Multiple Uses for a Simple SQL Procedure

Rebecca Larsen, University of South Florida, Tampa, FL

ABSTRACT

SAS users, who manage and analyze several secondary data sets from multiple sources, must prepare their data in order to provide standardized data sets that are suitable for pulling extracts and performing various analyses. A simple SQL procedure can be very effective as a tool for handling various data management & analysis steps. The CREATE TABLE and INSERT INTO…SELECT steps within the SQL procedure can be used to insert rows from multiple tables into one table, to insert rows from multiple queries into one table, to change the order of variables in a data set and to add formats, labels and change the names of variables in a data set. SAS users of all skill levels in a variety of disciplines can use these steps to manipulate their data sets. These steps are not specific to any one SAS version or operating system. The examples provided pertain to health care claims data, but the methods can be applied to other types of data as well.

INTRODUCTION

PROC SQL can be preferential to DATA steps and other SAS procedures in some circumstances. Many times it is just another way to do the same thing, but at other times it can cut down on the number of steps it takes to accomplish a certain task. It may also cut down on total lines of code, number of sorts and number of additional procedures required. Also, it may be a more intuitive language for users with more experience in other database systems, which may be important if collaborating on a project with persons in departments that do not regularly use SAS.

A procedure that comes in handy for a variety of different data management tasks is the CREATE TABLE and INSERT INTO…SELECT steps within PROC SQL. These steps can be used for the following purposes:

1. To insert rows from multiple tables into one table,
2. To insert rows from multiple queries into one table,
3. To change the order of variables in a data set, and
4. To add formats, labels and change the names of variables in a data set.

1. INSERTING ROWS FROM MULTIPLE TABLES INTO ONE TABLE

In the process of performing data analyses, you may need to create a master data set that combines information from multiple files. For example, in analyses of health care claims data, the data are often received in multiple files of different record type and variable layouts. In the development of a master data file, you can create a new, blank file using code similar to the following:

```
PROC SQL;
CREATE TABLE allclms
(
pers_id        char(11),
status_cd      char(2),
cat_cd         char(2),
race_cd               char(1),
svc_amt_cu     num,
svc_dt         num
)
;
quit;
```

Next, you can use a macro to insert the values from the different files into one master file. The file layouts in this example are similar, but have some differences. The id, service category code, status and race variables are named the same, but the service amount and service date variables have different names in the different source files.

```
%macro insert(amount, date, file);
  PROC SQL;
  INSERT INTO allclms
    SELECT
      pers_id,
      status_cd,
      cat_cd,
      race_cd,
      &amount.,
      &date.
    FROM &file.
    ;
  quit;
%mend insert;

%insert(serv_amt, serv_dt, hha_clm);
%insert(serv_amt, serv_dt, hsp_clm);
%insert(serv_amt, serv_dt, ip_clm);
```

```
%insert(serv_amt, serv_dt, snf_clm);
%insert(proc_amt, proc_dt, op_clm);
%insert(proc_amt, proc_dt, ptb_clm);
```

Notice in the macro calls that different source variables from the original files are inserted into a single variable in the new file (serv_amt vs. proc_amt and serv_dt vs. proc_dt). This works when the two variables in the original files are of the same type as the new variable in the new file.

2. INSERTING ROWS FROM MULTIPLE QUERIES INTO ONE TABLE

Once your master file has been created, you may then need to create a separate analysis file that has one record per person, rather than multiple service claim records per person. This blank file can be created in the same manner as the master, and records can be inserted by a series of queries.

The following code can be used to create a summary table from multiple queries:

```
PROC SQL;
CREATE TABLE min_cu
(
pers_id        char(11),
status_cd      char(2),
cat_cd         char(2),
minority_cd    char(4),
mean_cu        num
);

INSERT INTO min_cu
  SELECT
    distinct pers_id,
    status_cd,
    cat_cd,
    case
      when race_cd = '1'
      then '1'
      else '0'
    end as minority_cd,
    mean(serv_amt) as mean_cu
  FROM allclms
  WHERE serv_amt is not missing
  and cat_cd = 'mh' and count(distinct
  minority_cd) = 1
  GROUP BY pers_id;

INSERT INTO min_cu
  SELECT
    distinct pers_id,
    status_cd,
    cat_cd,
    case
      when race_cd = '1'
      then '1'
      else '0'
    end as minority_cd,
    mean(serv_amt) as mean_cu
  FROM allclms
  WHERE serv_amt is not missing
  and cat_cd = 'ph' and count(distinct
  minority_cd) = 1
  GROUP BY pers_id;
quit;
```

In this example, one record is created per person, and the race code is changed to reflect the more general categorizations of minority or non-minority. Also, a mean amount paid per person is created from the service cost variable in preparation for conducting some basic statistical analyses. If you want to conduct certain statistical tests at the person level, such as a t-test or an ANOVA, it is important that your class groupings are mutually exclusive and that you count a person in only one class. Otherwise the validity of your resulting statistics will be compromised. So, by specifying in the WHERE clause that you want only records where the person has one distinct value for the minority code, you eliminate the records where a person's minority status is indistinguishable. The count function in this WHERE clause is an example of SQL code that can accomplish in one step, what would take more than one step and several more lines of regular SAS code.

3. CHANGING THE ORDER OF VARIABLES IN A SAS DATA SET

In the process of managing data, you may want to change the order of variables in your data set. This can be accomplished using the SQL CREATE TABLE and INSERT INTO…SELECT steps.

Table 1 shows the order of variables in an example data set (preprodfile) before performing the reorder in SQL, and Table 2 shows the order of variables in the new data set (prodfile) after performing the changes.

Table 1. Order of variables *before* changing with PROC SQL (preprodfile)

ORDER OF VARIABLES	VARIABLE ATTRIBUTES
pin	char(11)
sex_cd	char(1)
race_cd	char(1)
dx_cd	char(8)
prcdr_cd	char(3)
dob	num
enroll_dt	num
termin_dt	num
death_dt	num
age	num

Table 2. Order of variables *after* changing with PROC SQL (prodfile)

ORDER OF VARIABLES	VARIABLE ATTRIBUTES
pin	char(11)
sex_cd	char(1)
race_cd	char(1)
dob	num
age	num
dx_cd	char(5)
prcdr_cd	char(5)
enroll_dt	num
termin_dt	num
death_dt	num

The code used to perform the variable reorder is as follows:

```
PROC SQL;
CREATE TABLE prodfile as
(
 pin           char(11),
 sex_cd        char(1),
 race_cd       char(1),
 dob           num,
 age           num,
 dx_cd         char(5),
 prcdr_cd      char(5),
 enroll_dt     num,
 termin_dt     num,
 death_dt      num
);

INSERT INTO prodfile
  SELECT
    pin,
    sex_cd,
    race_cd,
    dob,
    age,
    dx_cd,
    prcdr_cd,
    enroll_dt,
    termin_dt,
    death_dt
  FROM preprodfile;
quit;
```

4. CHANGING VARIABLE ATTRIBUTES AND/OR NAMES IN A SAS DATA SET

The names of the variables do not have to be identical in the newly created table as the table from which you are selecting records, so if the names of the variables need to be changed or amended, this is a good time to do so. Simply specify the new name you want in the CREATE TABLE statement, and then specify the existing variable name in the INSERT INTO…SELECT statement. Also, new formats and labels can be added to the newly created data sets.

For example, you decide you want to change the names of certain variables and add date formats to the example data set (prodfile) in Table 2. You want to change pin to pin_id, dob to dob_dt and age to age_iv. Also, you want to add a date format to the following variables: dob_dt, enroll_dt, termin_dt and death_dt. The following code will create a file (finfile) with changed variable names and added date formats.

```
PROC SQL;
CREATE TABLE finfile as
(
 pin_id        char(11),
 sex_cd        char(1),
 race_cd       char(1),
 dob_dt        num            format=mmddyy8.,
 age _iv       num,
 dx_cd         char(5),
 prcdr_cd      char(5),
 enroll_dt     num            format=mmddyy8.,
 termin_dt     num            format=mmddyy8.,
 death_dt      num            format=mmddyy8.
);

INSERT INTO finfile
  SELECT
    pin,
    sex_cd,
    race_cd,
    dob,
    age,
    dx_cd,
    prcdr_cd,
    enroll_dt,
    termin_dt,
    death_dt
  FROM prodfile;
quit;
```

SAS CODE HINTS & CAVEATS

The order in which you specify the variables in the INSERT INTO…SELECT statement must be the exact order of the variables that are created in the blank data set. Also, the attributes of the variables in the INSERT INTO…SELECT statement must be the same as the variables into which the values are being inserted, (e.g. numeric into numeric, character into character).

CONCLUSION

In conclusion, there are many applications for the CREATE TABLE, INSERT INTO…SELECT statements that can be used to perform multiple data management steps, including inserting rows from multiple tables into one table; inserting rows from multiple queries into one table; changing the order of variables in a data set; and adding formats, labels and changing the names of variables in a data set. The examples provided are just a few, simple ways you can use this procedure to manipulate your data to make it suitable for performing various analyses. Have fun and be CREATE-ive!

REFERENCES

SAS Institute Inc. (2000), *SAS® SQL Procedure User's Guide, Version 8*, Cary, NC: SAS Institute Inc.

ACKNOWLEDGEMENTS

I would like to express special thanks to everyone at the PSRDC, especially Paul Stiles, Shabnam Mehra, Diane Haynes and Keith Vossberg for their suggestions, encouragement and support.

CONTACT INFORMATION

Any questions or comments are welcomed and appreciated.

Rebecca Larsen
Phone: 813-974-7206
E-mail: rlarsen@fmhi.usf.edu

Policy & Services Research Data Center
Department of Mental Health Law & Policy
Louis de la Parte Florida Mental Health Institute
University of South Florida
13301 Bruce B. Downs Blvd., MHC2617
Tampa, Florida 33612