PharmaSUG 2015 - Paper TT02

Phantom of the ODS – How to run cascading compute blocks off of common variables in the data set for complex tasks.

Robin M. Sandlin, Cook Systems International, Inc., Memphis, TN

ABSTRACT

This paper teaches an SAS®-based Stored Process formatting technique to be used stand-alone or combined with other style techniques. This methodology is a prototype framework for use by SAS® programmers who may not be able to achieve similar fashion results using more sophisticated techniques (such as inline “^S” techniques) .

This paper gives examples and complete code, leveraging users knowledge in two different areas, the SQL procedure, using PROC SQL basic techniques and the REPORT procedure, using the PROC REPORT compute block framework (albeit multiple blocks) in what is called “Phantom of the ODS.” Due to the nature of the way that compute blocks resolve, the execution of the block prior to its application effectively results in only one style statement per block without resorting to advanced techniques such as inline style, etc. This effect, combined with the one compute block per variable rule, normally results in one style statement per column or grouping variable.

The technique introduces “phantom data” in PROC SQL which “cloned” from existing data using column aliases. The “phantom data” variables are then listed in the Proc Report column statement, but set as non printing in the PROC REPORT define statement, not visible to the report user, yet available for multiple compute block styling, which is visible, one block for each phantom variable (no limit actually). The one compute block per variable design paradigm is thus shifted. Anyone with basic SQL knowledge and general knowledge of single compute block processing should find this technique quickly and effectively implementable.

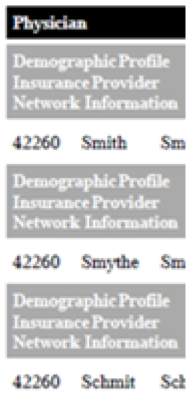
INTRODUCTION

This paper is the application of a SQL technique known as using pseudo-data or phantom-data, in combination with SAS PROC REPORT features for a nice and easy way to get formatting styled and aligned with output destinations that can be somewhat tricky such as the ODS PDF, ODS RTF and ODS HTML destinations. I have used this technique successfully on various clients and industries from medical research to distribution and variations on the general theme. It’s actually more of a work around than anything, when one compute block just isn’t enough. I have made every attempt to make the paper a “fun read” using a workplace scenario.aper body. Intended audience would be beginner or novice level developers.

MULTIPLE COMPUTE BLOCKS

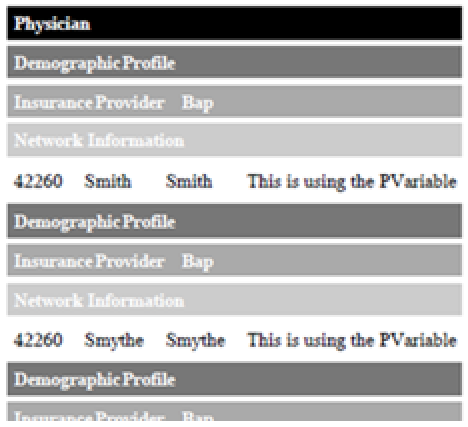
We will apply styling using multiple compute blocks which break before (or after) a common variable. Some call this “cascading” compute blocks, and it is typically accomplished by having each block call the next block using programming logic within the compute blocks themselves. We take a different approach and let data do the work of triggering the blocks. Note in the diagrams below (a report of patients by physician) there are multiple items of data that needs to be repeated each time the patients name changes.

In this example we use patients Smith, Smythe and Schmit for the Physician #42260. For each patient, we need a demographic profile, network information and insurance information. This is a typical theme for “operational” report used by operations personnel to go from detail (patient) up one level (demographic information, insurance provider and network). It may be the case that one department only uses part of the report, and frequently has to go to the same detail line over and over (for a different patient however). This is especially common when one report is used by multiple people or departments.



```
Physician
Demographic Profile
Insurance Provider
Network Information
42260   Smith    Sm
Demographic Profile
Insurance Provider
Network Information
42260   Smythe   Sm
Demographic Profile
Insurance Provider
Network Information
42260   Schmit   Sch
```

Figure 1 - Single compute block computed before each patient name changes.



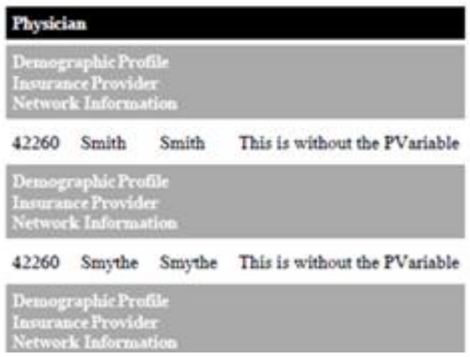
```
Physician
Demographic Profile
Insurance Provider   Bap
Network Information
42260   Smith    Smith    This is using the PVariable
Demographic Profile
Insurance Provider   Bap
Network Information
42260   Smythe   Smythe   This is using the PVariable
Demographic Profile
Insurance Provider   Bap
```

Figure 2 - Multiple compute blocks computed before each patient name changes.

Note that normally, in PROC REPORT, multiple compute blocks are not allowed to use the same variable. We are able to pull this off by setting up “phantom” values in the SQL procedure. The phantom values “mimmick” the actual values, and are dispensed with in PROC REPORT after they have served their purpose. A more advanced way of handling this scenario would be to use inline formatting. We will make the assumption that the user does not yet know inline techniques or you probably wouldn’t have read this far. There are several good source paper on inline techniques and I will not even attempt teach those techniques in this paper. When you are ready to learn inline, I would recommend, "Funny ^Stuff~ in My Code: Using ODS ESCAPECHAR" by Cynthia Zender. See reference below. For now, let’s relax, have some fun and learn a few tricks using PROC SQL and PROC REPORT.

COMPUTE BLOCKS: A WORKPLACE SCENARIO

Your boss is impressed by your SAS skills. She finally has the all the patient data she ever dreamed of in one consolidated report which you have written as a SAS stored process. Oh, just one question, it's pretty simple, "Can you make the report look similar to another report done by your predecessor which has a gray bar each time the patient name changes , so that high level information can quickly be found and differentiated from the core patient information?" Sure, no problem, you tell her about SAS's wonderful feature called compute blocks. The boss is impressed; you are off to the races. You use a compute block which is triggered each time the patient name changes, the results looks like the following.



Physician

Demographic Profile
Insurance Provider
Network Information

42260 Smith Smith This is without the PVariable

Demographic Profile
Insurance Provider
Network Information

42260 Smythe Smythe This is without the PVariable

Demographic Profile
Insurance Provider
Network Information

Figure 3 - Single Compute Block Example

This is great says the Boss. I knew it could be done. Now, I would like three lines, just like those, but each a different shade of gray, so that different users of the report can pick out their data from that of others. This is a realistic business need, a very reasonable and logical request, actually a brilliant idea. One problem, you can't do it based on your existing knowledge of SAS. Or, can you?

COMPUTE BLOCKS: BEHIND THE SCENES

In the example above the report first breaks on Physician name which is formatted in black (and which in this case is # 42260). Now notice that the next three lines are all specific to the patient, but each with a different purpose. In this example there are three lines to keep it simple. However, there could be three or three hundred. In any case they all are limited to the same traditional (non-inline) formatting for the entire compute block. Even using inline formatting, some things can be overridden (font, color) other things (alignment) cannot be overridden within the compute block.

Let's take a look at the source code for Figure 4.

First, within PROC SQL we set up a data table for the example.

```
proc sql;
CREATE TABLE WORK.SUG
  (
             Studynum num,
             Hospname char(20),
             Docnum num,
             Docname char(20),
             Patnum num,
             Patname char(20),
        Birth num informat=date7.
                  format=date7.,
        Added num informat=date7.
                  format=date7.);
```

Next, still within PROC SQL we will add some data lines:

```
proc sql;
insert into SUG
values(1123,'Bap',42260,'Lamb',1223,'Smith','26JUN71'd,'28JAN91'd)
values(1223,'Bap',42260,'Lamb',6320,'Smythe','25JUN72'd,'28JAN91'd)
values(32260,'Bap',42260,'Lamb',4604,'Schmit','27JUN73'd,'28JAN91'd)
values(32261,'StFr',42270,'Wolf',4211,'Jones','28JUN74'd,'28JAN91'd)
values(32261,'StFr',42270,'Wolf',2260,'Joan','29JUN75'd,'28JAN91'd)
values(32261,'StFr',42270,'Wolf',2346,'Johan','16JUN76'd,'28JAN91'd)
values(32262,'Presb',42280,'Hoot',3332,'Heartz','12JUN77'd,'28JAN91'd)
values(32262,'Presb',42280,'Hoot',6729,'Moonz','17JUN78'd,'28JAN91'd)
values(32262,'Presb',42280,'Hoot',5252,'Cloverz','18JUN79'd,'28JAN91'd)
;
```

Now, using PROC REPORT. We can add a compute block that computes before each patient record.

```
Compute Before Patname style=[background=grey left];
        Line  DemographicProfile ;
        Line  NetworkProvider;
        Line  InsuranceInformation;
Endcomp;
```

Now, the catch, the style and alignment is applied to the entire compute block. Again, inline styling can be used to change background color, etc. within the block, but other items such as alignment can not be changed inline within the block. Next, we will “tweak” the PROC SQL code to add two new columns, which are both alias values of the original patient column. This is the phantom data.

```
proc sql;
 CREATE TABLE WORK.SUG
  (
               Studynum num,
               Hospname char(20),
               Docnum num,
               Docname char(20),
               Patnum num,
               Patname char(20),
               Patname char(20) AS PatnamePhantom1,
               Patname char(20) AS PatnamePhantom2,



         Birth num informat=date7.
                   format=date7.,
         Hired num informat=date7.
                   format=date7.);
```

Now, back in PROC REPORT simply add the new phantom variables to the column statement.

```
Column        . . . Patname  PatnamePhantom1  PatnamePhantom2 . . .
```

Now, you're almost there. Define the phantom variable as "noprint" in the report definition.

```
define Patname / group ' '   ;
define PatnamePhantom1 / group ' ' noprint;
define PatnamePhantom2 / group ' ' noprint ;
```

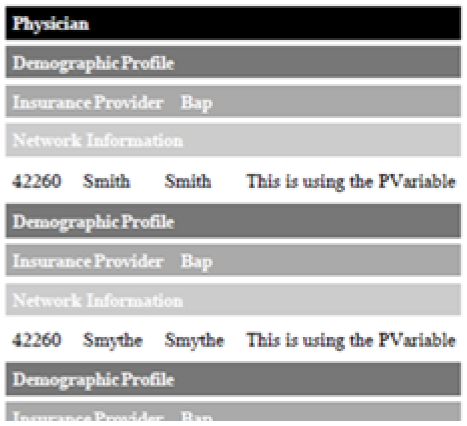
Last add a compute block for each phantom variable (you can actually add one before and one after each variable).

```
Compute Before Patname style=[background=grey left];
       Line  DemographicProfile;
Endcomp;

Compute Before PatnamePhantom1 style=[background=#999999 left];
       Line  NetworkProvider;
Endcomp;

Compute Before PatnamePhantom2 style=[background=#aaaaaa left];
       Line  InsuranceInformation;
Endcomp;
```

Now, you are able to ask the boss, "Which three shades of grey would you like?" Your report should look similar to the one in Figure 4 below.



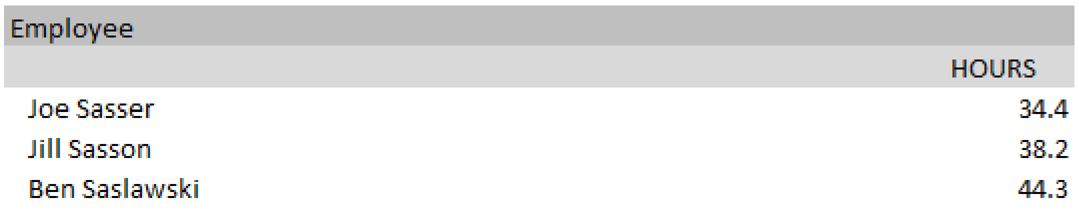
```
Physician
Demographic Profile
Insurance Provider   Bap
Network Information
42260   Smith   Smith   This is using the PVariable
Demographic Profile
Insurance Provider   Bap
Network Information
42260   Smythe   Smythe   This is using the PVariable
Demographic Profile
```

Figure 4 – Mission Accomplished

The boss is happy and you look like a superstar. Now might be a good to look into learing inline techniques.

ANOTHER VARIATION: NESTED COLUMN HEADING ALIGNMENT

Another variation of this strategy is in cases where there are column headings which print before or after a compute block, based on a grouping variable.



Employee	
	HOURS
Joe Sasser	34.4
Jill Sasson	38.2
Ben Saslawski	44.3

Figure 5 - Scalable Column Headings

Traditionally one could use a compute block that breaks before each new group variable value and a line statement to print the column headings. Spacing of the headings will vary when the code is run on multiple systems with different font sets. Again, using phantom variables, we can left justify column header such as “Employee” using one compute block before the variable, then right justify a column header such as “amount” using a second compute block before the phantom value of the same variable.

```
compute before emp   / style={background=#cccccc color=brbl  just=left} ;
       line justify=left 'Employee'   ;
endcomp;

compute before  empPhantom1  / style={background=#f0f0f0 color=brbl  just=right} ;
       line    justify=right 'HOURS'    ;
endcomp;
```

The result is column headers that “scale” to the width of the page.

CONCLUSION

This paper is intended as a general technique that can be used by users who do not yet use inline formatting techniques. These examples use styling and alignment because they are visual and easy to comprehend. But, anything that can be done in a compute block can be accomplished using multiple compute blocks and phantom data. Also, in cases where one compute block would get complicated and hard to follow (conditional logic for example), multiple compute blocks may make your code easirer to follow and to update. If it is truly style that you are focused on, inline techniques might be the way to go, once you are comfortable with the concept of compute blocks in general. For a great paper on inline techniques, I would highly recommend Funny ^Stuff~ in My Code: Using ODS ESCAPECHAR Cynthia L. Zender, SAS Institute Inc., Cary, NC. It’s very easy to understand and will point you in the right direction. See reference below.

REFERENCES

Zender, Cynthia. 2007. "Funny ^Stuff~ in My Code: Using ODS ESCAPECHAR". Technical Papers and Presentations. Cary, NC. Available at http://www2.sas.com/proceedings/forum2007/099-2007.pdf .

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Robin M. Sandlin
Cook System, International (SAS/SQL Contractor for FedEx, ALSAC/St. Jude)
6799 Great Oaks Rd # 200
Memphis, TN 38138
www.cooksys.com
(901) 649 - 3225
Rsandlin@CookSys.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.