Paper HOW-164 2018

Working with the SAS® ODS EXCEL Destination to Send Graphs, and Use Cascading Style Sheets When Writing to EXCEL Workbooks

William E Benjamin Jr, Owl Computer Consultancy, LLC, Phoenix AZ.

ABSTRACT

This Hands-On-Workshop will explore the new SAS® ODS EXCEL destination and focus on how to write Excel Worksheets with output from SAS Graph procedures and spice it up using Cascading Style Sheet features available on modern computer systems. Note that the ODS EXCEL destination is a BASE SAS product, which makes it available on all platforms. The workshop will be limited to the Windows platform, but it should be simple to port the code to other operating systems. The code will be on the computers and you will get a chance to see how it handles.

INTRODUCTION

The new ODS Excel Destination allows the direct output of Excel files in the new *.xlsx format. This feature of SAS is available to anyone using the third maintenance release of SAS 9.4 or greater as the base product, including SAS University Edition®. This SAS ODS destination is unlike the ODS tagset EXCELXP in that it cannot be modified and has a slightly different set of options. The most important two differences are that the ODS EXCELXP tagset does not directly support writing graphic images into an Excel W\workbook or create native format *.xlsx Excel workbooks.

PROBLEM

The ability to directly write graphs to Excel workbooks has long been something that has not been available with the EXCELXP tagset. Additionally, customizing the workbook has often been hard to accomplish. The new options and capabilities of the ODS EXCEL destination will simplify the customization of your output Excel workbooks.

ODS DESTINATION EXCEL

All ODS commands are described in the SAS online manual [1] “SAS® 9.4 Output Delivery System: User's Guide, Fourth Edition” including the commands that relate to the EXCEL destination. I found a copy at the following SAS web site: http://support.sas.com/documentation/cdl/en/odsug/67921/PDF/default/odsug.pdf In addition, as part of BASE SAS the ODS EXCEL destination also works on at least some “NON-Windows” based computer systems. See the SAS documentation to see which platforms are available. One system is the z/OS operating system that is usually found on IBM mainframes. This Excel destination works when using HFS files and directories, See the manual for more information.

WORKSHOP FEATURES AND EXAMPLES

OUTPUT USING SAS STYLE SHEET TEMPLATES

The simple way to spruce up your spreadsheet output is to use a SAS Stylesheet, but what are they called? Well it turns out that there is a simple way to get a listing of all of the SAS Styles supported on your computer. The following SAS code will write a nice report, listing the style names to the SAS log. I have reformatted the list into Table 1.

SAS Code 1 - SAS code to list all of the supported styles available to your computer.

```
ods _all_ close;
ods listing;
  proc template;
    list styles;
  run;
quit;
```

The following list of styles is listed alphabetically by row from the left to the right. I will leave it up to the student to experiment with these color schemes on your own time.

Table 1 List of SAS styles.

List of SAS Styles Supported (SAS 9.4 1M3)			
Analysis	BarrettsBlue	BlockPrint	DTree
Daisy	Default	Dove	EGDefault
Excel	FancyPrinter	Festival	FestivalPrinter
Gantt	GrayscalePrinter	HTMLBlue	Harvest
HighContrast	HighContrastLarge	Journal	Journal1a
Journal2	Journal2a	Journal3	Journal3a
Listing	Meadow	MeadowPrinter	Minimal
MonochromePrinter	Monospace	Moonflower	Netdraw
NoFontDefault	Normal	NormalPrinter	Ocean
Pearl	PearlJ	Plateau	PowerPointDark
PowerPointLight	Printer	Raven	Rtf
Sapphire	SasDocPrinter	SasWeb	Seaside
SeasidePrinter	StatDoc	Statistical	Word
vaDark	vaHighContrast	vaLight	

An alternate method of locating the available styles on your computer is by looking up the style templates. These styles are the output from PROC TEMPLATE and are provided be SAS. Since they are text files that are available for you to use, from the template storage, you can also change them and store them in a user storage directory. But that is a story for another day. Figure 1 and Figure 2 show the path to the style templates and part of the code for the HARVEST style used in Figure 3. The starting point for Figure 1 is The Results screen and the View option.

Figure 1 SAS Screens to locate the SAS Style Sheet PROC TEMPLATE code modules (part-1).

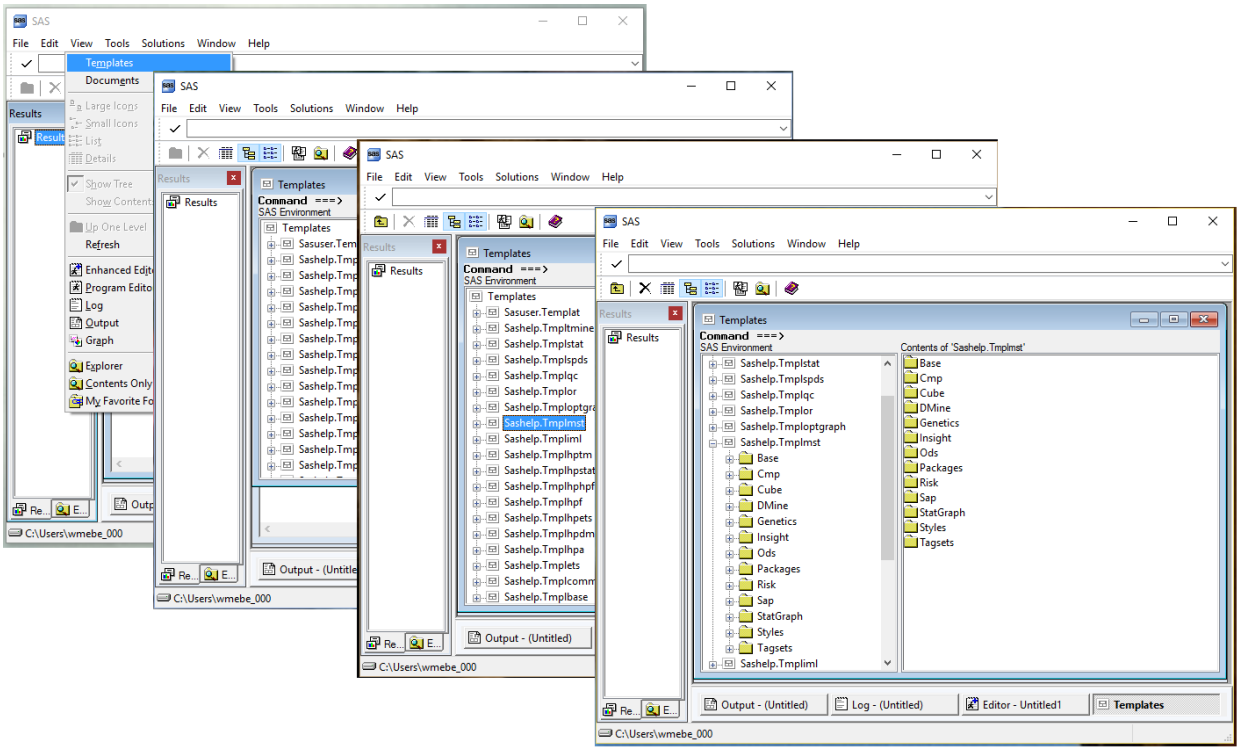
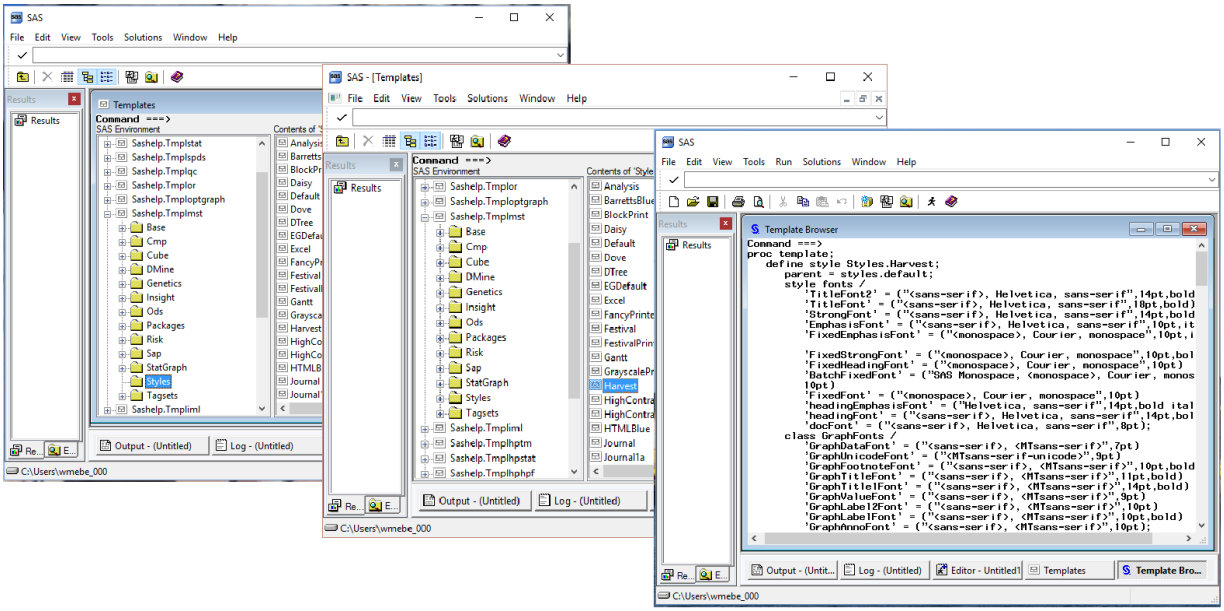


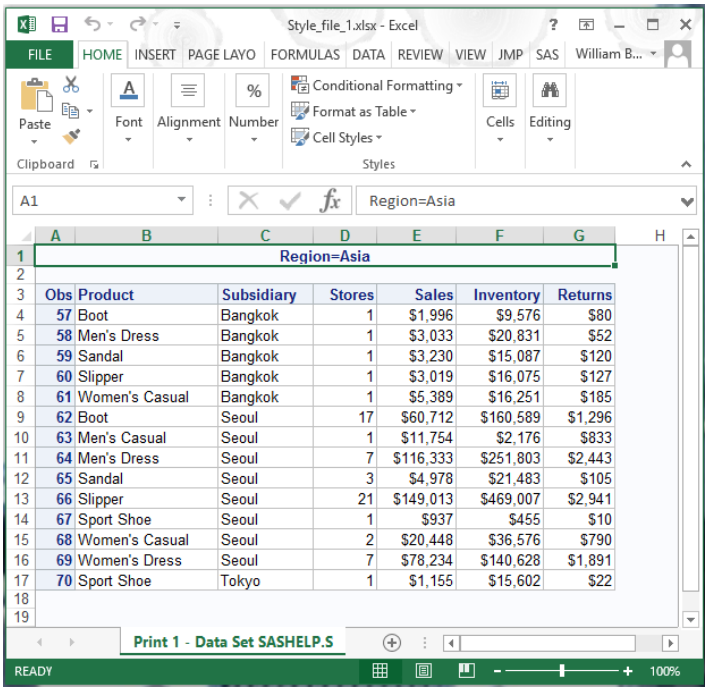
Figure 2 SAS Screens to locate the SAS Style Sheet PROC TEMPLATE code modules (part-2).



SAS Code 2 - Code to write 2 Excel workbooks, with and without the HARVEST style.

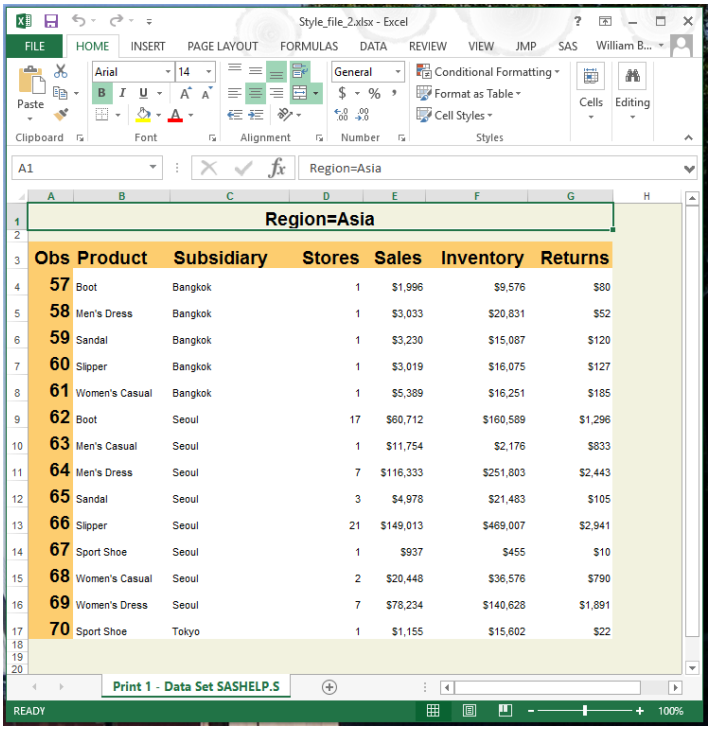
```
ods excel file = "i:\temp\Style_file_1.xlsx" ;
  proc print data=sashelp.shoes (where=(region="Asia"));
    by region;
  run;
ods excel close;

ods excel file = "i:\temp\Style_file_2.xlsx" style=Harvest ;
  proc print data=sashelp.shoes(where=(region="Asia"));
    by region;
  run;
ods excel close;
```

Figure 3- Output Excel Workbook without using the HARVEST style.



Obs	Product	Subsidiary	Stores	Sales	Inventory	Returns
57	Boot	Bangkok	1	$1,996	$9,576	$80
58	Men's Dress	Bangkok	1	$3,033	$20,831	$52
59	Sandal	Bangkok	1	$3,230	$15,087	$120
60	Slipper	Bangkok	1	$3,019	$16,075	$127
61	Women's Casual	Bangkok	1	$5,389	$16,251	$185
62	Boot	Seoul	17	$60,712	$160,589	$1,296
63	Men's Casual	Seoul	1	$11,754	$2,176	$833
64	Men's Dress	Seoul	7	$116,333	$251,803	$2,443
65	Sandal	Seoul	3	$4,978	$21,483	$105
66	Slipper	Seoul	21	$149,013	$469,007	$2,941
67	Sport Shoe	Seoul	1	$937	$455	$10
68	Women's Casual	Seoul	2	$20,448	$36,576	$790
69	Women's Dress	Seoul	7	$78,234	$140,628	$1,891
70	Sport Shoe	Tokyo	1	$1,155	$15,602	$22

Figure 4- Output Excel Workbook using the HARVEST style.



Obs	Product	Subsidiary	Stores	Sales	Inventory	Returns
57	Boot	Bangkok	1	$1,996	$9,576	$80
58	Men's Dress	Bangkok	1	$3,033	$20,831	$52
59	Sandal	Bangkok	1	$3,230	$15,087	$120
60	Slipper	Bangkok	1	$3,019	$16,075	$127
61	Women's Casual	Bangkok	1	$5,389	$16,251	$185
62	Boot	Seoul	17	$60,712	$160,589	$1,296
63	Men's Casual	Seoul	1	$11,754	$2,176	$833
64	Men's Dress	Seoul	7	$116,333	$251,803	$2,443
65	Sandal	Seoul	3	$4,978	$21,483	$105
66	Slipper	Seoul	21	$149,013	$469,007	$2,941
67	Sport Shoe	Seoul	1	$937	$455	$10
68	Women's Casual	Seoul	2	$20,448	$36,576	$790
69	Women's Dress	Seoul	7	$78,234	$140,628	$1,891
70	Sport Shoe	Tokyo	1	$1,155	$15,602	$22

While I have not used very many of the SAS Styles, I have found that the ones I have used do change the look of the output workbook, but they have that look and feel of a standardized output. Here we are looking for that customized look.

CASCADING STYLE SHEETS

Cascading Style Sheets were designed to enable the simple changing of output displayed to Web documents. A full explanation can be found at W3C Cascading Style Sheets Home Page. This is the W3C Home page with descriptions

and links to the CSS Standards. This Web page also describes drafts and proposed changes to the CSS Standards. SAS documentation indicates that the SAS usage of the CSS is as fully supported as they can make it. As with all things that are moving targets, the CSS Standards may change as time goes on. The W3C Web page has links to a Web Standards Curriculum which is a self-study course. While I cannot teach you everything about CSS coding in 20 pages, this will give you a place to start. The function of Cascading Style Sheet is to decorate the output, rather than modify the structure of the output.

I will jump ahead to show you a simple CSS code module, first let's write a simple Excel Workbook and create a simple "Document Object Module" (DOM). The SAS ODS "DOM" option (shown in the command "ODS EXCEL DOM;") created the output shown in Attachment 1, which can be used to find CSS elements. The SAS documentation [4, page 53] identifies the following CSS Style selectors. I will only address a small portion of these in this paper.

1. **Class Selectors**
2. **Element Selectors**
3. **Universal Selector**
4. **Pseudo-Class Selectors**
5. **ID Selectors**
6. **Attribute Selectors**
7. **Combinators**

The following brief definition of each of these selectors follows, see: SAS Institute Inc. 2014. SAS® 9.4 Output Delivery System: Advanced Topics. Cary, NC: SAS Institute Inc. pages 54 and following for more details. Since I am limited in space for this document I only have room to introduce the topics.

A Limited Selection of CSS Selector classes [4,page 54 el.]		
Selector Class	**Description**	**Example**
Class Selectors	**Class selectors are style selectors that select elements based on the value of the class= attribute in the markup of an ODS report. Class selectors must have a period (.) preceding the class name. For example, in the following rule set, the class style selector is .SYSTEMTITLE.**	**.systemtitle** **{ font-family: arial, helvetica, sans-serif;** **color: red;** **border: 1px solid black;** **}**
Element Selectors	**Element selectors are style selectors that select DOM elements based on the element name. For example, the following rule set selects elements with the name P:**	**p {color:green}**
Universal Selector	**The universal selector is a style selector that is a wildcard. It can match any element name. The syntax for the universal selector is an asterisk (*).**	*****
Pseudo-Class Selectors	**Pseudo-class selectors are style selectors that select elements based on the relationships between DOM elements. Pseudo-classes are represented by the pseudo-class name prefixed with a colon (:). The following are examples of some ways that you can use pseudo-class selectors.** **• select the first and last child of a parent element** **• select a specific child based on its positional index in the parent element** **• select an element by position of a particular element name**	**:root Selects the top-level element in the DOM.** **:first-child Selects the first element within the parent.** **:first-of-type Selects the first element of that type (that is, same element name) in the parent.** **:marker Selects a list item bullet in printer output.** **:nth-child(an+b) Selects an element based on the equation an +b. This equation selects every ath element starting with element at position b. The equation can be replaced with the keywords even or odd for the simple case of alternating the selection.** **:nth-of-type(an+b) Selects the :nth-child, except that only the same elements of the same type are used in the calculation.** **:empty Selects one or more empty elements. This only applies to elements that have been specified as empty by the procedure. *** **:before Inserts content before the element. *** **:after Inserts content after the element** **:not(…) Selects an element if the selector within the argument is not true.**
ID Selectors	**ID selectors are style selectors that select elements based on the id= attribute of a DOM element. The ID must be unique within a DOM and only one can be specified in the id= attribute. ID selectors are indicated by a "#" prefix. The following is a CSS rule set with an ID selector:**	**#idx1 { font-style: italic }**

Attribute Selectors	**Attribute selectors select DOM elements with the specified attribute. ID selectors and class selectors are special case attribute selectors. Attribute selectors use the following syntax to select attributes:** **[attribute operator "value"]** **operator** **specifies the operator.** **operator allows partial matches.**	**= Matches the entire attribute value.** **^= Matches the beginning of an attribute value.** **=$ Matches the end of an attribute value.** ***= Matches any substring in an attribute value.** **~= Matches any space-separated word in an attribute value. This operator can be used to emulate the class selector.** **|= Matches an attribute value and an optional value followed by a hyphen. This operator is used to match language codes such as enUS, en-GB, and so on.**
Combinators	**Combinators are characters that select an element based partially on its context within another element. This is done by combining selectors using one of the following characters.**	**" " (space) indicates that the selector to the left must match an element anywhere in the parentage of the currently selected element.** **> selects elements that are a direct descendent of the specified element.** **~ selects elements that have another sibling anywhere within the parent.** **+ selects elements that have a specified element immediately preceding them**

SAS Code 3- Use ODS "DOM" to make a Document Object Module output listing.

```
options linesize=255;
data test_css;
    Column_a =  'My test item';
run;

ods excel dom;
    proc print data=test_css noobs;
    run;
ods excel close;
```

The main output of this code is the listing in Attachment 1. From Attachment 1, some of the selector and property constructs of CSS can be identified. For instance "Header", "Body", "Rowheader", "Data", "Tbody" "TR", "TP", "Class", and "Type" to name just a few.

SOME SIMPLE CSS SYNTAX

CSS Code 1 - One example of CSS code syntax is the following:

```
selector {
  property1:value;
  property2:value;
  property3:value;
}
```

The CSS Code in the text box above is a sample of a simple syntax for CSS code. While other options exist this shows three types of values a "Sector", "Property", and a "Value". Because CSS was originally developed to modify HTML most of the elements of CSS are derived from HTML code.

- The “Sector” identifies an HTML element that we want to change.
- The “Property” is something we want to change.
- The “Value” is what we want the “Property” to become.

The pertinent parts are as follows [3]:

- The selector identifies the HTML elements that the rule will be applied to, using actual element names, eg. “body”, or another identifier such as class attribute values.
- The curly braces contain the property/value pairs, which are separated from each other by semi-colons; the properties are separated from their respective values by colons.
- The properties define what you want to do to the element(s) you have selected. These come in wide varieties, which can affect text color, background color, position on the page, font type, border color and thickness and many other things.
- The values are the values that you want to set for each property of the selected elements. The values are dependent on the property, for example properties that affect color can take hexadecimal colors like #336699, RGB values like rgb(12,134,22) or color names like red, green or blue. Properties that affect position, margins, width, height etc. can be measured in pixels, ems, percentages, centimeters or other such units.

Because I want to keep things simple I will show SAS code that builds the CSS file, and then uses it. This process will enable me to put all of the parts into one simple SAS program without needing to go into a lot of detail. As with other things I have written, I will show you how to get started, and you can continue from there. Here we will expand upon the example in Figure 5, which was built with SAS Code 3. The code in text box SAS Code 4, is a program that produces a simple file for creation of an Excel workbook, a CSS file to modify the output Excel Workbook, and two Excel workbooks one without modification and one with CSS formatting applied.

SAS Code 4 – Code to generate and apply CSS formatting to a simple Excel workbook.

```
 1  %let path = C:\HOW\Benjamin;
 2
 3    * Define a CSS output file;
 4   filename css "&path.\out\my_css_1.css";
 5
 6   * Write a css file;
 7   data _null_;
 8   file css noprint linesize=132;
 9   put
10   ".body  {background-color: lightblue;  } " /
11   ".header{background-color: gold;       } " /
12   ".rowheader {background-color: purple;   " /
13   "            color: white;             } " /
14   ".data  {background-color: lightgreen; } " ;
15    run;
16
17    * Create a simple file to write to Excel;
18    options linesize=255;
19    data test_css;
20        Column_a =  'My test item 1'; output;
21        Column_a =  'My test item 2'; output;
22        Column_a =  'My test item 3'; output;
23        Column_a =  'My test item 4'; output;
24        Column_a =  'My test item 5'; output;
25        Column_a =  'My test item 6'; output;
26        Column_a =  'My test item 7'; output;
27    run;
28
29    * Write a file to EXCEL without CSS changes;
30    ods excel file = "&path.\out\Css_file_1.xlsx";
31    proc print data=test_css;
32    run;
```

```
33    ods excel close;
34
35    * Write a file to EXCEL with CSS changes;
36    ods excel  file  = "&path.\out\Css_file_2.xlsx"
37            cssstyle = "&path.\out\my_css_1.css";
38    proc print data=test_css;
39    run;
40    ods excel close;
```

An explanation of the code in SAS Code 4 follows:

1. Line # 1. Defines a macro variable called PATH the allows you to make the code more portable.
2. Line # 4. Defines a file that will hold the CSS commands created.
3. Lines # 7 – 1 5 This code writes the CSS file with five color modifications listed.
4. Lines # 18 – 27 Creates a SAS data set to be written to Excel using ODS EXCEL commands.
5. Lines # 30 – 33 This writes an Excel workbook with no special processing (See Figure 6).
6. Lines # 36 – 40 This writes an Excel workbook with five modified fields defined by the CSS commands in step #3 (See Figure 7).

Figure 6 – Excel file output with no CSS modifications.

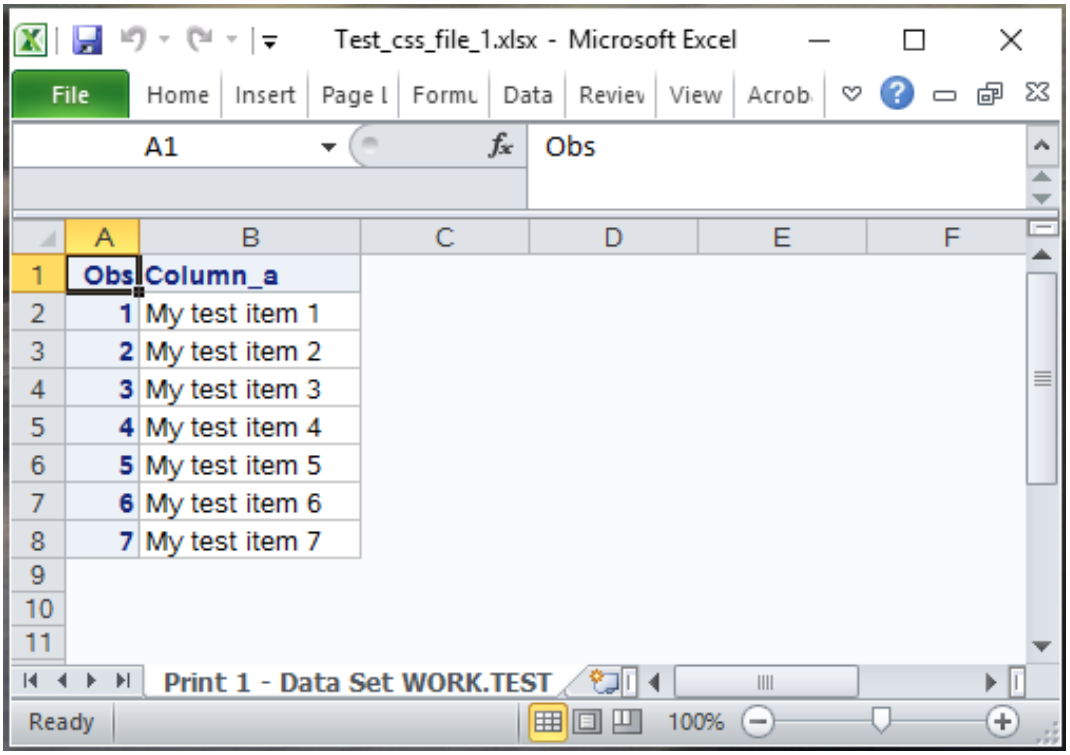
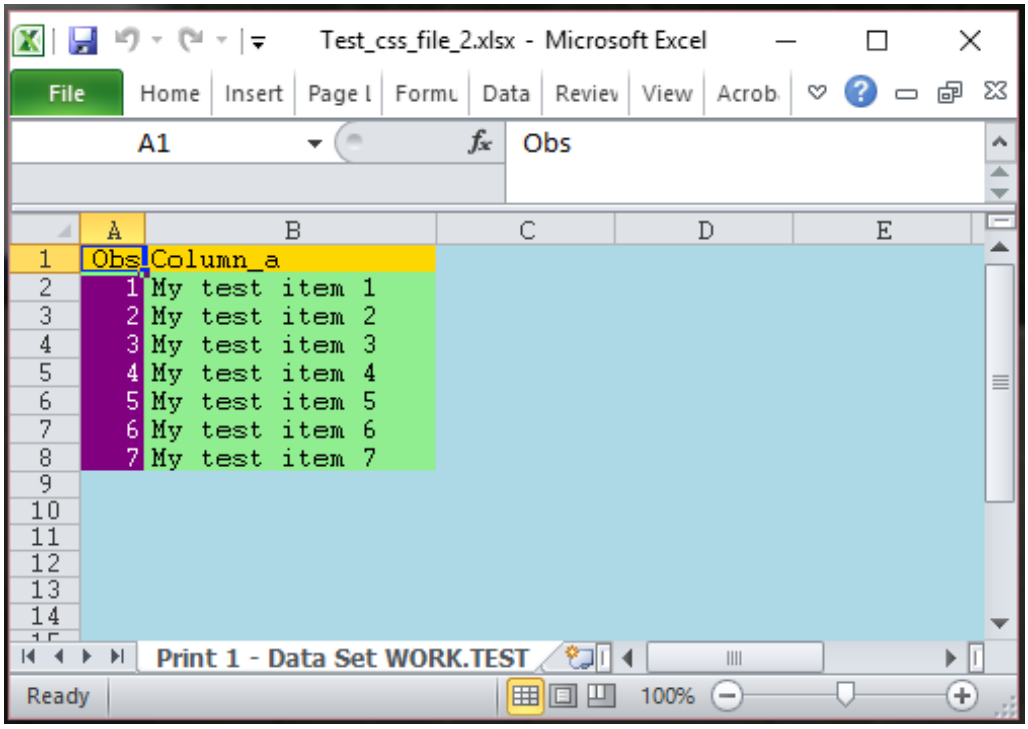


Figure 7 – Excel file output with CSS modifications to the output Excel Workbook.



GRAPHS

Graphs are far more complicated than simple PROC PRINT output listings. They have many more parts and options. In this paper I will stray away from the SGPLOT features that allow multiple charts and graphs on a single page, and focus of the few questions needed to get started. You may have noticed in the Fine print of Figure 2, there was a long list of “GRAPHFONTS”, well further down in the style are listed Style colors and a long list of classes of GraphColors that the style uses. Other Items that I will address are below, If I create a CSS file as shown below and use it to modify a graph to be placed into EXCEL Let’s see what it will do.

CSS Code 2 – Patriotic CSS Code to decorate an Excel Spreadsheet, named my_css_2.css.

```
  .graphbackground        {background-color: orange;    }
  .graphtitle1text        {color: blue; Font: 20pt arial; font-
weight: bold;}
  .graphfootnotetext      {color: blue; Font: 12pt arial; font-
weight: bold;}
  .graphvaluetext         {color: black;                }
  .graphdata1             {color: red;                  }
  .graphdata2             {color: white;                }
  .graphdata3             {color: blue;                 }
  .graphdata4             {color: red;                  }
  .graphdata5             {color: white;                }
  .graphdata6             {color: blue;                 }
  .graphdata7             {color: red;                  }
  .graphdata8             {color: white;                }
  .graphdata9             {color: blue;                 }
```

Let's see what happens to a pie chart graph when displayed with and without this CSS code.

SAS Code 5 – SAS Code to Create a Pie Chart of the SASHELP.SHOES (minus "ASIA").

```
%let path = C:\HOW\Benjamin;
%macro Graph_it;
      PROC SQL;
      CREATE VIEW WORK.Sorted_1 AS
         SELECT T.Region, T.Sales
              FROM SASHELP.SHOES(WHERE=(Region ne "Asia")) as T;
      QUIT;

      Legend1
         FRAME
         POSITION = (BOTTOM CENTER OUTSIDE);

      TITLE1 "Pie Chart of SASHELP.Shoes Return data by Region";
      TITLE2 "Execpt Asia";
      FOOTNOTE1 "Produced by William E Benjamin Jr";

      PROC GCHART DATA =WORK.Sorted_1;
         PIE3D  Region / SUMVAR=Sales
         TYPE=SUM
         LEGEND=LEGEND1
         SLICE=OUTSIDE
         PERCENT=OUTSIDE
         VALUE=OUTSIDE
         OTHER=4
         OTHERLABEL="Other"
         COUTLINE=BLACK
      NOHEADING;

      RUN;
      QUIT;

      TITLE;
      FOOTNOTE;
      RUN;
%mend Graph_it;

    /* without CSS style modification */
    ods excel file = "&path.\out\Test_css_file_1.xlsx";
    %graph_it;
    ods excel close;

    /* with CSS style modification */
    ods excel file = "&path.\out\Test_css_file_2.xlsx"
          cssstyle = "&path.\out\my_css_2.css";
    %graph_it;
    ods excel close;
```

Figure 8 – Excel file output without CSS modifications to the output Test_css_file_1.xlsx Excel Workbook.

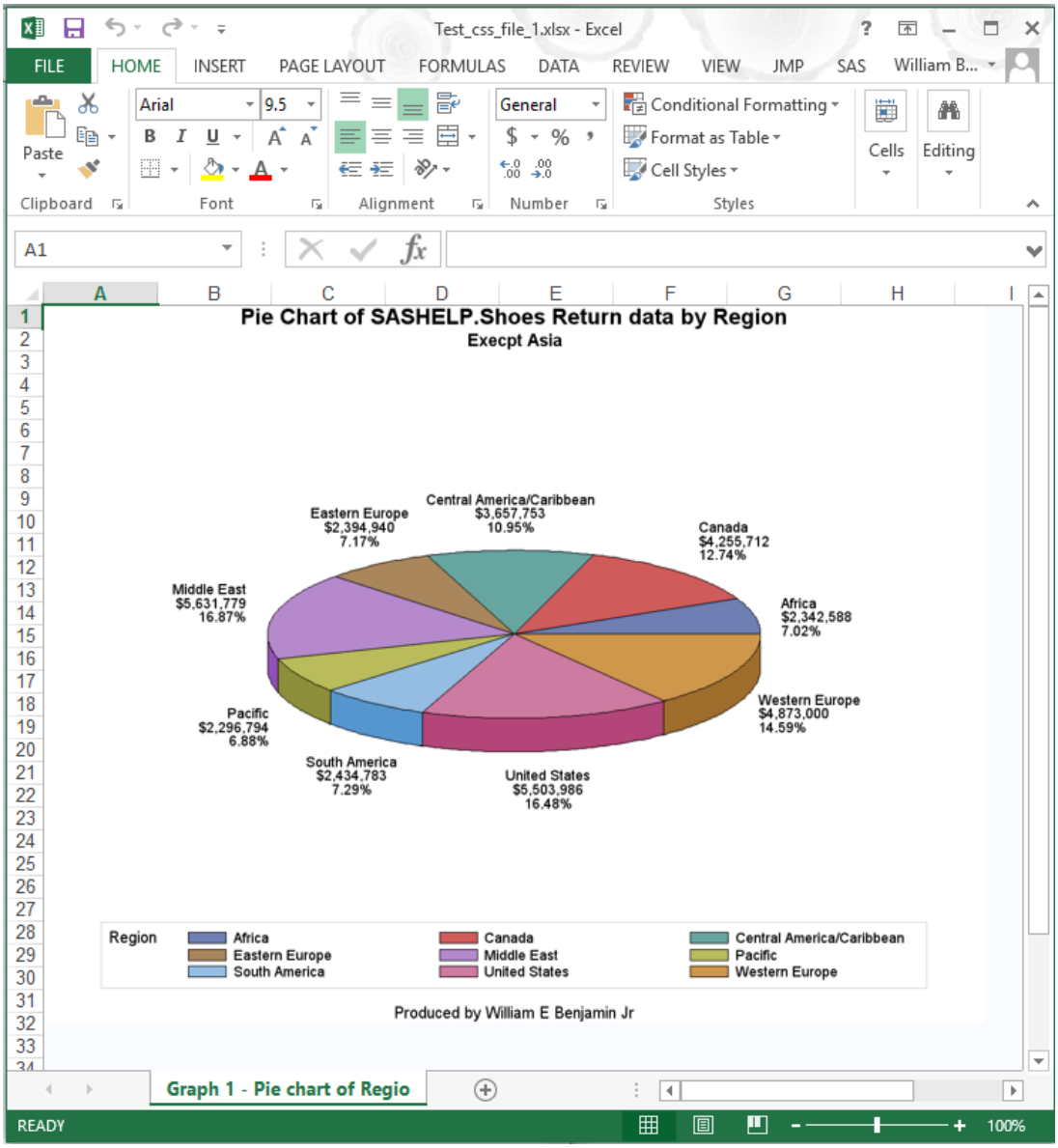
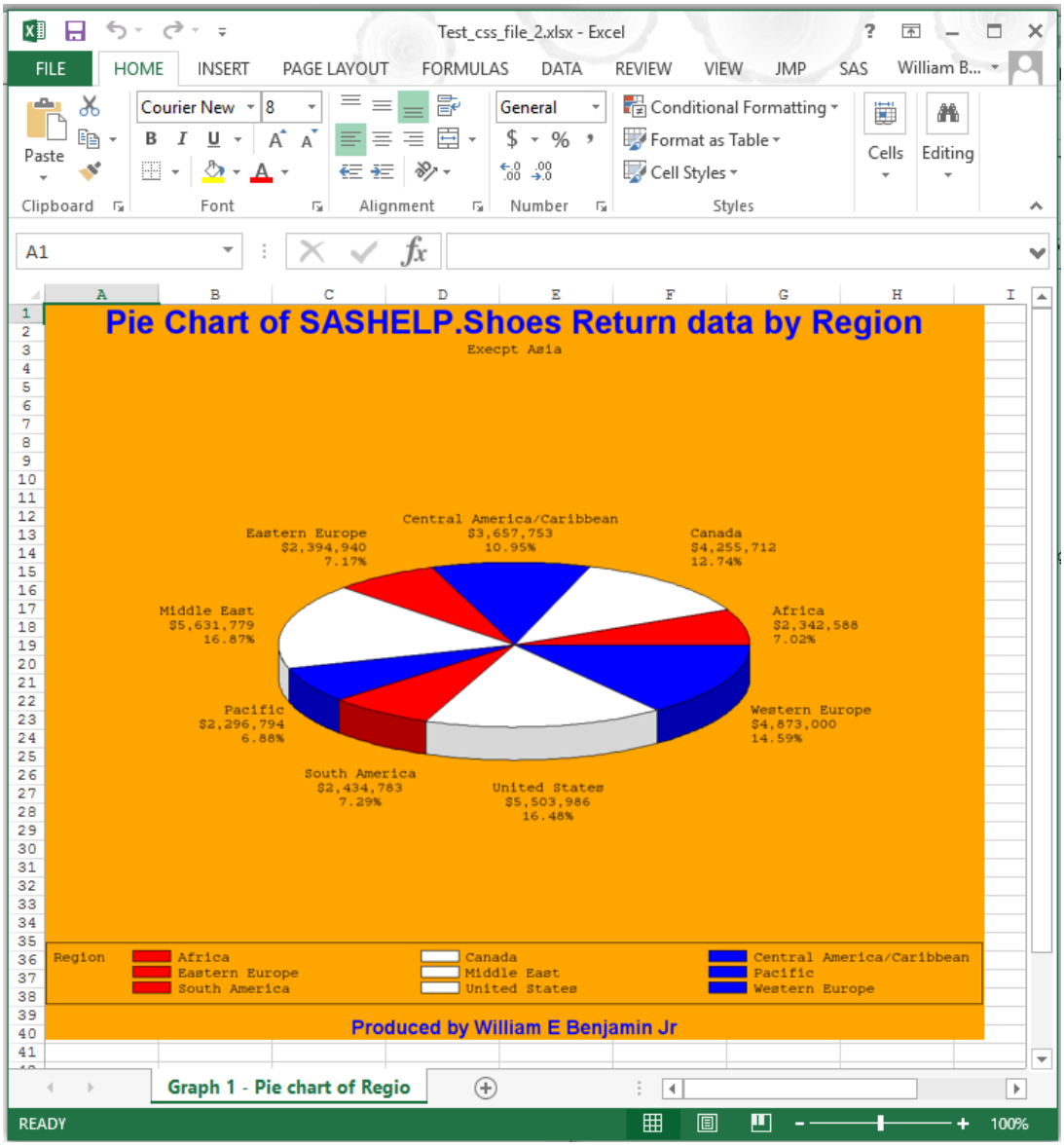


Figure 9 – Excel file output with CSS modifications to the output Test_css_file_2.xlsx Excel Workbook.



CSS Code 3 – The same CSS code as my_css_2.css with a ".body" command to add a background image.

```
.body    {background-image :
           url(&path\data\my_favoriate_photo.jpg);     }
.graphbackground        {background-color: orange;    }
.graphtitle1text        {color: blue;
                         Font: 20pt arial;
                         font-weight: bold;           }
.graphfootnotetext      {color: blue;
                         Font: 12pt arial;
                         font-weight: bold;           }
.graphvaluetext         {color: black;                }
.graphdata1             {color: red;                  }
```

```
.graphdata2            {color: white;              }
.graphdata3            {color: blue;               }
.graphdata4            {color: red;                }
.graphdata5            {color: white;              }
.graphdata6            {color: blue;               }
.graphdata7            {color: red;                }
.graphdata8            {color: white;              }
.graphdata9            {color: blue;               }
```

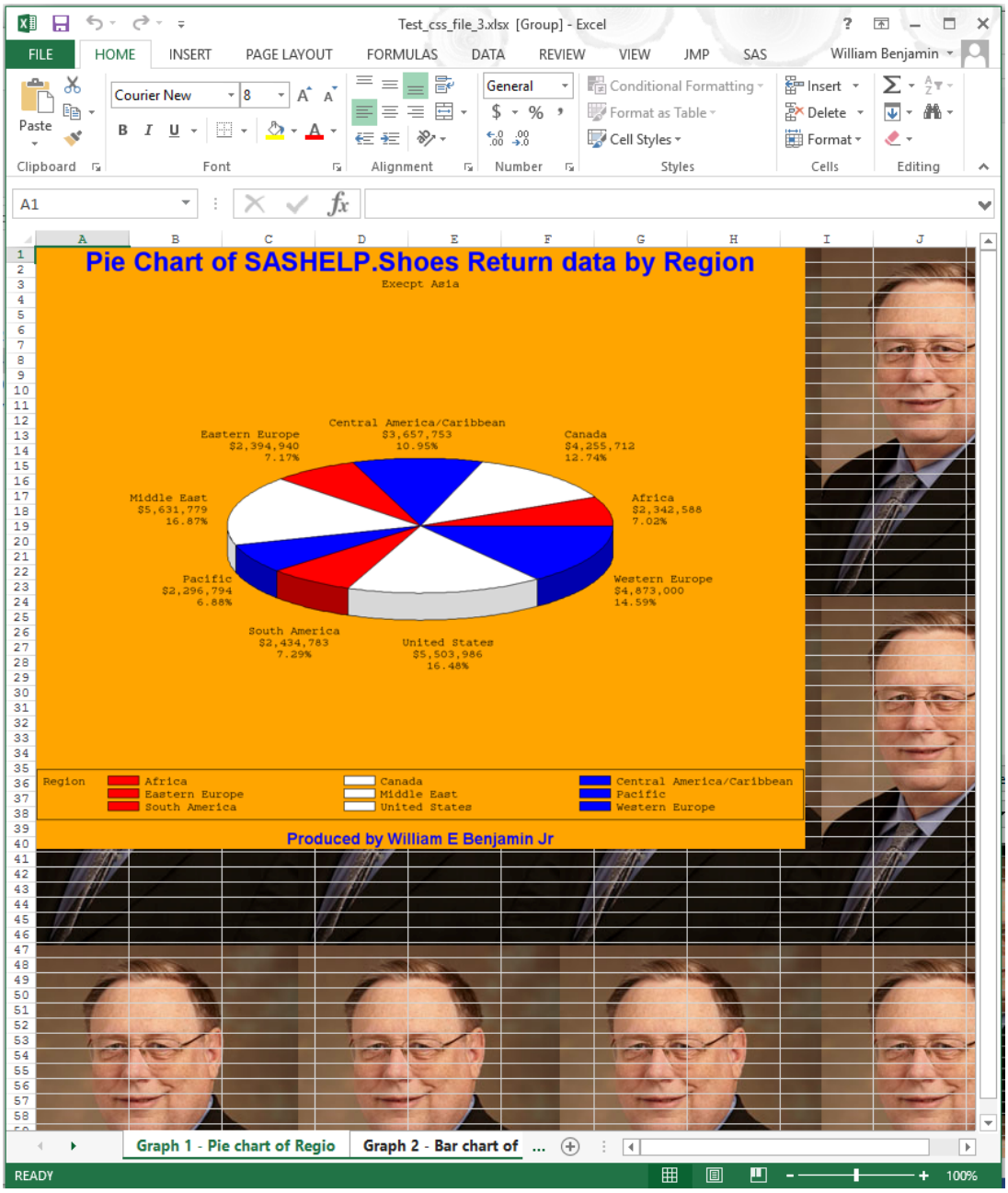
Note that the only change here is the addition of the ".body" command with the backgroung-image:url. This change added a background image[7] as a tile that covered the entire spreadsheet.

SAS Code 6 – SAS Code to create the same Pie Chart with a background image.

```
/* with CSS style modification */
ods excel file = "&path.\out\Test_css_file_3.xlsx"
      cssstyle = "&path.\out\my_css_3.css";
%graph_it;
ods excel close;
```

When the code in code box "SAS Code 6" is added to the code in "SAS Code 5" the output Excel workbook looks like the following. With the added image, which can either add to or detract from the overall beauty of the spreadsheet.

Figure 10 – Excel file output with CSS modifications and an image added to Excel Workbook.



CONCLUSION

I hope you have figured out by now that this new ODS destination called “EXCEL” is a powerful addition to the SAS system. The most important part of the tool is that it is part of the Output Delivery System (ODS) of Base SAS and creates EXCEL output files in the native EXCEL format for *.xlsx files. Additionally you can now write the native format EXCEL workbooks on non-windows computer systems. You can also write SAS Graphs to Excel Workbooks and apply Cascading Style Sheet (CSS) formatting to your output spreadsheets.

REFERENCES

[1] SAS Institute Inc. 2015. SAS® 9.4 Output Delivery System: User's Guide, Fourth Edition. Cary, NC: SAS Institute Inc.

[2] Benjamin, William E., Jr. 2015. *Exchanging Data Between SAS® and Microsoft Excel: Tips and Techniques to Transfer and Manage Data More Efficiently,* Cary, NC: SAS Institute Inc.

[3] Information about CSS Style Rules from the Web at address: https://www.w3.org/community/webed/wiki/CSS_basics

[4] SAS Institute Inc. 2014. SAS® 9.4 Output Delivery System: Advanced Topics. Cary, NC: SAS Institute Inc.

[5] Smith, Kevin D. 2011. "Unveiling the Power of Cascading Style Sheets (CSS) in ODS." Proceedings of the SAS Global Forum 2011 Conference. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings11/297-2011.pdf .

[6] Parker, Chevell. 2014. "Secrets from a SAS Technical Support Guy: Combining the Power of the SAS® Output Delivery System with Microsoft Excel Worksheets." Proceedings of the SAS Global Forum 2014 Conference. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings14/SAS177-2014.pdf

[7] Image drawn from my SAS Author page http://support.sas.com/publishing/authors/benjamin.html

CONTACT INFORMATION

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

Name: William E Benjamin Jr
Enterprise: Owl Computer Consultancy, LLC
Address: P.O.Box 42434
City, State ZIP: Phoenix AZ, 85080
Work Phone: 623-337-0269
E-mail: William@owlcomputerconsultancy.com

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

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

ATTACHMENTS

Attachment 1 - Log output of the ODS EXCEL DOM processing for a simple one variable data file sent to Excel. This output shows CSS items that can be changed. I have highlighted a few of the items that I changed above.

```
NOTE: Copyright (c) 2002-2012 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) Proprietary Software 9.4 (TS1M3)
NOTE: This session is executing on the X64_8HOME  platform.

NOTE: Additional host information:

 X64_8HOME WIN 6.2.9200  Workstation

NOTE: SAS initialization used:
      real time          3.50 seconds
      cpu time           2.65 seconds

1    options linesize=255;
2    data test_css;
3       Column_a =  'My test item';
4    run;

NOTE: The data set WORK.TEST_CSS has 1 observations and 1 variables.
NOTE: DATA statement used (Total process time):
      real time          0.04 seconds
      cpu time           0.03 seconds


5
6    ods excel dom;
<!DOCTYPE html>
<html>
  <head>
    <title>ODS EXCEL DOM</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body dest="excel" class="body">
    <div>
    </div>
    <div>
    </div>
    <body class="startupfunction">
    </body>
    <body class="shutdownfunction">
    </body>
    <div class="body">
7       proc print data=test_css noobs;
8       run;

       <div>
       </div>
       <section id="idx" class="oo" data-name="procprinttable" label="data set work.test_css"
proc="print" output="print" contents-label="data set work.test_css">
         <table class="pageno">
           <tbody>
             <tr>
               <td class="pageno">1
```

```
            </td>
          </tr>
        </tbody>
      </table>
      <table class="bodydate">
        <tbody>
          <tr>
            <td class="bodydate">Thursday, August 25, 2016
            </td>
          </tr>
        </tbody>
      </table>
      <h1 class="systemtitle">
      </h1>
      <table class="systitleandfootercontainer">
        <colgroup>
          <col>
        </colgroup>
        <tr>
          <td class="systemtitle">The SAS System
          </td>
        </tr>
      </table>
      <table class="pageno">
        <tbody>
          <tr>
            <td class="pageno">1
            </td>
          </tr>
        </tbody>
      </table>
      <table class="bodydate">
        <tbody>
          <tr>
            <td class="bodydate">Thursday, August 25, 2016
            </td>
          </tr>
        </tbody>
      </table>
      <h1 class="systemtitle">
      </h1>
      <table class="systitleandfootercontainer">
        <colgroup>
          <col>
        </colgroup>
        <tr>
          <td class="systemtitle">The SAS System
          </td>
        </tr>
      </table>
      <ul class="contentprocname">
      </ul>
      <div class="contentprocname" target="body"
url="c:\users\william\appdata\local\temp\sas temporary
files\_td13608_heber_pc_\_t0000000006d71780\[content_types].xml#idx">
        <ul class="contentitem">
        </ul>
        <div class="contentitem" target="body" url="c:\users\william\appdata\local\temp\sas
```

```
temporary files\_td13608_heber_pc_\_t0000000006d71780\[content_types].xml#idx">
            <div class="pagesitem">
            </div>
            <div>
            </div>
            <div class="data">
            </div>
            <table class="table">
              <colgroup>
                <col type="char" name="column_a">
              </colgroup>
              <div class="data">
                <div class="header">
                </div>
              </div>
              <thead>
                <tr>
                  <th class="header" type="char" unformatted-type="char" index="1"
name="column_a" data-name="column_a" label="column_a">Column_a
                  </th>
                </tr>
              </thead>
              <tbody>
                <tr>
                  <td class="data" type="char" unformatted-type="char" index="1"
name="column_a" data-name="column_a" label="column_a">My test item
                  </td>
                </tr>
              </tbody>
              <div>
              </div>
            </table>
          </section>
        </div>
      </div>
NOTE: There were 1 observations read from the data set WORK.TEST_CSS.
NOTE: PROCEDURE PRINT used (Total process time):
    real time          0.20 seconds
    cpu time           0.15 seconds


9   ods excel close;
    </div>
  </body>
  <div dest="excel" url="c:\users\william\appdata\local\temp\sas temporary
files\_td13608_heber_pc_\_t0000000006d71780\[content_types].xml">
  </div>
</html>
NOTE: Writing EXCEL file: .\sasexcl.xlsx
```