SESUG Paper 274-2018

`Automation Methods: Using SAS® to Write PROC SQL Joins for You

Imelda C. Go, Questar Assessment, Inc.

ABSTRACT

When the number of variables involved is large and there are a number of aliases, it can actually take some time to type the code for a PROC SQL join even though the task is inherently simple. We first apply PROC CONTENTS to the two data sets to be joined in order to get a full list of variables for each data set. We then focus on adding to each data set variables that have information that define the join. We can identify which variables are to be included in the join, specify aliases for each variable in the join, and in general, specify other information that could be useful for the join. We then use SAS to process the information so that SAS creates the programming code required for the join. This method allows the programmer to focus on specifying the features of the join instead of focusing on typing the SAS code flawlessly.

At one point or another, a programmer faced with mundane, boring, repetitive, and/or time-consuming manual work has to start thinking about how to automate one's work in order to save time, to prevent errors, and to maintain one's sanity. When you have repetitive code, macros are a natural solution.

Joining data sets is possible through the SAS Data Integration Studio using a point and click graphic user interface (GUI) similar to what might find in the Microsoft Access query design view, it is still a manual method that can be tedious and prone to error when a large number of variables is involved.

The proposed automation method has benefits:

- When a large number of variables is involved, typing these statements manually can introduce error into the code and increase your debugging time. Using this technique can reduce debugging time. All you need to do is validate the macro and it will execute correctly regardless of how many variables need to be included in the join.
- Save a lot of time and effort by letting SAS dynamically create the code for you.

This paper discusses the steps for a simple contrived example that illustrates the proposed automation method.

1. Let us suppose that you have two data sets from two different years and each of them has the exact same variables (date of birth, ID, last name, first name, and score).

Year1 Data Set

dob	Id	lname	fname	score
2007/01/15	1	Smith	Ann	100
2008/02/16	2	Doe	Jane	98
2008/02/16	3	Doe	John	85

Year2 Data Set

dob	Id	lname	fname	score
2007/01/15	1	Smith	Ann	65
2009/04/26	2	Walker	Jane	36
2007/10/24	4	Taylor	John	67

2. Run PROC CONTENTS on each of the two data sets. Without the KEEP statement, PROC CONTENTS will provide a multitude of variables for you. For this example, it suffices to keep the `name` variable for this example. Because both data sets have the same variables that are in the same order, the PROC CONTENTS output will be the same for both data sets.

```
proc contents data=year1 out=year1vars (keep=name) noprint;
proc contents data=year2 out=year2vars (keep=name) noprint;
```

Year1vars and Year2vars Data Set

NAME
dob
Id
fname
lname
score

3. Proceed to add information to each data set and combine them into one data set.
   - Flag each variable that needs to be included in the join (`flag = Y`).
   - Add an `alias` variable for each of the data sets. If there is no alias, the alias value can default to the original variable name.
   - Specify the source data set or data set name (`dsn`) for each variable.
   - This example will stop at these three additional pieces of information, but this technique lends itself well to variations and further customization to suit your needs.
   - Once you have recorded what you need for the join, you are ready to write the macro that writes the SQL join.

Allvars Data Set

name	flag	alias	dsn
id	Y	Id	year1
dob	Y	dob1	year1
fname	Y	fname1	year1
lname	Y	lname1	year1
score	Y	score1	year1
id	N	id1	year2
dob	Y	dob2	year2
fname	Y	fname2	year2
lname	Y	lname2	year2
score	Y	score2	year2

PROGRAMMING EXAMPLE

Using the sample data sets above, we continue with the coding.

SAS CODE	EXPLANATION

`data allvars;`
`length allvars $600.;`
`retain allvars varcount;`
`set allvars end=lastrecord;`
`if flag='Y' then`
`do; varcount+1;`
`if varcount=1 then`
`allvars=strip(dsn)||"."||strip(name)||" as`
`"||strip(alias);`
`else allvars=strip(allvars)||",`
`"||strip(dsn)||"."||strip(name)||" as`
`"||strip(alias);`
`end;`
	This code processes the data in `allvars` data set. It starts to build the allvars character variable by checking if a variable has been flagged for inclusion in the data set. If the variable has been flagged, the code adds the information to the `allvars` character variable using the appropriates PROC SQL syntax.
	
	If the last record of `allvars` has been reached, two macro variables will be populated. The `allvarlist` macro variable will contain the list of variables with corresponding aliases to be included in the join. The `allvarcount` variable contains the number of

SAS CODE	EXPLANATION

`if lastrecord then`
` do; call symput ('allvarlist',allvars);`
`  put "&allvarlist";`
`  call symput ('allvarcount',varcount);`
`  put "&allvarcount";`
`  end;`
`run;`
	variables included in the `allvarlist` macro variable value.

`%macro sqljoin;`
`%if &allvarcount>0 %then`
`%do;`
`  proc sql;`
`  select &allvarlist`
`  from year1, year2`
`  where year1.id=year2.id;`
`  run;`
`%end;`
`%mend sqljoin;`
	These next few statements define a macro, which creates the PROC SQL statement needed for the join. In this example, the records are joined whenever the `id` values from both data sets are equal.

`%sqljoin;`	This statement invokes the `sqljoin` macro.

The `allvars` data set contains the following variables after processing:

Allvars	varcount	name	flag	alias	dsn
year1.id as id	1	id	Y	id	year1
year1.id as id, year1.dob as dob1	2	dob	Y	dob1	year1
year1.id as id, year1.dob as dob1, year1.fname as fname1	3	fname	Y	fname1	year1
year1.id as id, year1.dob as dob1, year1.fname as fname1, year1.lname as lname1	4	lname	Y	lname1	year1
year1.id as id, year1.dob as dob1, year1.fname as fname1, year1.lname as lname1, year1.score as score1	5	score	Y	score1	year1
year1.id as id, year1.dob as dob1, year1.fname as fname1, year1.lname as lname1, year1.score as score1	5	id	N	id1	year2
year1.id as id, year1.dob as dob1, year1.fname as fname1, year1.lname as lname1, year1.score as score1, year2.dob as dob2	6	dob	Y	dob2	year2
year1.id as id, year1.dob as dob1, year1.fname as fname1, year1.lname as lname1, year1.score as score1, year2.dob as dob2, year2.fname as fname2	7	fname	Y	fname2	year2
year1.id as id, year1.dob as dob1, year1.fname as fname1, year1.lname as lname1, year1.score as score1, year2.dob as dob2, year2.fname as fname2, year2.lname as lname2	8	lname	Y	lname2	year2
year1.id as id, year1.dob as dob1, year1.fname as fname1, year1.lname as lname1, year1.score as score1, year2.dob as dob2, year2.fname as fname2, year2.lname as lname2, year2.score as score2	9	score	Y	score2	year2

The final value for the `allvarlist` macro variable is:
`year1.id as id, year1.dob as dob1, year1.fname as fname1, year1.lname as lname1, year1.score as score1, year2.dob as dob2, year2.fname as fname2, year2.lname as lname2, year2.score as score2`

The final value for the `allvarcount` macro variable is 9.

PROC SQL will produce the following:

id	dob1	fname1	lname1	score1	dob2	fname2	lname2	score2
1	2007/01/15	Ann	Smith	100	2007/01/15	Ann	Smith	65
2	2008/02/16	Jane	Doe	98	2009/04/26	Jane	Walker	36

This example was chosen to show that matching solely on an ID number can be dangerous. ID number 2 belongs to Jane Doe in year 1 and to Jane Walker in year 2. Based on the birth date, the two individuals are probably not the same person.

CONCLUSION

This automation method proposes a way of programming that has the potential to decrease human error especially when many variables are involved. The programmer will use SAS to write the portion of the PROC SQL macro code that is tedious to generate manually.

CONTACT INFORMATION

Imelda C. Go
igo@questarai.com
Working remotely from Columbia, SC

TRADEMARK NOTICE