

FAQ: PROC TEMPLATE

Katherine Burgess, Sanford Research, Sioux Falls, SD
 Ashley Miller, Sanford Research, Sioux Falls, SD

ABSTRACT

PROC TEMPLATE can be a very useful tool in SAS. It provides many options for organizing and presenting your ODS output. We will provide a brief introduction to PROC TEMPLATE and demonstrate its use through SAS table and style templates. Step by step examples of both will be given focusing on basic modifications such as changing fonts, colors, and traffic lighting. Not only will we discuss how to EDIT and DEFINE your own template from a beginner's perspective, we will also demonstrate how easy it is to save your template so that you can use it in future projects, and share your template across multiple users on a single network.

INTRODUCTION

How many times have you generated output in SAS and then wished there was a way to change the overall appearance? Have you ever wondered if there was a way to customize the appearance of your output in SAS? We have all been asked to generate reports from output in SAS, but how often do you find yourself making style and table changes in Word?

In this paper we will discuss the use of the TEMPLATE procedure, with a focus on style and table templates. PROC TEMPLATE is an underutilized tool that allows you to customize output into a specific format. Once you have a customized template, you can then save and apply the template to reports needed in the future. Previous papers have shown how to edit and define your own template, what we hope to do is build on how easy it is to create your template, save it, and use it again with other templates, and across multiple users.

PROC TEMPLATE is an efficient tool for creating visually appealing reports in SAS. As it is a powerful tool to use, it is intimidating at first glance. This paper will introduce you to PROC TEMPLATE; how it is used, and where templates are located and stored in SAS. Once this foundation is covered we will introduce you to two easy examples of both table and style templates. The goal of this paper is to demonstrate how easy it is to go from standard output in Figure 1 to more structured and formatted output using both style and table template as seen in Figure 2.

Figure 1. PROC MEANS Output

The SAS System
The MEANS Procedure

Product	N Obs	Variable	Label	Sum
Boot	52	Sales	Total Sales	\$2,350,543.00
		Returns	Total Returns	\$98,622.00
Men's Casual	45	Sales	Total Sales	\$7,933,707.00
		Returns	Total Returns	\$311,035.00
Men's Dress	50	Sales	Total Sales	\$5,507,243.00
		Returns	Total Returns	\$164,099.00
Sandal	49	Sales	Total Sales	\$868,436.00
		Returns	Total Returns	\$38,170.00
Slipper	52	Sales	Total Sales	\$6,175,834.00
		Returns	Total Returns	\$209,940.00
Sport Shoe	51	Sales	Total Sales	\$651,467.00
		Returns	Total Returns	\$25,179.00
Women's Casual	45	Sales	Total Sales	\$4,137,861.00
		Returns	Total Returns	\$131,394.00
Women's Dress	51	Sales	Total Sales	\$6,226,475.00
		Returns	Total Returns	\$193,653.00

Figure 2. PROC MEANS utilizing PROC TEMPLATE

Style Template Example
Including Table Template as well...

Based on PROC MEANS			
Type of Shoe	Calculated Totals		
	Stores	Sales	Returns
Boot	52	\$2,350,543.00	\$98,622.00
Men's Casual	45	\$7,933,707.00	\$311,035.00
Men's Dress	50	\$5,507,243.00	\$164,099.00
Sandal	49	\$868,436.00	\$38,170.00
Slipper	52	\$6,175,834.00	\$209,940.00
Sport Shoe	51	\$651,467.00	\$25,179.00
Women's Casual	45	\$4,137,861.00	\$131,394.00
Women's Dress	51	\$6,226,475.00	\$193,653.00

WHAT IS PROC TEMPLATE?

PROC TEMPLATE is a SAS procedure for output structuring. Just as PROC MEANS computes summary statistics and PROC REPORT displays tables, PROC TEMPLATE is a procedure for controlling the appearance and structure of SAS output. Think of it as the framework for the organization and presentation of your data.

Just as SAS provides different procedures for different purposes, SAS also provides different templates for different uses. There are five different types of templates provided by SAS, and each has its own specific purpose. The four template types are:

- Table: allows structural changes to your output
- Style: allows formatting changes to your output
- Tagset: allows markup language tags specific to your output file destination
- Graph: allows structural changes to your graphical output

For the remainder of this paper we will focus only on table and style templates, as covering all four would be beyond the confines of this paper.

WHAT ARE THE USES OF PROC TEMPLATE?

The table and style template components of PROC TEMPLATE can be used for a wide range of purposes. For example, if you work for a company that wishes to create a method of streamlining all reports so that they look the same and are reproducible, either or both style and table templates would be helpful. You can use PROC TEMPLATE to incorporate company logos, colors, or letterhead. Perhaps you have a paper to submit to a journal that has specifications for how your output should be displayed. PROC TEMPLATE can do this for you. Do you work with a group of people that would benefit from sharing templates? PROC TEMPLATE has the capability of being stored and easily shared with multiple users on the same network for consistent reporting across individuals.

PROC TEMPLATE gives you the flexibility of customizing your own unique template or modifying an existing template. By "modifying an existing template", we mean a template that is already made and provided by SAS. All procedures and data steps (except PRINT, TABULATE, and REPORT) in SAS come with a built-in default template, which SAS uses to present output by default. Sometimes it is easier to make changes to this default template than it is to customize a new template. The difference in defining a new template or editing an existing one will be discussed later in this paper with examples. The fundamental point we wish to make here is that in order to make these changes, you need to understand what templates your SAS procedure is currently using.

Figure 3. SAS TEMPLATE Examples



HOW DO YOU KNOW WHAT TEMPLATE SAS IS USING?

SAS provided a method of 'tracing' your output with the functionality of ODS. To determine what template SAS is using to create your output, you will need to use the ODS TRACE ON/OFF statement. For example, if you wanted to determine the template being used in the MEANS procedure you would submit this code:

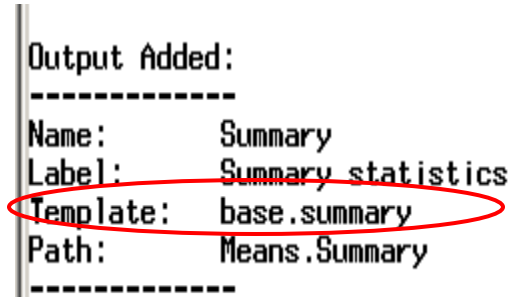
Example 1. PROC MEANS Using ODS TRACE ON/OFF

```
ODS TRACE ON;

PROC MEANS DATA=sashelp.shoes SUM;
    VAR sales returns;
    CLASS product;
RUN;

ODS TRACE OFF;
```

Figure 4. ODS TRACE ON/OFF in Log



```
Output Added:
-----
Name:        Summary
Label:       Summary statistics
Template:    base.summary
Path:        Means.Summary
-----
```

Figure 4 displays the ODS TRACE statement output in your log. You will notice that under 'OUTPUT ADDED' is the statement, 'TEMPLATE: BASE . SUMMARY'(circled above). This tells you that, for the MEANS procedure, the template SAS uses is called **BASE . SUMMARY**. So now what do you do with that information? The next step is to locate the template **BASE . SUMMARY**. Once you locate the template being used by SAS, you can access the code and make any changes. Knowing where the templates are stored allows you to view all the templates provided by SAS. This is particularly useful with SAS 9.3, as this version comes with over 50 default style templates.

WHERE CAN YOU LOCATE THE TEMPLATES?

SAS saves templates in locations known as item stores, which are created in the TEMPLATE procedure. These item stores can be default stores that come with SAS or can be user defined. There are two default item stores:

- 1) **SASHELP.TMPLMST**: location of all templates supplied with SAS
- 2) **SASUSER.TEMPLAT**: default location for templates that you create and modify

To locate and view the contents of an item store you can use either 1) the SAS windowing environment, 2) the SAS window command ODSTEMPLATES or 3) the TEMPLATE procedure.

Item stores can be made up of many different levels known as directories and subdirectories. These levels are then used to locate the correct template to be used in your procedure. For example, let's locate the **BASE . SUMMARY** template from our previous example. **BASE . SUMMARY** is a two level name, where **SASHELP.TMPLMST** store has directory **BASE** which contains template **SUMMARY**. Using the SAS windowing environment, if you were to go to **View** → **Results** → **View** → **Templates** (as shown in Figure 22 in the appendix), you would see:

Figure 5. Locating template stores by the SAS windowing environment

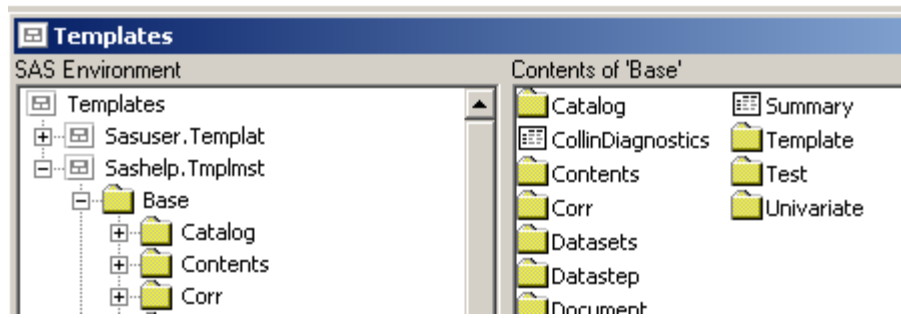


Figure 5 illustrates the location of the templates provided by SAS. What if you wanted a listing of the templates you have stored in `SASUSER.TEMPLAT`? You could locate the template store using the same method as shown in Figure 5, or you could code for an output listing of a specific store. For example, if you wanted to see a listing of all the directories and templates that you have created and saved to `SASUSER.TEMPLAT` you could run the code:

```
PROC TEMPLATE;
LIST/STORE=SASUSER.TEMPLAT;
RUN;
```

The `LIST/STORE=SASUSER.TEMPLAT` statement tells SAS that you want an output listing of all templates located in the `SASUSER.TEMPLAT` store. Figure 6 displays the listing from the SAS output of the above code. Notice that the system tells you what item store it is making a listing of as well as what type each item is within the store.

Figure 6. Output listing of `SASUSER.TEMPLAT` store

Listing of: `SASUSER.TEMPLAT`
Path Filter is: *
Sort by: `PATH/ASCENDING`

Obs	Path	Type
1	Base	Dir
2	My	Dir
3	My.Formatcolumn	Column
4	Styles	Dir
5	Styles.Styletemp1	Style
6	Styles.Styletemp2	Style
7	Styletemp1	Style
8	Tabtempl	Table
9	Tagsets	Dir
10	Tagsets.PT_alt	Tagset
11	Tagsets.Pt	Tagset
12	Tagsets.Show_class	Tagset
13	Tagsets.Show_style	Tagset

For example, Figure 6 shows that `STYLES.STYLETEMP1` is a style template within the directory '`STYLES`' of the item store `SASUSER.TEMPLAT`. Notice that '`DIR`' under '`TYPE`' specifies that the item in the path is a directory, and the first level in the name of all items within that directory. So within the directory '`STYLES`' we have two style templates: `STYLETEMP1` and `STYLETEMP2` (as shown by arrows).

WHAT IS TABLE TEMPLATE?

Now that you know where and how templates are stored, let's examine what you can do with PROC TEMPLATE starting with table template. Table template is what allows you to customize the appearance of your SAS output such as a table from a MEANS procedure. Each procedure has a table template specifically used for the ODS output.

The appearance of ODS output can be created or modified by using either a DEFINE or EDIT statement. An EDIT statement modifies a current table template while a DEFINE statement creates a new table template. When designing table templates there are certain elements and attributes you can modify that affect the appearance of your table, giving you the ability to customize the output. One can easily modify the table elements such as columns, headers, and footers. Table attributes, which order table elements, consist of things such as text appearance, justification of text, and formats and can be modified as well.

WHAT IS THE SYNTAX FOR TABLE TEMPLATE?

After learning the basics of table template, the next step is putting it to use. In order to use the table template, you need to start with the syntax. The basic syntax is:

```
PROC TEMPLATE;
    DEFINE TABLE table-name;
        statements/attributes/headers/footers/columns...
END;
RUN;
```

Now let's put this syntax to use with some examples of what you can do to customize a table.

HOW DO YOU CHANGE HEADERS AND CUSTOMIZE COLUMNS?

Have you wanted to quickly and easily change the headers in your table? Or format the columns so your table is more pleasing to the eye? Below is a step by step guide for doing just that.

Example 2. PROC TEMPLATE Table Template

```
PROC TEMPLATE;
    DEFINE TABLE tabtemp1; (1)
        COLUMN product _freq_ sales returns;
        HEADER header1 header2; (2)
        DEFINE HEADER header1; (3)
            TEXT 'Based on PROC MEANS';
            JUST=center;
            STYLE=HEADER{BACKGROUND=teal};
        END;
        DEFINE HEADER header2; (4)
            TEXT 'Calculated Totals';
            JUST=center;
            START=_freq_;
            STYLE=HEADER{BACKGROUND=yellow};
            END=returns;
        END;
        DEFINE product;
            HEADER="Type of Shoe";
        END;
        DEFINE _freq_;
            HEADER="Stores";
        END;
        DEFINE sales; (5)
            HEADER="Sales";
            FORMAT=DOLLAR14.2;
```

Figure 7. Table Template Columns

Based on PROC MEANS			
Type of Shoe	Calculated Totals		
	Stores	Sales	Returns
Boot	52	\$2,350,543.00	\$98,622.00
Men's Casual	45	\$7,933,707.00	\$311,035.00
Men's Dress	50	\$5,507,243.00	\$164,099.00
Sandal	49	\$868,436.00	\$38,170.00
Slipper	52	\$6,175,834.00	\$209,940.00
Sport Shoe	51	\$651,467.00	\$25,179.00
Women's Casual	45	\$4,137,861.00	\$131,394.00
Women's Dress	51	\$6,226,475.00	\$193,653.00

```

END;
DEFINE returns;
  HEADER="Returns";
  FORMAT=DOLLAR14.2;
END;
END;
RUN;

```

In step (1) the DEFINE statement is used to name the table template 'tabtemp1'. The default table settings are used for the MEANS procedure and only the elements and attributes that are addressed in the syntax will be modified. The HEADER (2) names the column headers to be used in the table. The first HEADER (3) 'header1' is given a title and the JUST= statement is used to center it. It can also be right or left justified by modifying the syntax. The second HEADER 'header2' (4) is titled 'Calculated Totals' and spans the columns '_freq_', 'sales' and 'returns'. This is accomplished by using a START= and END= statement identifying the appropriate columns. (5) The 'sales' and the variable 'returns' are formatted with the SAS FORMAT'DOLLAR14.2'. Changes in variable names also occurred by using a HEADER= statement. Now that the columns and headers are formatted let's move on to traffic lighting in the table.

HOW DO YOU USE TRAFFIC LIGHTING?

"Traffic lighting" refers to highlighting certain cells with color based on a rule defined using a CELLSTYLE AS statement. While our traffic lighting is not based on red, yellow, and green lights, certain cells are emphasized based on the data values.

Example 2. Table Template Continued

```

PROC TEMPLATE;
  DEFINE TABLE tabtemp1;
  COLUMN product _freq_ sales
  returns;

  DEFINE sales; (1)
  HEADER="Sales";
  FORMAT=DOLLAR14.2;
  CELLSTYLE sales >=7000000
  AS {FONT_WEIGHT=medium
  BACKGROUND= light yellow
  FOREGROUND=black};
END;
DEFINE returns; (2)
  HEADER="Returns";
  FORMAT=DOLLAR14.2;
  CELLSTYLE Returns<=30000 AS
  {FONT_WEIGHT=medium
  BACKGROUND=moderate blue
  FOREGROUND=black};
END;
END;
RUN;

```

Figure 8. Traffic Lighting using Table Template

Based on PROC MEANS			
Type of Shoe	Calculated Totals		
	Stores	Sales	Returns
Boot	52	\$2,350,543.00	\$98,622.00
Men's Casual	45	\$7,933,707.00	\$311,035.00
Men's Dress	50	\$5,507,243.00	\$164,099.00
Sandal	49	\$868,436.00	\$38,170.00
Slipper	52	\$6,175,834.00	\$209,940.00
Sport Shoe	51	\$651,467.00	\$25,179.00
Women's Casual	45	\$4,137,861.00	\$131,394.00
Women's Dress	51	\$6,226,475.00	\$193,653.00

In both 'sales' (1) and 'returns' (2) the code from above uses the CELLSTYLE AS statement. This allows the style of a cell to be set according to its value. For example, in the 'sales' column, the font, background, and foreground were modified to emphasize values greater than 7,000,000. The same is true for the 'returns' column; however it was modified to emphasize values less than 30,000.

WHAT IS STYLE TEMPLATE?

Let's go a little further in exploring PROC TEMPLATE and look at style template and some of its possibilities. SAS style template tells SAS how to display the output that it produces. A way to differentiate between the use of style template and table template is that style template affects the entire document while table template affects specific elements of the output. A style template affects the overall appearance of your output and can enhance both table and graphics and, like a table template, is made up of elements and attributes. Think of elements as the area of output you wish to change, such as headers, data display, and the table appearance. The attributes are what make up the elements. These include, but are not limited to, fonts or backgrounds, borders of the table, and images.

WHAT IS THE SYNTAX FOR STYLE TEMPLATE?

The syntax for style template is the same as for the table template; let's look at the basic syntax of style template:

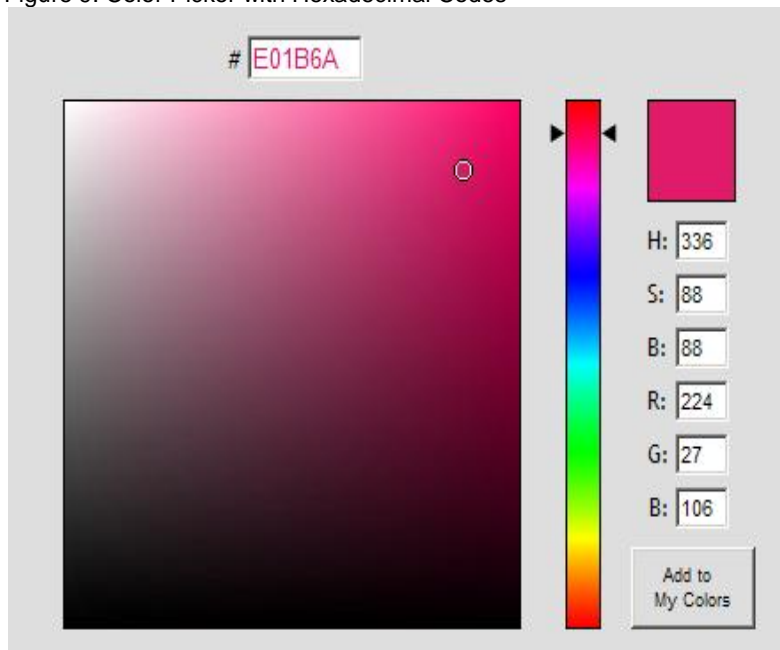
```
PROC TEMPLATE;  
    DEFINE STYLE style-name ;  
        .....statements/attributes...  
END;  
RUN;
```

We will look more closely at the syntax as we walk through an example, but first let's venture into the world of colors and fonts.

HOW DOES TEMPLATE USE COLOR AND FONTS?

Colors and fonts are both style attributes that can enhance the appearance of output. Colors can be specified using hexadecimal codes (RRGGBB) or Hue Light Saturation (HLS) values. An example of HLS is 'very light blue'. An example of hexadecimal code is cFFFFFFF for white. While white is simple and easy to remember, various other colors are not so easy. A color picker that is available online can be a very useful tool. In Figure 9 below the hexadecimal color is displayed in the box on top as 'E01B6A' for the color of pink that is showing in the top right corner.

Figure 9: Color Picker with Hexadecimal Codes



www.colorpicker.com

Another great way to get a list of accurate SAS color names is using PROC REGISTRY.

The syntax is :

```
PROC REGISTRY LIST
    STARTAT="COLORNAMES";
RUN;
```

As you can see this is a snippet of the entire list and a useful tool for finding hexadecimal values as well.

Figure 10. PROC REGISTRY List

```
proc registry list
    startat="COLORNAMES";
run;
```

NOTE: Contents of SASHELP REGISTRY starting at subkey [COLORNAMES]
COLORNAMES]
Active="HTML"
HTML]
AliceBlue=hex: F0,F8,FF
AntiqueWhite=hex: FA,EB,D7
Aqua=hex: 00,FF,FF
Aquamarine=hex: 7F,FD,D4
Azure=hex: F0,FF,FF
Beige=hex: F5,F5,DC
Bisque=hex: FF,E4,C4
Black=hex: 00,00,00
BlanchedAlmond=hex: FF,EB,CD
Blue=hex: 00,00,FF
BlueViolet=hex: 8A,2B,E2
BR=hex: A5,2A,2A
Brown=hex: A5,2A,2A
Burlywood=hex: DE,B8,87
CadetBlue=hex: 5F,9E,A0

Font style attributes are specified by FONT_FACE, FONT_WEIGHT, FONT_STYLE, FONT_SIZE. FONT_FACE are the fonts used such as 'Arial', 'Helvetica', 'Cambria', etc. When outputting, if you list fonts SAS will go through the list and pick the first available.

Examples of FONT_STYLE, FONT_WEIGHT, and TEXTDECORATION are given in Table 2 in the appendix. All are useful in customizing your output for your audience. An example of TEXTDECORATION is seen below.

Example 3 Style Template Decoration

Figure 11. Style Decoration Example

```
PROC TEMPLATE;
    DEFINE STYLE amstyle;
    PARENT=styles.sasweb;
    CLASS HEADER /
    FONT_FACE = "Impact"
    FONT_SIZE = 6
    FONT_WEIGHT = medium
    FONT_STYLE = roman
    FOREGROUND = cffffff
    BACKGROUND = teal
    TEXTDECORATION=underline;
```

Based on PROC MEANS			
Calculated Totals			
Type of Shoe	Orders	Sales	Returns

All of the above syntax will be explained in the example to follow. First let's look at how to easily create your own style template.

HOW CAN YOU EASILY CREATE YOUR OWN STYLE?

An easy tool to use to create your own style template is the ODS STYLE_POPUP.

```
ODS MARKUP TYPE=style_popup  
  PATH='c:\temp'  
  FILE='presentpu.html'  
  STYLE=styles.sasweb;
```

The PROC MEANS output from our above example is used. Notice when you hover over an element in the table, the output is highlighted by a salmon pink color; this is due to the use of the ODS STYLE_POPUP. Now when you click on the data or titles an html pop-up window appears with the syntax used for that specific element of the table. This is a quick and easy way to determine the syntax of the style template of the output. Figure 12 is an example of the data style syntax, and Figure 13 is an example of the header style syntax. To create your own style template, copy and paste what is in the pop-up into the basic syntax of the style template and change the elements that you would like to change. The next step is combining the table and style templates.

Figure 12: Data Style Element Pop Up Window

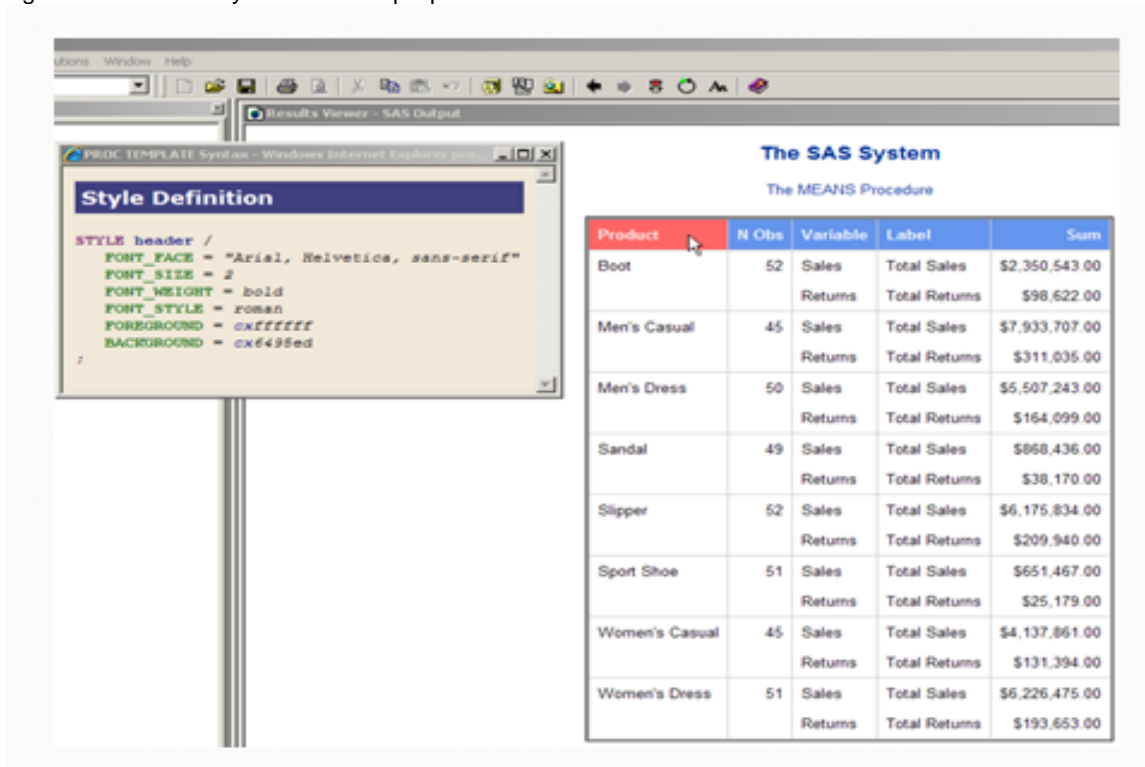
The screenshot shows the SAS Results Viewer interface. The main window displays the output of a PROC MEANS procedure, titled "The SAS System" and "The MEANS Procedure". The output is a table with columns: Product, N Obs, Variable, Label, and Sum. The table data is as follows:

Product	N Obs	Variable	Label	Sum
Boot	52	Sales	Total Sales	\$2,350,543.00
		Returns	Total Returns	\$98,622.00
Men's Casual	45	Sales	Total Sales	\$7,933,707.00
		Returns	Total Returns	\$311,035.00
Men's Dress	50	Sales	Total Sales	\$5,507,243.00
		Returns	Total Returns	\$164,099.00
Sandal	49	Sales	Total Sales	\$868,436.00
		Returns	Total Returns	\$38,170.00
Slipper	52	Sales	Total Sales	\$6,175,834.00
		Returns	Total Returns	\$209,940.00
Sport Shoe	51	Sales	Total Sales	\$651,467.00
		Returns	Total Returns	\$25,179.00
Women's Casual	45	Sales	Total Sales	\$4,137,861.00
		Returns	Total Returns	\$131,394.00
Women's Dress	51	Sales	Total Sales	\$6,226,475.00
		Returns	Total Returns	\$193,653.00

A pop-up window titled "PROC TEMPLATE Syntax - Windows Internet Explorer provide..." is overlaid on the table. It shows the style definition for the 'Boot' row, which is highlighted in salmon pink in the table. The style definition is:

```
STYLE data /  
  FONT_FACE = "Arial, Helvetica, sans-serif"  
  FONT_SIZE = 2  
  FONT_WEIGHT = medium  
  FONT_STYLE = roman  
  FOREGROUND = cx000000  
  BACKGROUND = cxffffff  
;
```

Figure 13: Header Style Element Pop Up Window



HOW DO YOU USE THE STYLE AND TABLE TEMPLATES TOGETHER?

The SAS code below is an example of using both the table template that was created in the previous example and the style template example that follows. Prior to the PROC MEANS statement the style template is referenced, telling SAS the specific template that will be used in the output. The DATA NULL step is used to reference the table template.

```

ODS PATH SASUSER.TEMPLATE (UPDATE)
  SASHELP.TMPLMST (READ);
ODS HTML PATH = 'c:\temp'
  FILE= 'test1.html'
  STYLE=styletemp1;
ODS PATH SHOW;
PROC MEANS DATA=sashelp.shoes SUM;
  VAR sales returns;
  CLASS product;
  OUTPUT OUT=example1 SUM=;
RUN;
DATA _null_;
  SET example1;
  IF product = ' ' THEN DELETE;
  FILE PRINT ODS=(TEMPLATE='tabtemp1');
  PUT _ODS_;
  TITLE 'Style Template Example';
  TITLE2 'Including Table Template as well...';
RUN;
ODS _ALL_ CLOSE;
  
```

← Referencing the style template to be used

← Referencing the table template created

HOW DO YOU ADD STYLE TO HEADERS, FOOTERS, TITLES AND THE DATA?

The style pop-up tool shown above was used to copy and paste the template into SAS.

Example 4. Style Template

Figure 14. Customized Style Template

```

PROC TEMPLATE;
  DEFINE STYLE styletempl;
  PARENT=styles.sasweb;
  CLASS SYSTEMTITLE / (1)
    FONT_FACE = "Impact"
    FONT_SIZE = 7
    FONT_WEIGHT = medium
    FONT_STYLE = roman
    FOREGROUND = teal
    BACKGROUND = cffffff;
  CLASS HEADER / (2)
    FONT_FACE = "Impact"
    FONT_SIZE = 6
    FONT_WEIGHT = medium
    FONT_STYLE = roman
    FOREGROUND = cffffff
    BACKGROUND = teal
  TEXTDECORATION=underline;
  CLASS DATA / (3)
    FONT_FACE = "Impact"
    FONT_SIZE = 2
    FONT_WEIGHT = medium
    FONT_STYLE = roman
    FOREGROUND = cffffff
    BACKGROUND = cx339999;
  CLASS FOOTER / (4)
    FONT_FACE = "Impact"
    FONT_SIZE = 2
    FONT_WEIGHT = light
    FONT_STYLE = slant
    FOREGROUND = cffffff
    BACKGROUND = teal;
END;
RUN;

```

Style Template Example

Including Table Template as well...

Based on PROC MEANS			
Type of Shoe	Calculated Totals		
	Stores	Sales	Returns
Boot	52	\$2,350,543.00	\$98,622.00
Men's Casual	45	\$7,933,707.00	\$311,035.00
Men's Dress	50	\$5,507,243.00	\$164,099.00
Sandals	48	\$868,436.00	\$38,170.00
Slipper	52	\$6,175,834.00	\$209,940.00
Sport Shoe	51	\$651,467.00	\$25,179.00
Women's Casual	45	\$4,137,861.00	\$131,394.00
Women's Dress	51	\$6,226,475.00	\$193,653.00
			<i>Yellow = Highest Total Sale</i>
			<i>Blue = Lowest Total Return</i>

Again this example was run using both the PROC MEANS example from above as well as the table template that was created. As you can see from the example the syntax is very similar. The SYSTEMTITLE (1) and HEADER (2) are identical with the exception of the changes in foreground and background. Using hexadecimal colors the font is colored white (cffffff) on the headers. Examining the FOOTER (4) shows similar syntax; however notice that the FONT_STYLE request is slant. The data style within the table is affected by the CLASS DATA statement (3).

HOW DO YOU ADD STYLE TO BORDERS AND CELLS?

To affect the borders of your table and its appearance there are several style options. The FRAME= option specifies which borders appear around the table. The BORDERSTYLE= option tells SAS which style of border will go around the table. The RULES= option affects the borders of the columns and rows within a table. For more options and to see examples of FRAME and BORDERSTYLE see Figures 23 and 24 in the appendix .

Style Template Example

Including Table Template as well...

```

PROC TEMPLATE;
  DEFINE STYLE styletempl;
    PARENT=styles.sasweb;
  CLASS TABLE /
    CELLPADDING=5
    CELLSPACING=3
    BACKGROUND= cxffffff
    FRAME= box
    RULES= all
    BORDERSPACING=3
    BORDERWIDTH=10
    BORDERSTYLE=double
    BORDERCOLOR=teal;
END;
RUN;
    
```

Based on PROC MEANS			
Type of Shoe	Calculated Totals		
	Stores	Sales	Returns
Boot	52	\$2,350,543.00	\$98,622.00
Men's Casual	45	\$7,533,707.00	\$311,035.00
Men's Dress	50	\$5,507,243.00	\$164,099.00
Sandal	49	\$868,436.00	\$38,170.00
Slipper	52	\$6,175,834.00	\$209,940.00
Sport Shoe	51	\$651,467.00	\$25,179.00
Women's Casual	45	\$4,137,861.00	\$131,394.00
Women's Dress	51	\$6,226,475.00	\$193,653.00
<i>Yellow = Highest Total Sale</i>			
<i>Blue = Lowest Total Return</i>			

HOW DO YOU SPECIFY WHICH TEMPLATE TO USE?

Now that you know where templates are stored and how to create your own templates you need to know how to specify which template to use in your code. To do this, SAS uses an ODS PATH statement. The function of the path statement is to define the location of the templates you wish to use.

The default path statement is:

```

ODS PATH SASUSER.TEMPLAT (UPDATE)
        SASHELP.TMPLMST (READ);
    
```

This path statement is telling SAS to first look for the template in the **SASUSER.TEMPLAT**, and if it does not exist there, to look in the **SASHELP.TMPLMST** next. You can also create a new item store with this path statement. For example, if we wanted to create a new template and save it in a store called, **EXAMPLE.MYTEMP**, all we would need to do is reset the path statement.

```

ODS PATH EXAMPLE.MYTEMP (UPDATE)
        SASHELP.TMPLMST (READ);
    
```

When SAS tries to locate the item store, **EXAMPLE.MYTEMP**, it will find that the store does not exist and will create it instead. Now any new templates you create will be stored here, because it is listed first in the path statement and has the access set to UPDATE.

Access options in the path statement define the manner in which SAS allows the user to access the item store. There are three access level options: READ, WRITE, and UPDATE. The key to remember is to always set the

SASHELP.TMPLMST to READ. The READ option protects the already made templates provided by SAS. With READ you can only access and use the templates in the store, you cannot delete or make changes. WRITE creates a new item store and provides READ access. UPDATE creates a new item store if the specified one does not exist, and allows you to add, delete, and make changes to the templates within the store.

There are two other options with the path statement that are very useful:

- 1) ODS PATH SHOW;
- 2) ODS PATH RESET;

The ODS PATH SHOW option allows you to keep track of the path order in your log. For example, if you wanted to see what path ODS is using for the following code:

Example 5. ODS PATH SHOW

```
ODS PATH EXAMPLE.MYTEMP (UPDATE)
SASHELP.TMPLMST (READ);

ODS PATH SHOW;
```

Figure 16. Example of ODS PATH SHOW in log

```
Log - (Untitled)
8   ODS PATH example.mytemp(UPDATE)
9       sashelp.tmplmst(READ);
10  ODS PATH SHOW;
    Current ODS PATH list is:
    1. EXAMPLE.MYTEMP(UPDATE)
    2. SASHELP.TMPLMST(READ)
```

When you run this code you get a comment in your log that lets you know what the current ODS PATH list is and a specification of the order. **EXAMPLE.MYTEMP** will be the first store SAS will use and then if the template is not found in **EXAMPLE.MYTEMP**, SAS will look in the **SASHELP.TMPLMST** store.

If you decide to use another template that is in **SASUSER.TEMPLAT** then the ODS PATH RESET option will allow you to set your path back to the SAS default path.

Example 6. ODS PATH SHOW and ODS PATH RESET

Figure 17. Example of ODS PATH RESET in log

```
ODS PATH EXAMPLE.MYTEMP (UPDATE)
SASHELP.TMPLMST (READ);

ODS PATH SHOW;

ODS PATH RESET;
ODS PATH SHOW;
```

```
Log - (Untitled)
8   ODS PATH example.mytemp(UPDATE)
9       sashelp.tmplmst(READ);
10  ODS PATH SHOW;
    Current ODS PATH list is:
    1. EXAMPLE.MYTEMP(UPDATE)
    2. SASHELP.TMPLMST(READ)
11  ODS PATH RESET;
12  ODS PATH SHOW;
    Current ODS PATH list is:
    1. SASUSER.TEMPLAT(UPDATE)
    2. SASHELP.TMPLMST(READ)
```

HOW DO YOU CREATE SHARED TEMPLATES?

Have you ever wanted to create a template to share among a group of users? Perhaps you work for a company that would prefer a certain style applied to each report generated by multiple users. The ODS PATH statement makes it possible to share templates across different users on a single network. The trick to this process is that you must store the template in a directory that is accessible to others.

```
LIBNAME group "W:\Shared\SDSUG Talks\Templates"; 1)
PROC TEMPLATE;
  DEFINE STYLE styletempl/STORE=group.templates; 2)
  PARENT=styles.sasweb;
```

The first step is to establish a permanent item store in a public folder (1). Then when you are creating a new template be sure to use the STORE= option in the define statement (2). This way your template is stored in the public directory referenced in the LIBNAME, and makes your template accessible to all users.

Figure 18. Log of Successful LIBNAME Statement

```
213
214 /*Creating a style template*/
215 libname group "W:\Shared\SDSUG Talks\Templates";
NOTE: Libref GROUP was successfully assigned as follows:
      Engine:          V9
      Physical Name:  W:\Shared\SDSUG Talks\Templates
216 proc template;
217     define style styletempl/store=group.templates;
218         parent=styles.sasweb;
219
```

It is very important to successfully assign your library because all users that will be sharing templates will need to specify the shared library within their LIBNAME statements. To make sure that the library reference was successfully created, you can check your log. Figure 18 above gives an example.

Figure 19. Log of Shared Template and Item Store

```
NOTE: STYLE 'styletempl' has been saved to: GROUP.TEMPLATES
275 run;
NOTE: PROCEDURE TEMPLATE used (Total process time):
      real time          0.07 seconds
      cpu time           0.03 seconds
```

You can also check your log to make sure that the template you create is saved to the item store in the shared user folder. Figure 19 shows that the style template, **STYLETEMPL**, was saved in the shared user item store **GROUP.TEMPLATES**. Now other users that have access to the folder can use the **STYLETEMPL** template by referencing the library 'GROUP' in their LIBNAME statement.

The following code is an example of using a shared template for a PROC PRINT with a different dataset than the **SASHELP.SHOES**. This example uses the same style template that we created before, and applies it to the **SASHELP.CLASS** data by accessing the template through the shared folder.

```
LIBNAME group "W:\Shared\SDSUG Talks\Templates"; 1)

ODS PATH RESET;
ODS PATH (PREPEND) group.templates(READ); 2)

ODS HTML FILE='C:\Documents and Settings\burgessk\My Documents\My Practice
Files\sharedtempl.html'
STYLE=styletempl; 3)

PROC PRINT DATA=sashelp.class (obs=5); RUN; 4)
ODS PATH SHOW; 5)
```

First, you reference the public file that contains the template in a LIBNAME statement (1). Make sure that you set the ODS PATH to read from the shared item store first (2). Next, you reference the style template you wish to use from the shared item store (3). Then you run your procedure (4) and view your current path in your log (5).

When you reference the shared templates in the LIBNAME statement, make sure to set the access to read only. This way the stored template cannot be changed when used by other users. This allows for consistency and reproducibility among those using the template. Next, set the path so that SAS will access and locate the templates stored in the **GROUP . TEMPLATES** store first. This can be done by using the (PREPEND) option. If you wanted to move the first item store to the end of the path statement you could use (APPEND), and if you wanted to remove an item store from the path you could use (REMOVE). The rest is simple! You just reference the type of file and location of where you want your output to go, state the style template you wish to use, and run the procedure.

Figure 20. Log of Current Path

```
10 ods path show;
Current ODS PATH list is:
1. GROUP . TEMPLATES(READ)
2. SASUSER . TEMPLAT(UPDATE)
3. SASHELP . TPLMST(READ)
```

Figure 21. PROC PRINT Output Using Shared Template

The SAS System

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
5	Henry	M	14	63.5	102.5

Figure 20 shows the updated path with **GROUP . TEMPLATES** being the first item store on the path. Figure 21 shows the PROC PRINT output with the **STYLETEMP1** style template.

CONCLUSION

PROC TEMPLATE is a highly useful tool for editing and customizing your data. Table template provides a way to structure your output and style template provides a way to format the appearance of your output. Together, both templates can be used to produce visually appealing reports all within SAS. Templates can be stored and used for future projects or shared with other SAS users, enhancing the flexibility and usability of this tool.

REFERENCES

Haworth, L.E. (2006), PROC TEMPLATE: The Basics, *Proceedings of the Thirty-First SAS Users Group International Conference*, paper 112-31. <http://www2.sas.com/proceedings/sugi31/112-31.pdf>.

Haworth, L.E., Zender, C.L., Burlew, M.M. 2009. *Output Delivery System: The Basics and Beyond*. Cary, NC: SAS Institute Inc.

Smith, K.D. (2007), PROC TEMPLATE Tables from Scratch, *Proceedings of the SAS Global Forum 2007 Conference*, paper 221-2007. <http://www2.sas.com/proceedings/forum2007/221-2007.pdf>.

Zender, C.L. (2009), Tiptoe through the Templates, *Proceedings of the SAS Global Forum 2009 Conference*, paper 227-2009. <http://support.sas.com/resources/papers/proceedings09/227-2009.pdf>.

Zender, C.L. (2010), SAS® Style Templates: Always in Fashion, *Proceedings of the SAS Global Forum 2010 Conference*, paper 033-2010. <http://support.sas.com/resources/papers/proceedings10/033-2010.pdf>.

ACKNOWLEDGMENTS

The authors wish to thank Dr. Paul Thompson and Dr. Susan Puumala in their review and guidance with this paper.

CONTACT INFORMATION

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

Katherine Burgess
Sanford Research
2301 East 60th Street N.
Sioux Falls, SD 57104
605-312-6465
Katherine.Burgess@sanfordhealth.org

Ashley Miller
Sanford Research
2301 East 60th Street N.
Sioux Falls, SD 57104
605-312-6463
Ashley.Miller@sanfordhealth.org

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

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

APPENDIX A.

Figure 22. Locating Templates using the SAS Windowing Environment

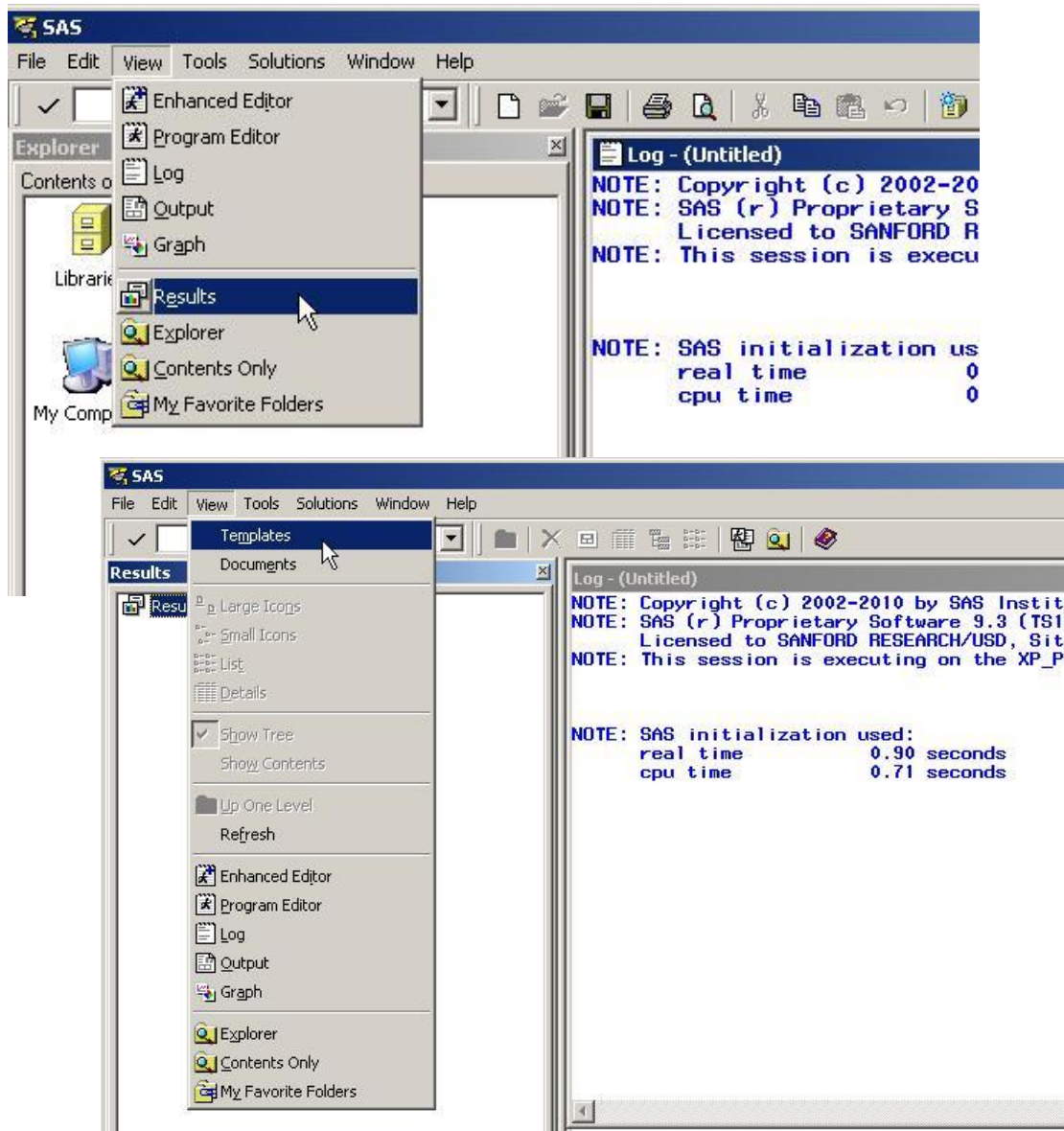


Table 1. Border Options

Frame	Rules	Borderstyle
Box	All	Dashed
Above	Cols	Dotted
Below	Rows	Double
Hsides	Groups	Groove
Vsides	None	Hidden
Lhs		Inset
Rhs		Outset
Void		Ridge
		Solid
		None

Table 2. Font Options

Font Style	Font Weight	Text Decoration
Italic	Bold	Line_through
Roman	Medium	Overline
Slant		Underline

Figure 23. Examples of FRAME Options

Frame = hside

Based on PROC MEANS			
Type of Shoe	Calculated Totals		
	Stores	Sales	Returns
Boot	52	\$2,350,543.00	\$98,622.00
Men's Casual	45	\$7,933,707.00	\$311,035.00
Men's Dress	50	\$5,507,243.00	\$164,099.00
Sandal	49	\$868,436.00	\$38,170.00
Slipper	52	\$6,175,834.00	\$209,940.00
Sport Shoe	51	\$651,467.00	\$25,179.00
Women's Casual	45	\$4,137,861.00	\$131,394.00
Women's Dress	51	\$6,226,475.00	\$193,653.00
<i>Yellow = Highest Total Sale</i>			
<i>Blue = Lowest Total Return</i>			

Frame = box

Based on PROC MEANS			
Type of Shoe	Calculated Totals		
	Stores	Sales	Returns
Boot	52	\$2,350,543.00	\$98,622.00
Men's Casual	45	\$7,933,707.00	\$311,035.00
Men's Dress	50	\$5,507,243.00	\$164,099.00
Sandal	49	\$868,436.00	\$38,170.00
Slipper	52	\$6,175,834.00	\$209,940.00
Sport Shoe	51	\$651,467.00	\$25,179.00
Women's Casual	45	\$4,137,861.00	\$131,394.00
Women's Dress	51	\$6,226,475.00	\$193,653.00
<i>Yellow = Highest Total Sale</i>			
<i>Blue = Lowest Total Return</i>			

Figure 24. Examples of BORDERSTYLE Options

Borderstyle = dotted

Based on PROC MEANS			
Type of Shoe	Calculated Totals		
	Stores	Sales	Returns
Boot	52	\$2,350,543.00	\$98,622.00
Men's Casual	45	\$7,933,707.00	\$311,035.00
Men's Dress	50	\$5,507,243.00	\$164,099.00
Sandal	49	\$868,436.00	\$38,170.00
Slipper	52	\$6,175,834.00	\$209,940.00
Sport Shoe	51	\$651,467.00	\$25,179.00
Women's Casual	45	\$4,137,861.00	\$131,394.00
Women's Dress	51	\$6,226,475.00	\$193,653.00
<i>Yellow = Highest Total Sale</i>			
<i>Blue = Lowest Total Return</i>			

Borderstyle = dashed

Based on PROC MEANS			
Type of Shoe	Calculated Totals		
	Stores	Sales	Returns
Boot	52	\$2,350,543.00	\$98,622.00
Men's Casual	45	\$7,933,707.00	\$311,035.00
Men's Dress	50	\$5,507,243.00	\$164,099.00
Sandal	49	\$868,436.00	\$38,170.00
Slipper	52	\$6,175,834.00	\$209,940.00
Sport Shoe	51	\$651,467.00	\$25,179.00
Women's Casual	45	\$4,137,861.00	\$131,394.00
Women's Dress	51	\$6,226,475.00	\$193,653.00
<i>Yellow = Highest Total Sale</i>			
<i>Blue = Lowest Total Return</i>			