SAS
nrd

SAS Cartesian Product with PROC SQL and the Data Step

May 3, 2019

If you browse the SAS Online Communities, you will see that quite a few problems involve Cartesian products. There are a few ways to create Cartesian products in SAS. In this post, I will demonstrate two examples of how to create a Cartesian product in SAS. One with PROC SQL and one with the Data Step.

From a data point of view, a Cartesian product involves all combinations of observations between two or more data sets. Let us imagine the simplest case with two data sets. Data set one with **n observations** and data set two with **m observations**. The Cartesian product between the two tables will contain **n*m observations**.

Before, let us create some example data. We will use this data in the examples to come

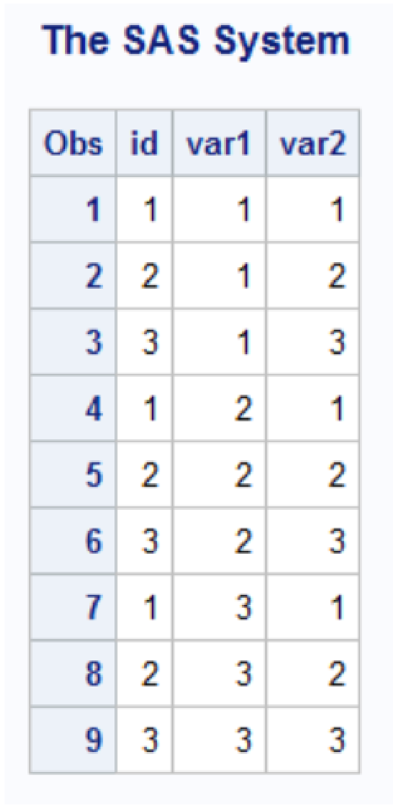
```
data test1;
input id var1;
datalines;
1 1
2 2
3 3
;

data test2;
input id var2;
datalines;
1 1
2 2
```

```
3 3
;
```

The PROC SQL Approach

The most common and straightforward way to create a Cartesian product in SAS is to use PROC SQL. A rule of thumb to remember is this: ***When you join two or more tables without a Where Clause, you create an internal Cartesian Product.***



The SAS System

Obs	id	var1	var2
1	1	1	1
2	2	1	2
3	3	1	3
4	1	2	1
5	2	2	2
6	3	2	3
7	1	3	1
8	2	3	2
9	3	3	3

Consider the code below. Here, I join the two tables **test1** and **test2**. I specify no join criteria in a Where Clause however. Both Test1 and Test2 contains 3 observations respectively. Consequently, the Cartesian product contains 9 observations.

You can see the result to the right. As you can see, the data set contains all possible combination of rows between the two tables.

If you examine the log after running the program below, you will see the following note: *"**NOTE:** The execution of this query involves performing one or more Cartesian product joins that can not be optimized."*. This note is quite important. If you do not intent to create a Cartesian product or if you are unaware that you do create one, then be careful. A Cartesian product can be *very* large and *very* unintended.

```
proc sql;
   create table CartSQL as
   select test1.*,
          test2.var2
   from test1, test2;
quit;
```

The Data Step Approach

In the next example, let us do the same with a SAS Data Step. This method is not as common. However, if you are more familiar with the Data Step than PROC SQL, then this will probably make more sense to you. In the code below, I start reading the **test1** data set in the first Set Statement. For each observation in **test1**, I read every single observation in **test2** with the Point= and the Nobs Options.

```
data CartDataStep;
   set test1;
   do i=1 to n;
      set test2 point=i nobs=n;
      output;
   end;
run;
```

Summary

In this post, I demonstrate two examples of how to create Cartesian products in SAS. One example with PROC SQL and one with the SAS Data Step. I recommend the SQL Procedure. However, if the situation applies, the Data Step approach can be just as fine. However, make sure that you actually *need* a Cartesian product. They are computationally heavy and usually, there is a smarter way.

For PROC SQL related posts, see Select Into Multiple Macro Variables in SAS Proc SQL and Using the _Method Option in PROC SQL. If you are serious about learning PROC SQL, get the book PROC SQL: Beyond the Basics Using SAS by Kirk Paul Lafler. As a related post, I also use the Point= Option to Reverse the Order of a SAS Data Set.

You can download the entire code from this post here.