Paper 59-26

ODS for Reporting with Style

Susan J. Slaughter, Avocet Solutions, Davis, CA

Lora D. Delwiche, University of California, Davis

Abstract

With the Output Delivery System (ODS) you no longer have to be satisfied with default output. Now you have many ways to produce eye-catching reports, with methods ranging from simple to complex. This paper starts with the simplest one, point-and-click menus, then shows how you can use the ODS statement to select an overall style, how you can control individual features of a report with the STLYE= option, and how you can create your own personalized style with PROC TEMPLATE. Along the way, we include a brief introduction to the TABULATE procedure, and show how to add images and highlighting to reports. The range of possibilities is enormous. Once you understand the capabilities of each feature, you can choose the technique that will give you the best results with the greatest ease.

The Output Delivery System

The Output Delivery System, introduced with Version 7 of the SAS® System, created a whole new way of handling output. In the past, each procedure produced its own output, but with ODS each procedure produces data, and sends that data to the Output Delivery System. Then the Output Delivery System decides where the data should go, and what it should look like when it gets there. Where the data go (such as to an HTML file or SAS data set) is called its "destination." What the data look like when it gets to its destination is determined by its template. These two concepts—destinations and templates—are important for understanding what you can do with ODS.

A template describes how ODS should format and present your data. There are two basic types of templates: style definitions and table definitions. A style definition specifies what the final report will look like (features such as color and font). A table definition specifies the basic structure of a report (such as what data and headers will be included). A few procedures, most notably TABULATE, REPORT, and PRINT, don't have ready-made table definitions. The syntax for these procedures acts as a custom template so a table definition is not needed.

The Data

The data used in this paper come from Jelly World, a fictitious manufacturer of gourmet jelly beans. Customers were asked to vote for their favorite flavor of jelly bean. For each vote, age and country of residence were also recorded. The data were typed into a file in comma-separated values (CSV) format, and read into a SAS data set using PROC IMPORT. A portion of the raw data appears in Table 1.

Table 1. A portion of the CSV file containing jelly bean data.

```
Country,Flavor,Age
Korea,cp,30
US,bn,13
US,mg,54
Korea,bb,32
Korea,fp,67
UK,cp,65
UK,fp,21
US,mg,43
US,bn,12
```

The following program reads the data file, jbdata.csv, and creates a SAS data set named JELLYBEANS, using the first line of the data file for the variable names. Then the program uses the PRINT procedure to list the first six observations of the data set. The results of the PROC PRINT appear in Table 2.

```
PROC IMPORT DATAFILE='c:\sugi26\jbdata.csv'
   OUT=jellybeans REPLACE;

TITLE 'Votes for Favorite Jelly Bean Flavor';
PROC PRINT DATA=jellybeans (OBS=6);
RUN;
```

Table 2. Listing output from PROC PRINT.

```
     Votes for Favorite Jelly Bean Flavor   1

     Obs    Country     Flavor        Age

       1    Korea       cp             30
       2    US          bn             13
       3    US          mg             54
       4    Korea       bb             32
       5    Korea       fp             67
       6    UK          cp             65
```

A Simple PROC TABULATE

To compare the popularity of each flavor, we need a summary report. Several procedures produce summary reports including PROC MEANS (and its nearly identical twin SUMMARY), FREQ, REPORT, and TABULATE. In this paper, we focus on PROC TABULATE because it is relatively easy to learn, it produces attractive tabular reports that are well-suited to the web, and its output can be customized using the powerful STYLE= option.

Here is a simple PROC TABULATE using the JELLYBEANS data:

```
TITLE 'Votes for Favorite Jelly Bean Flavor';
PROC TABULATE DATA=jellybeans FORMAT=5.;
   CLASS Country Flavor;
   TABLE Flavor, Country;
RUN;
```

In the PROC statement, the FORMAT= option specifies a format, in this case the standard numeric format with a length of 5, to be used for the data cells.

The CLASS statement lists the categorical variables (usually character variables) for SAS to use when dividing observations into groups.

The TABLE statement specifies how the report should be organized, and what numbers to compute. You can have more than one TABLE statement, and each TABLE statement will create a separate report. Each TABLE statement can specify up to three dimensions—a page dimension, a row dimension, and a column dimension, in that order—and these dimensions are separated by commas. If you specify only one dimension, SAS will use that for columns. With two dimensions, you get rows and columns. In this TABLE statement, there are only two dimensions. The report will contain one row for each value of Flavor, and one column for each value of Country. If there were a third dimension, the report would contain one page for each value of that variable.

A word of advice, you should always build your PROC TABULATE one feature at a time. That means you start with the column variable. When that is running correctly, then add the row variable. When the column and row dimensions are correct, then add the page variable, if you need one. Then you can add statistics and style options—one at a time. *Note that to avoid scrambling your table, you must insert the page and row variables in front of the column variable in your TABLE statement.*

The output from this procedure appears in Table 3. Notice that the row dimension is Flavor and the column dimension is Country. The values inside the cells of the table are simple counts, the number of votes for each combination of Flavor and Country. Since we didn't specify any statistics, SAS produced the default statistic for class variables, N.

Table 3. Listing output from a simple PROC TABULATE.

```
    Votes for Favorite Jelly Bean Flavor   2

     „ƒƒƒƒƒƒƒƒƒƒƒƒƒƒ…ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ†
     ,              ,     Country     ,
     ,              ‡ƒƒƒƒƒ…ƒƒƒƒƒ…ƒƒƒƒƒ‰
     ,              ,Korea, UK  , US  ,
     ,              ‡ƒƒƒƒƒˆƒƒƒƒƒˆƒƒƒƒƒ‰
     ,              ,  N  ,  N  ,  N  ,
     ‡ƒƒƒƒƒƒƒƒƒƒƒƒƒƒˆƒƒƒƒƒˆƒƒƒƒƒˆƒƒƒƒƒ‰
     ,Flavor        ,     ,     ,     ,
     ‡ƒƒƒƒƒƒƒƒƒƒƒƒƒƒ‰     ,     ,     ,
     ,bb            ,    5,    9,   11,
     ‡ƒƒƒƒƒƒƒƒƒƒƒƒƒƒˆƒƒƒƒƒˆƒƒƒƒƒˆƒƒƒƒƒ‰
     ,bn            ,    6,    3,   11,
     ‡ƒƒƒƒƒƒƒƒƒƒƒƒƒƒˆƒƒƒƒƒˆƒƒƒƒƒˆƒƒƒƒƒ‰
     ,cp            ,    8,    9,   19,
     ‡ƒƒƒƒƒƒƒƒƒƒƒƒƒƒˆƒƒƒƒƒˆƒƒƒƒƒˆƒƒƒƒƒ‰
     ,fp            ,    5,    4,    9,
     ‡ƒƒƒƒƒƒƒƒƒƒƒƒƒƒˆƒƒƒƒƒˆƒƒƒƒƒˆƒƒƒƒƒ‰
     ,mg            ,    5,   12,    9,
     ‡ƒƒƒƒƒƒƒƒƒƒƒƒƒƒˆƒƒƒƒƒˆƒƒƒƒƒˆƒƒƒƒƒ‰
     ,tf            ,    8,    8,   13,
     Šƒƒƒƒƒƒƒƒƒƒƒƒƒƒ‹ƒƒƒƒƒ‹ƒƒƒƒƒ‹ƒƒƒƒƒŒ
```

Destinations

The report in Table 3 is in the traditional form for SAS output. Because we didn't specify a destination, ODS sent this output to the default destination, the listing. The listing is what you see in the OUTPUT window in interactive mode, or in the LISTING or OUTPUT file, if you use batch mode. With ODS you can send your results to other destinations too. In release 8.1 the possible destinations include:

- LISTING
- OUTPUT
- HTML
- RTF
- PRINTER

The OUTPUT destination produces SAS data sets. HTML produces output in Hyper Text Markup Language, and RTF produces Rich Text Format. The PRINTER destination, by default, produces whatever type of output the current system printer uses such as PostScript. By default, PRINTER also sends output to your printer automatically. Using options, you can send PRINTER output to a file, or force it to produce PostScript output even if you don't have a PostScript printer. And, starting with 8.2, PRINTER can also produce PDF output.

Which destination you use depends on your needs and what printers are available to you. HTML gives you more choices (such as more colors) than other destinations, but LISTING, PRINTER, and RTF have advantages too. HTML reports are great for posting results on the Web, and for e-mailing reports to colleagues who can read HTML attachments. You can also import HTML files into Word documents, though some features may be lost. In this paper we used screen captures to show HTML output because we want you to see how the output looks on a screen. On the other hand, if you need to print long reports, PRINTER and RTF will give you logical page breaks. Because it's designed to be displayed on a screen rather than a page, HTML doesn't provide page breaks. RTF output becomes a Word table when opened in a Word document. As a Word table it can be easily resized and edited. And if you want consistency between screen and printed output, then the good old LISTING may be best.

Style Definitions

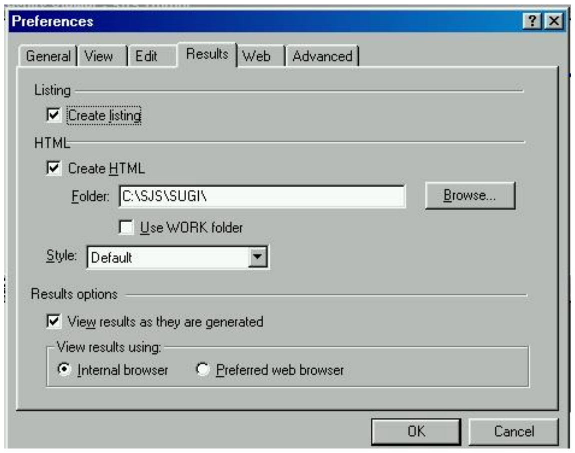
In addition to choosing a destination for your output, you can choose a style definition. Style definitions control things like color and font, and you can create your own style using PROC TEMPLATE. However, it's easier to choose from the long list of styles provided with the SAS System. The most commonly used styles include:

- DEFAULT
- MINIMAL
- BRICK
- STATDOC
- RTF
- PRINTER

Notice that RTF and PRINTER are names both of destinations and styles. Some styles work better with certain destinations than with others. DEFAULT is the default style for HTML output, RTF is the default style for RTF output, and PRINTER is the default style for PRINTER output.

Using Menus to Create HTML Output

If you are using interactive SAS in the Windows operating environment (starting with Version 7) or UNIX (starting with Version 8.1), then you can send your output to the HTML destination with a few clicks of your mouse. To do this, select **`Tools-Options-Preferences`** from the menus, click on the **`Results`** tab, then put a check in front of **`Create HTML`**.



Listing
[✓] Create listing
HTML
[✓] Create HTML
Folder: C:\SJS\SUGI\
[ ] Use WORK folder
Style: Default
Results options
[✓] View results as they are generated
View results using:
(●) Internal browser ( ) Preferred web browser

In the same window, you can also select a style. Now, when you run your programs, HTML output will automatically appear in the Results Viewer window. To save the HTML output to a file, make the Results Viewer window active (click on it), then click on the **`File`** menu, and choose **`Save As….`**

Using ODS Statements to Create HTML Output

If you are not using SAS in the Windows or UNIX operating environments, or if you want more control over the files SAS creates, you can use ODS HTML statements in your program. To generate HTML output all you need are two statements—one to open the HTML file, and one to close it. There are other types of ODS statements, and other types of HTML files that SAS can write, not to mention, other types of output. However, in this example we simply want to create an HTML file containing the report. You do that with these two basic statements:

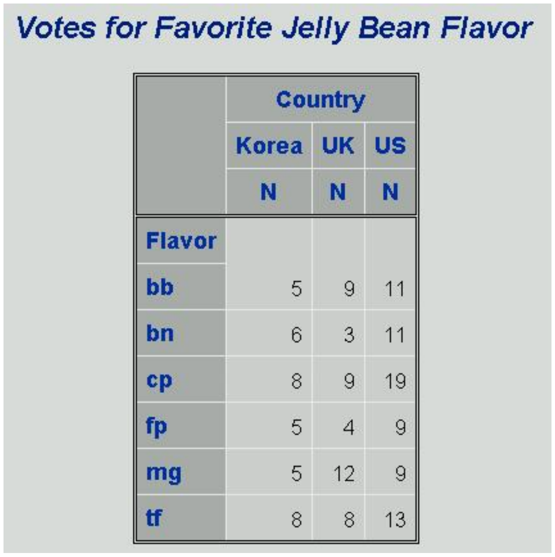
```
ODS HTML BODY = 'bodyfile.html'
   STYLE = style-name;

ODS HTML CLOSE;
```

The ODS HTML statement opens a file, specified in the BODY= option, which will contain the report. FILE= is an alias for BODY=. The ODS HTML CLOSE statement closes the file when the report is done. Generally speaking, you want to put the first statement just before the procedure, and the closing statement just after the RUN statement. Here is the TABULATE procedure used earlier, with ODS statements added.

```
ODS HTML BODY=‘c:\sugi26\jellybean1.html’;
TITLE 'Votes for Favorite Jelly Bean Flavor';
PROC TABULATE DATA=jellybeans FORMAT=5.;
   CLASS Country Flavor;
   TABLE Flavor, Country;
RUN;
ODS HTML CLOSE;
```

Table 4 shows a screen capture of the output from the Results Viewer window. Since we didn’t use the STYLE= option in the ODS statement, SAS used the DEFAULT style.

Table 4. HTML output from a simple PROC TABULATE.



Votes for Favorite Jelly Bean Flavor

	Country		
	Korea	UK	US
	N	N	N
Flavor			
bb	5	9	11
bn	6	3	11
cp	8	9	19
fp	5	4	9
mg	5	12	9
tf	8	8	13

A Compound PROC TABULATE

The previous PROC TABULATE contained only categorical variables. To add analysis variables (which must be numeric so you can compute statistics such as sums and means), list them in a VAR statement. You can have both a CLASS and a VAR statement, or just one, but all variables listed in a TABLE statement must appear in either a CLASS or a VAR statement.

By default, SAS will compute the number of observations in a group for CLASS variables (as it did in the preceding example), or the sum of the values for a variable listed in a VAR statement, but you can choose many other statistics. These are a few of the values that PROC TABULATE can compute and the keywords to request them.

Keyword	Value
ALL	adds a row, column, or page showing the total
MEAN	the arithmetic mean
MEDIAN	the median
N	the number of non-missing values
P90	the 90th percentile
PCTN	the percentage of observations for that group
PCTSUM	the percentage of a total sum for that group
SUM	the sum

You can insert these keywords into a TABLE statement right along with the variables. Also, variables and keywords can be crossed, concatenated, and grouped. To concatenate variables or keywords, simply separate them with a space. For example:

```
TABLE Flavor ALL;
```

To cross variables or keywords, separate them with an asterisk. For example:

```
TABLE Age * MEAN;
```

To group variables and keywords, enclose them in parentheses. This TABLE statement contains concatenating, crossing, and grouping.

```
TABLE Age * (MEAN  N);
```

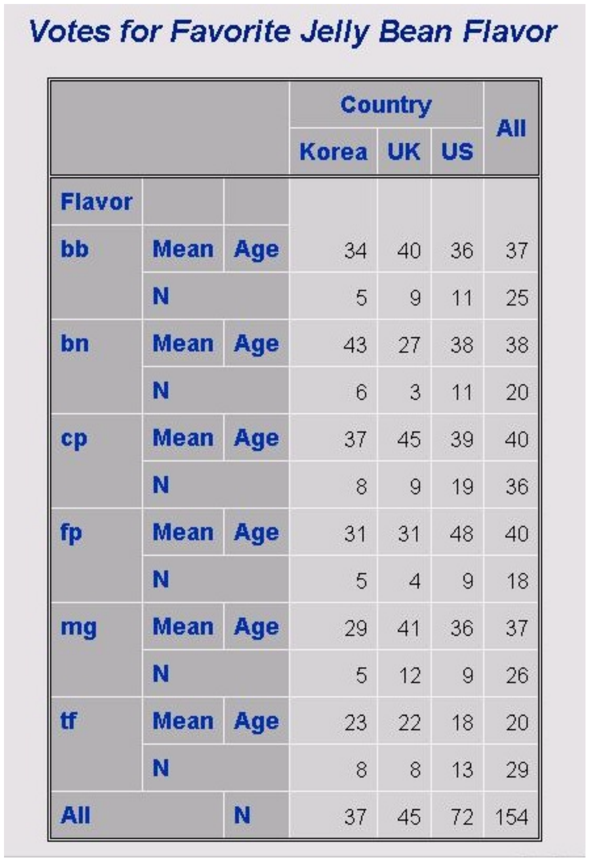
The following PROC TABULATE uses these features.

```
ODS HTML BODY='c:\sugi26\jellybean2.html';
TITLE 'Votes for Favorite Jelly Bean Flavor';
PROC TABULATE DATA=jellybeans FORMAT=5.;
   CLASS Country Flavor;
   VAR Age;
   TABLE Flavor * (MEAN*Age N) ALL, Country ALL;
RUN;
ODS HTML CLOSE;
```

This TABLE statement contains only two dimensions: rows and columns. In the row dimension, the variable, Flavor, is crossed with the mean of Age and with N. Flavor is also concatenated with the keyword ALL, so SAS will print the values for ALL below the values of Flavor. In the column dimension, the variable Country is concatenated with ALL so SAS will print the values of ALL alongside the values of Country.

When you run this program, SAS will produce the HTML output shown in Table 5. Because the STYLE= option was not specified in the ODS HTML statement, SAS used the DEFAULT style. If you are reading this paper in the printed version of the Proceedings, you will see a gray-scale version instead of color.

Table 5. HTML output from a compound PROC TABULATE with the DEFAULT style.



Votes for Favorite Jelly Bean Flavor

			Country			All
			Korea	UK	US	
Flavor						
bb	Mean	Age	34	40	36	37
	N		5	9	11	25
bn	Mean	Age	43	27	38	38
	N		6	3	11	20
cp	Mean	Age	37	45	39	40
	N		8	9	19	36
fp	Mean	Age	31	31	48	40
	N		5	4	9	18
mg	Mean	Age	29	41	36	37
	N		5	12	9	26
tf	Mean	Age	23	22	18	20
	N		8	8	13	29
All		N	37	45	72	154

If you don't like the DEFAULT style, you can choose another one by simply adding the STYLE= option to the opening ODS statement. In the preceding program, you would substitute this statement:

```
ODS HTML BODY = 'c:\sugi26\jellybean3.html'
   STYLE = BRICK;
```

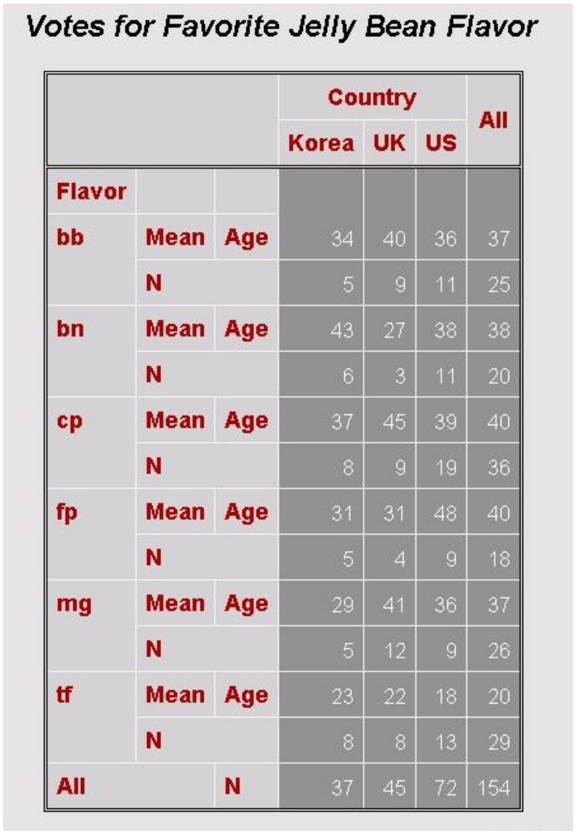
Table 6 shows the output in the BRICK style. If you are reading the printed Proceedings, these two tables may not look very different. The DEFAULT style has blue title and headers; the BRICK style has a black title and read headers.

Changing Headers in PROC TABULATE

These reports have good data, but the column headers are confusing. To start with, flavor names like "cp" are not terribly meaningful. "Cappuccino" would be much better. Also, the headers for the statistics are confusing. To replace the flavor codes with the names of the flavors, use PROC FORMAT to create a user-defined format. This procedure creates a format named $FLAV.

```
PROC FORMAT;
   VALUE $flav
   bb = 'Blue Berry'
   cp = 'Cappuccino'
   bn = 'Banana Bash'
   tf = 'Tutti Frutti'
   fp = 'Fruit Punch'
   mg = 'Margarita';
RUN;
```

Table 6. HTML output from a compound PROC TABULATE with the BRICK style.



Votes for Favorite Jelly Bean Flavor

			Country			All
			Korea	UK	US	
Flavor						
bb	Mean	Age	34	40	36	37
	N		5	9	11	25
bn	Mean	Age	43	27	38	38
	N		6	3	11	20
cp	Mean	Age	37	45	39	40
	N		8	9	19	36
fp	Mean	Age	31	31	48	40
	N		5	4	9	18
mg	Mean	Age	29	41	36	37
	N		5	12	9	26
tf	Mean	Age	23	22	18	20
	N		8	8	13	29
All		N	37	45	72	154

To apply the new format, you add a FORMAT statement to the PROC TABULATE. Don't confuse the FORMAT statement with the FORMAT= option. In PROC TABULATE, FORMAT statements specify formats for column or row headers. In contrast, the FORMAT= option specifies a format for the data cell values.

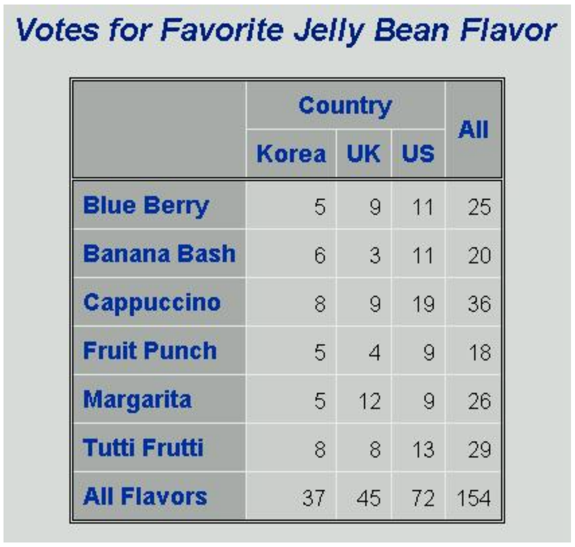
To customize headers for variable names and keywords, put an equal sign after the variable name or keyword in the TABLE statement followed by the new header between quotes. To eliminate a header, set it equal to blank (two quotes with nothing in between).

In the following PROC TABULATE, the mean of Age has been removed to simplify the report. A FORMAT statement has been added applying the $FLAV format to the variable flavor. And headers have been customized in the TABLE statement.

```
ODS HTML BODY='c:\sugi26\jellybean4.html';
TITLE 'Votes for Favorite Jelly Bean Flavor';
PROC TABULATE DATA=jellybeans FORMAT=5.;
   CLASS Country Flavor;
FORMAT Flavor $flav.;
   TABLE Flavor='' ALL='All Flavors',
      Country*N='' ALL*N='';
RUN;
ODS HTML CLOSE;
```

A screen shot of the results of the PROC FORMAT and PROC TABULATE appear in Table 7.

Table 7. HTML output with customized headers.



Votes for Favorite Jelly Bean Flavor

	Country			All
	Korea	UK	US	
Blue Berry	5	9	11	25
Banana Bash	6	3	11	20
Cappuccino	8	9	19	36
Fruit Punch	5	4	9	18
Margarita	5	12	9	26
Tutti Frutti	8	8	13	29
All Flavors	37	45	72	154

Producing Output for Other Destinations

To produce the preceding report for a different output destination, you would simply change the ODS statements. Here are the statements you would use to create RTF output. The RTF output appears in Table 8.

```
ODS RTF BODY = 'c:\sugi26\jellybean4.rtf';

ODS RTF CLOSE;
```

Table 8. RTF output with customized headers.

Votes for Favorite Jelly Bean Flavor

	Country			All
	Korea	UK	US	
Blue Berry	5	9	11	25
Banana Bash	6	3	11	20
Cappuccino	8	9	19	36
Fruit Punch	5	4	9	18
Margarita	5	12	9	26
Tutti Frutti	8	8	13	29
All Flavors	37	45	72	154

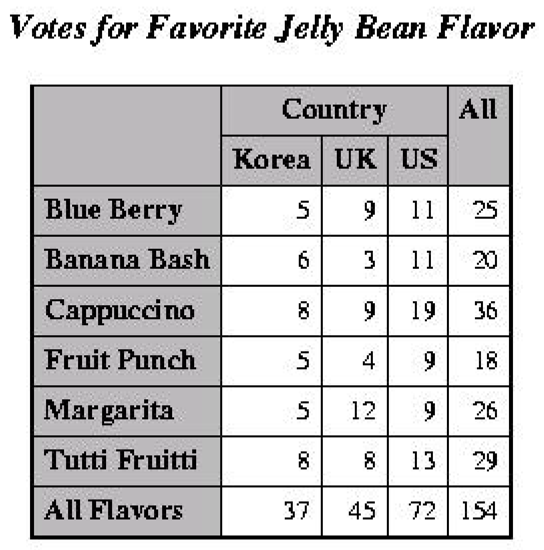
Notice that the colors and fonts are different in Table 8 than in 7. That's because no style was specified, and the default styles are different for HTML and RTF.

To produce output in PostScript format, use the ODS PRINTER statement.

```
ODS PRINTER PS BODY = 'c:\sugi26\jellybean4.ps';

ODS PRINTER CLOSE;
```

If your default printer uses PostScript, then you don't need the PS option. Adding the PS option tells SAS to create PostScript output regardless of the type of printer you have. The PostScript output in Table 9 looks a little rough because we've captured the screen image to put in a Word document. It looks sharper when printed by a PostScript printer.

Table 9. PRINTER output with customized headers.



Votes for Favorite Jelly Bean Flavor

	Country			All
	Korea	UK	US	
Blue Berry	5	9	11	25
Banana Bash	6	3	11	20
Cappuccino	8	9	19	36
Fruit Punch	5	4	9	18
Margarita	5	12	9	26
Tutti Fruitti	8	8	13	29
All Flavors	37	45	72	154

Adding Style to TABULATE Output

We have seen how you can use the STYLE= option on the ODS statement to change the overall look of your output. But, with the TABULATE procedure, you can also use the STYLE= option on various TABULATE statements to change the style of specific parts of your table. You can also use STYLE= options in PROC REPORT and, with release 8.2 of the SAS system, you can use the STYLE= option in PROC PRINT as well.

Table 10 shows which PROC TABULATE statements can use the STYLE= option and which parts of the table are affected.

Table 10. TABULATE statements that can use the STYLE= option.

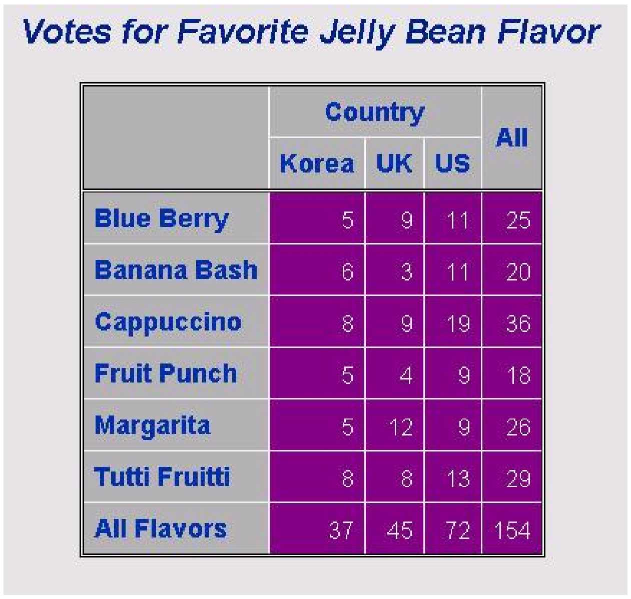
Statement	Table region affected
PROC TABULATE	Data cells
CLASS	Class variable name headings
CLASSLEV	Class level value headings
KEYWORD	Keyword headings
TABLE (as an option)	Table borders
TABLE (crossed with elements)	Element's data cell
VAR	Analysis variable name headings

The following program adds the STYLE= option to the PROC TABULATE statement while keeping the other statements the same as in the last example. Here we are changing the background color of the data cells to purple and the foreground (the text) to white.

```
ODS HTML BODY='c:\sugi26\sugipaper1.html';
TITLE 'Votes for Favorite Jelly Bean Flavor';
PROC TABULATE DATA=jellybeans
     STYLE={BACKGROUND=purple FOREGROUND=white};
  CLASS Country Flavor;
  TABLE Flavor='' ALL='All Flavors',
        Country*N='' ALL*N='';
  FORMAT Flavor $flav.;
RUN;
ODS HTML CLOSE;
```

The HTML output from this program appears in Table 11. This output uses the DEFAULT style. Notice how all the data cells are affected by the change.

Table 11. TABULATE HTML output using the DEFAULT style and the STYLE= option on the PROC TABULATE statement.


Votes for Favorite Jelly Bean Flavor

	Country			All
	Korea	UK	US	
Blue Berry	5	9	11	25
Banana Bash	6	3	11	20
Cappuccino	8	9	19	36
Fruit Punch	5	4	9	18
Margarita	5	12	9	26
Tutti Fruitti	8	8	13	29
All Flavors	37	45	72	154

There are many different attributes you can specify. Not all attributes are valid for all destinations. Table 12 lists some of the attributes you can change, and for which destinations they are valid. For a complete list, see the *SAS Procedures Guide, Version 8,* Chapter 2, "Fundamental Concepts for Using Base SAS Procedures".

Table 12. Selected attributes and their valid destinations.

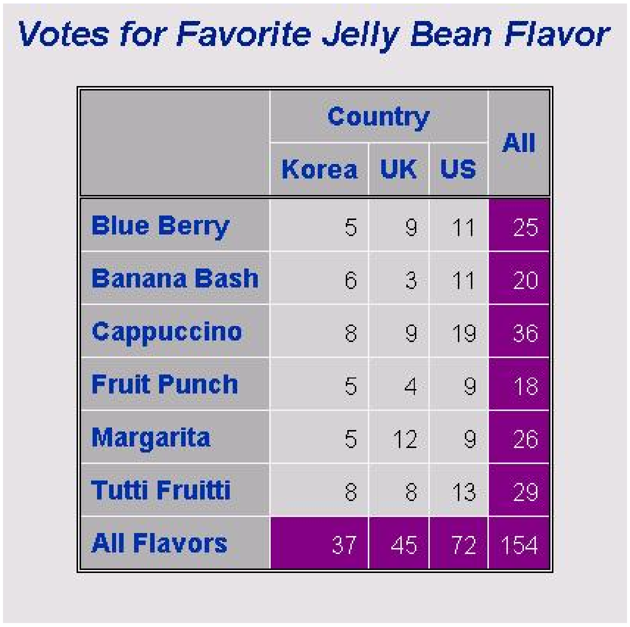
	Valid Output Destinations		
Attribute	**HTML**	**PRINTER**	**RTF**
BACKGROUND	X	X	X
BACKGROUNDIMAGE	X		
BORDERCOLOR	X	X	X
FLYOVER	X		
FONT	X	X	X
FOREGROUND	X	X	X
JUST	X	X	X
PREIMAGE	X		

The following program, instead of using the STYLE= option on the PROC TABULATE statement, crosses the STYLE= option with an element of the TABLE statement. The ALL keywords in the row and column dimensions are crossed with the style.

```
ODS HTML BODY='c:\sugi26\sugipaper2.html';
TITLE 'Votes for Favorite Jelly Bean Flavor';
PROC TABULATE DATA=jellybeans;
  CLASS Country Flavor;
  TABLE  Flavor='' ALL='All Flavors'*
   {STYLE={BACKGROUND=purple FOREGROUND=white}},
       Country*N='' ALL*N=''*
   {STYLE={BACKGROUND=purple FOREGROUND=white}};
  FORMAT Flavor $flav.;
RUN;
ODS HTML CLOSE;
```

The HTML output from this program appears in Table 13. Notice how now only the data cells corresponding to the ALL row and column have the new style. By placing the STYLE= option in the row or column dimension of the TABLE statement, you have more control over which data cells are affected by the style.

Table 13. TABULATE HTML output using the DEFAULT style and the STYLE= option crossed with the ALL keywords in the TABLE statement.


Votes for Favorite Jelly Bean Flavor

	Country			All
	Korea	UK	US	
Blue Berry	5	9	11	25
Banana Bash	6	3	11	20
Cappuccino	8	9	19	36
Fruit Punch	5	4	9	18
Margarita	5	12	9	26
Tutti Fruitti	8	8	13	29
All Flavors	37	45	72	154

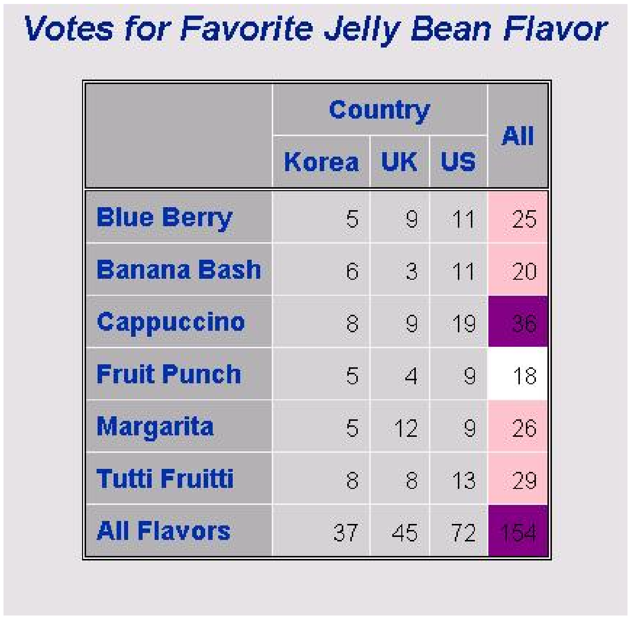
You can have even more control over the style of the data cells by using a feature known as traffic-lighting. Here the value of the data cell determines the style that the data cell displays. Using traffic-lighting, you can draw attention to important values, or highlight relationships between values. To add traffic-lighting, you create a user-defined format specifying different style attributes for each range of values, and then use that format as the value of the style element.

The following program defines a format, VOTEFMT, which assigns colors (white, pink and purple) to value ranges. The idea is that the low vote getters would show a white background, the high vote getters would be purple and the rest would be pink. To apply this format, you set the value of the BACKGROUND= style element equal to the VOTEFMT format. This style is crossed with the ALL table element in the column dimension.

```
PROC FORMAT;
  VALUE votefmt
         0-19 = 'white'
        20-29 = 'pink'
      30-HIGH = 'purple';
ODS HTML BODY='c:\sugi26\sugipaper3.html';
TITLE 'Votes for Favorite Jelly Bean Flavor';
PROC TABULATE DATA=jellybeans;
  CLASS Country Flavor;
  TABLE  Flavor='' ALL='All Flavors',
         Country*N='' ALL*N=''*
         {STYLE={BACKGROUND=votefmt.}};
  FORMAT Flavor $flav.;
RUN;
ODS HTML CLOSE;
```

Table 14 shows the HTML output from this program. Notice how the background color of the ALL column changes with the data values. (The pink background looks a lot like the default gray background when you print this in black and white).

Table 14. TABULATE HTML output using the DEFAULT style and traffic-lighting.



Votes for Favorite Jelly Bean Flavor

	Country			All
	Korea	UK	US	
Blue Berry	5	9	11	25
Banana Bash	6	3	11	20
Cappuccino	8	9	19	36
Fruit Punch	5	4	9	18
Margarita	5	12	9	26
Tutti Fruitti	8	8	13	29
All Flavors	37	45	72	154

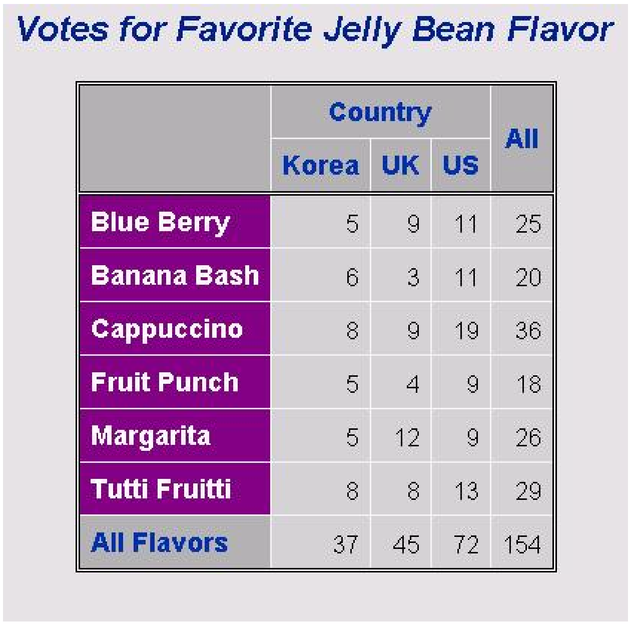
So far we have only changed the style of the data cells by placing the STYLE= option either in the PROC TABULATE statement or by placing it in a TABLE statement crossed with another element. You can also change the style of other regions of the table by placing the STYLE= option in other statements (see Table 10).

The CLASSLEV statement allows you to specify styles for the headings of the levels of the class variables. The following program uses the STYLE= option on the CLASSLEV statement to change the style of the headings for the class variable Flavor.

```
ODS HTML BODY='c:\sugi26\sugipaper4.html';
TITLE 'Votes for Favorite Jelly Bean Flavor';
PROC TABULATE DATA=jellybeans;
  CLASS Country Flavor;
  CLASSLEV Flavor /
    STYLE={BACKGROUND=purple FOREGROUND=white};
  TABLE  Flavor='' ALL='All Flavors',
         Country*N='' ALL*N='';
  FORMAT Flavor $flav.;
RUN;
ODS HTML CLOSE;
```

The HTML results of this program are shown in Table 15. Notice that none of the data cells are affected by the CLASSLEV statement, only the headings.

Table 15. TABULATE HTML output using the DEFAULT style and the STYLE= option on the CLASSLEV statement.



Votes for Favorite Jelly Bean Flavor

	Country			All
	Korea	UK	US	
Blue Berry	5	9	11	25
Banana Bash	6	3	11	20
Cappuccino	8	9	19	36
Fruit Punch	5	4	9	18
Margarita	5	12	9	26
Tutti Fruitti	8	8	13	29
All Flavors	37	45	72	154

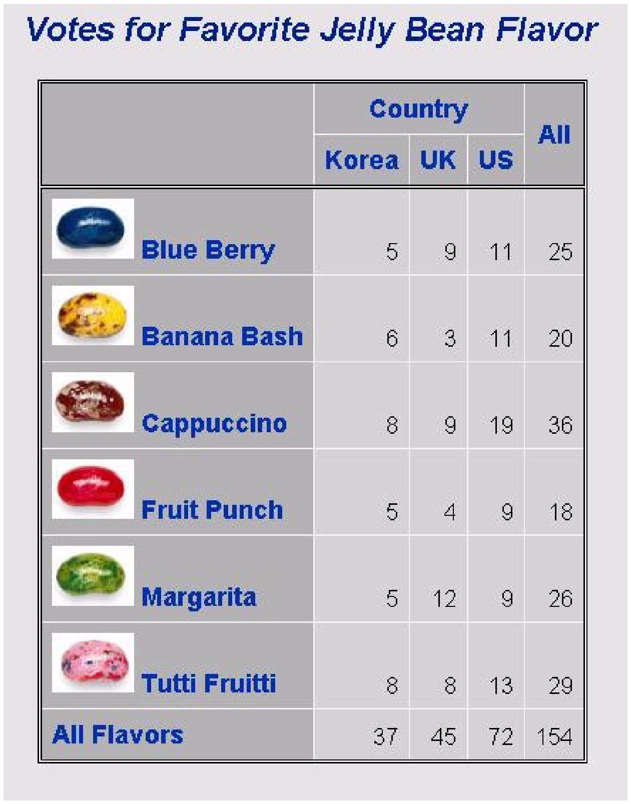
One of the interesting things you can do with the CLASSLEV statement and the HTML destination, is include images in the headings for the class variables. To do this, create a format that assigns the file names of your images to the different levels of your class variable. Then set the PREIMAGE style element equal to this format in the STYLE= option.

The following program creates a format, $IMAGE, that assigns file names to the different values of the Flavor variable. Then the PREIMAGE style element is set equal to the $IMAGE format in the CLASSLEV statement.

```
PROC FORMAT;
  VALUE $image
    bb='c:\sugi26\blueberry.jpg'
    cp='c:\sugi26\cappuccino.jpg'
    fp='c:\sugi26\hawaiian.punch.jpg'
    mg='c:\sugi26\margarita.jpg'
    bn='c:\sugi26\top.banana.jpg'
    tf='c:\sugi26\tutti.fruitti.jpg';

ODS HTML BODY='c:\sugi26\sugipaper5.html';
TITLE 'Votes for Favorite Jelly Bean Flavor';
PROC TABULATE DATA=jellybeans;
  CLASS Country Flavor;
  CLASSLEV Flavor /STYLE={PREIMAGE=$image.};
  TABLE  Flavor='' ALL='All Flavors',
         Country*N='' ALL*N='';
  FORMAT Flavor $flav.;
RUN;
ODS HTML CLOSE;
```

The HTML output resulting from this program is shown in Table 16. You can see the pictures of the various jelly beans are now included in the headings of the table. Obviously you would want to keep your images pretty small if you are going to include them in a table. Jelly beans, it turns out, fit quite nicely in a table. This feature only works for the HTML destination, but as you can see, it is possible to include these HTML files in a Word document if you turn your HTML file into an image. (We used print screen, then used the Image Editor in SAS to crop the image and save it as a jpg file.) If you simply import the HTML file into Word, then you won't see the images.

Table 16. TABULATE HTML output using the DEFAULT style and the PREIMAGE attribute of the STYLE= option of the CLASSLEV statement.



Votes for Favorite Jelly Bean Flavor

	Country			All
	Korea	UK	US	
Blue Berry	5	9	11	25
Banana Bash	6	3	11	20
Cappuccino	8	9	19	36
Fruit Punch	5	4	9	18
Margarita	5	12	9	26
Tutti Fruitti	8	8	13	29
All Flavors	37	45	72	154

Creating Your Own Style

You have a lot of control over style using the STYLE= options in PROC TABULATE, but you can only affect the table itself using these options. If you want to change the style of other elements of your output, such as the overall background, titles, footnotes, or procedure titles, then you need to choose a different style template. As mentioned earlier, there are many different styles that come with SAS that you can choose from. But sometime you may what to develop your own style. You may want your company logo to appear on every piece of output, you may want to always use the same color scheme, or your company may want you to use a particular font.

To create your own style, use the TEMPLATE procedure. The TEMPLATE procedure basically does two different things: creates or modifies style definitions, and creates or modifies table definitions. A **table** definition tells a SAS procedure how to layout the output: how many columns there are, how they should be labeled, and what format should be used for the data. A **style** definition determines the colors to use, which fonts, and the characteristics of the fonts such as weight and width. We will talk about modifying a style definition. Table definitions are entirely different animals, and in fact, the TABULATE procedure is one of the few procedures (also REPORT and PRINT) that does not use a table definition to format its output.

You can start out from scratch and create a new style definition, but if you can find a style that is close to what you want, and modify it, that is probably easier. You actually don't want to make changes to any of the styles that come with SAS, but it is easy to create new styles that are replicas of existing styles. PROC TEMPLATE uses a concept called inheritance, where styles and style elements can inherit traits from other styles and style elements. This makes programs shorter, because you do not have to specify every attribute for every element if there are other elements with the desired traits. But PROC TEMPLATE programs can be difficult to follow if the element you wish to modify, inherits traits from another element, which in turn inherits traits from another element, which in turn inherits traits from another element.

A style definition consists of style elements. A style element has attributes and those attributes are assigned values. For example, the DEFAULT style has a style element named HeadersAndFooters. The HeadersAndFooters element has the attributes: Font, Foreground, and Background all of which are assigned values such as Courier, White, and Purple.

If you want to look at the PROC TEMPLATE code for the styles that SAS provides (such as the DEFAULT style), you can click on the **`Results`** window, then choose **`Templates`** from the **`View`** menu. Open the Sashelp.Tmplmst folder and then the Styles folder. Then you can double click on the style that you want to view. All the templates SAS provides are in the Sashelp.Tmplmst folder. This folder is write protected so you can't accidentally write over a SAS template. There is also a Sasuser.Tmplmst folder – this is where any user defined templates are stored (by default).

When used to define a style definition, the TEMPLATE procedure, in its simplest form, has just four statements. Here is an example of a complete TEMPLATE procedure that creates a new style called JELLYBEAN.

```
PROC TEMPLATE;
  DEFINE STYLE styles.jellybean;
    PARENT=styles.default;
  END;
RUN;
```

The DEFINE STYLE statement says that you are going to create a new style definition. The name of the new style has two parts: STYLES.JELLYBEAN. This notation is similar to the way permanent SAS data sets are referenced. The first part of the name is the storage location (STYLES is a pre-defined location in the Sasuser.Tmplmst storage area for user defined templates), and the second part, JELLYBEAN, is the name you create for your style. The PARENT= statement says that you want your new style to inherit all traits from the DEFAULT style. So, the JELLYBEAN style is an exact replica of the DEFAULT style. By default, SAS first looks in the Sasuser.Tmplmst folder for templates. If it does not find the storage location and template in the Sasuser.Tmplmst folder, then it will look in the Sashelp.Tmplmst folder. So, in this case, the STYLES.JELLYBEAN template will get stored in Sasuser.Tmplmst (the first place SAS looks), and since there is no STYLES.DEFAULT template in Sasuser.Tmplmst, SAS will look in Sashelp.Tmplmst for the DEFAULT style.

Of course you don't normally want to create an exact copy of a style; you want to make some change to the style. There are an amazing number of style elements that you can change in the DEFAULT style. This is good, because it gives you a lot of control, but it can be difficult to locate the style element that you want to change. Table 17 lists **some** of the style elements from the DEFAULT style that you may want to change. Style elements inherit attributes from the style element listed in the column to the left of the style element. Please note that this table is not complete, there are many other style elements in the DEFAULT style. Also, for some of the elements in this table, there are additional elements that inherit from them. For example the elements Header and Footer inherit from HeadersAndFooters.

Table 17. Selected style elements from the DEFAULT style. Elements inherit attributes from the element listed in the column to the left.

Container	Cell	Data HeadersAndFooters
	Output	Table Graph
	TitlesAndFooters	ProcTitle SystemTitle SystemFooter
	Document	Body Pages Contents Frame

The first change we want to make to the DEFAULT style, is to use an image for the background of the HTML body file. To do this, we need to change the attributes of the Body element. The REPLACE statement allows you to replace the definition of a style element. You can see from Table 17 that Body inherits from Document (which in turn inherits from Container). We want our new Body element to inherit attributes from Document, so we start the REPLACE statement with:

```
 REPLACE Body FROM Document /
```

Style elements don't have to inherit from the same element they inherited from in the parent style, but you will have more predictable results if you always specify the same element. Next, we specify attributes for the Body element after the slash (/). In this case, we just want to set the background of the HTML body file to an image. So we set the BACKGROUNDIMAGE attribute equal to the filename of our image file.

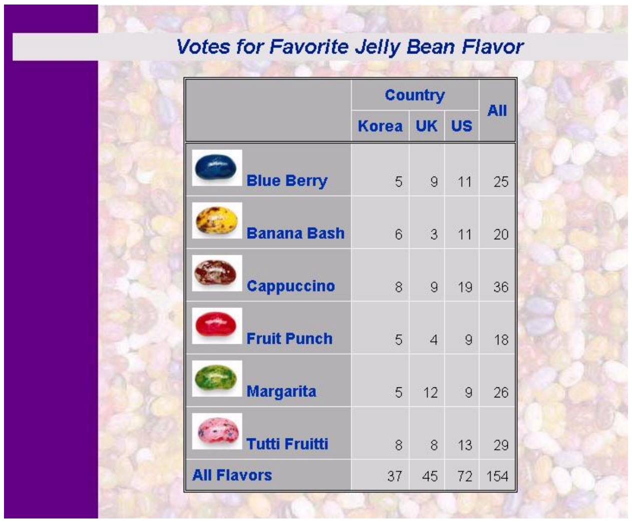
```
PROC TEMPLATE;
  DEFINE STYLE styles.jellybean;
  PARENT=styles.default;
  REPLACE Body FROM Document/
    BACKGROUNDIMAGE='c:\sugi26\beans.jpg';
  END;
RUN;
```

Now, to use the new style you just created, you need to specify it in the STYLE= option of the ODS statement. Here in the ODS HTML BODY statement we set STYLE= to JELLYBEAN. Then we run the same PROC TABULATE statements.

```
ODS HTML BODY='c:\sugi26\sugipaper6.html'
    STYLE=jellybean;
TITLE 'Votes for Favorite Jelly Bean Flavor';
PROC TABULATE DATA=jellybeans;
  CLASS Country Flavor;
  CLASSLEV Flavor / STYLE={PREIMAGE=$image.};
  TABLE  Flavor='' ALL='All Flavors',
         Country*N='' ALL*N='';
  FORMAT Flavor $flav.;
RUN;
ODS HTML CLOSE;
```

Table 18 shows the output from the PROC TABULATE using the new JELLYBEAN style.

Table 18. TABULATE HTML output using the JELLYBEAN style.



Votes for Favorite Jelly Bean Flavor

	Country			All
	Korea	UK	US	
Blue Berry	5	9	11	25
Banana Bash	6	3	11	20
Cappuccino	8	9	19	36
Fruit Punch	5	4	9	18
Margarita	5	12	9	26
Tutti Fruitti	8	8	13	29
All Flavors	37	45	72	154

Now suppose we want to match the color of the heading text to the purple color of the solid bar on the left of the background image. (In the printed proceedings this looks black, but it really is purple). We also want to change the font of the headings. To make changes to the heading text, we can alter the HeadersAndFooters element. The REPLACE statement starts:

```
  REPLACE HeadersAndFooters FROM Cell/
```

since HeadersAndFooters inherits from Cell in the DEFAULT style. The font, foreground color, and background color are all defined in the HeadersAndFooters element in the DEFAULT style. So, we need to specify all those attributes in the REPLACE statement. If we only specify for example the font, then we will get whatever foreground and background colors HeadersAndFooters inherits from Cell in the DEFAULT style which is probably not what you want. Here is the complete REPLACE statement with the FONT, BACKGROUND and FOREGROUND attributes defined:

```
REPLACE HeadersAndFooters FROM Cell/
    FONT= ("Comic Sans MS,Helvetica,Helv",4,BOLD)
    BACKGROUND=silver
    FOREGROUND=#640082;
```

The FONT attribute specifies Comic Sans MS as the first choice for font. This is probably not a font you would normally choose for HTML because it is not that commonly available, but for this paper we wanted to choose a font that was different looking enough that you could see how the style has changed. We also use the #640082 color for the foreground, which is the RGB notation for the same purple color that appears in the background image.

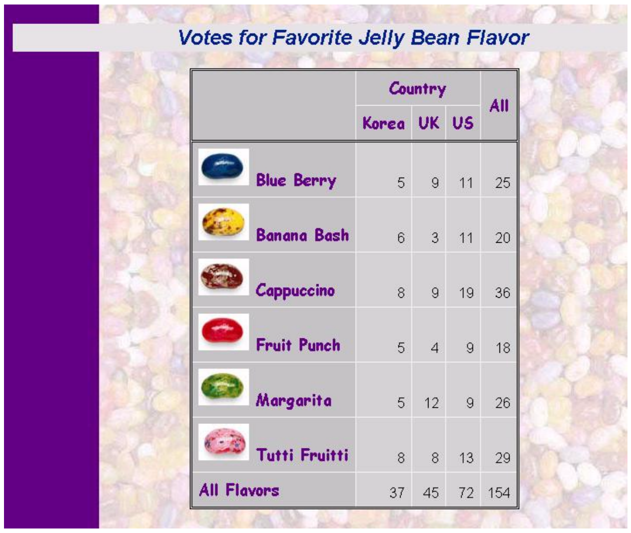
Because we used the REPLACE statement, all elements that inherit from HeadersAndFooters (such as Header and Footer) will inherit the values of these attributes. You can also use the STYLE statement to create style elements. But, if you use the STYLE statement instead of the REPLACE statement, then there is no inheritance of the new attributes.

Here is the complete program that creates the new JELLYBEAN2 style and the same PROC TABULATE statements that were used in the last example.

```
PROC TEMPLATE;
  DEFINE STYLE styles.jellybean2;
  PARENT=styles.default;
  REPLACE Body FROM Document/
    BACKGROUNDIMAGE='c:\sugi26\beans.jpg';
  REPLACE HeadersAndFooters FROM Cell/
    FONT= ("Comic Sans MS,Helvetica,Helv",4,BOLD)
    BACKGROUND=silver
    FOREGROUND=#640082;
  END;
RUN;

ODS HTML BODY='c:\sugi26\sugipaper7.html'
    STYLE=jellybean2;
TITLE 'Votes for Favorite Jelly Bean Flavor';
PROC TABULATE DATA=jellybeans;
  CLASS Country Flavor;
  CLASSLEV Flavor / STYLE={PREIMAGE=$image.};
  TABLE  Flavor='' ALL='All Flavors',
         Country*N='' ALL*N='';
  FORMAT Flavor $flav.;
RUN;
ODS HTML CLOSE;
```

Table 19 shows the TABULATE output using the new JELLYBEAN2 style.   Notice that the headings in the table have the new font and color.

Table 19. TABULATE HTML output using the JELLYBEAN2 style.



Votes for Favorite Jelly Bean Flavor

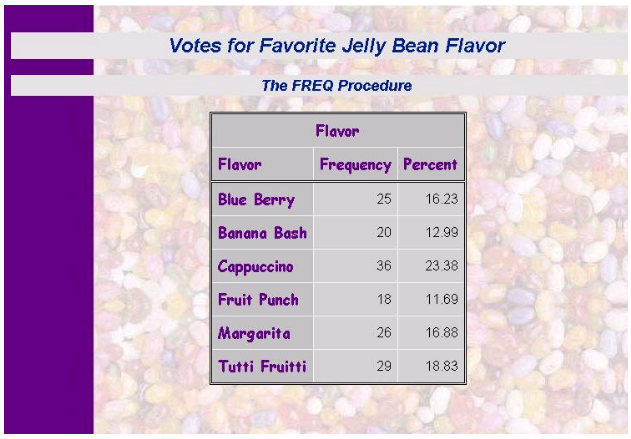
	Country			All
	Korea	UK	US	
Blue Berry	5	9	11	25
Banana Bash	6	3	11	20
Cappuccino	8	9	19	36
Fruit Punch	5	4	9	18
Margarita	5	12	9	26
Tutti Fruitti	8	8	13	29
All Flavors	37	45	72	154

You could have also made these font and color changes using the STYLE= options in PROC TABULATE. But when you create a new style, you can apply it to any procedure output. The following program uses this new JELLYBEAN2 style for a PROC FREQ output.

```
ODS HTML BODY='c:\sugi26\sugipaper8.html'
    STYLE=jellybean2;
TITLE 'Votes for Favorite Jelly Bean Flavor';
PROC FREQ DATA=jellybeans;
  TABLE Flavor/NOCUM;
  FORMAT Flavor $flav.;
RUN;
ODS HTML CLOSE;
```

Table 20 shows the results of the PROC FREQ using the JELLYBEAN2 style. See how we have the color, font, and background that we want by just using the new style in the ODS statement.

Table 20. FREQ HTML output using the JELLYBEAN2 style.



Votes for Favorite Jelly Bean Flavor

The FREQ Procedure

Flavor		
Flavor	Frequency	Percent
Blue Berry	25	16.23
Banana Bash	20	12.99
Cappuccino	36	23.38
Fruit Punch	18	11.69
Margarita	26	16.88
Tutti Fruitti	29	18.83

Conclusions

The Output Delivery System opens up a whole new world of ways you can style your output. Some methods are quickly implemented, such as changing the style in the results window. Other methods, such as creating your own style, may take a little more effort, but the rewards are sweet.

References

Fehlner, William (1999). Making the Output Delivery System (ODS) Work for You. *Proceedings of the Twenty-fourth Annual SAS Users Group International Conference*. SAS Institute Inc., Cary, NC.

Olinger, Chris (2000). ODS for Dummies. *Proceedings of the Eighth Annual Western Users of SAS Software*. SAS Institute Inc., Cary, NC.

SAS Institute Inc. (1999). *SAS Procedures Guide, Version 8*. SAS Institute Inc., Cary, NC.

SAS Institute Inc. (1999). *The Complete Guide to the SAS® Output Delivery System, Version 8.* SAS Institute Inc., Cary, NC.

About the Authors

Susan Slaughter and Lora Delwiche are authors of *The Little SAS Book: A Primer* published by SAS Institute. They may be contacted at:

Susan J. Slaughter	(530) 756-8434 susan@avocetsolutions.com
Lora D. Delwiche	(530) 752-6285 llddelwiche@ucdavis.edu