

PROC SQL, to collapse many-to-one relationship

Tadesse Haileyesus

Centers for Disease Control and Prevention, Atlanta, GA¹

ABSTRACT

Many researchers, especially those working with complex data sets are often challenged with collapsing and aggregating their data. The most frequent question is how to collapse or combine multiple relationships to get aggregates at a respondent level. In this paper, Structured Query Language (SQL) procedure will be utilized to communicate with a database and collapse many to one relationship. SQL is a powerful tool that talks to a relational database management system. Although SQL syntax is relatively simple and compact it is a powerful procedure for creating, combining and managing data. The SQL procedure collapses observations in an order defined by the user, keeps a record of all collapsed information, and retains the original and the aggregated record values. This paper will describe real-world applications as well as tips and techniques on effective uses so that others can adapt and put them into practice.

KEYWORDS: collapse, combine, aggregate, SQL.

INTRODUCTION

Many researchers, especially those working with complex data sets are often challenged with collapsing and aggregating their data. The most frequent question is how to collapse or combine multiple relationships to get aggregates at a respondent level. In this paper, simple PROC SQL (Structured Query Language), a powerful tool, will be utilized to communicate with a database and collapse many to one relationship. SQL is a language that talks to a relational database management system. Although SQL syntax is relatively simple and compact it is a powerful procedure for creating, combining and managing data. The SQL procedure collapses observations in an order defined by the user, keeps a record of all collapsed information, and retains the original and the aggregated record values. This paper will describe real-world applications as well as tips and techniques on effective uses so that others can adapt and put them into practice.

The example presented herein will demonstrate how to use PROC SQL to view, create, manage, summarize, and combine data. The example shown was done in the SAS® system for PC, version 9.1. The intended audience for this paper is beginner to intermediate level SAS users.

CHARACTERSTICS OF THE SAMPLE DATA SET

For simplicity, consider a data set that contains only three variables; the structure of this data set is similar to adolescent health studies and medical claims database.

Below is the code, for the data set called RELATION that will be used through out this paper. It contains a respondent or patient unique identifier number (ID), responses about the respondent (Victim), and responses about their partner (Perpetrate). In this sample data set ID has been encrypted for masking purposes. See below under DATA ENCRPTION EXAMPLE for details.

Notice that in the data set given below the first observation (ID=10570815) has one relationship (since there is only one record in the file). The second observation (ID=42506310) had three relationships (since there are three records in the file), etc.

¹ The findings and conclusions in this report (presentation) have not been formally disseminated by the Centers for Disease Control and Prevention and should not be construed to represent any agency determination or policy.

```

DATA relation;
  INPUT ID $ Victim Perpetrate;
CARDS;

10570815 1 1
42506310 0 0
42506310 . .
42506310 0 0
18606137 0 1
52316922 1 1
54316620 1 1
54316620 0 1
54316620 . .
54316711 0 0
;
RUN;

```

NOTE: The data set WORK.RELATION has 10 observations and 3 variables.

```

PROC PRINT DATA=relation;
RUN;

```

NOTE: PROCEDURE PRINT is used to output the data set RELATION, and it looks like the following:

Obs	ID	Victim	Perpetrate
1	10570815	1	1
2	42506310	0	0
3	42506310	.	.
4	42506310	0	0
5	18606137	0	1
6	52316922	1	1
7	54316620	1	1
8	54316620	0	1
9	54316620	.	.
10	54316711	0	0

Notice that the SORT procedure is used below to order the RELATION data set by the ID variable.

```

PROC SORT DATA=relation;
  BY ID;
RUN;

```

Below, PROC SQL followed by a semi-colon issued to start the SQL procedure. The second command line creates a table called AGGREGATE by selecting ID, count of the number of partners (Num_partner), sum of the number of partners that ever victimized the respondent (Ever_victim), and sum of the number of partners ever perpetrated by the respondent (Ever_perpetrate) from RELATION table.

Notice that count function is used with ID to keep track and count the number of partners, while the sum function is used with Victim and Perpetrate to summarize the number of (Victim) and (Perpetrate) variables at the respondent level. Hence, this procedure will collapse many to one relationship. Group by ID statement is used to specify how to combine the data for summarizing; the quit statement ends the SQL procedure.

Note that the summary functions include such useful tools as: SUM, MEAN, FREQ or COUNT, MAX, MIN, RANGE, STD and VAR, among others.

Here is the basic syntax:

```
PROC SQL;
  CREATE TABLE aggregate AS
  SELECT ID,
         COUNT(ID) AS Num_partner,
         SUM(victim) AS Ever_victim,
         SUM(perpetrate) AS Ever_perpetrate
  FROM relation
  GROUP BY ID;
QUIT;
```

NOTE: Table WORK.AGGREGATE created, with 6 rows and 4 columns. Notice that the rows collapsed from 10 to 6.

```
PROC PRINT DATA=aggregate;
RUN;
```

NOTE: PROCEDURE PRINT used to output the data set AGGREGATE, and it looks like the following:

Obs	ID	Num_ partner	Ever_ victim	Ever_ perpetrate
1	10570815	1	1	1
2	18606137	1	0	1
3	42506310	3	0	0
4	52316922	1	1	1
5	54316620	3	1	2
6	54316711	1	0	0

Notice that the first respondent (ID=10570815) has one partner; it has been victimized once as well as perpetrated the partner. The third respondent (ID=42506310) has three partners and has never been victimized nor perpetrated the partner, etc.

DATA ENCRPTION EXAMPLE

Here is an example of the use of substring function to encrypt ID or similar variables:

```
DATA temp;
  SET aggregate;
  ID_masked = SUBSTR(ID,7,2) || SUBSTR(ID,1,2) || SUBSTR(ID,3,4);
RUN;
```

NOTE: The data set WORK.TEMP has 6 observations and 5 variables. The last variable (ID_masked) is the encrypted ID. The encryption formula given here is not the actual one; this is for the illustration purpose only.

```
PROC PRINT DATA=temp;
RUN;
```

Here is how the data set WORK.TEMP looks:

Obs	ID	Num_ partner	Ever_ victim	Ever_ perpetrate	ID_masked
1	10570815	1	1	1	15105708
2	18606137	1	0	1	37186061
3	42506310	3	0	0	10425063
4	52316922	1	1	1	22523169
5	54316620	3	1	2	20543166
6	54316711	1	0	0	11543167

CONCLUSION

The SQL procedure is a powerful tool to communicate with a database and manipulate data. Now that you know a simple way of how to collapse or combine multiple relationships to get aggregates at a respondent level, you can adapt and put them into practice.

REFERECES

- SAS® Institute Inc. (1999), SAS Procedures Guide, Version 8, Cary, NC: SAS Institute Inc.
- Tamara Hagemeyer, Yue-Hwa Chang, "Use PROC SQL to Collapse Cells – It's Easy", Proceedings of the Twenty-Fourth Annual SAS Users Group International Conference (1999), Paper 74.
- Thomas J. Winn Jr., "Intermediate PROC SQL", Proceedings of the Twenty-Third Annual SAS Users Group International Conference (1998), Advanced Tutorials, Paper 35.
- Kirk Paul Lafler, "Manipulating Data with PROC SQL", Proceedings of the Thirteenth Annual SAS Users Group International Conference (2005), Tutorials, Paper 261-30.

ACKNOWLEDGMENTS

The author thank and express appreciation to J. Lee Annest, Ph.D., Chester Pogostin, DVM, MPA and Kevin Webb for their review and helpful comments on this paper, and all other colleagues for their continued support.

CONTACT INFORMATION

Tadesse Haileyesus, M.S.
Mathematical Statistician
Office of Statistics and Programming, NCIPC/CDC
4770 Buford Hwy (MS-K59), Atlanta, GA 30341
Phone (770) 488-1304, Fax (770) 488-1665
Email: THaileyesus@cdc.gov