Paper SA-14-2014

An Introduction to the Mighty DATASETS Procedure

Ben Cochran, The Bedford Group, Raleigh, NC

ABSTRACT

On occasions, a SAS user might find themselves in the position where they need to do a number of things to a SAS dataset, like copying it or renaming it or even deleting it. All of these tasks, and many more can be done with the incredible DATASETS procedure. The purpose of this paper is to show a step by step approach to accomplishing these tasks.

INTRODUCTION

The DATASETS procedure is a utility procedure that manages your SAS files. With the DATASETS procedure you can do the following:

- copy SAS files from one location to another.
- rename SAS files,
- repair SAS files,
- delete SAS files,
- list a SAS library's content,
- list the attributes of a SAS data set,
- append SAS datasets,
- modify attributes of SAS data sets and variables within the data set,
- create and manage audit files for SAS data sets.

The typical syntax of the DATASETS procedure is:

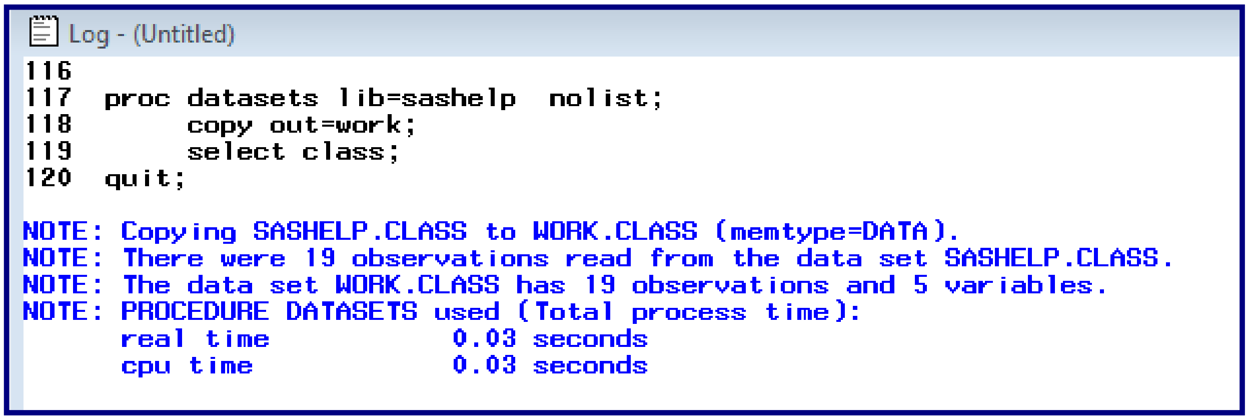
```
PROC DATASETS << option – 1  <… option – n >>> ;

          AGE  current-name  related-SAS-file-1
             < current-name related-SAS-file-n> ;
          APPEND BASE= SAS-dataset
                    DATA= SAS-dataset ;
          AUDIT  SAS-file < SAS-password) < GENNUM = integer > ) > ;
          CHANGE  old-name-1 = new-name-1
                    < old-name-n = new-name-n > ;
          CONTENTS  < option-1 < option-n >> ;
          COPY OUT = SAS-file
                 IN =  SAS-file
                 EXCLUDE  SAS-file < SAS-file-n >
                 SELECT   SAS-file  <  SAS-file-n > ;
          DELETE  SAS-file-1  <  SAS-file-n > ;
          MODIFY SAS-file < option-1  < … option-n  >   ;
                 ATTRIB   variable list(s) attribute lists(s) ;
                 FORMAT variable-1  format-1  ;
                 IC CREATE  < constraint-name = >  constraint ;
                 IC DELETE   constraint-name-1  < constraint-name-n > |  _ALL_  ;
             INDEX CREATE   index-specification  < options > ;
             INDEX DELETE    index-1  < index-n > |  _ALL_   ;
             INFORMAT  variable-1  < informat-1 > ;
             LABEL  variable-1 = < 'label-1' >  ;
             RENAME  old-name = new-name ;
          REBUILD  SAS-file  <  options > ;
          REPAIR   SAS-file-1  <  SAS-file-n > ;
          SAVE   SAS-file-1  < SAS-file-n > ;

 run ;
```

Figure 1. – Proc DATASETS Syntax.

Example 1: Use the DATASETS procedure to **copy** the CLASS dataset from the SASHELP library to the WORK library.



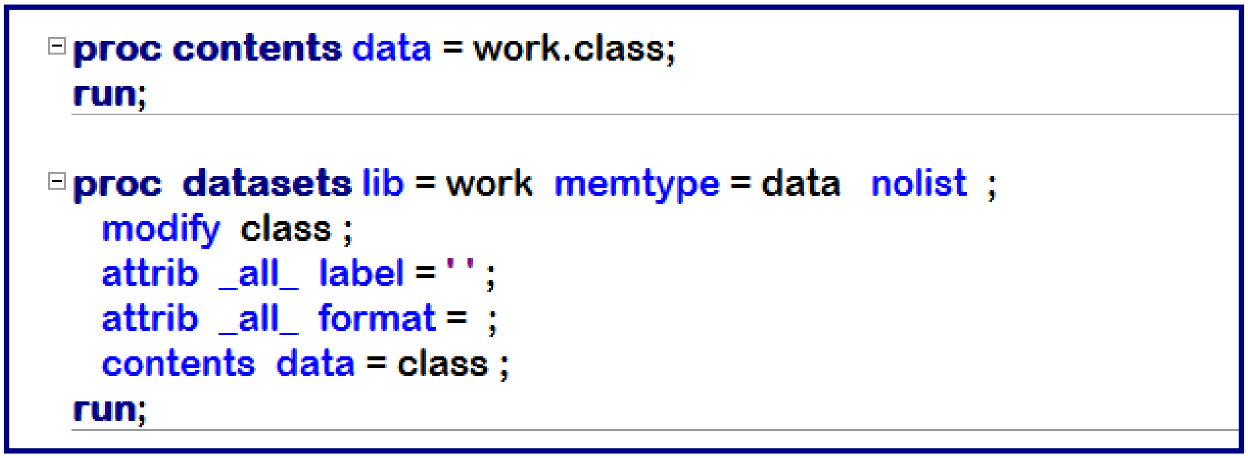
```
Log - (Untitled)
116
117   proc datasets lib=sashelp  nolist;
118        copy out=work;
119        select class;
120   quit;

NOTE: Copying SASHELP.CLASS to WORK.CLASS (memtype=DATA).
NOTE: There were 19 observations read from the data set SASHELP.CLASS.
NOTE: The data set WORK.CLASS has 19 observations and 5 variables.
NOTE: PROCEDURE DATASETS used (Total process time):
      real time           0.03 seconds
      cpu time            0.03 seconds
```

Figure 2. Copying a data set.

The NOLIST option on the PROC statement suppresses information on the contents of the SASHELP library. If this information is not desired, it makes the SAS/Log less cluttered.

Example 2: Use the DATASETS procedure to remove all labels and formats from the WORK.CLASS dataset. Use the CONTENTS procedure to see the initial attributes. Use the CONTENTS statement in the DATASETS procedure to verify the removal of the labels and the formats from all variables.

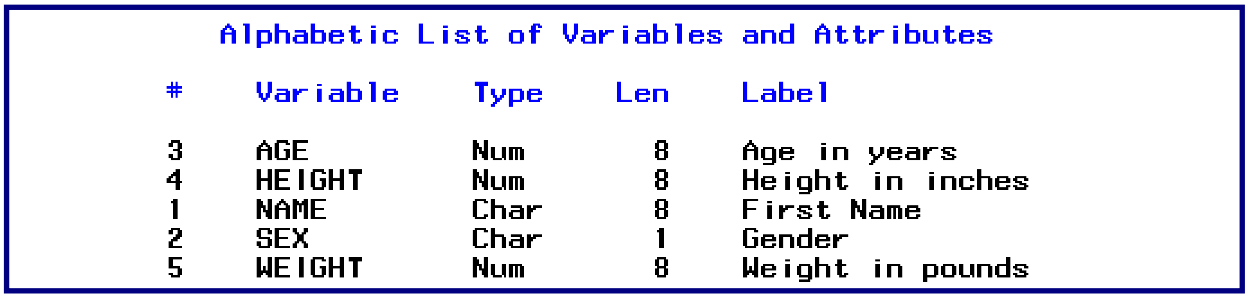


```
proc contents data = work.class;
run;

proc  datasets lib = work  memtype = data   nolist ;
  modify  class ;
  attrib  _all_  label = '' ;
  attrib  _all_  format =  ;
  contents  data = class ;
run;
```

Figure 3. Removing attributes.

The output from the CONTENTS procedure shows the original labels.

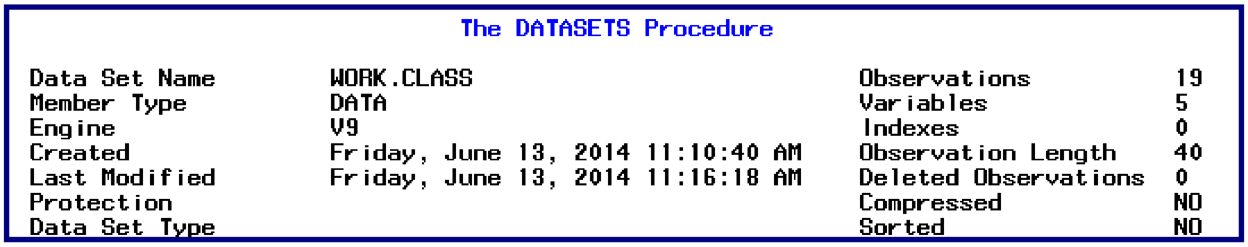


```
        Alphabetic List of Variables and Attributes

  #    Variable      Type    Len    Label

  3    AGE           Num       8    Age in years
  4    HEIGHT        Num       8    Height in inches
  1    NAME          Char      8    First Name
  2    SEX           Char      1    Gender
  5    WEIGHT        Num       8    Weight in pounds
```

Figure 4. Partial Proc CONTENTS output.

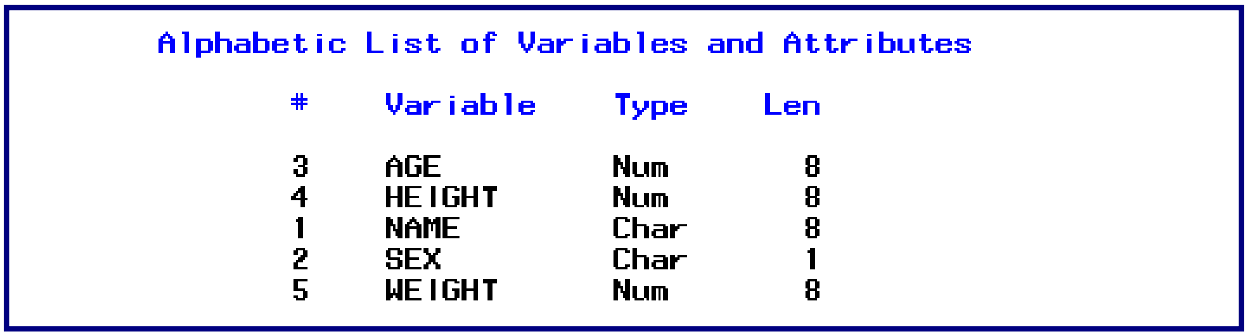


```
                         The DATASETS Procedure

Data Set Name          WORK.CLASS                              Observations          19
Member Type            DATA                                    Variables             5
Engine                 V9                                      Indexes               0
Created                Friday, June 13, 2014 11:10:40 AM       Observation Length    40
Last Modified          Friday, June 13, 2014 11:16:18 AM       Deleted Observations  0
Protection                                                     Compressed            NO
Data Set Type                                                  Sorted                NO
```

Figure 5. Partial Proc DATASETS output.

The effects of the CONTENTS statement in the DATASETS procedure shows no labels (or formats).



```
        Alphabetic List of Variables and Attributes

                #     Variable     Type     Len

                3     AGE          Num        8
                4     HEIGHT       Num        8
                1     NAME         Char       8
                2     SEX          Char       1
                5     WEIGHT       Num        8
```

Figure 6. Partial Proc DATASETS output.

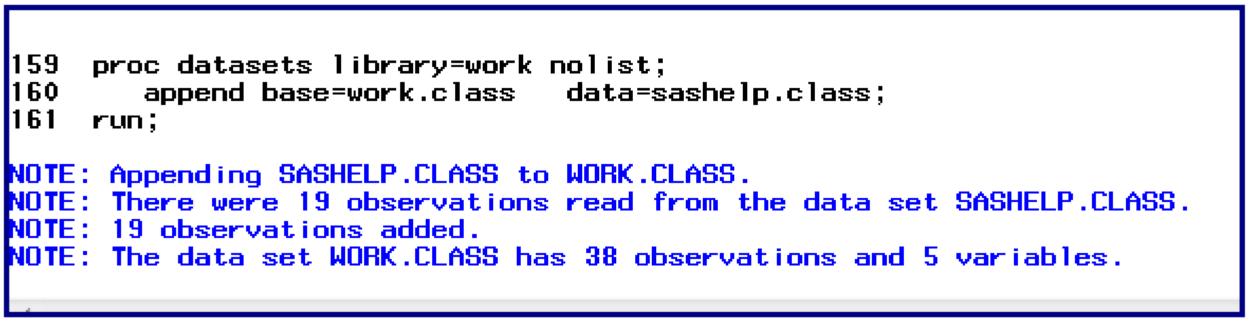
Example 3: Using the MOVE option on the COPY statement removes the data set from the original library

```
proc datasets lib=work;
   copy out = sas_3  move;
   select  catalog_sales;
quit;
```

Figure 7.   The MOVE option.

First, this step copies the CATALOG_SALES data set from the WORK library to the SAS_3 library. Once the copy is successfully made, the **MOVE** option deletes the data set from the WORK library. If the CATALOG_SALES data set already exists in the SAS_3 library, it will be overwritten by the above step.

Example 4: Use the DATASETS procedure to append the work.class data set to the SASHELP.CLASS data set.



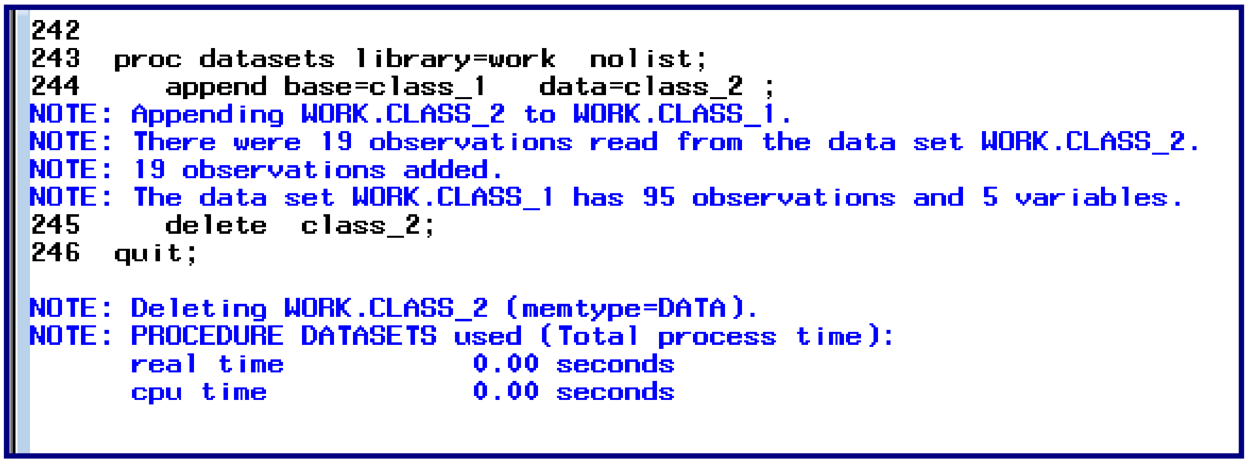
```
proc datasets library=work nolist;
   append base=work.class   data=sashelp.class;
run;

NOTE: Appending SASHELP.CLASS to WORK.CLASS.
NOTE: There were 19 observations read from the data set SASHELP.CLASS.
NOTE: 19 observations added.
NOTE: The data set WORK.CLASS has 38 observations and 5 variables.
```

Figure 8. The APPEND statement.

At the conclusion of this step, there are still NO labels on the variables. This is because the WORK.CLASS was listed as the **BASE** dataset. If the SASHELP.CLASS was listed as the **BASE** dataset, the variables would have labels.

Example 5: Modify the Proc DATASETS step by adding a DELETE statement to the step with the APPEND statement. After the concatenation of the data sets is complete, the data set named in the DELETE statement is may no longer be needed.



```
242
243   proc datasets library=work  nolist;
244      append base=class_1   data=class_2 ;
NOTE: Appending WORK.CLASS_2 to WORK.CLASS_1.
NOTE: There were 19 observations read from the data set WORK.CLASS_2.
NOTE: 19 observations added.
NOTE: The data set WORK.CLASS_1 has 95 observations and 5 variables.
245      delete  class_2;
246   quit;

NOTE: Deleting WORK.CLASS_2 (memtype=DATA).
NOTE: PROCEDURE DATASETS used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

Figure 9. The DELETE statement.

Example 6: Use the DATASETS procedure to create and delete indexes. The general form of the syntax is:

```
proc datasets library = libref ;
    modify dataset ;
       index delete index – name ;
       index create index – specification / options ;
quit;
```

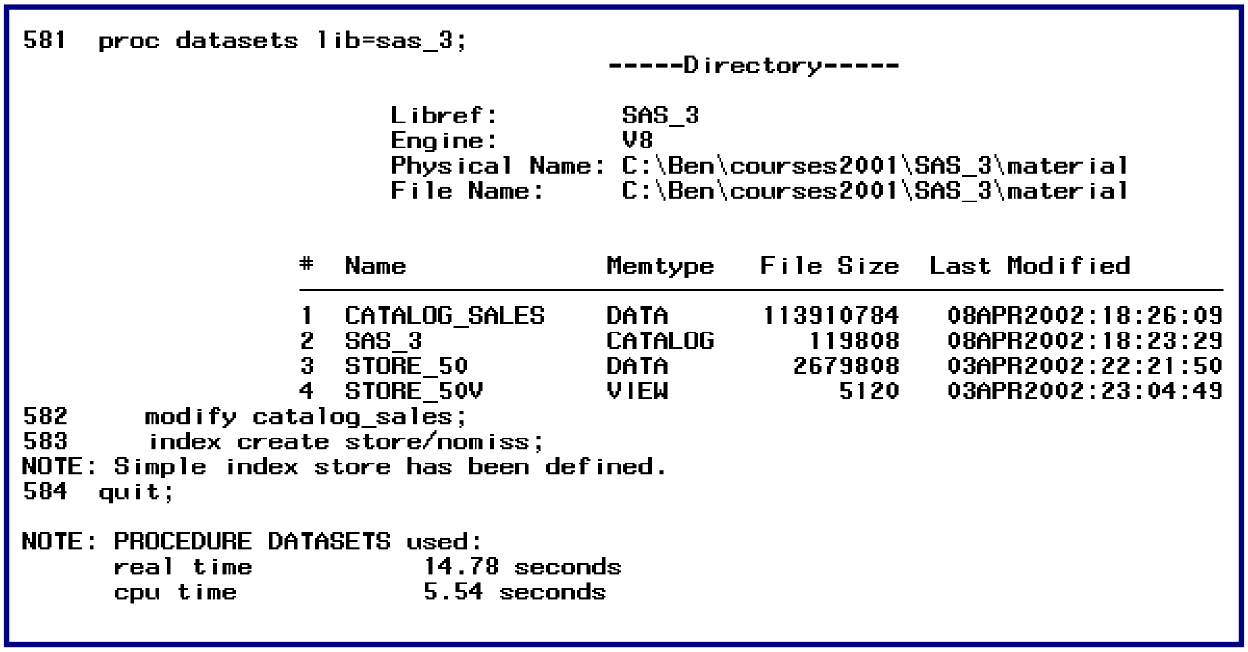
.
Figure 8. General syntax for creating and deleting indexes.

Example 6b: Specifically use the DATASETS procedure to create a **simple** index on an existing SAS data set (sas_3.catalog_sales) using STORE as the key variable.

```
proc datasets lib = sas_3;
   modify catalog_sales;
   index create store / nomiss;
quit;
```

Figure 9. Specific syntax for creating an index.

Notice that the NOLIST option is NOT used. The SAS log on the next page shows some of the information that is displayed by default. The contents of the SAS_3 library are displayed in the SAS log..


```
581  proc datasets lib=sas_3;
                                            -----Directory-----

                        Libref:        SAS_3
                        Engine:        V8
                        Physical Name: C:\Ben\courses2001\SAS_3\material
                        File Name:     C:\Ben\courses2001\SAS_3\material


                  #  Name              Memtype   File Size  Last Modified
                  ----------------------------------------------------------
                  1  CATALOG_SALES     DATA      113910784  08APR2002:18:26:09
                  2  SAS_3             CATALOG      119808  08APR2002:18:23:29
                  3  STORE_50          DATA        2679808  03APR2002:22:21:50
                  4  STORE_50V         VIEW           5120  03APR2002:23:04:49
582     modify catalog_sales;
583     index create store/nomiss;
NOTE: Simple index store has been defined.
584  quit;

NOTE: PROCEDURE DATASETS used:
      real time           14.78 seconds
      cpu time            5.54 seconds
```

Figure 10. The SAS Log.

Example 6c: Use the DATASETS procedure to create a **composite** index on an existing SAS data set (sas_3.catalog_sales) using STORE and YEAR as the key variables.

```
proc datasets lib = sas_3  nolist;
   modify catalog_sales;
   index create Store_Year = ( Store  Year ) / nomiss;
quit;
```

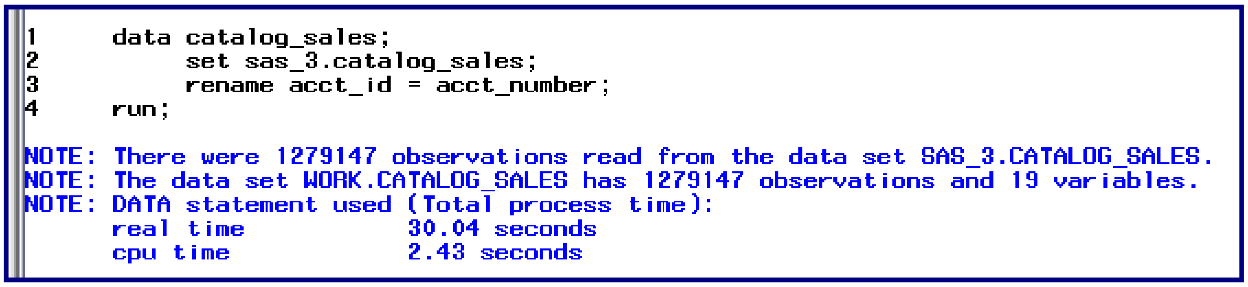
Figure 11. Creating a composite index.

In this example, the name of the index is STORE_YEAR and it is build on the two variables, STORE and YEAR.

EFFICIENT SAS PROGRAMMING

One way that you can write efficient SAS code is to use the DATASETS procedure to do some 'house keeping' tasks such as renaming variables.

Example 7a: Use the DATA step to INEFFICIENTLY rename a variable. Notice the amount of time it took the DATA step to run. (real time – 30.4 seconds, cpu time – 2.43 seconds).


```
1      data catalog_sales;
2           set sas_3.catalog_sales;
3           rename acct_id = acct_number;
4      run;

NOTE: There were 1279147 observations read from the data set SAS_3.CATALOG_SALES.
NOTE: The data set WORK.CATALOG_SALES has 1279147 observations and 19 variables.
NOTE: DATA statement used (Total process time):
      real time           30.04 seconds
      cpu time            2.43 seconds
```

Figure 11. Renaming a variable using the DATA step.

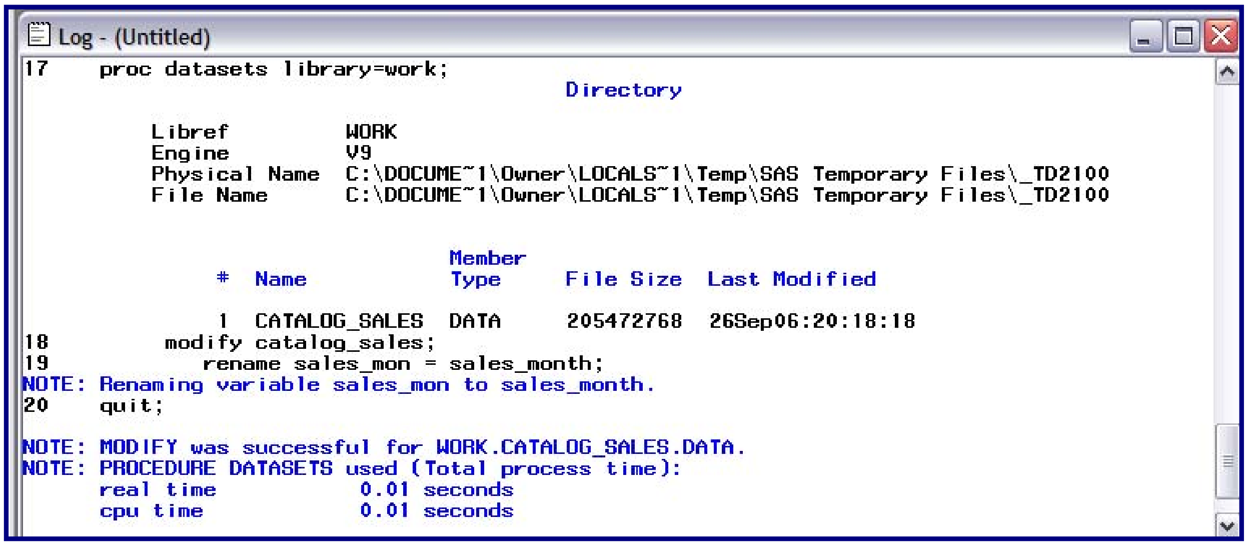
Example 7b: Use the DATASETS procedure to EFFICIENTLY rename a variable.

```
proc datasets library = work;
   modify catalog_sales;
   rename  sales_mon = sales_month;
quit;
```

Figure 12. Using the DATASETS procedure to rename a variable.

Look at the SAS log below. Notice the amount of time it took the DATASETS procedure to run. (real time –0.1 seconds, cpu time – 0.01 seconds).

A look at the SAS log reveals how quickly the above step runs.



```
Log - (Untitled)
17     proc datasets library=work;
                                                    Directory

           Libref         WORK
           Engine         V9
           Physical Name  C:\DOCUME~1\Owner\LOCALS~1\Temp\SAS Temporary Files\_TD2100
           File Name      C:\DOCUME~1\Owner\LOCALS~1\Temp\SAS Temporary Files\_TD2100


                                         Member
                #  Name              Type      File Size  Last Modified

                1  CATALOG_SALES     DATA      205472768  26Sep06:20:18:18
18          modify catalog_sales;
19             rename sales_mon = sales_month;
NOTE: Renaming variable sales_mon to sales_month.
20     quit;

NOTE: MODIFY was successful for WORK.CATALOG_SALES.DATA.
NOTE: PROCEDURE DATASETS used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

Figure 13. The SAS Log.

CONCLUSION

This paper illustrates some of the more useful tasks that can be done by the DATASETS procedure. Not only can one do many things with the DATASETS procedure, but it can perform them efficiently as well.

The author can be reached at:
Ben Cochran
The Bedford Group
3224 Bedford Avenue
Raleigh, NC 27607
(919) 741-0370
bencochran@nc.rr.com

sas. SAS Alliance Affiliate Member