Paper 105-26

Name: **Clayton Wells**
Arkansas Foundation for Medical Care
401 W Capitol Suite 410
Little Rock AR 72201
(501) 375-5700
cwells@afmc.org

Tips For Using PROC SQL to Extract Information From Medical Claims Data Sets

Introduction

This paper is aimed at beginning SAS users who are writing SAS code used to manipulate and analyze medical claims data sets. Often, raw claims data sets contain one observation per claim. The SQL procedure provides a quick and efficient way to manipulate data sets of this type and allow the programmer to extract lots of information with minimal coding.

Problem

For simplicity, consider a data set that contains four variables – **id_number** (an identification number assigned to each unique person in the data set), **mem_month** (a date variable that represents for each person, the month the person was enrolled in a particular health care program), **provider** (health care provider where the person was enrolled for a given month), and **dob** (date of birth – this will obviously be the same within a given id_number). This data set, **DATA_1**, is shown in Figure 1. This data is fairly representative of a raw medical claims data set. Each observation in the set represents a month of enrollment in a health care plan by a particular individual. There are several questions that we can answer with this data:

1. How many unique members (ID numbers) are there in the data set?
2. How many months of membership does each unique member have?
3. How many unique providers are in the data set?
4. For each unique provider, how many unique members did the provider have enrolled for at least one month?
5. How many total “member months” did each unique provider have?

The SQL procedure is well suited to answer all of the questions above. For comparison purposes, an alternative (non-SQL) method will be demonstrated as well for questions 1 and 2.

Method 1 – PROC TRANSPOSE

One way to answer questions 1 and 2 above if to transpose the data in Figure 1 on the **mem_month** variable by the **id_number** variable. The result of this transformation is shown in Figure 1, **DATA_2**. Since **id_number** 1001 and 1006 contain 6 months of membership, more than the other unique identification numbers in the data set, there are 6 new variables in the resulting data set, labeled COL1 through COL6. The number of observations in **DATA_2** in Figure 1 is equal to the number of unique members and question 1 is answered. So we must now answer question 2. There are many ways to do this. The only one discussed here is as follows: create 6 new variables corresponding to COL1 through COL6. Call these new variables **m1** through **m6**. Assign these variables a value of 1 if its corresponding COL variable is non-missing, 0 otherwise. Then we can sum **m1** to **m12**. The resulting sum is the number of membership months and question 2 is now answered. The resulting data set containing the sum variable is called **DATA_3** in Figure 1.

The code for this method is as follows:

```
proc transpose data=lib.data_1 out=lib.data_2;
var mem_month; by id_number;
run;


data lib.data_3; set lib.data_2;
m1=0; m2=0; m3=0; m4=0; m5=0; m6=0;
if col1 ne . then m1=1;
if col2 ne . then m2=1;
if col3 ne . then m3=1;
if col4 ne . then m4=1;
if col5 ne . then m5=1;
if col6 ne . then m6=1;
sum=m1+m2+m3+m4+m5+m6;
run;
```

This method is not terribly inefficient (for this particular problem), but the SQL required to acquire the same information is definitely more compact.

Method 2 – SQL

A technique using the count function in SQL requires less code. The code shown below will create a new data set containing one observation for each unique member and will contain a variable indicating total number of membership months. The resulting data can be seen in **DATA_4** in Figure 1. The code needed to create the data set is as follows:

```
proc sql;
create table lib.data 4 as
select unique(id number),
       count(mem month) as member months
from lib.data_1 group by id_number;
quit;
```

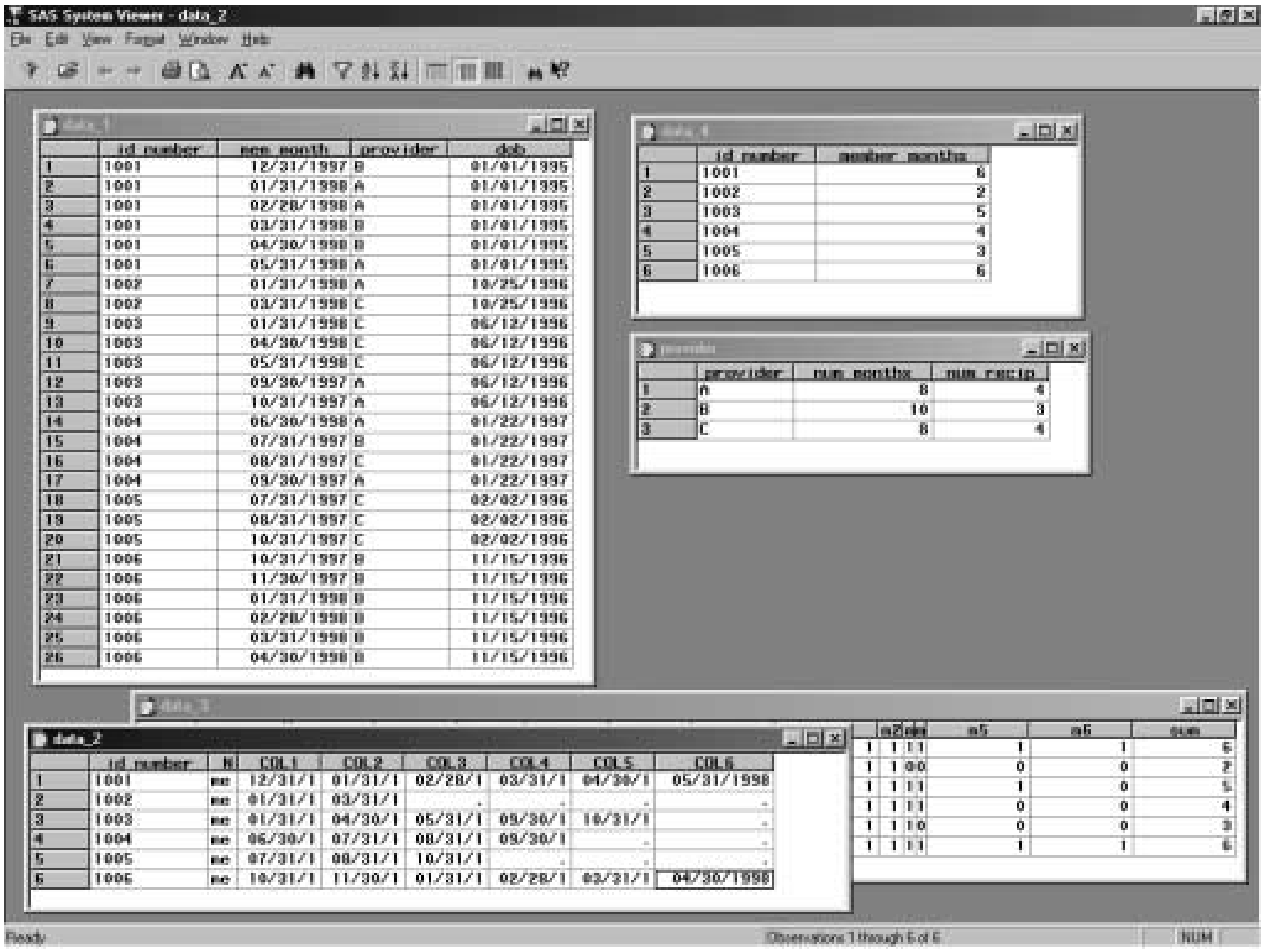
To answer questions 3, 4, and 5, we can again use the SQL procedure to create a new data set containing one observation per unique provider that contains variable representing the total number or members enrolled at least one month at that particular provider and a variable to represent the total number of member months at the particular provider. This data can be found in the **PROVIDER** data set in Figure 1. The code to create this set is as follows:

```
proc sql;
create table lib.provider as
select provider,
       count(id number) as num months,
         count(unique(id number)) as num_recip
from lib.data_1 group by provider;
quit;
```

Discussion

From inspection we are able to identify the answer to all of the questions listed above (since the raw data set only contains 26 observations). For question 1, we can see that the raw data set contains 6 unique identification variables. Also note that the data sets **DATA_2** and **DATA_4** contain six observations, as they should. For the next three questions, which pertain to the providers in the raw data set, we can note first that there are 3 unique providers, A, B, and C. Provider A had four unique members enrolled at one point - members 1001, 1002, 1003, and 1004. Provider B only had three unique members enrolled at one point - members 1001, 1004, and 1006. Hence the values of **num_recip** in the **PROVIDER** data set. The values for the variable **num_months** in **PROVIDER** is obtained by counting the number of times a particular provider appears in the raw data set, since each observation represents a month of membership. So provider A had 8 months of membership, provider B had 3, and so on.

Note in the problem presented here that the maximum number of member months per unique ID number was 6. Often this number is 12 or more. Method 1 requires at least one line of additional code for each additional member month added to the data set. The method utilizing PROC SQL, however, is independent of the number of member months.


SAS System Viewer - data_2
File Edit View Format Window Help
Ready
Observations 1 through 6 of 6
NUM

data_1

	id_number	mem_month	provider	dob
1	1001	12/31/1997	B	01/01/1995
2	1001	01/31/1998	A	01/01/1995
3	1001	02/28/1998	A	01/01/1995
4	1001	03/31/1998	B	01/01/1995
5	1001	04/30/1998	B	01/01/1995
6	1001	05/31/1998	A	01/01/1995
7	1002	01/31/1998	A	10/25/1996
8	1002	03/31/1998	C	10/25/1996
9	1003	01/31/1998	C	06/12/1996
10	1003	04/30/1998	C	06/12/1996
11	1003	05/31/1998	C	06/12/1996
12	1003	09/30/1997	A	06/12/1996
13	1003	10/31/1997	A	06/12/1996
14	1004	06/30/1998	A	01/22/1997
15	1004	07/31/1997	B	01/22/1997
16	1004	08/31/1997	C	01/22/1997
17	1004	09/30/1997	A	01/22/1997
18	1005	07/31/1997	C	02/02/1996
19	1005	08/31/1997	C	02/02/1996
20	1005	10/31/1997	C	02/02/1996
21	1006	10/31/1997	B	11/15/1996
22	1006	11/30/1997	B	11/15/1996
23	1006	01/31/1998	B	11/15/1996
24	1006	02/28/1998	B	11/15/1996
25	1006	03/31/1998	B	11/15/1996
26	1006	04/30/1998	B	11/15/1996

data_4

	id_number	member_months
1	1001	6
2	1002	2
3	1003	5
4	1004	4
5	1005	3
6	1006	6

provider

	provider	num_months	num_recip
1	A	8	4
2	B	10	3
3	C	8	4

data_3

	a2	[illegible]	[illegible]	a5	a6	sum
1	1	1	1	1	1	6
1	1	0	0	0	0	2
1	1	1	1	1	0	5
1	1	1	1	0	0	4
1	1	1	0	0	0	3
1	1	1	1	1	1	6

data_2

	id_number	N	COL1	COL2	COL3	COL4	COL5	COL6
1	1001	me	12/31/1	01/31/1	02/28/1	03/31/1	04/30/1	05/31/1998
2	1002	me	01/31/1	03/31/1	.	.	.	.
3	1003	me	01/31/1	04/30/1	05/31/1	09/30/1	10/31/1	.
4	1004	me	06/30/1	07/31/1	08/31/1	09/30/1	.	.
5	1005	me	07/31/1	08/31/1	10/31/1	.	.	.
6	1006	me	10/31/1	11/30/1	01/31/1	02/28/1	03/31/1	04/30/1998

Figure 1 - Data sets used by the SAS program