Paper FF-11

Evolve from a Carpenter's Apprentice to a Master Woodworker: Creating a Plan for Your Reports and Avoiding Common Pitfalls in REPORT Procedure Coding

Allison McMahill Booth, SAS Institute Inc., Cary, NC

ABSTRACT

Do you sometimes get into a routine of using the same techniques to create PROC REPORT code only to end up with reports that no longer suit your needs? You're sure that there's a better way to code in PROC REPORT, but you don't know where to start. If you're frustrated with the amount of time that you spend creating reports that are less than ideal, consider taking the time to create a coding plan before you begin to code.

Although PROC REPORT coding is flexible and can be used with various SAS® applications, there are pitfalls. Just like an inexperienced carpenter's apprentice, you might code in familiar ways that you know will work, rather than in ways that work optimally with PROC REPORT. Making a coding plan and being aware of the pitfalls will enable you to code PROC REPORT easily, thereby taking you from an apprentice-level carpenter to a master-level woodworker.

INTRODUCTION

PROC REPORT combines features of the PRINT, TABULATE, and MEANS procedures with features of the DATA step in a single report-writing tool that you can use to produce a variety of reports. Although PROC REPORT has several options that are easy to use, there are pitfalls that are specific to using PROC REPORT, which differs from using other SAS procedures.

As output-reporting needs change, existing PROC REPORT code is often modified many times until the desired output is produced. Without a coding plan, this trial-and-error process can be frustrating and time consuming, and might not produce the desired output, especially if the pitfalls are not taken into account. The techniques in this paper enable you to evolve from impulsively hammering away at code to skillfully establishing a plan that creates masterful PROC REPORT code easily.

THE CARPENTER'S APPRENTICE

Carpentry covers a wide range of woodworking tasks, from minor wood repair, to house construction, to furniture building. Most carpenters start out as an apprentice under the guidance of a skilled and experienced woodworker and acquire their skills from on-the-job training. Until the apprentice learns more about the trade, he works on routine projects, using basic tools and standard methodologies. In the end, his work is acceptable and useful, but his lack of knowledge of the pitfalls of carpentry could result in mistakes as he works. Making mistakes hampers efficiency.

PRODUCING ROUTINE CODE

The woodworking analogy can also be applied to coding with PROC REPORT. You can run the same PROC REPORT code with few modifications and produce a report that looks nice. Consider the simplest form of PROC REPORT code and LISTING output (Figure 1):

```
proc report nowd data=sashelp.prdsale ls=120;
run;
```

```
Actual   Predicted                          Product
Sales    Sales    Country  Region Division  Type       Product Quarter  Year  Month
$925.00  $850.00  CANADA   EAST   EDUCATION FURNITURE  SOFA    1        1993  JAN
$999.00  $297.00  CANADA   EAST   EDUCATION FURNITURE  SOFA    1        1993  FEB
$608.00  $846.00  CANADA   EAST   EDUCATION FURNITURE  SOFA    1        1993  MAR
```

Figure 1. Basic PROC REPORT Code and LISTING Output

You can obtain the entire code by adding the LIST option to the PROC REPORT statement. The LIST option generates basic PROC REPORT code in the SAS log for the current PROC REPORT submission. Figure 2 shows the PROC REPORT code that is generated in the SAS log when the LIST option is added to the PROC REPORT code shown in Figure 1.

```
proc report data=sashelp.prdsale ls=120 ps=55  split="/" center ;
column  actual predict country region division prodtype product quarter year month;
   define  actual / sum format= dollar12.2 width=12 spacing=2 right "Actual Sales" ;
   define  predict / sum format= dollar12.2 width=12 spacing=2 right "Predicted Sales" ;
   define  country / display format= $char10. width=10 spacing=2 left "Country" ;
   define  region / display format= $char10. width=10 spacing=2 left "Region" ;
   define  division / display format= $char10. width=10 spacing=2 left "Division" ;
   define  prodtype / display format= $char10. width=10 spacing=2 left "Product Type" ;
   define  product / display format= $char10. width=10 spacing=2 left "Product" ;
   define  quarter / sum format= 8. width=8 spacing=2 right "Quarter" ;
   define  year / sum format= 4. width=4 spacing=2 right "Year" ;
   define  month / sum format= monname3. width=3 spacing=2 right "Month" ;
run;
```

Figure 2. PROC REPORT in the SAS Log Produced by Using the LIST Option in the PROC REPORT Statement

The LIST option is very useful for debugging, especially when using macros. It shows defaults that might not have been specified in the original code. Using the LIST option is also a quick way to produce code that can be copied and pasted into the SAS editor for further modification. Keep in mind that the LIST option excludes SAS system options, statements that are not specific to PROC REPORT such as the WHERE or TITLE statements, and the following PROC REPORT statement options: LIST, OUT=, OUTREPT=, PROFILE=, REPORT=, WINDOWS, and NOWINDOWS.

Thus far, the sample code has suited your needs, much like the work that is produced by the apprentice carpenter. However, as your needs change, you might need to acquire coding skills that are beyond your current skill level. What you now want is customized output. To achieve this goal, you need to evolve as a carpenter's apprentice to a master woodworker.

THE MASTER WOODWORKER

Master woodworkers can complete the same tasks as the average carpenter. However, what distinguishes them as master-level craftsmen is that their completed projects reveal artistry and imagination that typically is not seen with routine carpentry. They can envision the final product, which comes only from experience and knowing what can be accomplished. With this knowledge and experience, a master woodworker can create more elaborate projects that require more skill and use more advanced tools.

In order to account for potential pitfalls and minimize errors, master woodworkers always work from plans that they carefully design. These plans typically include drawings of the final product, materials lists, and step-by-step instructions.

CREATING A CODING PLAN

Before you can create customized output, you need to consider several factors:

- What do you want the output to look like?
- Who will use the final product and for what purpose?
- Where is the data that will be used as input for PROC REPORT?

Once you have the answers to these questions, you can then formulate a plan that will guide you through each part of the building process to produce the final customized product.

WHAT DO YOU NEED TO GET STARTED?

If you do not already have an example of the final output that you would like to create, then take the time to sketch a sample structure of the final output. Keep in mind that it is difficult to make a plan if you do not know what you want or what you need to produce. The sample output sketch is what you will use as a template or a guide to build upon. Be sure to put in as much detail as possible at this step. The more detail you have, the easier it will be to create a code plan. When creating the sketch of the sample structure, be sure to include your company or customer report standards for titles, logos, and footnotes.

Once you have a sketch of the sample structure, think about who is going to be using your final output and what they expect to do with the information. If your goal is to produce a report for management or a customer, you might want to consider routing the output to an Output Delivery System (ODS) destination such as HTML, PDF, RTF, Microsoft Excel, and many others. Compared to using LISTING output, ODS gives you more flexibility and control over the look of your output.

Finally, you need to consider the input data. If the data that you need is not in the input data set, you will need to determine if the values can be calculated. If the values cannot be calculated within PROC REPORT, then you will

need to preprocess your data so that all the necessary variable values are in the same data set. Also, because PROC REPORT is a reporting tool that creates columns, you will need to look at how the input data is structured. This means that you cannot stack the variables within a column and that you might need to reshape your data before coding with PROC REPORT.

To illustrate this process, consider this example. Suppose that you are employed by a specialty woodworking company that provides custom-made furniture to customers in the U.S.A, Canada, and Germany. You have been asked to provide a regional report that spans three years of total actual sales, predicted sales, and the differences between the two. Senior management and sales analysts will use the report to determine a future pricing model. ODS HTML is chosen as the output destination because the final report will be published on the company Web site. Reports that are published on the company Web site use the color scheme that is found in the sketch style that is supplied with your SAS software. A requested feature of the final report is the ability to link to a monthly report of actual sales, predicted sales, and the difference for each product type within each division for the region. Another requested feature is to highlight total differences that are either negative for a three-year period or over the target of $2,300 for a three-year span.

Given the requirements, you envision the following report structure as shown in Figure 3:

Actual Sales and Predicted Sales Difference for 2008 - 2010 (2010 sales are estimated)

Country: CANADA, EAST Region													
		Sales Year											
		2008			2009			2010 *Estimate*			Total Sales		
Product Type	Product	Actual	Predict	Diff	Actual	Predict	Diff	Actual	Predict	Diff	Actual	Predict	Diff
Division Name: CONSUMER													
FURNITURE	BED	$7,142	$5,216	$1,926	$6,477	$6,372	$105	$6,736	$6,736	$0	$20,355	$18,324	$2,031
	SOFA	$5,341	$5,865	$-524	$6,413	$6,486	$-73	$6,670	$6,670	$0	$18,424	$19,021	$-597*
* Total Actual Sales of CONSUMER SOFA FURNITURE less than Total Projected Sales - Projections too high													
FURNITURE	Total	$12,483	$11,081	$1,402	$12,890	$12,858	$32	$13,406	$13,406	$0	$38,779	$37,345	$1,434
Division Name: EDUCATION													
FURNITURE	BED	$6,660	$6,140	$520	$6,917	$5,071	$1,846	$7,194	$7,194	$0	$20,771	$18,405	$2,366**
** Total Actual Sales of $2,300 over Total Projected Sales for EDUCATION BED FURNITURE - Great Job													
	SOFA	$7,807	$6,832	$975	$4,563	$5,104	$-541	$4,746	$4,746	$0	$17,116	$16,682	$434
FURNITURE	Total	$14,467	$12,972	$1,495	$11,480	$10,175	$1,305	$11,939	$11,939	$0	$37,886	$35,086	$2,800
EAST Total		$26,950	$24,053	$2,897	$24,370	$23,033	$1,337	$25,345	$25,345	$0	$76,665	$72,431	$4,234

Figure 3. A Sample of the Final Output Report

Notice that the output reports that are shown in Figures 1 and 3 are created from the same input data set. However, the final output report that is illustrated in Figure 3 shows a better picture of the actual and predicted sales than the output report in Figure 1.

For this example, the majority of the input data for the final report is included in the SASHELP.SALEPRD data set[1]. The DIFF variable value, which is calculated by subtracting the PREDICT variable value from the ACTUAL variable value, will be created within the PROC REPORT code. You can also preprocess the data set to calculate the differences. The YEAR variable with the values of **`1993`**, **`1994`**, and **`1995`** will need to be mapped to the year labels of **`2008`**, **`2009`**, and **`2010`** by using a format. The YEAR variable does not contain information for the value **`1995`**, which will be mapped to the **`2010`** year label. The data for **`1995`**, which is mapped to the **`2010`** year label, will need to be calculated based on a 4% increase of the **`1994`** ACTUAL variable value, which is mapped to the **`2009`** year label. The years will need to span across the columns. PROC REPORT allows this type of transposing of a variable. Alternatively, you can reshape the data so that each **`Actual`** and **`Predict`** column for each year is a separate variable. Finally, there is a link for a detailed HTML report for each product within a region for each month. For the links to work correctly, you will need to create these detailed HTML reports before you run the final code from the coding plan. For this paper, assume that the detailed HTML reports already exist.

HOW DO YOU PUT THE PIECES TOGETHER?

As you create your coding plan, it is important to remember that PROC REPORT processes data in a left-to-right, top-to-bottom direction when building reports. The remainder of this paper takes you through each part of the final output report to determine the code that you can use to create the final product. Where applicable, the discussion addresses common pitfalls and alternative ways to code.

[1] This data set is supplied by SAS with your SAS software. You can access this data from the SASHELP library.

Step 1. Build the COLUMN Statement

When you consider the header portion of the final report (Figure 4), you can determine which variables are needed from the input data set and their placement in the COLUMN statement.

Country: CANADA, EAST Region													
		Sales Year											
		2008			2009			2010 *Estimate*			Total Sales		
Product Type	Product	Actual	Predict	Diff	Actual	Predict	Diff	Actual	Predict	Diff	Actual	Predict	Diff

Figure 4. The Header Portion of the Sample Final Output Report

The COUNTRY and REGION variable labels and values are at the top of Figure 4. These variables are the first two variables that are listed in the COLUMN statement. If you look at the columns from left to right in Figure 4, you see **`Product Type`** and **`Product`**. Under each year, there are columns with these labels: **`Actual`**; **`Predict`**; and **`Diff`**. Under the spanned header **`Total Sales`**, there are columns with these labels: **`Actual`**, **`Predict`**, and **`Diff`**.

The values of the YEAR variable span over three columns each. All the years that you need for the final output report are in the variable called YEAR, with the exception of the value labeled **`*Estimate*`**. You can either transpose or reshape the data set so that for each year you have the variables of ACTUAL and PREDICT or you can use PROC REPORT to make the transposition by defining YEAR as an ACROSS variable. You can also precalculate the **`Diff`** variables values or choose to have these values calculated within a PROC REPORT COMPUTE block. Although the output report will look the same, there are differences in coding.

If you choose to transpose the data set before running PROC REPORT, then you do not have to use an ACROSS variable. Any computation will then enable you to use the variable name or alias rather than the column number in the form of _C***n***_ where ***n*** represents the column number. If you need to create an output data set from PROC REPORT, your output data set will contain the actual column name instead of the column number in the form of _C***n***_. This method does require preprocessing the data.

On the other hand, you can define YEAR as an ACROSS variable to enable PROC REPORT to transpose the data. In the COLUMN statement, the ACROSS variable is usually followed by a comma, and multiple variables that are beneath the ACROSS value are in parentheses. If a comma is not included after the ACROSS variable, the N statistic will be used for the column values. With this method, any column under an ACROSS variable has to be referenced in a COMPUTE block by the column number in the form of _C***n***_. Any NOZERO or NOPRINT variable has to be accounted for when counting the columns for ***n*** in _C***n***_.

In the COLUMN statement, you can add the ACTUAL, PREDICT, and DIFF variables within parentheses after the ACROSS variable YEAR and a comma. Each year value now becomes a column header that spans over the ACTUAL, PREDICT, and DIFF variables that are associated with that year value. The ACTUAL, PREDICT, and DIFF2 variables that are under the spanned header of **`Total Sales`** are also be added to the COLUMN statement. The resulting COLUMN statement looks like this:

```
column country region prodtype product year,(actual predict diff) actual predict
diff2;
```

However, are you missing any other variables that you need in the report code in the COLUMN statement? Are there any underlying variables that are not shown in the final report but are necessary for calculations, conditional statements, or for ordering the output data? For this final output report, you do not have any such variables. If you did, you would need to be aware of the variable placement in the COLUMN statement. You also would need to make sure that the DEFINE statement contained a NOPRINT option in order to keep the value from printing on the output report. Keep in mind that **any** variable that is from the input data set and that you need to use in the PROC REPORT code has to be in the COLUMN statement. Look at the body of the report for any other variable values that are not yet included in the COLUMN statement. Figure 5 shows a line above each section of data that contains the division information:

Country: CANADA, EAST Region													
		Sales Year											
		2008			2009			2010 *Estimate*			Total Sales		
Product Type	Product	Actual	Predict	Diff	Actual	Predict	Diff	Actual	Predict	Diff	Actual	Predict	Diff
Division Name: CONSUMER													
FURNITURE	BED	$7,142	$5,216	$1,926	$6,477	$6,372	$105	$6,736	$6,736	$0	$20,355	$18,324	$2,031
	SOFA	$5,341	$5,865	$-524	$6,413	$6,486	$-73	$6,670	$6,670	$0	$18,424	$19,021	$-597*

Figure 5. Report Row Containing the Division Value above the Summary Data

Because all the variables except for COUNTRY and REGION are within a specific division, you need to add DIVISION to the left of all the variables and to the right of COUNTRY and REGION. The COLUMN statement looks like this:

```
column country region division prodtype product year,(actual predict diff) actual
predict diff2;
```

In the COLUMN statement, there is a duplication of the ACTUAL and PREDICT variables. This is because you want to see the actual and predicted values for each year and then a total of the actual and predicted values. You could calculate the totals by adding up the actual and predicted values for every year but, because you already have the data, it is better to avoid the calculation, especially when using ACROSS variable columns.

If you do not assign an alias to the duplicated variables in the COLUMN statement, PROC REPORT will do so behind the scenes. Adding your own alias will give you better control in the rest of the report code. You can choose any of the duplicate variable names in which to add the alias. The alias label can be any valid variable name. In this sample code, you are choosing the first set of ACTUAL and PREDICT variables to add the alias. The COLUMN statement looks like this:

```
column country region division prodtype product year,(actual=actual1
predict=predict1 diff) actual predict diff2;
```

Now look at the header portion of the report to see if there is anything else you need to add to the COLUMN statement.

Figure 6 shows text that spans multiple columns:

Country: CANADA, EAST Region													
		Sales Year											
		2008			2009			2010 *Estimate*			Total Sales		
Product Type	Product	Actual	Predict	Diff	Actual	Predict	Diff	Actual	Predict	Diff	Actual	Predict	Diff

Figure 6. First Four Rows of the Column Header Showing Spanned Headers on the Second and Third Row

You can create text that spans multiple columns above the column headers by using a LINE statement, an ACROSS variable, a column header label in a DEFINE statement, or a spanned header. Figure 6, shows the first four rows of the final report output with text spanning multiple columns. However, the spanning texts are created differently.

Spanning text created by a LINE statement within a COMPUTE BEFORE _PAGE_ statement can show different values for every page. The value for the ACROSS variable can be changed by a spanned header or with a column header label in the DEFINE statement. The spanning text values under the ACROSS variable can be changed with a FORMAT=option in the DEFINE statement or a FORMAT statement. All other spanning text can be created with a spanned header.

The label **`Total Sales`** is created by the spanned header and needs to be added to the COLUMN statement. The COLUMN statement looks like this:

```
column country region division prodtype product year,(actual=actual1
predict=predict1 diff) ('Total Sales' actual predict diff2);
```

Step 2. Create the DEFINE Statements

You now have the first part of your coding plan, which defines the order of the columns in the final report. Next you need to decide how to define the variables that are listed in the COLUMN statement. In a SAS editor, if you add the COLUMN statement from your coding plan after a PROC REPORT statement that contains the LIST option and DATA=SASHELP.PRDSALE, the default PROC REPORT code is produced in the log. The code will look like this:

```
proc report data=sashelp.prdsale list ;
column country region division prodtype product year,( actual=actual1
predict=predict1 diff ) ("Total Sales"  actual predict diff2 ) ;
```

If you run the preceding code, the SAS log shows the default code and an error. The error is produced because you have not yet defined year as an ACROSS variable even though you have identified it as an ACROSS variable in the COLUMN statement by adding a comma. The error message looks like this in the SAS log.

```
ERROR: There is more than one ANALYSIS usage associated with
       the column defined by the following elements.
       Name                            Usage
       -------------------------------  --------
       YEAR                            ANALYSIS
       ACTUAL                          ANALYSIS
```

The default code in the SAS log shows the DEFINE statements, as shown in Figure 7.

```
define  country / display format= $char10. width=10 spacing=2 left "Country" ;
define  region / display format= $char10. width=10 spacing=2 left "Region" ;
define  division / display format= $char10. width=10 spacing=2 left "Division" ;
define  prodtype / display format= $char10. width=10 spacing=2 left "Product Type" ;
define  product / display format= $char10. width=10 spacing=2 left "Product" ;
define  year / sum format= 4. width=4 spacing=2 "Year" ;
define  actual1 / sum format= dollar12.2 width=12 spacing=2 "Actual" ;
define  predict1 / sum format= dollar12.2 width=12 spacing=2 "Predict" ;
define  diff / computed format= best9. width=9 spacing=2 "Diff" ;
define  actual / sum format= dollar12.2 width=12 spacing=2 right "Actual Sales" ;
define  predict / sum format= dollar12.2 width=12 spacing=2 right "Predicted Sales" ;
define  diff2 / computed format= best9. width=9 spacing=2 right "Diff2" ;
```

Figure 7. DEFINE Statements Produced from the LIST Option in the PROC REPORT Statement

Notice that YEAR by default is defined as SUM. Unless otherwise specified, by default, all numeric variables are defined as SUM and all character variables are defined as DISPLAY. The error message will be eliminated in the SAS log once the YEAR variable is defined as an ACROSS variable.

Because you are creating a summary report, you need to define all of your character variables as GROUP. If there are any character variables that are defined as DISPLAY or ORDER, PROC REPORT will not consolidate the rows of detail data and will treat the GROUP as ORDER. Also, if there are both DISPLAY and GROUP defined variables, a note is produced. For example, for the data set SASHEP.PRDSALW, if all the character variables are defined as GROUP except DIVISION remains as a DISPLAY variable, a note such as the following occurs in the SAS log:

```
NOTE: Groups are not created because the usage of DIVISION is DISPLAY.
```

To decide how to define each variable, you need to look at the sample of the final output report and coding plan in the COLUMN statement (Figure 8) and the DEFINE statements from the SAS log that are produced by the LIST option, which is shown in Figure 7.

```
column country region division prodtype product year,(actual=actual1
predict=predict1 diff) (‘Total Sales’ actual predict diff2);
```

Country: CANADA, EAST Region													
		Sales Year											
		2008			2009			2010 *Estimate*			Total Sales		
Product Type	Product	Actual	Predict	Diff	Actual	Predict	Diff	Actual	Predict	Diff	Actual	Predict	Diff
Division Name: CONSUMER													
FURNITURE	BED	$7,142	$5,216	$1,926	$6,477	$6,372	$105	$6,736	$6,736	$0	$20,355	$18,324	$2,031
	SOFA	$5,341	$5,865	$-524	$6,413	$6,486	$-73	$6,670	$6,670	$0	$18,424	$19,021	$-597*
* Total Actual Sales of CONSUMER SOFA FURNITURE less than Total Projected Sales - Projections too high													
FURNITURE	Total	$12,483	$11,081	$1,402	$12,890	$12,858	$32	$13,406	$13,406	$0	$38,779	$37,345	$1,434
Division Name: EDUCATION													
FURNITURE	BED	$6,660	$6,140	$520	$6,917	$5,071	$1,846	$7,194	$7,194	$0	$20,771	$18,405	$2,366**
** Total Actual Sales of $2,300 over Total Projected Sales for EDUCATION BED FURNITURE - Great Job													
	SOFA	$7,807	$6,832	$975	$4,563	$5,104	$-541	$4,746	$4,746	$0	$17,116	$16,682	$434
FURNITURE	Total	$14,467	$12,972	$1,495	$11,480	$10,175	$1,305	$11,939	$11,939	$0	$37,886	$35,086	$2,800
EAST Total		$26,950	$24,053	$2,897	$24,370	$23,033	$1,337	$25,345	$25,345	$0	$76,665	$72,431	$4,234

Figure 8. A Sample of the COLUMN Statement and Final Output Report as Shown in Figure 3

The first three variables in the COLUMN statement are not columns in the final output report. The values from the variable are spanned across the report in different places. Because you do not need a column for these three variables, you need to add the NOPRINT option to the DEFINE statement. All the character variables with the default value of DISPLAY need to be defined as GROUP. The DEFINE statements for the character variables that are shown in Figure 7 can be modified by changing the DISPLAY to GROUP. Here is the modified DEFINE statement:

```
define country / group format= $char10. width=10 spacing=2 left "Country" noprint;
define region / group format= $char10. width=10 spacing=2 left "Region" noprint ;
define division /group format= $char10. width=10 spacing=2 left "Division" noprint;
define  prodtype / group format= $char10. width=10 spacing=2 left "Product Type" ;
define  product / group format= $char10. width=10 spacing=2   left "Product" ;
```

In Figure 7, notice that the numeric variables are defined as SUM. Also notice that the alias variable labels of **`actual1`** and **`predict1`** take on the same characteristics as the original values for ACTUAL and PREDICT including the header labels. Because you have a spanned header of **`Total Sales`**, you can remove the word **`sales`** from the header label for ACTUAL and PREDICT. The header label for PREDICT is changed from **`Predicted`** to **`Predict`**. The SUM statistic can be changed to any valid PROC REPORT statistic. For this report, we want to use the default statistic of SUM. Keep in mind that if you use a different statistic and need to reference the variable in a COMPUTE block, the statistic will need to be used instead of SUM. The only other modification that you need to make to these DEFINE statements is to change the format from DOLLAR12.2 to DOLLAR10 in the FORMAT= option as follows:

```
define  actual / sum format= dollar10. width=12 spacing=2 right "Actual" ;
define  predict / sum format= dollar10. width=12 spacing=2 right "Predict" ;
define  actual1 / sum format= dollar10. width=12 spacing=2 right "Actual" ;
define  predict1 / sum format= dollar10. width=12 spacing=2 right "Predict" ;
```

In the COLUMN statement, you added variable names of DIFF and DIFF2 because you want to have a calculated value. In the DEFINE statements in Figure 7, both DIFF and DIFF2 variables are defined as COMPUTED. PROC REPORT automatically defines any variable that is not from the input data set as COMPUTED. As a default, a COMPUTED type variable is assigned as numeric. For the purposes of this example, you want the COMPUTED type variable to be numeric. If you want the type to be character, you need to change the default format to a valid character format. Also, in the COMPUTE block for a character COMPUTED type variable, you need to add the /CHARACTER or CHAR options at the end of the COMPUTE statement. In the sample of the final report output in Figure 8, DIFF and DIFF2 columns are labeled **`Diff`**. You can modify the header labels in the DEFINE statement to **`Diff`** and change the format from Best9. to Dollar10. in the FORMAT= option as follows:

```
define  diff / computed format= dollar10. width=9 spacing=2 right "Diff" ;
define  diff2 / computed format= dollar10. width=9 spacing=2 right "Diff" ;
```

The remaining variable that needs to be defined is YEAR. The values for YEAR will span the columns of ACTUAL, PREDICT and DIFF. To span the variable YEAR value over other columns, define YEAR as an ACROSS variable. Look back at the sample final output for YEAR in Figure 8. There are two issues that you need to address. First, the input data for year contains the values **`1993`** and **`1994`**, which need to be mapped to the format labels of **`2008`** and

2009, respectively. The input data set does not contain the value of **1995**. You know that you need to estimate the ACTUAL and PREDICT columns under the **1995** column and that you need to map the **1995** column to the **2010** format label.

You could preprocess the input data set to add estimated actual and predicted values for **1995** and change the values of **1993**, **1994**, and **1995** to the format labels of **2008**, **2009**, and **2010**. As an alternative, you could take advantage of using the PRELOADFMT option and add the missing **1995** year value through a format. With the format, you can map the value that you need without having to change the input data set. You can use PROC FORMAT to create a user-defined format that will add all the start values that you need and the desired labels. The PROC FORMAT code looks like this:

```
proc format;
value yearfmt 1993='2008'
              1994='2009'
              1995='2010 *Estimate*';
```

In the DEFINE statement for the YEAR variable, you can replace FORMAT=4. with FORMAT=*yearfmt.* (*yearfmt* is a user-defined format from the preceding code). Also in the DEFINE statement for YEAR, modify the header label from **YEAR** to **Sales Year** and add the PRELOADFMT option. When you look at the sample of the final output report in Figure 8, you want the order of the ACROSS variables values to be based on the unformatted values. By default, PROC REPORT orders GROUP, ORDER, and ACROSS variables based on the formatted values. In this case, both the formatted and unformatted values render in the same order. But if you added a space to the label for **1995**, such as in **2010 *Estimate***, by default the value of **2010 *Estimate*** will print first. Adding the ORDER=INTERNAL option in the DEFINE statement will eliminate a potential ordering problem. If YEAR had been a SAS date value, using the ORDER=INTERNAL option prevents the SAS date from being ordered by the formatted date label. The modified DEFINE statement for YEAR is as follows:

```
define  year / across format=yearfmt15. order=internal width=4 spacing=2 right
        "Sales Year" preloadfmt;
```

So far you have a COLUMN statement, the DEFINE statements, and a PROC FORMAT in your coding plan. The COLUMN and DEFINE statements create the structure of the report and the PROC FORMAT code adds the year columns that are not in the input data set.

Step 3. Set up the BREAK Statements

The final report output in Figure 8, shows two types of summary rows. There is a summary row after each product type and after each region. The last two summary rows of your sample output report are shown in Figure 9:

Country: CANADA, EAST Region													
		Sales Year											
		2008			2009			2010 *Estimate*			Total Sales		
Product Type	Product	Actual	Predict	Diff	Actual	Predict	Diff	Actual	Predict	Diff	Actual	Predict	Diff
FURNITURE	Total	$14,467	$12,972	$1,495	$11,480	$10,175	$1,305	$11,939	$11,939	$0	$37,886	$35,086	$2,800
EAST Total		$26,950	$24,053	$2,897	$24,370	$23,033	$1,337	$25,345	$25,345	$0	$76,665	$72,431	$4,234

Figure 9. The Last Two Summary Rows of the Sample Output Report

The first row of the two summary rows shown in Figure 9 is created after each product type. You want to create a summary row after each product type within a division. You can achieve this with a BREAK AFTER statement on the product type variable of PRODTYPE. The break row for the product type has a different background color than the detail rows of the final output report. Changing the attributes of a cell, row, or column provides a visual way to have that portion of the report stand out. You can change the background color with the BACKGROUND= attribute within a STYLE= option in the BREAK statement. The FONT_WEIGHT=BOLD option within the STYLE= option displays the text in bold. The STYLE options are valid only for ODS destinations, not the LISTING output. The BREAK statement looks like this:

```
break after prodtype / summarize style=[font_weight=bold background=cxFFFEDB];
```

The second row of the two summary rows shown in Figure 9 is created after each region. You not only want a summary row for the region, but you also want to create a new page per region. You can add these functionalities in

a BREAK AFTER statement with the SUMMARIZE and PAGE options for the REGION variable. You can change the background to a lighter color and bold the text by using the STYLE= option in the BREAK statement as follows:

```
break after region / page summarize style=[font_weight=bold background=cxCC9966];
```

Step 4. Create the COMPUTE BEFORE _PAGE_ Statement

When you defined the COUNTRY and REGION variables, you added the NOPRINT option in the DEFINE statement because you wanted to see only the values from these variables above the column headers. PROC REPORT provides a way to add text above the column header by specifying the _PAGE_ option in a COMPUTE BEFORE statement. In this report, you want to see a mixture of text and variable values in the row above the column headers.

In a COMPUTE BEFORE _PAGE_ block, PROC REPORT uses the first detail row of data on the current page whenever a variable is referenced within the compute block. For the first page and for any new page that is created by the PAGE option, there generally is not a problem obtaining the variable values.

However, when the first detail row of a page contains a blank value for a GROUP or ORDER variable that has continued to a new page, the result is a blank value as well (or '.' for a numeric). In this case, you can use a DATA step variable that has been created in a COMPUTE block to hold the value. In the reverse situation, a COMPUTE AFTER _PAGE_ statement uses the last detail row values of the page for referenced variables.

Because you have requested a BREAK for every region value, the values for first detail row in the report are populated. If you look at the text above the column headers, Figure 10, you can see specific text and variable values that span over the column headers:

Country: CANADA, EAST Region													
		Sales Year											
		2008			2009			2010 *Estimate*			Total Sales		
Product Type	Product	Actual	Predict	Diff	Actual	Predict	Diff	Actual	Predict	Diff	Actual	Predict	Diff

Figure 10. Country and Region Text above the Column Headers

In a COMPUTE BEFORE _PAGE_ block, you need to create a DATA step variable that you can use in the LINE statement. Creating a DATA step variable, sometimes referred to as a text string, is a handy way to print out values that you determine within with code for the final output report.

Because the DATA step variable value can change, it is a good practice to use a LENGTH statement to establish the variable type and maximum length. If the DATA step variable is being used in the same COMPUTE block that it is created in, then a best practice is to add the LENGTH statement after the COMPUTE statement. A best practice for placing a LENGTH statement when the DATA step variable is used in different COMPUTE blocks is discussed later in this paper.

Keep in mind that a LINE statement will always print. There is no way to conditionally execute a LINE statement; it will always print. A DATA step variable is controlled by the code and is not added in the COLUMN statement. If it does list in the COLUMN statement, the value of the DATA step variable reinitializes at every row. The best way to keep a value over multiple rows is to create a DATA step variable.

You can create a DATA step variable using any valid SAS variable name. For this sample final output report, you can create a DATA step variable that will contain the value of the country, a comma, the region value, and a text value of **`Region`**. Using a combination of LEFT and TRIM functions ensures that there are no blanks within the DATA step variable. The LEFT function left justifies the character value. The TRIM function removes the trailing blanks from the character string. You need to add a space in the text portion of the string. In the LINE statement, you can add the text string 'Country: ' and then the DATA step variable. In Figure 10, the text string is left justified. You can add a STYLE option with the JUST=L style attribute so that the text string will be left justified. The STYLE= option in the COMPUTE statement applies to all the LINE statements within that COMPUTE block. The COMPUTE BEFORE _PAGE_ block looks like this:

```
compute before _page_/style=[just=l];
length region_text $25;
region_text=trim(left(country))||', '||trim(left(Region))||' Region';
line    ' Country: ' region_text $25.;
endcomp;
```

Step 5. Create the Detail Data for the Product Type COMPUTE Block

Figure 11 shows the detail rows for the first product type:

Country: CANADA, EAST Region													
		Sales Year											
		2008			2009			2010 *Estimate*			Total Sales		
Product Type	Product	Actual	Predict	Diff	Actual	Predict	Diff	Actual	Predict	Diff	Actual	Predict	Diff
FURNITURE	BED	$7,142	$5,216	$1,926	$6,477	$6,372	$105	$6,736	$6,736	$0	$20,355	$18,324	$2,031
	SOFA	$5,341	$5,865	$-524	$6,413	$6,486	$-73	$6,670	$6,670	$0	$18,424	$19,021	$-597*
* Total Actual Sales of CONSUMER SOFA FURNITURE less than Total Projected Sales - Projections too high													
FURNITURE	Total	$12,483	$11,081	$1,402	$12,890	$12,858	$32	$13,406	$13,406	$0	$38,779	$37,345	$1,434

Figure 11. Detail Rows for the First Product Type

One of the requirements for this final output report is to provide a hyperlink to a report with monthly sales information. The hyperlink needs to link to an HTML file that contains the product type, within a division, region, and country. The HTML file that you will link to has to be created by the time the final code is completed and run. PROC REPORT enables you to use a CALL DEFINE statement to create the link.

The CALL DEFINE statement requires three arguments. The first argument is the location. This can be a column number, a column variable name, a row, or the COMPUTE block column. The second argument is the attribute name. In this case, you are using the URL attribute name, which creates the link. The third argument is the value that the attribute name will use and place at the location in the first argument.

To create the location, you can use a DATA step variable with all the values. Keep in mind that any variable that you reference in the COMPUTE block has to be to the left of the COMPUTE block variable or PROC REPORT considers it to be missing or blank. Also keep in mind that any of the GROUP variables that are part of the DATA step text string might show as missing if the row value is blank. This is because PROC REPORT looks at the created report in memory to obtain values. Because GROUP and ORDER variables do not repeat the same value unless there is a GROUP or ORDER variable with a value located to the left on the same row, any reference will be blank.

You can obtain the GROUP value in different ways. In a COMPUTE BEFORE block, assign a DATA step variable the value of the GROUP variable. As an alternative approach, work within the current COMPUTE block by adding an IF statement to determine when the GROUP variable is populated and assign the GROUP variable value to a DATA step variable when the IF condition is **`true`**. For this sample code, all the processing is done in the current COMPUTE block to create the URL string value.

Also, the link is only at the detail level row and not at the break row. You can use the automatic variable _BREAK_ to determine when you are on a detail row. If the _BREAK_ variable value is blank, then you know you are on a detail row. If the _BREAK_ value is not blank, then _BREAK_ will be the value of the break value that created the row. For an rbreak row, the value is **_RBREAK_**. You can create an output data set by adding OUT= to the PROC REPORT statement to see the values that are contained in the _BREAK_ variable for the different break rows. This is a quick way to verify the _BREAK_ variable value when there are multiple BREAK statements.

Because you are in the **`Product Type`** column, you can also create a DATA step variable that will contain the value of the product code. The code to create the DATA step variables that are used in creating the URL string, the DATA step variable that contains the product code value, and the CALL DEFINE statement is as follows:

```
if _break_ eq ' ' then do;
  if country ne ' ' then country1=country;
  if region ne ' ' then region1=region;
  if division ne ' ' then division1=division;
  if prodtype ne ' '  then do;
     urlstring=trim(left(country1))||'_'||
                      trim(left(region1))||'_'||
                      trim(left(division1))||'_'||
                       trim(left(prodtype))||'_'||'detail.htm';
  call define('prodtype','URL',urlstring);
  hold_prodtype=prodtype;
  end;
end;
```

Look back at the output in Figure 11. There is a highlighted string that contains the value for the product type. The highlighted string also shows the value for DIVISION. The LINE statement that creates this highlighted string is in a COMPUTE AFTER block. However, the DATA step variable is assigned in a different COMPUTE block. PROC

REPORT will establish the variable type and length when completing the coded program statements within the COMPUTE variable block. You can add a LENGTH statement for the DATA step variables that you will be creating in any COMPUTE variable block. You need a LENGTH statement for the URL string that is long enough to hold the longest possible amount of text values of the DATA step variables. It is a best practice to put the LENGTH statement after the COMPUTE statement as follows:

```
compute prodtype;
length urlstring $ 75 hold_division $ 10 hold_prodtype $ 10;
```

Step 6. Develop the 2010 Estimate Data

You used the PRELOADFMT option in the DEFINE statement to create the columns under the year labeled **2010 *Estimate***. You also created columns for the difference between ACTUAL and PREDICT for each year and total. If you need to create an output data set, you would need to use a COMPUTE block for each column in which you need to add a calculation statement. For this paper, you are not creating an output data set, therefore, you can use the last variable in the COLUMN statement to add your calculations.

Using the last variable in the COLUMN statement to put in all your calculations is helpful when there are columns under an ACROSS variable. For columns under an ACROSS variable, PROC REPORT has to reprocess all the columns under the ACROSS variable until all the values are entered. When there are accumulation calculations, this reprocessing can cause the values to increase by the number of times the column has been reprocessed. PROC REPORT has no problem moving to the left and replacing values in the output report. By using the last variable in the COLUMN statement as the COMPUTE block variable, the value under ACROSS is replaced once.

In this coding plan, you will use the compute block for DIFF2 for the calculations. When referencing any column that is under an ACROSS variable in a COMPUTE block, the column number has to be used in the form of _C***n***_. You need to calculate the difference between the ACTUAL and PREDICT variable values. To find the column number of the first DIFF under the label of first year value, you need to look at the COLUMN statement as follows:

```
column country region division prodtype product year,(actual=actual1
predict=predict1 diff) ('Total Sales' actual predict diff2);
```

Even though COUNTRY, REGION, and DIVISON are not printed columns in the output report, the columns still exist in memory. The NOPRINT and NOZERO options in the DEFINE statement are added right before the report is released from memory to the output destination or LISTING output. The SHOWALL option can be used in the PROC REPORT statement to show all the columns, even the variables with NOPRINT and NOZERO options in the DEFINE statement.

In the COLUMN statement, the first value of the DIFF variable is in column 8, the next value of the DIFF variable is in column 11, and the last value of the DIFF variable is in column 14. Keep in mind that there is no way to add just one calculation that will apply to all the ACROSS variable columns. Each column under an ACROSS variable requires its own calculation statement. You also need to calculate the **actual** and **predict** values for the year label of the **2010 *Estimate***. You know that you want to increase the **2010 *Estimate*** column by 4% using the **2009** ACTUAL variable value. You also know that until the real data is available, the ACTUAL and PREDICT values for the **2010 *Estimate*** will be the same value. The calculations for the columns under ACROSS look like this:

```
compute diff2;
_c8_=_c6_-_c7_;
_c11_=_c9_-_c10_;
_c12_=_c9_*1.04;
_c13_=_c12_;
_c14_=_c12_-_c13_;
```

The totals for the ACTUAL and PREDICT variables are the totals for the values from the input data set. Because you added estimated ACTUAL and PREDICT values for the **2010 *Estimate*** values, you need to also add the **2010 *Estimate*** values to the **Total Sales** ACTUAL and PREDICT values. You can then add a statement using the variable names in the calculations to get the difference in the totals for the DIFF2 variable. The calculations that are added to the COMPUTE DIFF2 block look like this:

```
actual.sum=actual.sum+_c12_;
predict.sum=predict.sum+_c13_;
diff2=actual.sum-predict.sum;
```

Look again at the detail rows from the sample final report output in Figure 12:

Country: CANADA, EAST Region													
		Sales Year											
		2008			2009			2010 *Estimate*			Total Sales		
Product Type	Product	Actual	Predict	Diff	Actual	Predict	Diff	Actual	Predict	Diff	Actual	Predict	Diff
Division Name: CONSUMER													
FURNITURE	BED	$7,142	$5,216	$1,926	$6,477	$6,372	$105	$6,736	$6,736	$0	$20,355	$18,324	$2,031
	SOFA	$5,341	$5,865	$-524	$6,413	$6,486	$-73	$6,670	$6,670	$0	$18,424	$19,021	$-597*
* Total Actual Sales of CONSUMER SOFA FURNITURE less than Total Projected Sales - Projections too high													
FURNITURE	Total	$12,483	$11,081	$1,402	$12,890	$12,858	$32	$13,406	$13,406	$0	$38,779	$37,345	$1,434
Division Name: EDUCATION													
FURNITURE	BED	$6,660	$6,140	$520	$6,917	$5,071	$1,846	$7,194	$7,194	$0	$20,771	$18,405	$2,366**
** Total Actual Sales of $2,300 over Total Projected Sales for EDUCATION BED FURNITURE - Great Job													

Figure 12. A Sample of the Detail Rows

Some of the differences are being highlighted under the column labeled **`Diff`** for **`Total Sales`**. You can use IF statements to determine when the **`Diff`** column values under the **`Total Sales`** column header are less than **`0`** or greater than **`2300`**. When the IF statement is **`true`**, you can use a CALL DEFINE statement to highlight the cell, change the foreground color, and add a pretext value. This time, instead of using the URL as the attribute name, you use the STYLE as the attribute name. The IF and CALL DEFINE statements look like this:

```
if diff2 < 0 and upcase(_break_) eq ' ' then do;
  call define('diff2','style',
      'style=[foreground=red background=cxFFFFCC posttext="*"] ');
end;
if diff2 > 2300 and upcase(_break_) eq ' ' then do;
  call define('diff2','style',
       'style=[foreground=green background=cxFFFFCC  posttext="**"] ');
end;
```

Step 7. Add the Division Name

In the sample output report before the detail data for the PRODUCT TYPE value per division, there is a spanning row that contains the DIVISION variable value and descriptive text (Figure 13).

Country: CANADA, EAST Region													
		Sales Year											
		2008			2009			2010 *Estimate*			Total Sales		
Product Type	Product	Actual	Predict	Diff	Actual	Predict	Diff	Actual	Predict	Diff	Actual	Predict	Diff
Division Name: CONSUMER													
FURNITURE	BED	$7,142	$5,216	$1,926	$6,477	$6,372	$105	$6,736	$6,736	$0	$20,355	$18,324	$2,031
	SOFA	$5,341	$5,865	$-524	$6,413	$6,486	$-73	$6,670	$6,670	$0	$18,424	$19,021	$-597*
* Total Actual Sales of CONSUMER SOFA FURNITURE less than Total Projected Sales - Projections too high													
FURNITURE	Total	$12,483	$11,081	$1,402	$12,890	$12,858	$32	$13,406	$13,406	$0	$38,779	$37,345	$1,434
Division Name: EDUCATION													
FURNITURE	BED	$6,660	$6,140	$520	$6,917	$5,071	$1,846	$7,194	$7,194	$0	$20,771	$18,405	$2,366**
** Total Actual Sales of $2,300 over Total Projected Sales for EDUCATION BED FURNITURE - Great Job													

Figure 13. Output Showing the DIVISION Variable Value and Descriptive Text

You can use a COMPUTE BEFORE division block to create the text string. You are interested in only one variable value; there is no need to create a DATA step text string because you can put all of the values in the LINE statement. In the division row output, the text string is highlighted. You can create this effect by using inline formatting syntax.

To use inline formatting, you must first declare the ODS ESCAPECHAR character value that you intend to use. Keep in mind that the default for the ESCAPECHAR value is the same value as the default PROC REPORT SPLIT='/' value. If you intend to use inline formatting, then choose an ESCAPECHAR value that typically would not be included

in your data. The carat (^) and the tilde (~) characters are good values to use. To declare the ODS ESCAPECHAR character value, add a statement like this before any ODS destination statement:

```
ods escapechar='^';
```

You can add inline formatting syntax to the text that you are creating in the LINE statement. For example, you can use ^*S={style-attribute}*. Only the curly style of left and right brackets is used with inline formatting syntax, not square style brackets. To revert back to the default style attributes, add ^S{} at the position where you want to end the inline formatting.

If a COMPUTE BEFORE GROUP or ORDER variable is used, the value of that GROUP or ORDER variable is always available. This is another place where you can create a value for a DATA step variable to hold the value of **`Division`**. Here is the resulting COMPUTE BEFORE DIVISION code:

```
compute before division;
line @1 "^S={background= cxFFFEDB} Division Name: " division $10.;
hold_division=division;
endcomp;
```

Step 8. Create the Total Text in the Region and Product Type Break Rows

Figure 14 shows the text of **`Total`** added to the break row for the PRODTYPE and REGION variables. For the PRODTYPE variable labeled as **`Product Type`**, the **`Total`** text is displayed in the **`Product`** column. For the **`Region`** break row, the **`Total`** text is added to the **`Region`** value and placed in the **`Product Type`** column.

Country: CANADA, EAST Region													
		Sales Year											
		2008			2009			2010 *Estimate*			Total Sales		
Product Type	Product	Actual	Predict	Diff	Actual	Predict	Diff	Actual	Predict	Diff	Actual	Predict	Diff
FURNITURE	Total	$14,467	$12,972	$1,495	$11,480	$10,175	$1,305	$11,939	$11,939	$0	$37,886	$35,086	$2,800
EAST Total		$26,950	$24,053	$2,897	$24,370	$23,033	$1,337	$25,345	$25,345	$0	$76,665	$72,431	$4,234

Figure 14. `Product Type` and `Region` Rows with Added `Total` Text

In a COMPUTE AFTER PRODTYPE block, use an IF statement to evaluate the automatic variable _BREAK_ for a value. When the IF statement is **`true`**, then assign the cell in the **`Product`** column the value of **`Total`** as follows:

```
compute after prodtype;
if _break_ ne ' ' then product='Total';
endcomp;
```

In a COMPUTE AFTER REGION block, use an IF statement to evaluate the _BREAK_ automatic variable for a value. When the IF statement is **`true`**, then assign PRODTYPE the value of **`Region`** concatenated with the text of **`Total`** as follows:

```
compute after region;
if _break_ ne ' ' then do;
prodtype=trim(left(region))||' Total';;
end;
endcomp;
```

Step 9. Add Text to Explain the Total Difference Cell Highlighting

Figure 15 shows a line of text that explains the highlighted difference value from the previous row:

Country: CANADA, EAST Region													
		Sales Year											
		2008			2009			2010 *Estimate*			Total Sales		
Product Type	Product	Actual	Predict	Diff	Actual	Predict	Diff	Actual	Predict	Diff	Actual	Predict	Diff
Division Name: CONSUMER													
FURNITURE	BED	$7,142	$5,216	$1,926	$6,477	$6,372	$105	$6,736	$6,736	$0	$20,355	$18,324	$2,031
	SOFA	$5,341	$5,865	$-524	$6,413	$6,486	$-73	$6,670	$6,670	$0	$18,424	$19,021	$-597*
* Total Actual Sales of CONSUMER SOFA FURNITURE less than Total Projected Sales - Projections too high													
FURNITURE	Total	$12,483	$11,081	$1,402	$12,890	$12,858	$32	$13,406	$13,406	$0	$38,779	$37,345	$1,434
Division Name: EDUCATION													
FURNITURE	BED	$6,660	$6,140	$520	$6,917	$5,071	$1,846	$7,194	$7,194	$0	$20,771	$18,405	$2,366**
** Total Actual Sales of $2,300 over Total Projected Sales for EDUCATION BED FURNITURE - Great Job													

Figure 15. A Sample of the Detail Rows

The red text indicates a higher total-predicted value over the total actual value for a particular product row. The green text indicates a higher total-actual value over a total predicted value that is greater than $2,300.

Both of the text rows are created by a LINE statement. As discussed earlier in this paper, a LINE statement always prints. But when the format for the entire line statement is **0**, the line statement does not appear in the output report, even though it is in memory. Only the LINE statement behaves this way.

You can use the SAS format $VARYING, which requires you to add a LENGTH statement. In the COMPUTE AFTER block for PRODUCT, you add a LENGTH statement for the text variable that is created to hold the appropriate text value. You also need to initialize the text to a blank and the DATA step variable NUM to **0**. The NUM DATA step variable contains the length value for the $VARYING format.

In an IF condition, you can evaluate the value of **Diff** under the **Total Sales** column header and whether the row that you are currently on is from the _BREAK_ row created from the COMPUTE AFTER PRODUCT statement. Because you are in a COMPUTE AFTER PRODUCT block, the break row is created in memory but is not shown in the report, because you do not have a BREAK AFTER PRODUCT statement. You do have access to any value that would have been printed on the break row.

When the IF condition is **true**, you can create the text string for the DATA step variable that you named in the LENGTH statement. You can create the text string by concatenating both text and DATA step variables that you created previously in other COMPUTE blocks. Inline formatting is added to the text string to achieve the highlighted background, change the foreground text color, and display the text in bold. You also set the DATA step variable of NUM to **250**, the value that is used in the LENGTH statement.

In the LINE statement, the DATA step variable that contains the text string is listed with the SAS format $VARYING and the NUM DATA step variable. On the COMPUTE AFTER PRODUCT statement, you change the background, the justification, and the cell height for the LINE statement. The code looks like this:

```
compute after product/style=[just=c cellheight=6.5pt background=cxFCCF7B];
length text $ 250;
num=0;
text=' ';
if diff2 < 0 and upcase(_break_) eq 'PRODUCT' then do;
  text='^S={foreground=red background=cxFFFFCC}* Total Actual Sales of '||
       trim(left(hold_division))||' '||trim(left(product))
       ||' '||trim(left(hold_prodtype))||
       ' less than Total Projected Sales - Projections too high';
num=250;
end;
if diff2 > 2300 and upcase(_break_) eq 'PRODUCT' then do;
   text='^S={foreground=green background=cxFFFFCC font_weight=bold}
        ** Total Actual Sales of $2,300 over Total Projected Sales for '
        ||trim(left(hold_division))||' '||trim(left(product))||' '||
        trim(left(hold_prodtype))||' - Great Job';
   num=250;
end;
line text $varying. num;
endcomp;
```

Step 10. Tidy the Loose Ends

You are almost finished creating your coding plan, but you still need to add the most important statement of all—the PROC REPORT statement. From the SAS log with the PROC REPORT code from the LIST option, you have this PROC REPORT statement:

```
proc report data=sashelp.prdsale ls=120 ps=55  split="/" center ;
```

You need to add the NOWINDOWS option, which is typically referred to as NOWD, in the PROC REPORT statement. If NOWD is not included in the PROC REPORT statement, the PROC REPORT window opens. This window shows an instant report output. However, because the report window does not support ODS destinations, it is a best practice to add the NOWD option, even if code is run in batch.

In the PROC REPORT statement, change the CENTER option to NOCENTER to left justify the report output. Any style option that is added in the PROC REPORT statement is applied to the location that is specified. In the STYLE(LINES)=, STYLE(HEADER)=, and STYLE(COLUMN)= options, you want to modify the background colors. To ensure that all the observations in the data set are processed, you need to add the MISSING option. Without the MISSING option, any GROUP, ORDER, or ACROSS variable that contains a missing or a blank value is excluded from the report output. The modified PROC REPORT statement looks like this:

```
proc report data=sashelp.prdsale split="/" nocenter nowd  missing
style(report column summary calldef)=[font_size=6.5pt cellpadding=1 borderwidth=1]
style(lines)=[background=cxfccf7b]
style(header)=[background=cxfffedb ]
style(column)=[background=cxfccf7f]
;
```

You can add the STYLE=SKETCH option to the ODS HTML statement to create the final output in the desired style.

```
ods html  file='c:\actual_predicted_sales.htm' style= sketch;
```

The last addition to the coding plan is the TITLE statement as follows:

```
title 'Actual Sales and Predicted Sales Difference for 2008 - 2010.  (2010 sales
are estimated)';
```

Step 11. Assemble the Final Product

Now that you have created all the pieces for your customized output, you need to put them all together. Here is the final coding plan that creates the customized output table that is shown in Figure 3:

```
ods escapechar='^';
ods html  file='c:\actual_predicted_sales.htm' style= sketch;
proc format;
value yearfmt 1993='2008'
              1994='2009'
              1995='2010 *estimate*';
run;
title 'Actual Sales and Predicted Sales Difference for 2008 - 2010.  (2010 sales
       are estimated)';
proc report data=sashelp.prdsale ls=256 ps=42  split="/" nocenter nowd  missing
style(report column summary calldef)=[font_size=6.5pt cellpadding=1 borderwidth=1]
style(lines)=[background=cxfccf7b font_size=6.5pt cellpadding=1 ]
style(header)=[font_size=6.5pt cellpadding=1 background=cxfffedb ]
style(column)=[font_size=6.5pt cellpadding=1 background=cxfccf7f];
column country  region  division prodtype  product  year,(actual=actual1
       predict=predict1  diff)
 ('Total Sales' actual predict diff2);
define  actual / sum format= dollar10. width=12 spacing=2   right "Actual" ;
define  predict / sum format= dollar10. width=12 spacing=2   right "Predict" ;
define  actual1 / sum format= dollar10. width=12 spacing=2   right "Actual" ;
define  predict1 / sum format= dollar10. width=12 spacing=2   right "Predict" ;
define  country / group format= $char10. width=10 spacing=2   left "Country" noprint;
```

```
define  region / group format= $char10. width=10 spacing=2   left "Region" noprint ;
define  division / group format= $char10. width=10 spacing=2 left "Division" noprint;
define  prodtype / group format= $char10. width=10 spacing=2   left "Product Type"  ;
define  product / group format= $char10. width=10 spacing=2   left "Product" ;
define  diff / computed format= dollar10. width=9 spacing=2   right "Diff" ;
define  diff2 / computed format= dollar10. width=9 spacing=2   right "Diff" ;
define  year / across format= yearfmt15. order=internal width=4 spacing=2 right
        "Sales Year" preloadfmt;

break after region / page summarize style=[font_weight=bold background=cxfffedb];
break after prodtype / summarize style=[font_weight=bold background=cxcc9966];

compute before division;
line @1 "^s={background= cxfffedb} Division Name: " division $10.;
hold_division=division;
endcomp;

compute after product/style=[just=c cellheight=6.5pt background=cxfccf7b];
length text $ 250;
num=0;
text=' ';
if diff2 < 0 and upcase(_break_) eq 'product' then do;
   text='^s={foreground=red background=cxffffcc}* Total Actual Sales of '||
         trim(left(hold_division))||' '||trim(left(product))
         ||' '||trim(left(hold_prodtype))||
         ' less than Total Projected Sales - Projections too high';
num=250;
end;
if diff2 > 2300 and upcase(_break_) eq 'product' then do;
   text='^s={foreground=green background=cxffffcc font_weight=bold}
        ** Total Actual Sales of $2,300 over Total Projected Sales for '
        ||trim(left(hold_division))||' '||trim(left(product))||' '||
        trim(left(hold_prodtype))||' - Great Job';
   num=250;
end;
line text $varying. num;
endcomp;

compute after prodtype;
if _break_ ne ' ' then product='Total';
endcomp;

compute after region;
if _break_ ne ' ' then do;
prodtype=trim(left(region))||' Total';;
end;
endcomp;

compute before _page_/style=[just=l];
   /* Establish the variable type and maximum length of the DATA step variable */
   /* used to create a text string for the line statement. */
length region_text $25;
region_text=trim(left(country))||', '||trim(left(region))||' Region';
line    ' Country: ' region_text $25.;
endcomp;

compute prodtype;
   /* Establish the variable type and maximum length of DATA step variables. */
length urlstring $ 75 hold_division $ 10 hold_prodtype $ 10;
   /* When the row is a detail row, create the DATA step variable for country, */
   /* region, division and prodtype. */
if _break_ eq ' ' then do;
   if country ne ' ' then country1=country;
   if region ne ' ' then region1=region;
   if division ne ' ' then division1=division;
```

```
      /* Also, when prodtype has a value then create a URL string to use as the */
      /* value for the call define.  Create a link with the URL attribute name */
      /* and apply the link to the PRODTYPE variable. */
   if prodtype ne ' '  then do;
      hold_prodtype=prodtype;
      urlstring=trim(left(country1))||'_'||
                trim(left(region1))||'_'||
                trim(left(division1))||'_'||
                trim(left(prodtype))||'_'||'detail.htm';
       call define('prodtype','url',urlstring);
   end;
end;
endcomp;

compute diff2;
   /* Calculations for the DIFF variable under the ACROSS variable. */
_c8_=_c6_-_c7_;
_c11_=_c9_-_c10_;
   /* Calculations for the estimated columns. */
_c12_=_c9_*1.04;
_c13_=_c12_;
_c14_=_c12_-_c13_;
   /* Calculations to add the estimate to the total and the difference. */
actual.sum=actual.sum+_c12_;
predict.sum=predict.sum+_c13_;
diff2=actual.sum-predict.sum;
   /* Highlight total values under 0. */
if diff2 < 0 and upcase(_break_) eq ' ' then do;
  call define('diff2','style',
      'style=[foreground=red background=cxffffcc posttext="*"] ');
end;
   /* Highlight total difference values over 2300. */
if diff2 > 2300 and upcase(_break_) eq ' ' then do;
  call define('diff2','style',
       'style=[foreground=green background=cxffffcc  posttext="**"] ');
end;
endcomp;

run;

ods html close;
```

CONCLUSION

PROC REPORT is a powerful reporting tool. With a few lines of code you can produce an output report. However, there are times when you need to create a report that goes beyond your current knowledge of and skills for building customized reports. This paper explains how creating a coding plan can help you expand your knowledge and manage the task of building reports that advance your skills to a higher level. The coding plan needs to be based on what type of report you need, who needs the report, and which tools are available to create the plan. As you go through each section of the sample final output to practice the techniques that are presented in this report, you will find yourself advancing from an apprentice-level carpenter to a master-level woodworker.

RESOURCES

McMahill, Allison. 2007. "Beyond the Basics: Advanced PROC REPORT Tips and Tricks." Available at support.sas.com/rnd/papers/sgf07/sgf2007-report.pdf.

SAS Institute Inc. 2008. "The REPORT Procedure: Getting Started with the Basics." Available at support.sas.com/resources/papers/ProcReportBasics.pdf.

SAS Institute Inc. 2008. "Using Style Elements in the REPORT and TABULATE Procedures." Available at support.sas.com/resources/papers/stylesinprocs.pdf.

SAS Institute Inc. 2009. "The REPORT Procedure." *Base SAS® 9.2 Procedures Guide*. Cary, NC: SAS Institute Inc. Available at support.sas.com/documentation/cdl/en/proc/61895/PDF/default/proc.pdf.

SAS Institute Inc. 2010. Base SAS, SAS Notes and Concepts for ODS Web site. Available at support.sas.com/rnd/base/ods/templateFAQ/index.html.

CONTACT INFORMATION

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

Allison McMahill Booth
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
E-mail: support@sas.com
Web: support.sas.com