SECDRY_TYP_CD Char      3     59    $3.     $3.      PRD_SECDRY_TYP_CD
 5  acctbalance       Num       8     24    17.2    17.2     Average Account

                           -----Sort Information-----

                             Sortedby:      CCT_NO
                              Validated:     YES
                             Character Set: ANSI
```

OUTPUT 1.3

```
                    The DATASETS Procedure

Data Set Name: MYLIB.SECUREDLOANS    Observations:      20000
Member Type:   DATA                  Variables:             8
Engine:        V8                    Indexes:               0
Created:15:10 Tuesday, July 11, 2000 Observation Length:   64
Last Modified: 22:03 Wednesday, July 19, 2000
Deleted Observations: 0
Protection:                          Compressed:           NO
Data Set Type:                       Sorted:               NO
Label:
```

OUTPUT 1.4

```
                    The DATASETS Procedure

Data Set Name: MYLIB.SECUREDLOANS      Observations:     20000
Member Type:   DATA                    Variables:            8
Engine:        V8                      Indexes:              0
Created:15:10 Tuesday, July 11, 2000   Observation Length:  64
Last Modified: 22:19 Wednesday, July 19, 2000
Deleted Observations: 0
Protection:                            Compressed:          NO
Data Set Type:                         Sorted:              NO
Label:         Secured Loans
```

OUTPUT 1.5

```
                    -----Directory-----

                    Libref:        MYLIB
                    Engine:        V8
                    Physical Name: c:\temp
                    File Name:     c:\temp


                                  File
 #  Name            Memtype      Size    Last Modified
 ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ
 1  MORTGAGES       DATA       656384   19JUL2000:22:26:26
 2  PLUS            DATA        74752   20JUL2000:00:28:04
    PLUS            INDEX       21504   20JUL2000:00:28:04
 3  PLUSCUSTOMERS   DATA       123904   19JUL2000:22:05:48
 4  SECUREDLOANS    DATA      1950720   20JUL2000:00:28:02
    SECUREDLOANS    INDEX      865280   20JUL2000:00:28:02
```

OUTPUT 1.6

```
     -----Alphabetic List of Variables and Attributes-----

# Variable          Type    Len  Pos   Format  Informat  Label
ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ
1 ACCNO             Char     21  32    $21.     $21.     Account Number
3 ACC_OPN_DT        Num       8   8    DATE9.   DATE9.   ACC_OPN_DT
6 ACC_PD_CTGY_CD    Char      3  53    $3.      $3.      ACC_PD_CTGY_CD
2 CCT_NO            Num       8   0    11.       11.     Cost Center
4 CLS_DT            Num       8  16    DATE9.    DATE9.  CLS_DT
7 PRD_PRMRY_TYPE_CD Char      3  56    $3.       $3.     PRD_PRMRY_TYPE_CD
8 PRD_SECDRY_TYP_CD Char      3  59    $3.       $3.     PRD_SECDRY_TYP_CD
5 acctbalance       Num       8  24    17.2      17.2    Average Account
```

OUTPUT 1.7

```
            -----Alphabetic List of Variables and Attributes-----

 # Variable          Type    Len  Pos   Format   Informat  Label

ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ
 3 ACC_OPN_DT        Num       8    8   DATE9.    DATE9.    ACC_OPN_DT
 6 ACC_PD_CTGY_CD    Char      3   53    $3.       $3.      ACC_PD_CTGY_CD
 4 CLS_DT            Num       8   16   DATE9.    DATE9.    CLS_DT
 7 PRD_PRMRY_TYPE_CD Char      3   56    $3.       $3.      PRD_PRMRY_TYPE_CD
 8 PRD_SECDRY_TYP_CD Char      3   59    $3.       $3.      PRD_SECDRY_TYP_CD
 1 accountnumber     Char     21   32    $12.      $21.     Account Number
 2 costcenter        Num       8    0    F8.       11.      Cost Center
 5 acctbalance       Num       8   24    17.2      17.2     Average Account

                -----Alphabetic List of Indexes and Attributes-----

                                          # of
                                          Unique
                          #    Index      Values
                         ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ
                           1    accountnumber     10000
                           2    costcenter          116
```

OUTPUT 1.8

```
                    -----Directory-----

                    Libref:        MYLIB
                    Engine:        V8
                    Physical Name: c:\temp
                    File Name:     c:\temp


                                      File
    #  Name           Memtype      Size  Last Modified
    ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ
    1  PLUS           DATA        74752   20JUL2000:00:33:34
       PLUS           INDEX       21504   20JUL2000:00:33:34
    2  PLUSCUSTOMERS  DATA       123904   19JUL2000:22:05:48
    3  SECUREDLOANS   DATA      1950720   20JUL2000:00:33:34
       SECUREDLOANS   INDEX      865280   20JUL2000:00:33:34


Data Set Name: MYLIB.SECUREDLOANS          Observations:   20000
Member Type:   DATA                        Variables:          8
Engine:        V8                          Indexes:            2
Created:15:10 Tuesday, July 11, 2000       Observation Length:64
Last Modified: 0:28 Thursday, July 20, 2000
Deleted Observations: 0
Protection:                                Compressed:        NO
Data Set Type:                             Sorted:            NO
Label:         Secured Loans



          -----Engine/Host Dependent Information-----

      Data Set Page Size:         8192
      Number of Data Set Pages:   238
      First Data Page:            1
      Max Obs per Page:           127
      Obs in First Data Page:     96
      Index File Page Size:       4096
      Number of Index File Pages: 211
      Number of Data Set Repairs: 0
      File Name:                  c:\temp\securedloans.sas7bdat
      Release Created:            8.0000M0
      Host Created:               WIN_NT
```

OUTPUT 1.8 CONTINUED

-----Alphabetic List of Variables and Attributes-----

#	Variable	Type	Len	Pos	Format	Informat	Label
3	ACC_OPN_DT	Num	8	8	DATE9.	DATE9.	ACC_OPN_DT
6	ACC_PD_CTGY_CD	Char	3	53	$3.	$3.	ACC_PD_CTGY_CD
4	CLS_DT	Num	8	16	DATE9.	DATE9.	CLS_DT
7	PRD_PRMRY_TYPE_CD	Char	3	56	$3.	$3.	PRD_PRMRY_TYPE_CD
8	PRD_SECDRY_TYP_CD	Char	3	59	$3.	$3.	PRD_SECDRY_TYP_CD
1	accountnumber	Char	21	32	$12.	$21.	Account Number
5	acctbalance	Num	8	24	17.2	17.2	Average Account
2	costcenter	Num	8	0	F8.	11.	Cost Center

-----Alphabetic List of Indexes and Attributes-----

#	Index	# of Unique Values
1	accountnumber	10000
2	costcenter	116