PharmaSUG 2013 - Paper CC37

Turn Your Plain Report into a Painted Report Using ODS Styles

Cynthia L. Zender and Allison M. Booth, SAS Institute, Inc., Cary, NC

ABSTRACT

In order to use STYLE= statement level overrides, you need to understand which pieces or areas of PRINT, REPORT, and TABULATE output you can change. And then you need to understand how and where in your procedure syntax to use the STYLE= override syntax. Last, but not least, you need to know the names of the style attributes that you want to change.

This presentation illustrates with concrete examples how to use STYLE= overrides with PRINT, REPORT, and TABULATE. As the examples move from the simple to the complex, you will learn how to change fonts, add text decoration, alter the interior table lines, perform trafficlighting, and insert images into your ODS output files using some ODS magic to improve your reports.

INTRODUCTION

Everybody's heard about the San Francisco "painted ladies," Victorian houses with colors that enhance their architectural details.

These paint schemes call attention to the house's details, but there are still a lot of steps involved in turning a plain house with a "regular" paint job into a "painted lady." The same holds true for your reports. From the plain text LISTING file to the highly customized report in PDF, RTF, or HTML result files, you can follow some of the same steps. First, you need to select your basic or main color, then you select the contrasting or complementary colors and decide which details to highlight on the house. After all the prep and setup and selection, you finally set up the scaffolding and start painting.

When we decided to write this paper, it was because our customers always ask about how to enhance the results of the SAS report procedures. Many times, it comes down to using STYLE= overrides with the procedure syntax. Between the two of us, we have over 40 years of cumulative experience with SAS and reporting using PROC PRINT, PROC REPORT, and PROC TABULATE. We have both worked with ODS since it was first introduced in SAS 7. This paper represents the basic building blocks for enhancing your reports with ODS and these popular procedures.

So get ready for our examples and recommendations on how to turn your plain reports into "painted" reports using ODS.

WHAT IS THE OUTPUT DELIVERY SYSTEM?

First, a brief overview of the Output Delivery System (ODS), which enables you to run your procedures and SAS programs and direct the report results to HTML, RTF, or PDF files. There is also a feature of ODS that enables you to create output data sets and ODS DOCUMENT stores, but those are more advanced uses of ODS that fall outside the scope of this paper.

If you think for a moment about the basic differences between reading something online versus reading it on paper, you will quickly come to appreciate all that ODS does for you automatically. For example, on the Internet, when you're looking at a product catalog or at a report, the output can be as wide (150 columns) or as long (5000 rows) as it needs to be, and you can just scroll back and forth and up and down on the web page. On the Internet, you can call attention to some piece of text by making it blink or having a little box pop up when the mouse is positioned over a word or cell. But, on a printed piece of paper, there is no blinking text or pop-up box. You can't display 150 columns of anything on a piece of paper unless you reduce the font size to be very, very tiny.

So, when you use the default ODS invocation for output, using either Display Manager or SAS Enterprise Guide, each of those interfaces has a default type of destination output that it "likes" to produce. And each interface has a default style that it will use for each type of output. By default, if you have SAS 9.3 or higher, ODS produces HTML output in SAS Display Manager or the SAS Windowing Environment. In SAS Enterprise Guide, ODS produces SASReport XML by default. It is possible, in either interface to take control of the type of output that ODS produces.

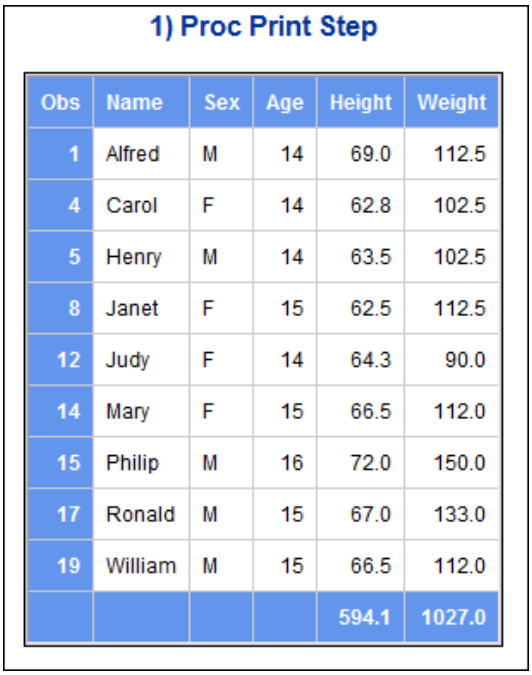
The basic format that we are going to use is the ODS "sandwich" technique, in which we take "full control" of the ODS invocation. In this usage, we will always "sandwich" our SAS code examples between ODS statements that tell ODS three important things:

1) which destination we want to use
2) the name of the file we want to create
3) the basic style (or paint job) that we want for the output report

Here's the basic invocation code and the partial output from the program is shown in Figures 1 through 5:

```
ods _all_ close;
ods html file='c:\temp\demo1.html' style=sasweb;
. . . SAS code. . .
ods html close;
```

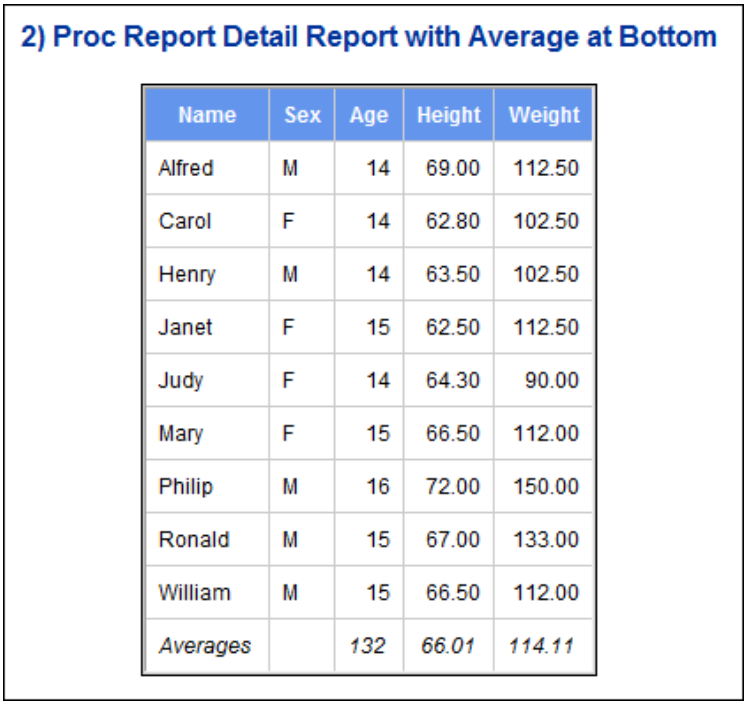
We use STYLE=SASWEB so that no matter which interface you use, or which version of SAS you use (or which version of SAS Enterprise Guide), the output will look the same, because in some versions of SAS and SAS Enterprise Guide, the new (9.3) HTMLBLUE style was not used by all interfaces. Our SAS code has five separate steps: PROC PRINT, several PROC REPORT steps, and a PROC TABULATE step. And each report has the basic "paint job" that comes by default with the use of the SASWEB style.



1) Proc Print Step

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
8	Janet	F	15	62.5	112.5
12	Judy	F	14	64.3	90.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
17	Ronald	M	15	67.0	133.0
19	William	M	15	66.5	112.0
				594.1	1027.0

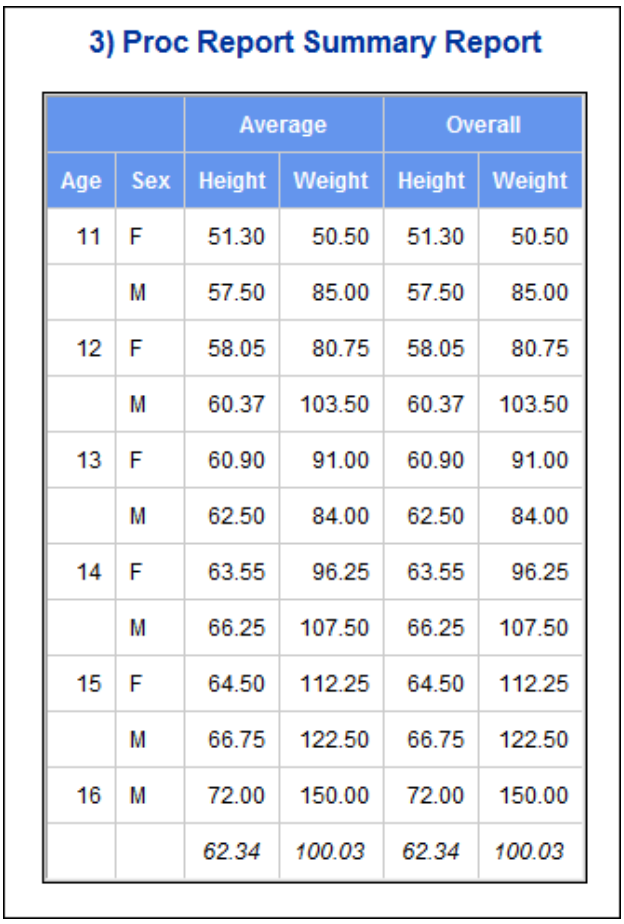
Figure 1. PROC PRINT Results



2) Proc Report Detail Report with Average at Bottom

Name	Sex	Age	Height	Weight
Alfred	M	14	69.00	112.50
Carol	F	14	62.80	102.50
Henry	M	14	63.50	102.50
Janet	F	15	62.50	112.50
Judy	F	14	64.30	90.00
Mary	F	15	66.50	112.00
Philip	M	16	72.00	150.00
Ronald	M	15	67.00	133.00
William	M	15	66.50	112.00
Averages		*132*	*66.01*	*114.11*

Figure 2. PROC Report Detail Report Results



3) Proc Report Summary Report

		Average		Overall	
Age	Sex	Height	Weight	Height	Weight
11	F	51.30	50.50	51.30	50.50
	M	57.50	85.00	57.50	85.00
12	F	58.05	80.75	58.05	80.75
	M	60.37	103.50	60.37	103.50
13	F	60.90	91.00	60.90	91.00
	M	62.50	84.00	62.50	84.00
14	F	63.55	96.25	63.55	96.25
	M	66.25	107.50	66.25	107.50
15	F	64.50	112.25	64.50	112.25
	M	66.75	122.50	66.75	122.50
16	M	72.00	150.00	72.00	150.00
		62.34	*100.03*	*62.34*	*100.03*

Figure 3. PROC Report Summary Report Results

4) Proc Report Crosstab Report

	Gender Avg					
	F		M		Overall	
Age	Height	Weight	Height	Weight	Height	Weight
11	51.30	50.50	57.50	85.00	54.40	67.75
12	58.05	80.75	60.37	103.50	59.44	94.40
13	60.90	91.00	62.50	84.00	61.43	88.67
14	63.55	96.25	66.25	107.50	64.90	101.88
15	64.50	112.25	66.75	122.50	65.63	117.38
16	.	.	72.00	150.00	72.00	150.00
	60.59	*90.11*	*63.91*	*108.95*	*62.34*	*100.03*

Figure 4. PROC Report Crosstab Report Results

5) Proc Tabulate Crosstab Report

	Gender Avg				Overall Avg	
	F		M			
	Height	Weight	Height	Weight	Height	Weight
Age						
11	51.30	50.50	57.50	85.00	54.40	67.75
12	58.05	80.75	60.37	103.50	59.44	94.40
13	60.90	91.00	62.50	84.00	61.43	88.67
14	63.55	96.25	66.25	107.50	64.90	101.88
15	64.50	112.25	66.75	122.50	65.63	117.38
16	.	.	72.00	150.00	72.00	150.00
All	60.59	90.11	63.91	108.95	62.34	100.03

Figure 5. PROC TABULATE Results

In case you're wondering, Report #1 and #2 used a subset of observations to keep the screenshot smaller. And, PROC REPORT has three examples. That's because PROC REPORT is the "Grande Dame" of the group—it can do quite a few things that PRINT and TABULATE can't do. But given that basic starting point, let's look at PROC PRINT.

STARTING WITH PROC PRINT

Technically, a Victorian house can't be a painted lady unless it has three or more colors. So the basic color scheme (black, white, and blue) displayed in Figure 1 does use three colors. But we can jazz up the PROC PRINT output a ***lot*** by introducing some STYLE= overrides into the PROC PRINT statement. When a style template is used with an ODS destination that supports style characteristics, the style template contains style element definitions that are used for each discrete "piece" or "area" of the final report. So, for example, in the default output, the blue background and white foreground colors come from the Header style elements. And that makes sense, because every place in the PROC PRINT output that is white on blue is styled by the Header style element.

The Header style element has a "package" or "collection" of style characteristics, and each characteristic is called a style attribute. Figure 6 contains a side-by-side display of the code that is used in PROC PRINT with the HTML results opened in a browser.

Code

```
proc print data=sashelp.class
    style(table)={rules=rows frame=void
                  cellspacing=0
                  bordercolor=cyan}
    style(header)={background=purple
                   font_size=12pt}
    style(grandtotal)={background=pink
                       font_size=14pt}
    style(obs)={background=pink color=black
                font_size=12pt}
    style(obsheader)={background=green
                      font_size=12pt};
  title '2.1) Proc Print Step';
  where age gt 13;
  sum height weight;
run;
```

Results

2.1) Proc Print Step

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
8	Janet	F	15	62.5	112.5
12	Judy	F	14	64.3	90.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
17	Ronald	M	15	67.0	133.0
19	William	M	15	66.5	112.0
				594.1	1027.0

Figure 6: PROC PRINT Code and Results Using STYLE Overrides

The ODS HTML statement is not shown in this code snippet because it is the same ODS HTML statement that produced Figures 1 through 5. But, except for the background color and foreground color of the data cells, you can see that every piece of this report had its style changed through the use of STYLE= statement level options. These options act to override the style attributes that are set by the STYLE= option in the destination invocation. The "areas" or "locations" that you see in parentheses are the only pieces of the PROC PRINT output that will be touched by the STYLE= overrides. Because the style overrides are specified in the PROC PRINT statement, the style changes will apply to all the areas in the report. For example, in the HEADER area, all the header areas will be purple and 12 point. But, because the OBS area is overridden to be pink, black, and 12 point and the OBSHEADER area is overridden to be green, those overrides are applied to the only OBSHEADER and the OBS column values. You can see from this example that an "area" could have an impact on one cell (OBSHEADER) or on multiple cells (GRANDTOTAL, OBS, and HEADER).

Some simpler code is shown in Figure 7, with the PROC PRINT results. In this simplified example, the default for all the headers is set to white on purple. Then, in a VAR statement for the NAME variable, only the foreground color is overridden to be red. Notice that the OBSHEADER and the OBS columns are unaffected by the STYLE= override.

That is because those two areas are styled by their own area keywords.

Code	Partial Results

`proc print data=sashelp.class`
`  style(header)=`
`  {background=purple`
`  font_size=12pt};`
`  title '2.2) Proc Print Step';`
`  where age gt 13;`
`  var name /`
`  style(header)={color=red};`
`  var sex age height weight;`
`  sum height weight;`
`run;`
	2.2) Proc Print Step

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
4	Carol	F	14	62.8	102.5

Figure 7: PROC PRINT Results Override One Header

The STYLE= option in the VAR statement needs to specify only the attributes that are being overridden. So, BACKGROUND=PURPLE (in the PROC PRINT statement) overrides the background for the HEADER, as specified in the STYLE template. But it does not change the SASWEB foreground color of white. Then, COLOR=RED in the VAR statement for NAME overrides the foreground color of WHITE, but only for the NAME variable. The header cells for the other variables retain the foreground color of WHITE. If the same program were submitted using a different STYLE= option on the ODS invocation, the results would be different, as shown in Figure 8.

Code	Partial Results

`ods html file='demo_2_2a.html'`
`  style=htmlblue;`
`  . . . same code with different`
`  title statement . . .`
`ods html close;`
	2.2a) Proc Print Step--Different Style

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
4	Carol	F	14	62.8	102.5

Figure 8: PROC PRINT Results Override One Header

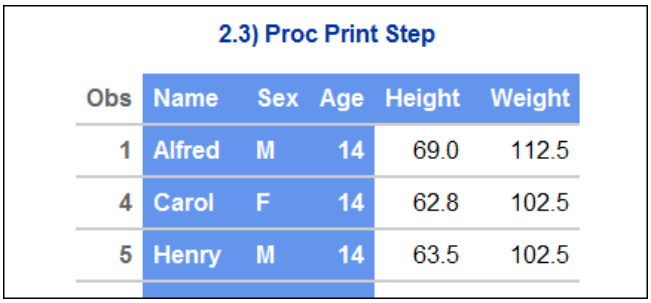
Notice how the purple and red colors override the style characteristics from the HTMLBLUE style that is specified in the ODS invocation. The attributes and areas that were unchanged in Figure 7 are also unchanged in Figure 8. This shows the fundamental ODS style behavior in which attributes that are not specifically overridden in the procedure code will use the style template attributes of whatever style is specified in the ODS invocation.

Figure 9 revisits Proc PRINT output, but with a more subdued color scheme. The important set of attributes that are illustrated in this example are the TABLE area attributes for controlling the interior table lines. First, in Figure 7, = notice that there is a black outline around the whole table. This is known as the table's frame (like a picture frame) and it is controlled by the FRAME style attribute. FRAME=VOID, as seen in the Figure 6 code, turns off the frame around the report table. FRAME has an interaction with borderwidth and cellspacing. (The black outline in the screenshot is the outline from the image capture software.)

Code

```
proc print data=sashelp.class
     style(table)={rules=rows
           frame=void
           cellspacing=0
           bordercolor=cxcccccc}
     style(grandtotal)=
           Header{font_size=14pt}
     style(obs)={background=white
                 color=cx666666
                 font_size=12pt}
     style(obsheader)=
          {background=white
           color=cx666666
           font_size=12pt};
```

Partial Results



2.3) Proc Print Step

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5

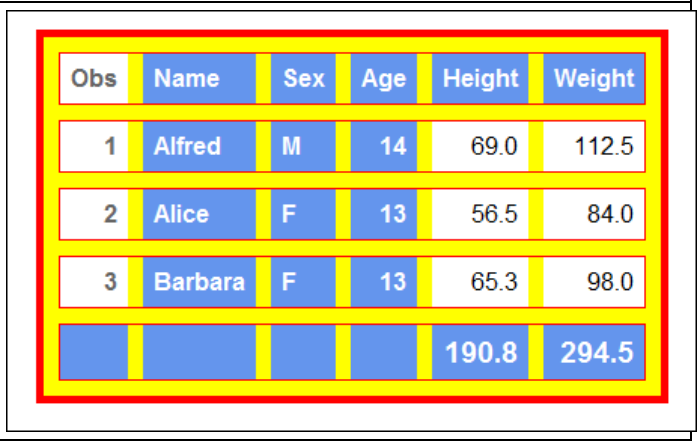
Figure 9: Partial PROC PRINT Results with Interior Table Rules Changed

In order to understand the interactions of attributes that control the report output structure, consider Figure 10. With just a few changes, the appearance of the output report has been dramatically impacted.

Code

```
proc print data=sashelp.class(obs=3)
     style(table)={rules=rows
           frame=box
           cellspacing=10
           background=yellow
           borderwidth=5
           bordercolor=red}
```

Partial Results



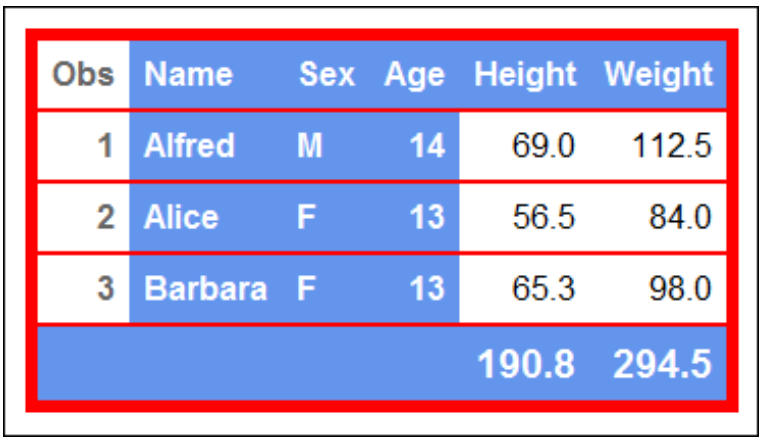
Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
				190.8	294.5

Figure 10: Three Observations in PROC PRINT Results with Interior Table Rules Changed

Notice how the bordercolor value of red has been used for the FRAME=BOX attribute. When FRAME=VOID, as in the code for Figure 9, the border color was not used for a frame around the box, only for the interior rows. But after the FRAME= value was set to BOX, the bordercolor of red and the border width of 5 were used for the frame around the report. You can see that the red border color is also used for the interior table lines, which are controlled by the RULES attribute. In previous screen shots, the setting RULES=ROWS was rendered as horizontal lines between each report row, using the border color of light gray (cxcccccc).

In the earlier example for Figure 9, however, cellspacing was set to 0, which meant that the normal background color of the table did not show through the space between the cells (CELLSPACING or BORDERSPACING attribute). In the code that produced Figure 10, the value for the CELLSPACING attribute was set to 10. This means that the background color of yellow will show through the 10 pixels of space between the boundary of each cell. You can tell that RULES=ROWS is still in effect because the red line for the interior rule is crossing only the horizontal sides of each cell and does not touch the vertical sides of each cell.

If the only change to the above code is to change CELLSPACING=0, then the output would appear as shown in Figure 11.



Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
				190.8	294.5

Figure 11: Three Observations with CELLSPACING=0

In the zip file of programs that you will be able to download after the SAS Global Forum, there are some other examples of using RULES and FRAME to impact the table output. To show all the possible values of RULES and all the possible values of FRAME falls outside the scope of this paper. The last example for PROC PRINT will be a trafficlighting example.

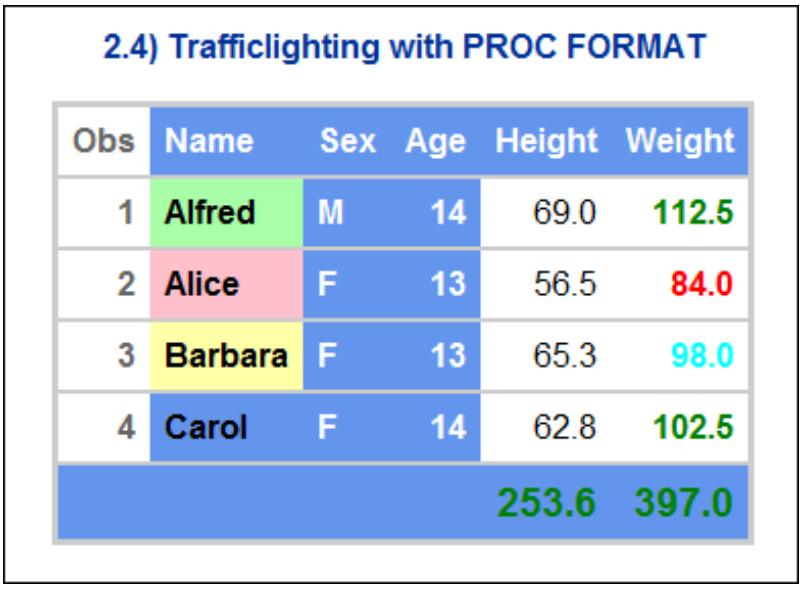
To perform trafficlighting with PROC PRINT, you need a user-defined format to specify style attribute values. Note in Figure 12 how Alfred, Alice, and Barbara all have different background colors based on the value of NAME. In this example, there is no user-defined format defined for Carol. So the values that control that data cell are the default values in the style template and the other overrides in the VAR statement, but not the user-defined format overrides. In addition, notice the foreground color changes for the WEIGHT variable; these are based on the variable values. In addition, the color of the GRANDTOTAL summary line values was changed because the user-defined format was also used on the GRANDTOTAL style attribute override. The entire code that produces Figure 12 is shown below.

```
proc format;
  value $nfmt 'Alfred' = 'verylightgreen'
              'Alice' = 'pink'
              'Barbara' = 'verylightyellow';

  value wfmt low-85 = 'red'
             86-99 = 'cyan'
            100-high='green';
run;

ods html file='c:\temp\demo_2_4.html' style=sasweb;
  proc print data=sashelp.class(obs=4)
       style(table)={rules=rows frame=box
             cellspacing=0 borderwidth=2 bordercolor=cxcccccc }
       style(grandtotal)=Header{font_size=14pt foreground=wfmt.}
       style(obs)={background=white color=cx666666 font_size=12pt}
       style(obsheader)={background=white color=cx666666
                         font_size=12pt};
  title '2.4) Trafficlighting with PROC FORMAT';
  var name/style(header)=Header{font_size=12pt}
           style(data)=Header{font_size=12pt
                       background=$nfmt.
                       foreground=black};
  var sex age/style(header)=Header{font_size=12pt}
              style(data)=Header{font_size=12pt };
  var height/style(header)=Header{font_size=12pt}
              style(data)={font_size=12pt};
  var weight/style(header)=Header{font_size=12pt}
              style(data)={font_size=12pt
                           foreground=wfmt. font_weight=bold};
  sum height weight ;
```

```
  run;
ods html close;
```



2.4) Trafficlighting with PROC FORMAT

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
				253.6	397.0

Figure 12: Three Observations with CELLSPACING=0

The format values are applied in the PROC PRINT step by using the format name to supply the attribute value. Because NAME is a character variable, the format needed to be a character format. Because WEIGHT is numeric, the format needed to be numeric. When you use a format name to supply the attribute value, the only thing that you specify is the value itself, in the form: **attribute=*format-name.*** The rule of thumb is that the more specific the change overrides the less specific the change. So the style attributes in the SASWEB style are overridden by the style attributes that are specified in the PROC PRINT statement. Then the style attributes that are specified in the VAR statements apply to either the column heading or the column data. Finally, at the individual cell level, the user-defined format values are used to override what is specified for the particular column (or header).

PROC PRINT is good at producing detail reports with subtotals and grand totals. But there is not a lot you can do with reports if you want to show the detail rows and get a different statistic at the break. User-defined formats only allow you to apply trafficlighting based on the values in one cell. What if you want to highlight the WEIGHT value, but only for Alfred and Barbara? That is something else that you can't do with PROC PRINT. There are other ways to work with style attributes and style overrides using PROC REPORT and PROC TABULATE, so it's time to move on to the PROC REPORT topic.

DOING MORE WITH PROC REPORT

It will be easy to replicate the Figure 12 results using PROC REPORT. The overall approach of STYLE= overrides in PROC REPORT is the same as in PROC PRINT. The user-defined formats are the same, although instead of a VAR statement, the STYLE overrides are primarily specified in the DEFINE statement. PROC REPORT uses the same concept of areas or locations as PROC PRINT. That way in one DEFINE statement (as with the VAR statement), you can specify overrides for both the column heading cells and the column data cells. Table 1 shows the difference between PROC PRINT areas and PROC REPORT areas or locations for style overrides.

PROC PRINT Areas	PROC REPORT Areas
TABLE	REPORT
HEADER (HDR)	HEADER (HDR)
DATA	COLUMN
GRANDTOTAL/TOTAL	SUMMARY

OBS/OBSHEADER	LINES
BYLABEL	CALLDEF
N	

Table 1: The Difference between PROC PRINT and PROC REPORT Area or Location Style Overrides

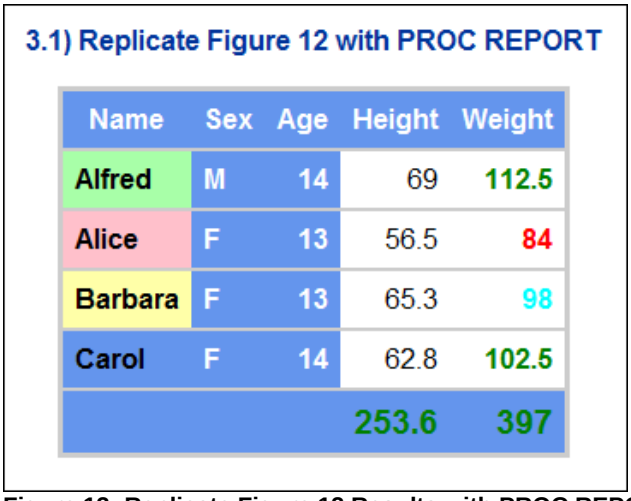
Each area or location can be used in the PROC PRINT or the PROC REPORT statement But, as you've seen, they can also be used in other procedure statements as well. Consult the documentation for each procedure for more information about these locations and the other statements they can be used in.

The PROC REPORT code that creates Figure 13 is shown below. It illustrates the major differences. For example, STYLE(GRANDTOTAL) in PROC PRINT becomes STYLE(SUMMARY) in PROC REPORT. And, STYLE(DATA) in PROC PRINT is used for the data cells, just as STYLE(COLUMN) is used in PROC REPORT. Other locations like OBS and OBSHEADER are unique to PROC PRINT, just as LINES and CALLDEF are unique to PROC REPORT.

The same user-defined formats that are demonstrated with PROC PRINT are used for the PROC REPORT trafficlighting. The difference, of course, is that PROC REPORT does not have an OBS column. So the OBS and OBSHEADER locations are not needed in the REPORT code. The internal structure of the table lines is changed in the STYLE(REPORT) location. And the SUMMARY location is used for the summary at the end of the report. Initially, the SUM statistic is requested for the HEIGHT and WEIGHT variables in order to replicate the PROC PRINT report. However, in the next example, we'll change the statistic.

```
   ods html file='c:\temp\demo_3_1.html' style=sasweb;

     proc report data=sashelp.class(obs=4) nowd
          style(report)={rules=rows
                frame=box
                cellspacing=0
                borderwidth=2
                bordercolor=cxcccccc }
          style(summary)=Header{font_size=14pt
                                foreground=wfmt.};
     title '3.1) Replicate Figure 12 with PROC REPORT';
     column name sex age height weight;
     define name / order
         style(header)=Header{font_size=12pt}
         style(column)=Header{font_size=12pt
                      background=$nfmt.
                      foreground=black};
     define sex / display
         style(header)=Header{font_size=12pt}
         style(column)=Header{font_size=12pt };
     define age/ display
         style(header)=Header{font_size=12pt}
         style(column)=Header{font_size=12pt };
     define height / sum style(header)=Header{font_size=12pt}
                    style(column)={font_size=12pt};
     define weight / sum style(header)=Header{font_size=12pt}
                      style(column)={font_size=12pt
                                     foreground=wfmt.
                                     font_weight=bold};
     rbreak after / summarize;
     run;
ods html close;
```

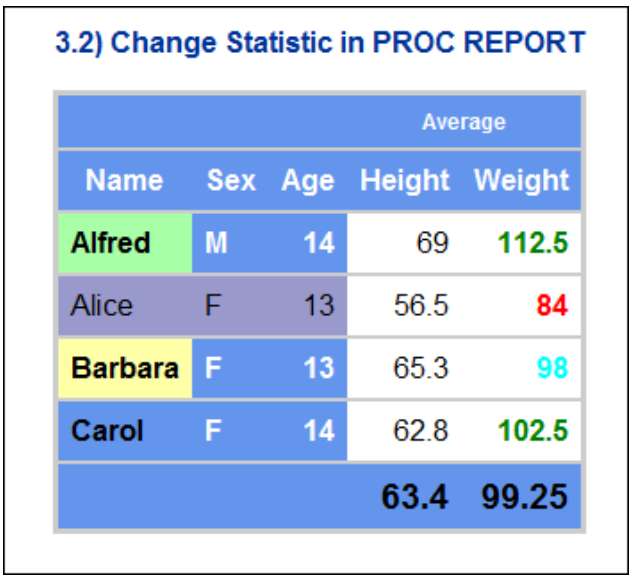


3.1) Replicate Figure 12 with PROC REPORT

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
Carol	F	14	62.8	102.5
			253.6	397

Figure 13: Replicate Figure 12 Results with PROC REPORT

Now that the basics of STYLE= overrides with PROC REPORT give us the same basic paint scheme as PROC PRINT, let's make a few changes as shown in Figure 14.



3.2) Change Statistic in PROC REPORT

			Average	
Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
Carol	F	14	62.8	102.5
			63.4	99.25

Figure 14: Different Statistic and Different Style

You can see a couple of things that are different. First, the statistics at the report break represent the MEAN statistic. That happened because these two statements were changed:

```
define height / mean style(header)=Header{font_size=12pt}
                    style(column)={font_size=12pt};
define weight / mean style(header)=Header{font_size=12pt}
                    style(column)={font_size=12pt foreground=wfmt.
                                   font_weight=bold};
```

Then, the trafficlighting for the NAME, SEX, and AGE columns was accomplished in a COMPUTE block for AGE:

```
compute age;
  if age = 13 and name = 'Alice' then do;
     call define(_col_,'style','style={background=cx9999cc}');
     call define('name','style','style={background=cx9999cc}');
     call define('sex','style','style={background=cx9999cc}');
  end;
endcomp;
```

A COMPUTE block is a way to create computed columns (a capability not being used in the above code) and perform other processing such as changing the style conditionally. With trafficlighting, the value in one and only one cell is used to set some style characteristic. But in a COMPUTE block, you can make a multi-condition test and then change style characteristics based on whether the test was true or false. In the above code, if the value of AGE is 13 and the value of NAME is Alice, then the three CALL DEFINE statements are executed.

Basically, the CALL DEFINE statement takes three arguments: 1) which variable or item to change in the report; 2) the specific type of change to make; and 3) the syntax that is required for the change. You can actually change more than just the style of a report element. For example, other valid values for the second argument might be 'FORMAT' or 'URL'.

The code above shows two of the methods for providing values for the first argument. These are probably the most common methods: using the automatic _COL_ as the argument or using a text string that resolves to the name of one of the other items in the report. Another popular value for the first argument is _ROW_; consider this change to the COMPUTE block for AGE, which produces Figure 15.

In addition, note how the format for AGE was changed through the use of 'FORMAT' in the CALL DEFINE statement. Only the changed COMPUTE block is shown below:

```
compute age;
 if age = 13 and name = 'Alice' then do;
    call define(_row_,'style','style={background=cx9999cc}');
```

```
      call define(_col_,'format','5.2');
    end;
  endcomp;
```

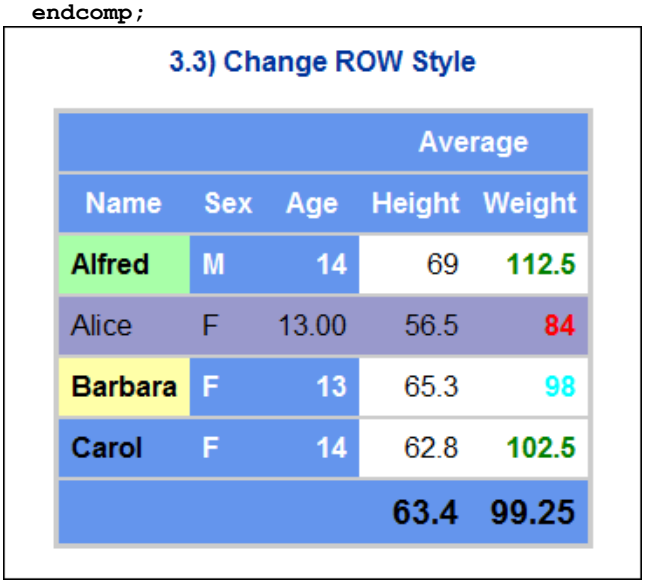


Figure 15: Using _ROW_ for the CALL DEFINE Statement

In the zip file of programs that you can download for this paper, you will also find a PROC REPORT program that performs alternate row highlighting using CALL DEFINE techniques.

Sometimes you will find that a SAS style is adequate for what you need to produce, but you just need to make a few modifications. For example, the summary report in Figure 3 shows Average and Overall Height and Weight. Using the JOURNAL style accomplishes almost everything that is desired for the output report. The only changes to Journal style that are required are to reduce some of the white space in the report, which can be done by changing the CELLPADDING style attribute and underlining the words "Average" and "Overall" in the column headings.

Starting in SAS 9.2, there is a style attribute called TEXTDECORATION that you can use to perform an underline of a cell value. The results shown in Figure 16 were produced by the following code. Note the use of ODS ESCAPECHAR to underline the text strings in the COLUMN statement.

```
ods html file='c:\temp\demo_3_4.html' style=journal;
  ods escapechar='^';
  proc report data=sashelp.class nowd
    style(report)={cellpadding=2pt}
    style(summary)=Header;
    column age sex ('^{style[textdecoration=underline]Average}' height weight)
              ('^{style[textdecoration=underline]Overall}' height=hta weight=wta);
    title '3.4) Proc Report Summary Report';
    define age / group;
    define sex / group;
    define height / mean f=7.2;
    define weight / mean f=7.2;
    define hta / mean f=7.2 style(column)={font_weight=bold};
    define wta / mean f=7.2  style(column)={font_weight=bold};
    rbreak after / summarize;
  run;
ods html close;
```

		Average		Overall	
Age	Sex	Height	Weight	Height	Weight
11	F	51.30	50.50	**51.30**	**50.50**
	M	57.50	85.00	**57.50**	**85.00**
12	F	58.05	80.75	**58.05**	**80.75**
	M	60.37	103.50	**60.37**	**103.50**
13	F	60.90	91.00	**60.90**	**91.00**
	M	62.50	84.00	**62.50**	**84.00**
14	F	63.55	96.25	**63.55**	**96.25**
	M	66.25	107.50	**66.25**	**107.50**
15	F	64.50	112.25	**64.50**	**112.25**
	M	66.75	122.50	**66.75**	**122.50**
16	M	72.00	150.00	**72.00**	**150.00**
		62.34	*100.03*	***62.34***	***100.03***

Figure 16: Summary Report with Underline

In the code for Figure 16, ODS ESCAPECHAR was used to apply an underline to the strings "Average" and "Overall". But as you can see in Figure 17, the text string "Confidential Report" appears twice in the output. And it is also underlined. This time, the change was accomplished using a STYLE= override in the COMPUTE statements.

3.5) Proc Report Summary Report

		Average		Overall	
Age	Sex	Height	Weight	Height	Weight
		Confidential Report			
11	F	51.30	50.50	**51.30**	**50.50**
	M	57.50	85.00	**57.50**	**85.00**
12	F	58.05	80.75	**58.05**	**80.75**
	M	60.37	103.50	**60.37**	**103.50**
13	F	60.90	91.00	**60.90**	**91.00**
	M	62.50	84.00	**62.50**	**84.00**
14	F	63.55	96.25	**63.55**	**96.25**
	M	66.25	107.50	**66.25**	**107.50**
15	F	64.50	112.25	**64.50**	**112.25**
	M	66.75	122.50	**66.75**	**122.50**
16	M	72.00	150.00	**72.00**	**150.00**
		62.34	*100.03*	***62.34***	***100.03***
		Confidential Report			

Figure 17: STYLE= Override on COMPUTE Block

The only changes to the previous code were the addition of these two COMPUTE blocks. The COMPUTE BEFORE and COMPUTE AFTER statements will each be executed only one time in a report. The COMPUTE BEFORE statement executes after the headers have been written, but before the data cells for the first report row are written.

The COMPUTE AFTER statement executes after the last data cells for the last report row are written. The STYLE override specified after the slash does not have a location in parentheses. This is because there is only one place that can be impacted by this type of STYLE override.

```
compute before / style={textdecoration=underline just=c color=red};
  line 'Confidential Report';
endcomp;
compute after / style={textdecoration=underline just=c color=red};
  line 'Confidential Report';
endcomp;
```

However, while this does produce the phrase "Confidential," perhaps it is necessary to call attention to the confidential nature of the report by inserting an image into the output file.

In addition, what if instead of having both AGE and SEX in the rows, there is a request to have only AGE in the rows and to have columns for each unique value of the SEX variable. To accomplish this, let's make PROC REPORT create a crosstab report first.

Code

```
proc report data=sashelp.class nowd out=abscols
  style(summary)=Header;
  where age le 13;
  title '4.1) Proc Report Crosstab Report';
  column age ( sex,(height weight))
             ('Overall' height=hta weight=wta);
  define age / group
         style(column)=Header;
  define sex / across 'Gender Avg';
  define height / mean f=7.2;
  define weight / mean f=7.2;
  define hta / mean f=7.2 style(column)=Header;
  define wta / mean f=7.2 style(column)=Header;
  rbreak after / summarize;
  compute before _page_/
          style={textdecoration=underline
                 just=c color=red};
    line 'Confidential Report';
  endcomp;
  compute after /
          style={textdecoration=underline
                 just=c color=red};
    line 'Confidential Report';
  endcomp;
run;
```

Results

Confidential Report

	Gender Avg					
	F		M		Overall	
Age	Height	Weight	Height	Weight	Height	Weight
11	51.30	50.50	57.50	85.00	54.40	67.75
12	58.05	80.75	60.37	103.50	59.44	94.40
13	60.90	91.00	62.50	84.00	61.43	88.67
	57.84	78.80	60.22	95.90	59.03	87.35

Confidential Report

Figure 18: Code to Produce Crosstab Report with PROC REPORT

Notice how the COMPUTE AFTER works the same as the previous COMPUTE example. The text string at the top of the report is accomplished with the COMPUTE BEFORE _PAGE_ block of code. This statement positions the extra report row from the LINE statement at the top of the report (above the table and above the first header row). However, the "box" around the table extends both above and below the extra text.

Next, to get an image into the report, the PREIMAGE style attribute will be used to insert a predefined image in a few different places. Figure 19 shows the image placed above the table. Figure 20 shows the image placed below the table. The image at the top of the table was placed there by altering the STYLE(REPORT) location in order to produce Figure 19. The changed PROC REPORT statement is the only difference from the program that produced Figure 18:

```
proc report data=sashelp.class nowd out=abscols
            style(report)={preimage="confidential.png"}
            style(summary)=Header;
```

	Gender Avg					
	F		M		Overall	
Age	Height	Weight	Height	Weight	Height	Weight
11	51.30	50.50	57.50	85.00	54.40	67.75
12	58.05	80.75	60.37	103.50	59.44	94.40
13	60.90	91.00	62.50	84.00	61.43	88.67
	57.84	78.80	60.22	95.90	59.03	87.35

Figure 19: Preimage before Report Table

The image at the bottom of the table was placed there by altering the style for the COMPUTE AFTER code block. Note that the COMPUTE statement does not use a location in parentheses, as seen in the Figure 17 code. That's because the only location that can be impacted by the COMPUTE AFTER is the text (a space) that is written by the LINE statement. Notice that the image at the bottom of the table is also contained within the box or frame of the table. The code that produces Figure 20 is shown below:

```
compute after / style={preimage="confidential.png"};
  line ' ';
endcomp;
```

Figure 20: Preimage after Report Table

In Figure 20, you also see that the summary line that is produced by the RBREAK statement is styled with the HEADER style element. In addition, the "Overall" columns are also styled with the HEADER style element. The use of a style element directly in the code causes the whole group of style attributes that are collected in the style element definition to be used in the report for the location specified. When you use STYLE= overrides with PROC PRINT, PROC REPORT, or PROC TABULATE, this method of referring to a style element is handy if you like the style used for the report headers and you want to use the whole "package" of attributes by referring to the element name.

The method of making style overrides is very similar for PROC PRINT and PROC REPORT. As we've seen, the major differences between these two procedures are the areas or locations that you use inside the procedure syntax. Last, but certainly not least, when we talk about making "painted" reports, no discussion of style overrides would be complete without a discussion of PROC TABULATE.

ENDING WITH PROC TABULATE

PROC TABULATE, like PROC PRINT and PROC REPORT, enables you to use statement level style overrides to impact the report style. The difference between PROC REPORT and PROC PRINT is that PROC TABULATE does not need a location or area specified because each TABULATE statement can impact only a specific area of the report table. Let's look at a specific example. In PROC REPORT, for example, you might have something like this:

```
define charvar / group
       style(header)={background=pink}
       style(column)={font_face='Courier New'};
define numvar / sum
       style(header)={just=c}
       style(column)={color=red};
```

But the equivalent statements in PROC TABULATE would be like this:

```
class charvar / style={background=pink};
classlev charvar / style={font_face='Courier New'};
var numvar / style={just=c};
table charvar, numvar*sum*{s={color=red}};
```

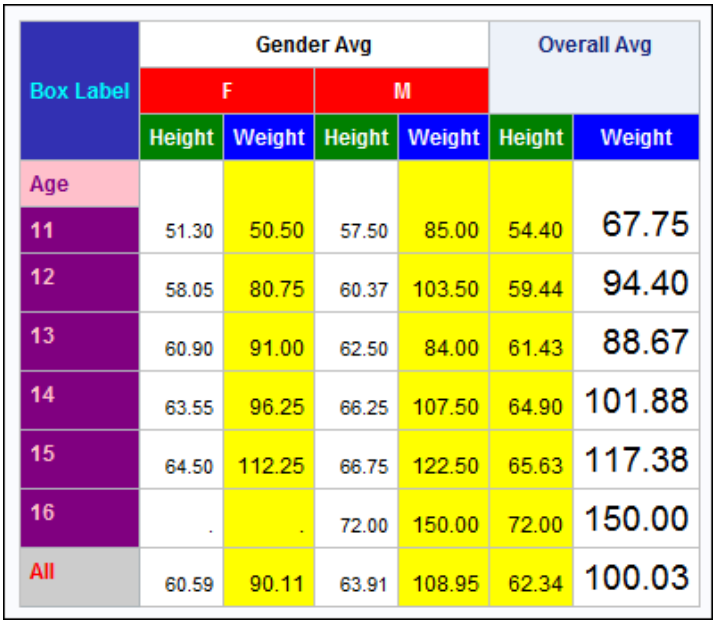
The fundamental TABULATE syntax provides statements that specify variable usage (CLASS or VAR statements) and a statement that specifies table structure (TABLE statement). In addition, there are statements like CLASSLEV and KEYWORD that were designed explicitly for PROC TABULATE and style overrides. Figure 21 shows a somewhat overpainted report to drive home the fact that every TABULATE statement "touches" a piece of the output report.

Note how the style option used in the PROC TABULATE statement changes the background color of the calculated data cells to yellow. However, a more specific change in the TABLE statement overrides the specification in the TABULATE statement. This is how TABULATE works, from the outside—from the less specific statements—to the inside. By the time a change is applied in the TABLE statement, it completely overrides any style changes that you previously specified.

In some instances, you might have conflicting style specifications because you have specified a style override in the PAGE and/or ROW and/or COLUMN dimension. Just as PROC TABULATE has a FORMAT_PRECEDENCE statement, it also has a STYLE_PRECEDENCE statement. In this way, you can control which style (or which format) will win over the other styles or formats. The code that produces Figure 21 is shown below:

```
  proc tabulate data=sashelp.class f=7.2
       style={background=yellow};
    title '5.1) Proc Tabulate Crosstab Report';
    var height/style={background=green color=white};
    var weight/style={background=blue color=white};
    class age /style={background=pink color=purple};
    classlev age / style={background=purple color=pink};
    class sex /style={background=white color=black} ;
    classlev sex / style={background=red color=white};
    table age all,
          mean=' '*sex='Gender Avg'*(height*{s={font_size=8pt}} weight)
          mean='Overall Avg'*(height weight*{s={font_size=14pt}}) /
          box={label='Box Label' s={background=bib color=cyan}};
    keyword all / style={background=cxcccccc color=red};
```

```
  run;
```



Box Label	Gender Avg				Overall Avg	
	F		M			
	Height	Weight	Height	Weight	Height	Weight
Age						
11	51.30	50.50	57.50	85.00	54.40	67.75
12	58.05	80.75	60.37	103.50	59.44	94.40
13	60.90	91.00	62.50	84.00	61.43	88.67
14	63.55	96.25	66.25	107.50	64.90	101.88
15	64.50	112.25	66.75	122.50	65.63	117.38
16	.	.	72.00	150.00	72.00	150.00
All	60.59	90.11	63.91	108.95	62.34	100.03

Figure 21: PROC TABULATE Style Overrides

The more subdued output shown in Figure 22 uses the JOURNAL style for the majority of the style changes. The only embellishments were to underline selected header strings and change the justification of the levels of the AGE and the ALL class variables. Because the POSTIMAGE style attribute was used in the TABLE statement, the image appears after or at the bottom of the report table. The code that produces Figure 22 is shown below, including the ODS invocation. In this code, a macro variable was used for the location of the image file, in order to make the physical path specification more flexible and changeable.

```
%let imgloc=%str(C:\Courses\SAS_Global_Forum_2013\Painted_Report\confidential.png);
ods html file='c:\temp\demo_5_2.html' style=journal;
  proc tabulate data=sashelp.class f=7.2;
    where age le 13;
    title '5.2) Proc Tabulate Crosstab Report';
    var height weight / style={vjust=b};
    class age / style={just=c textdecoration=underline};
    classlev age /style={just=c};
    class sex / style={vjust=b textdecoration=underline};
    classlev sex / style={textdecoration=underline};
    table age all*{s={just=c}},
          mean=' '*sex='Gender Avg'*(height weight)
          mean='Overall Avg'*(height weight) /
          style={postimage="&imgloc"};
   keyword mean / style={textdecoration=underline};
  run;
ods _all_ close;
```

In the report, there is heavy use of the TEXTDECORATION style attribute. As with the previous table, the syntax of TABULATE places AGE in the ROW dimension and the MEAN of HEIGHT and WEIGHT for each value of the SEX variable in the COLUMN dimension. Then, the Overall Average of HEIGHT and WEIGHT, without regard to SEX, is placed in the report as the final set of columns.

In the table, the WHERE statement limits the number of AGE values that are shown in the output table. However, notice how the CLASS statement for AGE and the CLASSLEV statement for AGE both impact the report table shown in Figure 22. Because all of these values were center justified, the value for the ALL class variable was also center justified so that it would look correctly justified in the row heading area.

	Gender Avg				Overall Avg	
	F		M			
	Height	Weight	Height	Weight	Height	Weight
Age						
11	51.30	50.50	57.50	85.00	54.40	67.75
12	58.05	80.75	60.37	103.50	59.44	94.40
13	60.90	91.00	62.50	84.00	61.43	88.67
All	57.84	78.80	60.22	95.90	59.03	87.35

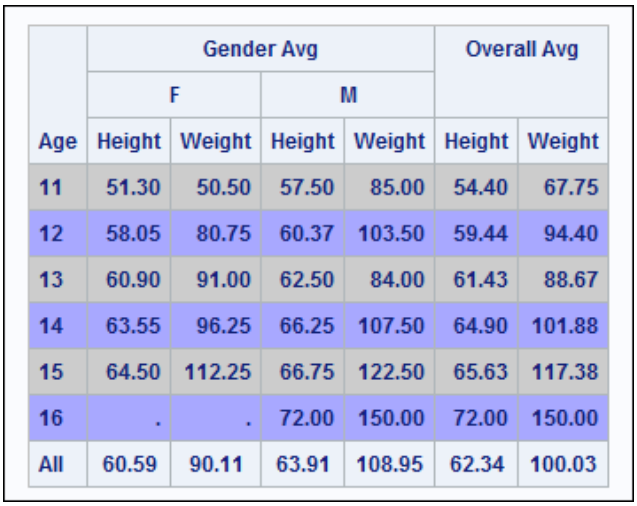
Figure 22: Postimage Added in the TABLE Statement

One feature that PROC TABULATE has, that is not shared by either PROC REPORT or PROC PRINT, is the ability to specify that one table element should inherit style specifications from a parent style element in the table. This is accomplished through a syntax specification unique to PROC TABULATE, as shown in the code below:

```
proc format;
  value agef 11, 13, 15 = 'cxccccccc'
             12, 14, 16 = 'verylightblue';
run;

ods html file='c:\temp\demo_5_3.html';
  proc tabulate data=sashelp.class f=7.2;
    title '5.3) Proc Tabulate Crosstab Report';
    var height weight ;
    class age sex;
    classlev age /style={background=agef.};
    class sex ;
    table age=' '*{style=<parent>} all*{s=<parent>{just=c}},
          mean=' '*sex='Gender Avg'*(height weight)
          mean='Overall Avg'*(height weight) /
          style_precedence=row box={label='Age' s={vjust=b}};
  run;
ods _all_ close;
```

The horizontal banding as shown in Figure 23 was accomplished by using a format to establish the background color for every age value. Then the <PARENT> element was specified for the AGE and ALL class variables. Each AGE row heading was a different color. And because STYLE_PRECEDENCE=ROW was used in the TABLE statement, each row of calculated statistics inherited the color specification from the AGE row heading value. The keyword ALL in the row heading area was unchanged by the format, so it has the same color scheme as the column headings. Then the last summary row inherited those same header characteristics. This is a different type of highlighting than was used in PROC REPORT, where a report row could be changed based on one condition or on multiple conditions. In PROC TABULATE, the parent cell sets the style attributes, and the child cell inherits those attributes, as shown in Figure 23.



Age	Gender Avg				Overall Avg	
	F		M			
	Height	Weight	Height	Weight	Height	Weight
11	51.30	50.50	57.50	85.00	54.40	67.75
12	58.05	80.75	60.37	103.50	59.44	94.40
13	60.90	91.00	62.50	84.00	61.43	88.67
14	63.55	96.25	66.25	107.50	64.90	101.88
15	64.50	112.25	66.75	122.50	65.63	117.38
16	.	.	72.00	150.00	72.00	150.00
All	60.59	90.11	63.91	108.95	62.34	100.03

Figure 23: Using STYLE=<PARENT>

CONCLUSION

The truth is that you probably don't want to over-embellish your reports. Unlike Victorian houses that come with gingerbread details that beg to be painted, your reports primarily need to convey information so that decisions can be made or facts can be documented. The important things to decide are your destination of interest, the primary style of interest,and the style embellishments that will be just enough to enhance your message and not obscure it.

SAS provides three powerful report procedures, and ODS gives you extensive control over your report output. After you have exhausted all you can do with PROC PRINT, PROC REPORT, and PROC TABULATE, you can still explore TABLE templates and DATA _NULL_ or you can explore ODS LAYOUT and the DATA step interface for report writing. Whichever report method you choose, your investment in using STYLE overrides will only help you understand ODS better.

To learn more about all the style attributes that you can change and the values that you can specify, refer to the ODS documentation topic entitled "ODS Style Elements" in the Appendix to the *SAS Output Delivery System: User's Guide*.

Let us leave you with one final thought—before you paint your report, make sure that it is accurate and that your foundation is solid. No amount of style embellishment will make up for inaccurate or incorrect data.

REFERENCES

Booth, Allison McMahill. 2010. "Evolve from a Carpenter's Apprentice to a Master Woodworker: Creating a Plan for Your Reports and Avoiding Common Pitfalls in REPORT Procedure Coding." *Proceedings of the SAS Global Forum 2010 Conference.* Cary, NC: SAS Institute Inc. (Paper 133-2010). Available at http://support.sas.com/resources/papers/proceedings10/133-2010.pdf.

Booth, Allison McMahill. 2011. "Beyond the Basics: Advanced REPORT Procedure Tips and Tricks Updated for SAS® 9.2." *Proceedings of the SAS Global Forum 2011 Conference.* Cary, NC: SAS Institute Inc. (Paper 246-2011). Available at http://support.sas.com/resources/papers/proceedings11/246-2011.pdf.

Booth, Allison McMahill. 2012. "PROC REPORT Unwrapped: Exploring the Secrets behind One of the Most Popular Procedures in Base SAS® Software." *Proceedings of the Pharmaceutical Industry 2012 SAS Users Group Conference*. Cary, NC: SAS Institute Inc. (Paper TF20-SAS). Available at http://www.pharmasug.org/proceedings/2012/TF/PharmaSUG-2012-TF20-SAS.pdf.

Huntley, Scott, and Bari Lawhorn. 2010. "Getting the Right Report (Again): Your Compatibility Guide for ODS PDF 9.2." *Proceedings of the SAS Global Forum 2010 Conference.* Cary, NC: SAS Institute Inc. (Paper 035-2010). Available at http://support.sas.com/resources/papers/proceedings10/035-2010.pdf.

McMahill, Allison. 2007. "Beyond the Basics: Advanced PROC REPORT Tips and Tricks." *Proceedings of the SAS Global Forum 2007 Conference.* Cary, NC: SAS Institute Inc. (Paper 276-2007). Available at http://support.sas.com/rnd/papers/sgf07/sgf2007-report.pdf.

O'Connor, Daniel, and Scott Huntley. 2009. "Breaking New Ground with SAS® 9.2 ODS Layout Enhancements." *Proceedings of the SAS Global Forum 2009 Conference*. Cary, NC: SAS Institute Inc. (Paper 043-2009). Available at http://support.sas.com/resources/papers/proceedings09/043-2009.pdf.

Smith, Kevin D. 2006. "The TEMPLATE Procedure Styles: Evolution and Revolution." *Proceedings of the SAS Users Group International 31 Conference.* Cary, NC: SAS Institute Inc. (Paper 053-31). Available at http://www2.sas.com/proceedings/sugi31/053-31.pdf.

Zender, Cynthia L. 2010. "SAS® Style Templates: Always in Fashion." *Proceedings of the SAS Global Forum 2010 Conference*. Cary, NC: SAS Institute Inc. (Paper 033-2010). Available at http://support.sas.com/resources/papers/proceedings10/033-2010.pdf.

Zender, Cynthia L. 2009. "Tiptoe through the Templates." *Proceedings of the SAS Global Forum 2009 Conference*. Cary, NC: SAS Institute Inc. (Paper 227-2009). Available at http://support.sas.com/resources/papers/proceedings09/227-2009.pdf.

Zender, Cynthia L. 2008. "Creating Complex Reports." *Proceedings of the SAS Global Forum 2008 Conference*. Cary, NC: SAS Institute Inc. (Paper 173-2008). Available at http://www2.sas.com/proceedings/forum2008/173-2008.pdf.

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

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

Zender, Cynthia L. "CSSSTYLE: Stylish Output with ODS and SAS 9.2." *Proceedings of the SAS Global Forum 2009 Conference*. Cary, NC. SAS Institute Inc. (Paper 014-2009).Available at http://support.sas.com/resources/papers/proceedings09/014-2009.pdf.

ACKNOWLEDGMENTS

The authors would like to Michele Ensor and Linda Jolley, whose review comments made this a better paper. In addition, thanks are due to Kathy Underwood who edited this paper and improved its readability.

CONTACT INFORMATION

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

Cynthia L. Zender
SAS Institute, Inc.
SAS Campus Drive
Cynthia.Zender@sas.com

Allison Booth
SAS Institute, Inc.
SAS Campus Drive
Allison.Booth@sas.com

DOWNLOAD INFORMATION

To download the zip file of programs that accompany this paper, go to http://support.sas.com/rnd/papers. Then look for the paper title in the list of SAS Global Forum 2013 papers