# Creating That Perfect Data Grid Using the SAS® Output Delivery System

Revised

Chevell Parker, SAS Institute Inc., Cary, NC

## ABSTRACT

Using the SAS Output Delivery System (ODS), you can create that perfect data grid with functionality that resembles .NET files or Microsoft Excel applications. Using ODS tagset options or dialog boxes, the data grid dynamically sorts, filters, freezes rows and columns, creates tables and panels, hides columns, applies styles, applies alternating row and columns barriers, and saves data to various formats. Microsoft Office applications also include other functionality such as sending tables from a Web page to individual sheets within a Microsoft Excel workbook, modifying column widths, modifying orientation, adding headers and footers, rotating headers, adding graphics, updating existing worksheets with tables, saving table as native excel files, creating pivot tables, applying Excel formats, and creating Microsoft PowerPoint slides with tables.

## INTRODUCTION

A *data grid*, as defined in this document, is simply one or more tables of data that are generated with a SAS procedure or with the DATA step. The following sections illustrate ways you can make your tables, or grids, of data more useful by using options to do the following:

- enhance the presentation or readability.
- maximize the ability to analyze and manipulate your data
- generate useful reports with limited effort and knowledge of the TEMPLATE procedure and the SAS Output Delivery System (ODS).

Most of the examples in this paper use the TableEditor tagset (or, destination) to enhance the functionality of the data grid. *Tagsets* consist of events (such as System_Title, Header, Data, System_Footer, and others) that are triggered in a particular order based on a SAS procedure or DATA step. The TableEditor tagset inherits from the HTML4 tagset that is used by the HTML destination.

**Note:** To download the TableEditor, see the link in the section Resources. The download file contains a readme file with instructions on how to download the TableEditor. This file also includes all the options that can be used with the TableEditor.

The TableEditor tagset is not a replacement for current methods of generating styles (such as style templates) using ODS. Instead, you can use the tagset in conjunction with current methods to create styles and enhancements for your data grid. The TableEditor also enables you to generate reports using a common set of options across all procedures. Regardless of whether a procedure (such as the PRINT, REPORT, or TABULATE procedures) supports style overrides, you can modify the grid by selecting options of the tagset's parameters in the procedure options.

Much of the functionality generated by the data grid examples that are discussed in this paper is mainly reserved for use in applications such as .NET, Java, or Excel. However, the functionality that the grid generates is created by using the ODS Markup Language, cascading style sheets (CSS), and scripting. Much of the output generated by the data grid is for the Web, and this type of output is supported in the Microsoft Internet Explorer browser.

## ODS MARKUP LANGUAGE

*ODS Markup Language* gives you the ability to modify tagset events in ways that range from modifying a tag to changing the order in which events are triggered, and even to preventing an event from being triggered at all. A good tool for modifying tagsets is the EVENT_MAP tagset. *EVENT_MAP* is a diagnostic tagset that displays the flow of the procedure or the DATA step to include how the events are triggered with all of the metadata. You can use the information produced by this diagnostic tagset to modify and generate other tagsets. The EVENT_MAP tagset was used heavily in the creation of the TableEditor tagset.

The examples presented in this paper were created with ODS Markup Language. ODS Markup Language provides the ability to access and modify all of the data and the metadata that generated from a procedure or a DATA step. Based on this information, you can decide how the output should be generated. In SAS® 9.1, DATA step functions that enable you to generate any type of output that you want to produce were added to the ODS Markup Language: for example, ExcelXP, CSV, LaTex, HTML, various types of XML, and other formats were generated at SAS with the ODS Markup Language.

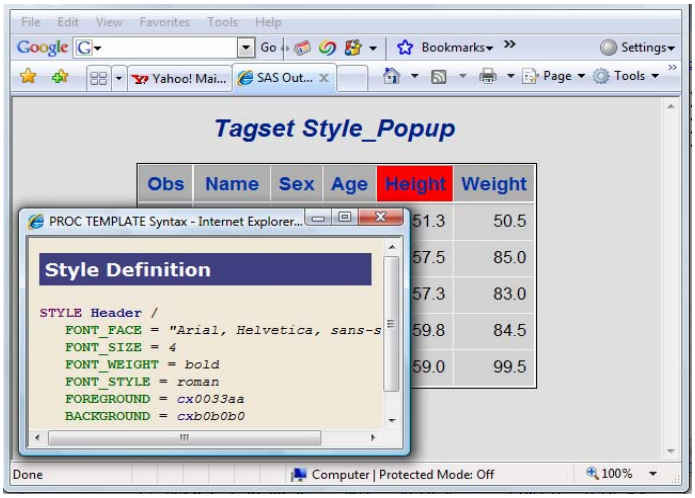
## USING STYLES TO ENHANCE YOUR DATA GRID

You can use ODS to modify styles and enhance your data grid in a variety of ways. For example, one way is to use style templates, which are created by the TEMPLATE procedure. *Style templates* consist of style elements, which modify the overall appearance of output that is generated with ODS. A *style element* is made up of style attributes that modify the behavior of the style element or some part of the output (such as the color or font). PROC TEMPLATE provides various types of templates (such as style, tagset, table, and statgraph templates) that you can apply to data grids. All of these templates are stored in an *item store* (which is a member of a SAS library). You can find detailed information about these style elements and attributes in SAS OnlineDoc 9.1.3 (support.sas.com/onlinedoc/913/docMainpage.jsp) and other SAS documentation.

In addition to using PROC TEMPLATE, you can create styles using other methods, such as cascading style sheets for markup output and style overrides for procedures that support style overrides.

When you are ready to modify a grid, you need to determine what style elements and attributes you want to modify. To do that, you need to know what particular elements and attributes are associated with the output. You can dynamically determine the style elements and attributes that are associated with output by using the diagnostic tagset Style_Popup. Click on any item on the page to display a pop-up window with the style element currently being used along with the attributes specified for the element, as shown in Display 1. You can also mouse over items of the page to see the element names. The following example illustrates how to use the Style_Popup as a diagnostic tagset:

```
ods tagsets.style_popup file='temp.html';

proc print data=sashelp.class;
   title "Tagset Style_Popup";
run;

ods tagsets.style_popup close;
```




**Display 1. Determining Style Elements and Attributes Using the Style_Popup Tagset**

### GENERATING STYLES WITH THE TableEditor TAGSET (DESTINATION)

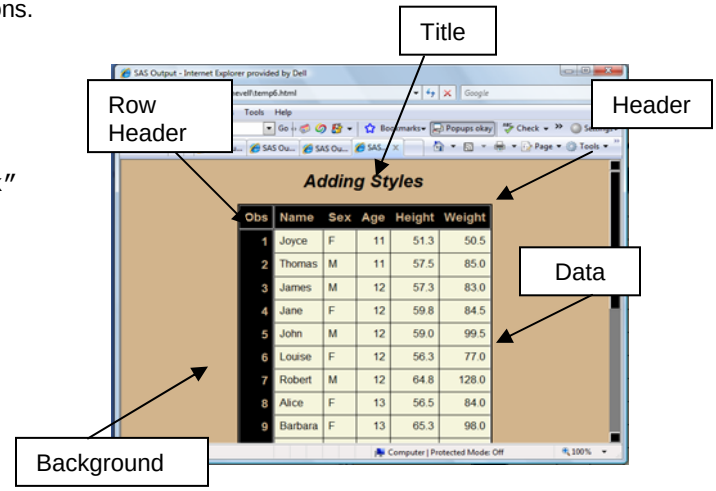
After you determine which style elements and attributes you want to modify, you can easily modify them using options in the TableEditor tagset. This method is an easy approach to creating styles for Web pages because the only thing you need to provide is the value of the option. Specifying options to the tagsets enables you to modify various parts of the output. To generate the perfect data grid, the data needs to stand out in a way that shows the meaning behind the data. People often achieve this effect with graphics. However, you can do an effective job using just data and styles, such as the Styles.Mystyles template, which is provided by default when you download the TableEditor tagset.

This style provides a similar appearance to Windows applications as shown in Display 4. Regardless of the style that you decide to use, you can modify or override the chosen styles using options. You can make modifications using style overrides in some SAS procedures, CSS or with PROC TEMPLATE, as mentioned previously. This method (overriding the style by modifying the options) does not require any knowledge of ODS to generate some very aesthetic and useful reports. You can also use this method in conjunction with the other methods of style generation, as well.

### Creating Styles by Modifying Options

In the following example code, the default style for a data grid is changed by modifying options. The title size is changed, and colors are changed for the header, row headers, page background, foreground, grid line, and scroll bar. Display 2 shows the new output with the modifications.

```
ods tagsets.tableeditor
      file="temp6.html"
      options(header_bgcolor="black"
              header_fgcolor="tan"
              rowheader_bgcolor=”black”
              rowheader_fgcolor="tan"
              data_bgcolor="beige"
              gridline_color="black"
              background_color="tan"
              scrollbar_color="black"
              title_size="20pt" );

proc print data=sashelp.class;
   title c=black "Adding Styles";
run;

ods tagsets.tableeditor close;
```

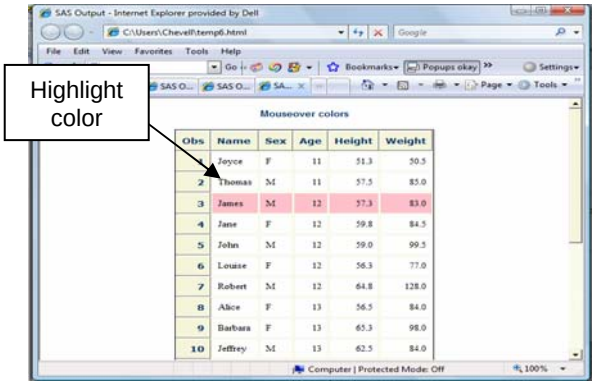



**Display 2. Creating Styles by Modifying Options**

### Highlighting a Row with Color

The next example shows how to enhance the data grid by applying the Styles.Mystyle template, which is installed with the tagset. If you use Styles.Mystyle, or any other style that does not have a background color, you can include the HIGHLIGHT_COLOR= option to highlight a row in a specified color when you move your mouse over the row, as shown here in Display 3.

```
   ods tagsets.tableeditor
       file="temp6.html"
       style=mystyle
       options(highlight_color="pink"
               scrollbar_color="beige"
               title_style="normal" );

   proc print data=sashelp.class;
      title "Mouseover colors";
   run;

   ods tagsets.tableeditor close;
```




**Display 3. Highlighting a Row with Color**

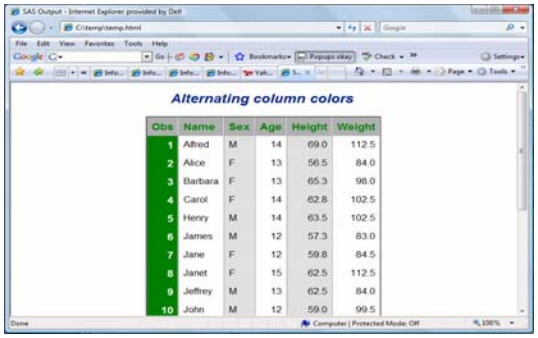
#### ACCENTUATING DATA BY ALTERNATING COLORS

Another way to accentuate your data and make the data grid more readable is to use alternating colors for rows and columns. You can apply alternating colors in a numbers of ways. For example, you can incorporate a table template that uses the CELLSTYLE statement or use a CALL DEFINE statement in PROC REPORT. You can also create this same effect using options in TableEditor, regardless of the procedure that is used.

### Alternating Column Colors

The following example illustrates how to make data grid columns more visible by using alternating column colors that are created with the COL_COLOR_EVEN= and COL_COLOR_ODD= options. As shown in Display 4, this example uses other options, such as ROWHEADER_BGCOLOR= and GRIDLINES=, to create other color effects.

```
ods tagsets.tableeditor
    file="temp.html"
    options(rowheader_bgcolor="green"
            background_color="white"
            rowheader_fgcolor="white"
            header_bgcolor="#b0b0b0"
            header_fgcolor="green"
            data_bgcolor="#b0b0b0"
            col_color_even="#e0e0e0"
            col_color_odd="white"
            gridlines="cols");

proc print data=sashelp.class;
   title "Alternating column colors";
run  ods _all_ close;
```

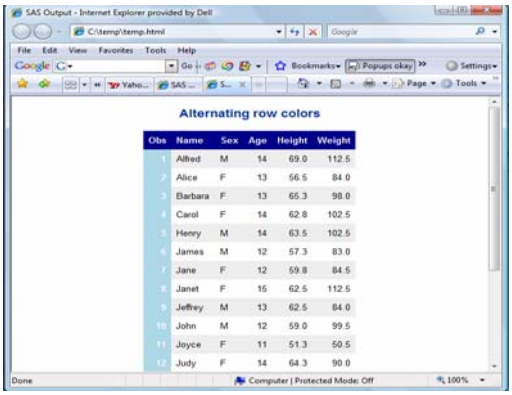



**Display 4. Alternating Column Colors**

### Alternating Row Colors

The next example demonstrates how to modify the background colors for the data rows using the BANNER_COLOR_EVEN= and the BANNER_COLOR_ODD= options to create alternating row colors. You can also change the foreground colors by using the FBANNER_COLOR_EVEN= and the FBANNER_COLOR_ODD= options.

```
ods tagsets.tableeditor
    file="temp.html"
    options(header_bgcolor="darkblue"
            header_fcolor="white"
            rowheader_bgcolor="lightblue"
            banner_color_even="white"
            banner_color_odd="#eeeeee"
            gridlines="no");

proc print data=sashelp.class;
   title "Alternating column colors";
run;

ods_all_close;
```

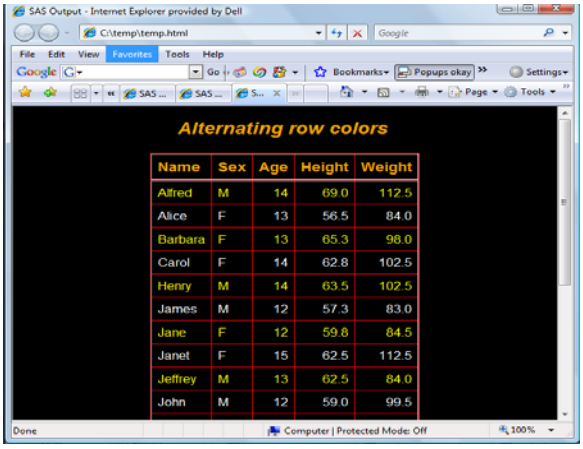



**Display 5. Alternating Row Colors**

### Alternating Foreground Banner Colors

Display 6 generates alternating foreground banner colors and modification of other colors.

```
ods tagsets.tableeditor file="temp.html"
    options(fbanner_color_even="white"
            fbanner_color_odd="yellow"
            background_color="black"
            header_bgcolor="black"
            header_fgcolor="orange"
            data_bgcolor="black"
            gridline_color="red");

proc report data=sashelp.class nowd;
   title c=orange "Alternating row
                   colors";
run;

ods _all_ close;
```

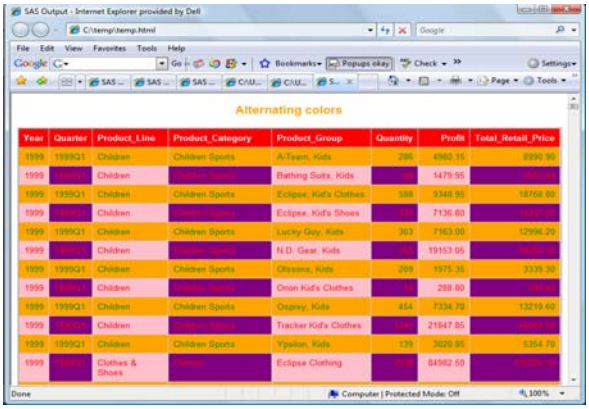



**Display 6. Alternating Foreground Colors**

**ALTERNATING FOREGROUND, BACKGROUND AND COLUMN COLORS**

This example creates a table that uses a combination of alternating foreground and background colors as well as alternating column colors.

```
ods tagsets.tableeditor file="temp.html"
    options(col_color_odd="pink"
            col_color_even="purple"
            banner_color_odd="gold"
            fbanner_color_even="red"
            fbanner_color_odd="green"
            header_bgcolor="red"
            header_fgcolor="white"
            background_color="white");
proc print data=sashelp.orsales noobs;
   title c=gold "Alternating colors";
run;

ods _all_ close;
```




**Display 7. Alternating Foreground, Background, and Column Colors**

**IMAGES AND CAPTIONS**

You can also use Images and captions to enhance your data grid. Images are used, in particular, as a signature or a logo on a page. You can add images in various ways, including with ODS modifying styles and with tagset options. Captions are also useful for enhancing your data table, enabling you to provide descriptive elements to parts of the table. Such descriptions might include a date, information about the table, or a simple explanation.

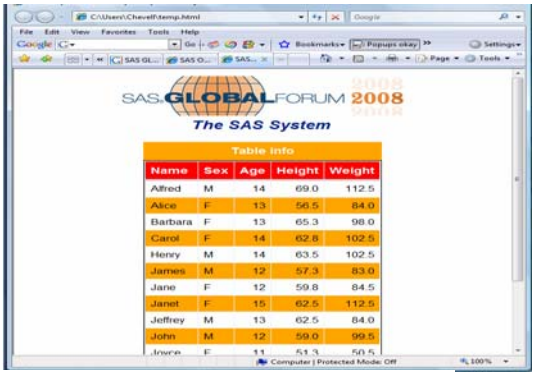
### Adding an Image and a Caption

This example uses the IMAGE_PATH= option in the TableEditor tagset to add an image as a logo. The example also uses the IMAGE_JUST= option, the CAPTION_TEXT= option, and the CAPTION_BACKGROUND= option. You specify your caption text in the CAPTION_TEXT= option. In this example, the caption, Table info, appears in a banner above the table.

**Note:** If you have multiple captions, you must separate each caption string with a comma in the CAPTION_TEXT= option.

The CAPTION_BACKGROUND= option enables you to modify the color of the caption background. You can also modify the captions with the CAPTION_BGCOLOR=, CAPTION_FGCOLOR=, CAPTION_FONT=, and CAPTION_JUST= and the BACKGROUND_IMAGE= options.

```
ods tagsets.tableeditor file="temp.html"
    options(background_color="white"
            image_path="c:\temp\title08.jpg"
            image_just="center"
            header_bgcolor="red"
            header_fgcolor="white"
            banner_color_even="orange"
            banner_color_odd="white"
            caption_text="Table info"
            caption_background="orange");

proc print data=sashelp.class(obs=15) noobs;
run;

ods tagsets.tableeditor close;
```

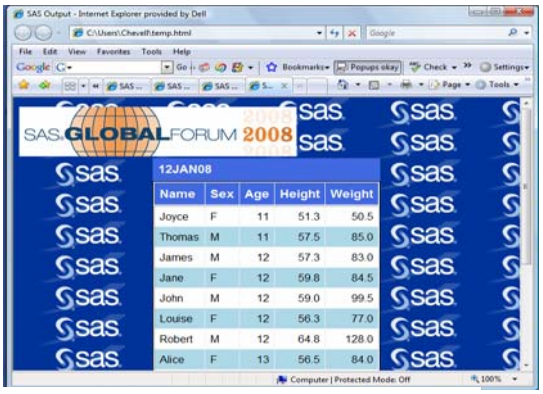



**Display 8. Adding an Image and a Caption**

### Adding an Image as a Background

This example includes a logo image, but it also adds a separate image as the background for the page.

```
ods tagsets tagsets.tableeditor
    file="temp.html"
    options(background_image="c.\sas.gif"
            image_path="c:\temp\title08.jpg"
            header_bgcolor="royalblue"
            header_fgcolor="white"
            banner_color_odd="white"
            banner_color_even="lightblue"
            caption_text="&sysdate"
            caption_just="left");

proc print data=sashelp.class(obs=15) noobs;
run;

ods tagsets.tableeditor.close;
```




**Display 9. Adding an Image as a Background**

**DYNAMIC STYLES**

Styles can be generated that give you the ability to dynamically modify the presentation of your data. To generate dynamic styles, you specify the TableEditor tagset's STYLE_SWITCH= option. This option generates an options list on your Web page from which you can choose and load different styles (CSS files). To load the CSS files, you use the same method that is used with the ODS HTML destination; that is, the URL= option within the STYLESHEET= option. You specify existing CSS files in the URL= option.

You can create CSS files (with a .css extension) that are comparable to SAS template files by using STYLESHEET="*filename*.css" along with the SAS template name in the STYLE= option.

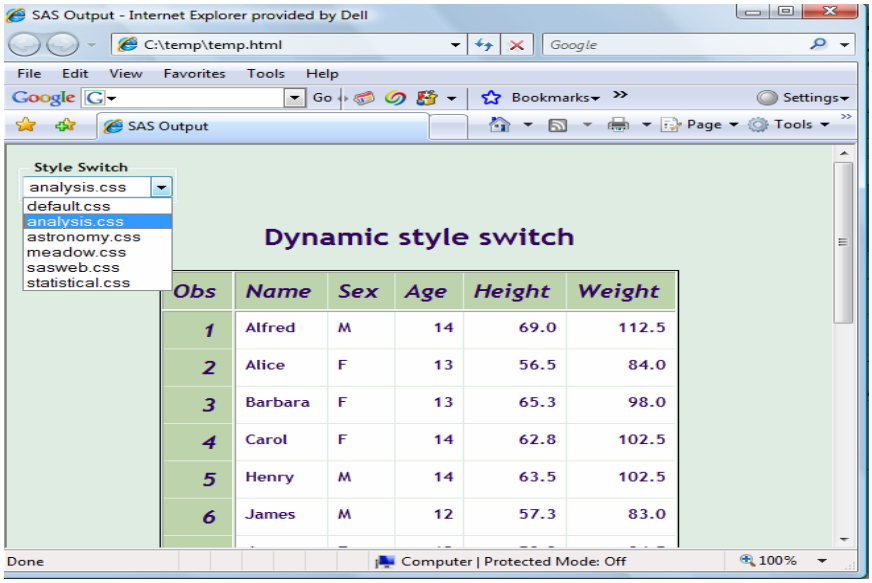
**Note:** If you have SAS® Enterprise Guide® installed, you will have a .css file comparable to the template styles (in the SASHELP location) that are shipped with the SAS® System.

**Creating a CSS File from a Single Template Style**

The following example illustrates how to use the STYLESHEET= option to create a CSS file from a single template style. In this example, the first ODS statement uses the STYLESHEET= option to create the .css file. Later in the program, a macro is included that reads all of the SAS template styles and queries the DICTIONARY.STYLES table in order to convert the template styles to .css files. Once the files are converted, they can be included in the URL= option within the STYLESHEET= option in the ODS statement that appears before the PROC PRINT statement.

```
ods html stylesheet="c:\temp\sasweb.css"  style=styles.sasweb;

   /* CSS setup */
options mprint sgen;
proc sql;
   select scan(style,2,".") into:temp separated by " "
          from dictionary.styles;
run;
quit;

%macro test(path=c:\temp);
%let x=1;

%do %while(%scan(&temp,&x," ") ne);
%let tmp=%scan(&temp,&x," ");

ods html stylesheet="&path\&tmp..css" style=styles.%scan(&tmp,1,".");

%let x=%eval(&x+1);

%end;
%mend;
%test()




ods tagsets.tableeditor path="c:\temp\" file="temp.html"
```

```
    options( style_switch="yes")
            stylesheet=(url="default.css analysis.css astronomy.css meadow.css
                             sasweb.css statistical.css") ;

proc print data=sashelp.class;
   title "Dynamic style switch";
run;

ods tagsets.tableeditor close;
```




**Display 10. Using Dynamic Styles to Enhance Your Data Presentation**

## HIGHLIGHTING DATA

One of the jobs of a good data grid is to enable you to get the most information from the data and effectively analyze it. When data is not in a format that can be easily used, that is when graphs are usually added or data is exported to other applications. You can enhance your data grid in a number of ways that facilitate effective presentation and analysis. This section explains some of these ways to enhance your data grid, including the following:

- scaling tables
- adding Web tabs
- adding filters
- sorting data
- positioning data
- modifying columns
- adding check buttons to highlight data

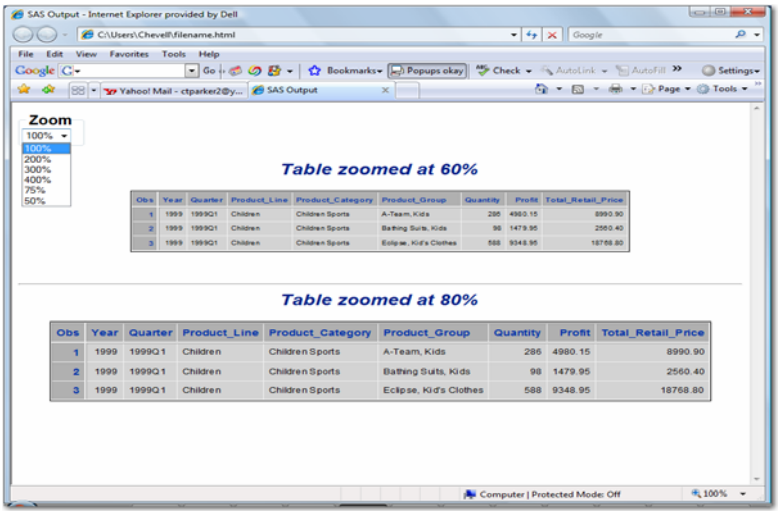
### SCALING TABLES

In certain situations, you might need to scale your table. You can scale tables using the following options:

- **ZOOM= option—**scales the overall output according to the value of the argument that you use. This type of scaling is helpful when you need to reduce the size of your table in order to keep all of your data in a viewable area of the browser.
- **ZOOM_TABLE= option—**enables you to specify the size of individual tables. In the option syntax, you must separate the values with commas.
- **ZOOM_TOGGLE= option—**enables you to control the zooming feature interactively.

The following program uses the ZOOM_TABLE= and the ZOOM_TOGGLE= options to scale two tables.

```
ods tagsets.tableeditor file="filename.html" options(zoom_table="60%,80%"
                                                     zoom_toggle="yes"
                                                     background_color="white");

proc print data=sashelp.orsales(obs=3);
   title "Table zoomed at 60%";
run;

proc print data=sashelp.orsales(obs=3);
   title "Table zoomed at 80%";
run;

ods tagsets.tableeditor close;
```



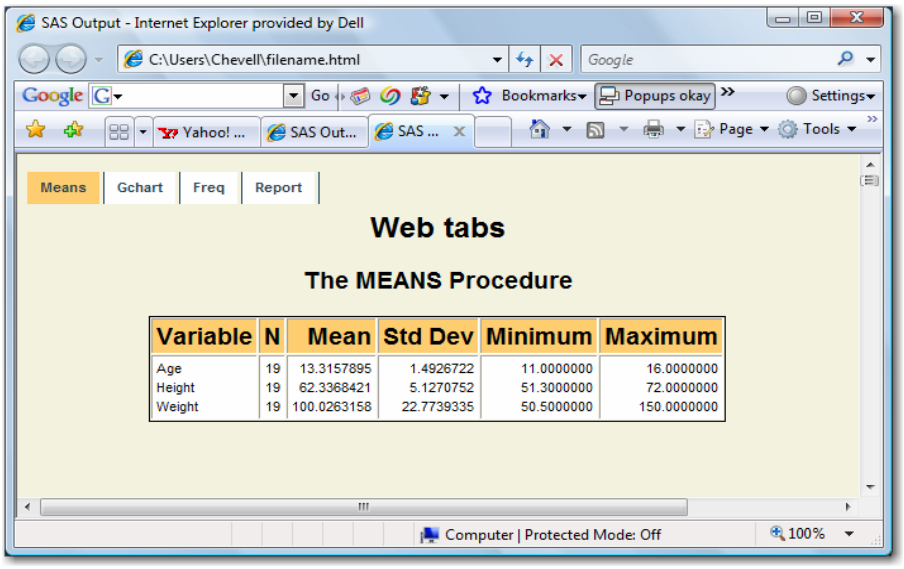

**Display 11. Scaling Tables Using Zoom Options**

**ADDING WEB TABS**

Web tabs enable you to display a single graph or table by selecting its respective tab. You use the WEB_TABS= option to generate tabs. Multiple tabs in the WEB_TABS= option must be separated by commas. While they look similar to worksheets you would see in Microsoft Excel, these sheets are actually generated in Base SAS® software. This is also a good way to print the information in the tab that you are interested in.

The following example illustrates how to create Web tabs with the WEB_TABS= option.

```
ods tagsets.tableeditor file="filename.html"
   options(web_tabs="Means,Gchart,Freq,Report")
   style=statistical;

proc means data=sashelp.class;
run;

proc gchart data=sashelp.class;
   vbar age;
run;
quit;

proc freq data=sashelp.class;
run;

proc report data=sashelp.orsales nowd;
run;

ods tagsets.tableeditor close;
```




**Display 12. Adding Web Tabs to Your Data Grid**

**SORTING**

Another key to an effective data grid is having the ability to view and analyze the data in several ways without having to resubmit the code. The TableEditor destination enables you to do just that—you can dynamically sort table columns by clicking on the column headers. By specifying the SORT= option, you can obtain several views of the data without leaving the page. You can sort on five different data types:

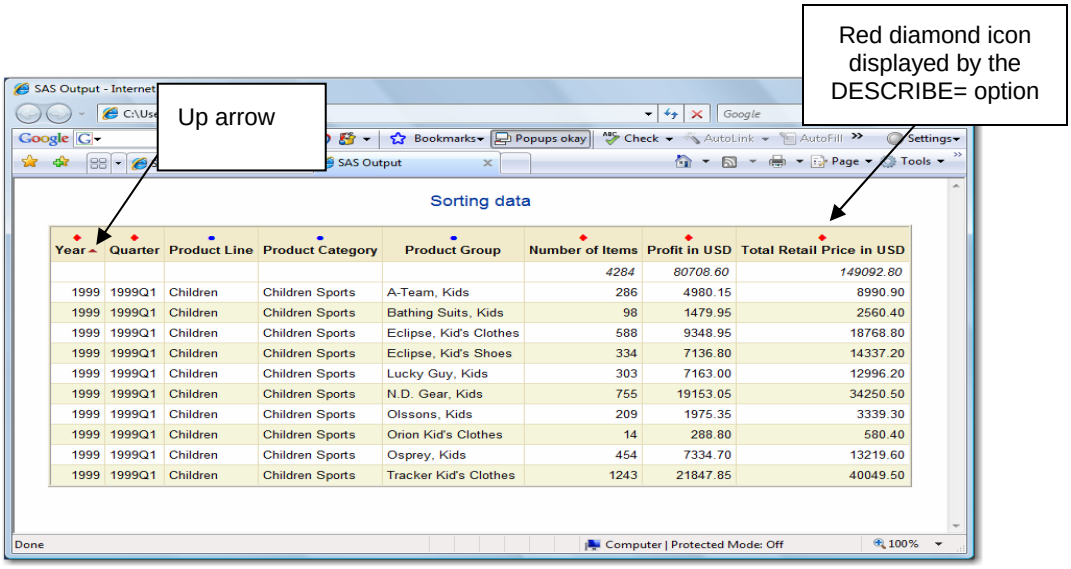
- **String—**used for character data.
- **Number—**used for numbers that are integers and real numbers.
- **Numberx** (Number Extended)**—**used for numbers that contain a comma or a dollar sign
- **Date—**used with the DATE9., MONNY., and MMYYDD. date formats.
- **None—**prevents the columns from being sorted.

These data types are handled automatically with the exception of the SAS 9.1 REPORT procedure which has a default type of String for all fields. To override the default data types or to add a data type in SAS 9.1 PROC REPORT, use the DATA_TYPE= option. This option specifies or overrides the data types of the columns. Each data type must be separated by a comma. Note: You can also set a column's data type using the TAGATTR= style attribute with the appropriate data type for the column. To determine the default data types for a column, use the DESCRIBE= option. This option displays a red diamond in columns that are of type Number or Numberx, a blue circle for columns of type String, and an hourglass for columns of type Date.

**Determining Column Data Types and Sorting the Columns**

The following program uses the SORT= and DESCRIBE= options to sort table columns and to display icons that denote the data type for each column.

```
ods tagsets.tableeditor  file="filename.html" style=sasweb2
    options(sort="yes" sort_arrow_color="brown" describe="yes"
            data_type="Number,Number,String,String,String,Number,Number,Number"
            banner_color_even="beige");

  proc report data=sashelp.orsales(obs=10) nowd;
     title "Sorting data";
     define year / display;
     rbreak after / summarize;
  run;

  ods tagsets.tableeditor close;
```




**Display 13. Using the DESCRIBE= Option to Determine Column Data Types and the SORT= Option to Sort Columns**

### Adding Images, Underlining, and Color to Sorted Columns

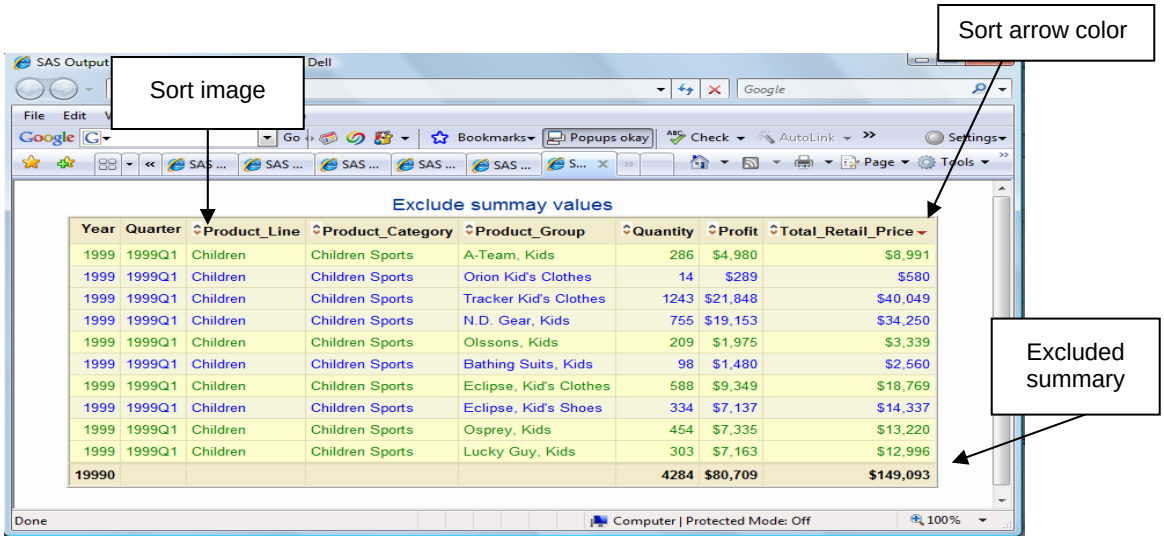
When you specify the SORT= option, you can also include other sort options that enable you to add images, modify the arrow colors, and underline column headers.

- **SORT_IMAGE= option—**displays an image in the column header when you also specify the SORT= option. If the data type for a column is None, that column cannot be sorted. Therefore, a sort image will not be displayed.
- **SORT_UNDERLIN**E= option**—**adds an underline to a column header when you also specify the SORT= option.
- **SORT_COLOR= option—**enables you to specify colors for the Up and Down arrow images.

You can exclude summaries from the sort by using a combination of the EXCLUDE_SUMMARY="YES" option with HTMLCLASS="noFilter".

In the following example, the data type `None` is specified for the first two columns, which means that those two columns cannot be sorted. The other columns use the other various data types. This is a client-side sort, therefore more observations might result in lower performance.

```
ods tagsets.tableeditor file="filename.html" style=sasweb2
    options(sort="yes"
            sort_arrow_color="brown"
            exclude_summary="yes"
            sort_image="c:\temp\newfolder\sort10.jpg"
            data_type="None,None,String,String,String,Number,Numberx,Numberx"
            banner_color_even="beige"
            banner_color_odd="ffffcc"
            fbanner_color_even="blue"
            fbanner_color_odd="green");

proc print data=sashelp.orsales(obs=10) style(gtotal)={htmlclass="noFilter"} noobs;
   title "Exclude summay values";
   format year 5.;
   format profit total_retail_price dollar10.;
   sum _numeric_;
run;

ods tagsets.tableeditor close;
```




**Display 14. Adding Images and Arrow Colors as Well as Excluding the Summary from the Sorting Process**

## FILTERS

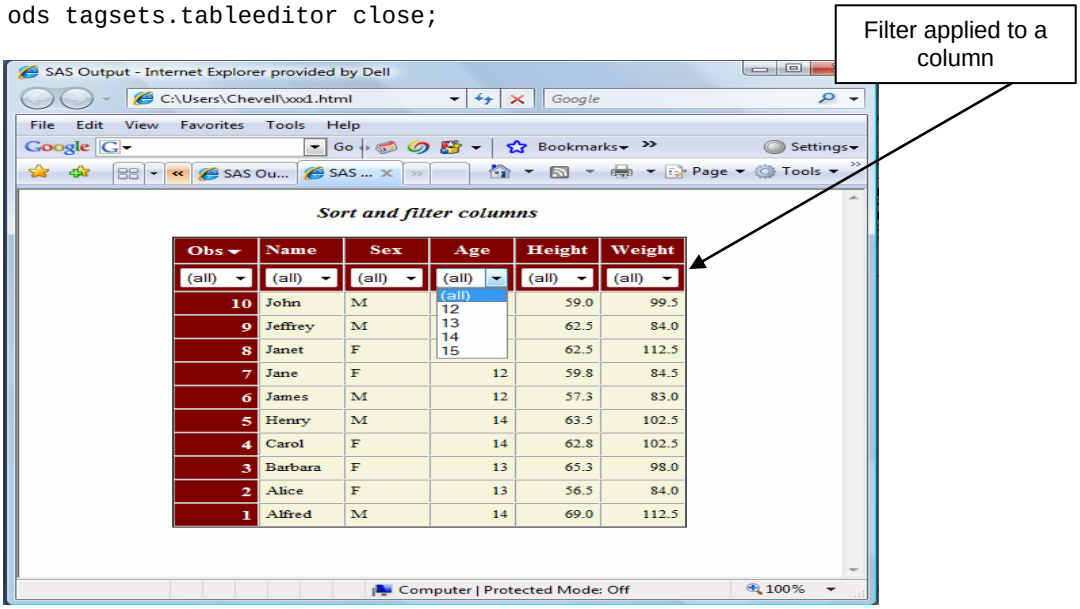
Microsoft Excel has the capability of adding filters to worksheets, which is helpful when you want to drill down and look at a specific group of data. You can add filters that enable you to drill down to a specific value or group of values without leaving the page. Using the TableEditor, you can also add filters that enable you to drill down to specific values or groups of values without leaving the Web page. The TableEditor tagset uses the following options to create a filtering effect:

- **AUTOFILTER= option—**indicates specific tables to filter. If this option is not specified, filters will be applied to all tables. Values for the **AUTOFILTER_TABLE**= option are a list of table numbers, separated by commas. When you specify this option along with sorting options, you can create a dynamic and powerful grid.
- **FILTER_COLS= option—**enables you to select specific columns to filter. You specify the column numbers, separated by commas.
- **AUTOFILTER_ENDCOL= option—**enables you to specify the last column to which filters are applied.
- **AUTOFILTER_WIDTH= option—**enables you to add a uniform width to the filters.

The following program applies filters to the table columns:

```
ods tagsets.tableeditor  file="filename.html"
    style=styles.printer
    options(header_bgcolor="maroon"
            header_fgcolor="white"
            rowheader_bgcolor="maroon"
            rowheader_fgcolor="white"
            data_bgcolor="beige"
            sort="yes"
            autofilter="yes"
            autofilter_width="4em");
```

```
proc print data=sashelp.class(obs=10);
   title “Sort and filter columns”;
run;

ods tagsets.tableeditor close;
```




**Display 15. Filtering Table Columns**

**POSITIONING**

The ability to position data is important to any data grid. Using the following TableEditor options, you have the capability to freeze column headers and row headers, much like you can in Excel.

- **FROZEN_HEADERS= option—**enables you to lock, or freeze, column headers. When the headers are locked, they cannot be moved when you scroll the page.
- **FROZEN_HEADERS_ALL= option---**Enable you to lock you to lock, or freeze, column headers for multiple tables.
- **FROZEN_ ROWHEADERS**= **option—**enables you to freeze row headers. If FROZEN_ROWHEADERS= is set to “YES”, procedures or statements with columns that use the ROWHEADER style element will be frozen when the page width has been reached. Examples of procedures that use the ROWHEADER style element include the following:
    - PROC PRINT with the OBS column or with variables that are specified in the ID statement.
    - PROC TABULATE for variables on the row dimension. This option can also take values for the column numbers for the columns that should use this ROWHEADER style allowing it to freeze.

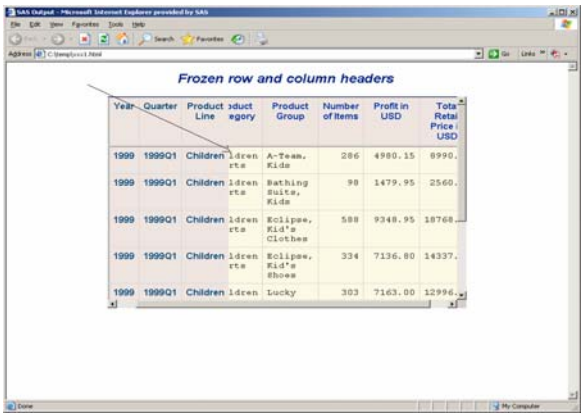
The FROZEN_HEADERS= and the FROZEN_ROWHEADERS= options are single-table options; that is, they are only applicable for a single table on the page with Internet Explorer. The FROZEN_HEADERS= option is designed for use on a single long table. The FROZEN_ROWHEADERS= option is designed for use on a single, wide table.

The FROZEN_HEADERS= and FROZEN_ROWHEADERS= options can work as standalone options, or in conjunction with the PAGEHEIGHT= and the PAGEWIDTH= options. When the PAGEHEIGHT and PAGEWIDTH options are not specified with the FROZEN_ROWHEADERS and the FROZEN_HEADERS options, the page width and the page height are generated dynamically which is the width and the height of the viewable screen. When the height or width is reached, the browser adds scrollbars vertically and horizontally, and column headers or rows are frozen.

The PAGEWIDTH= option works well with either percentages or a width; the PAGEHEIGHT= option works better with a defined width. These options help create a uniform look for each table. When you use the FROZEN_ROWHEADERS= option, the browser recalculates for each row, which might cause performance problems for long tables. (This does not happen with the FROZEN_HEADERS= option.) Therefore, you should not use the FROZEN_ROWHEADERS= option for long tables, but rather for wide tables.

In the following example, the column headers and the first three columns are frozen.

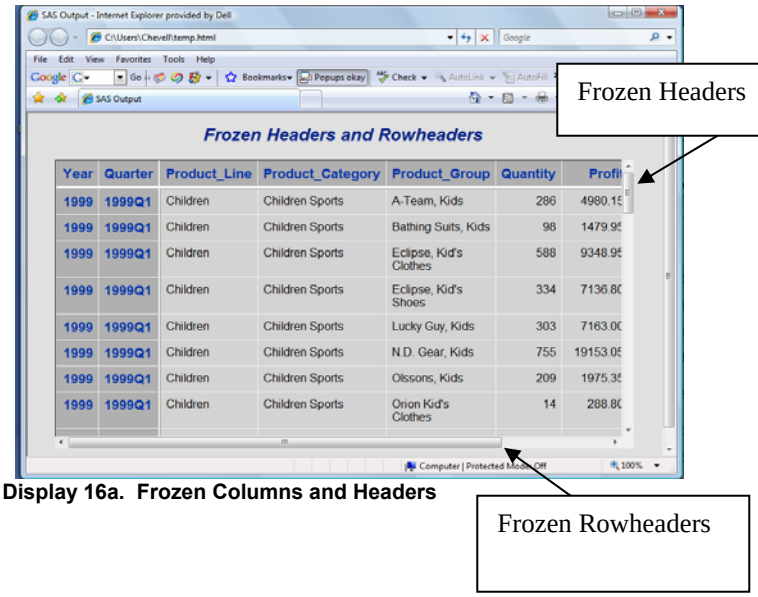
```
ods tagsets.tableeditor
    file="filename.html"
    options(frozen_headers="yes"
            pageheight="300"
            frozen_rowheaders="1,2,3"
            pagewidth="50%");

proc report data=sashelp.orsales(obs=10)
            nowd;
   title "frozen rows and titles";
run;

ods tagsets.tableeditor close;
```




**Display 16. Frozen Columns and Headers**

In the following example, the column headers and the first two columns are frozen. By not applying a pageheight= and the pagewidth= option, it is applied dynamically based on the viewable area.

```
ods tagsets.tableeditor
file="temp.html"
options(frozen_headers="yes"
        frozen_rowheaders="yes");


proc print
data=sashelp.orsales(obs=50);
title "Frozen Headers and
Rowheaders";
id year quarter;
run;

ods tagsets.tableeditor close;
```



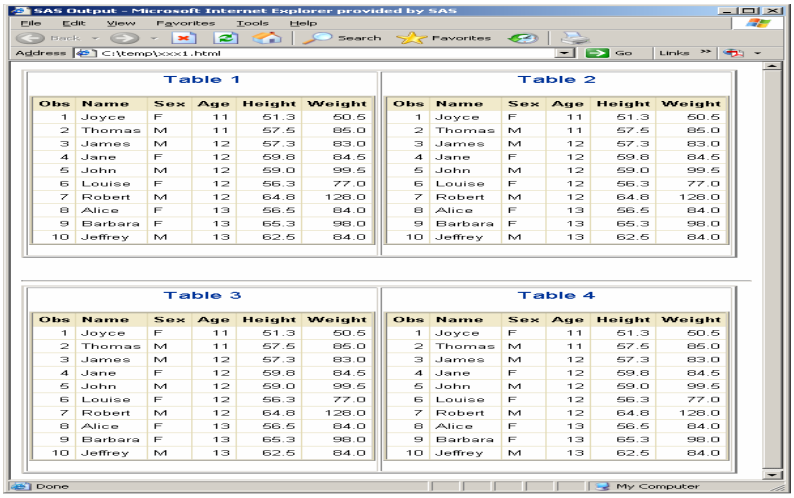

**Display 16a. Frozen Columns and Headers**

In addition to freezing rows and columns headers, another great way to enhance your grid is to use panels. You can convert tables and graphs into panels in order to display multiple tables or graphs on one page or to display a table and a graph together. To create panels, you use the PANELCOL= option.

The following code generates two tables side-by-side on each page.

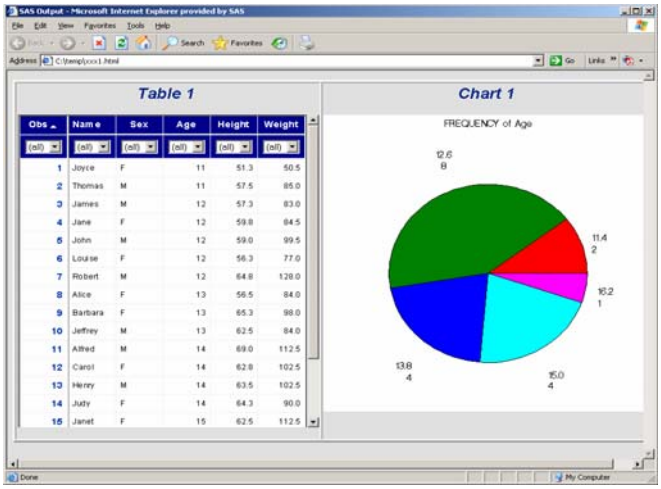
```
ods tagsets.tableeditor file="filename.html"
options(panelcols="2");

proc print data=sashelp.class;
run;

proc print data=sashelp.class;
run;

ods tagsets.tableeditor
    options(panelcols="2");
```

```
proc print data=sashelp.class;
   title "Table 1";
run;

proc print data=sashelp.class;
   title "Table 2";
run;

ods tagsets.tableeditor close
```




The next example displays a graph and a table side by side. The table has frozen headers as well as sorted and filtered columns.

```
goptions hzise=5in vsize=5in ftext=swiss;
ods tagsets.tableeditor file="filename.html"
    notitle
    options(header_bgcolor="darkblue"
            header_fgcolor="white"
            rowheader_bgcolor="white"
            data_bgcolor="white"
            frozen_headers="yes"
            pageheight="500"
            sort="yes"
            autofilter="yes"
            autofilter_width="4em"
            zoom_table="80%"
            panelcols="2");

proc print data=sashelp.class;
   title "Table 1"
run;

proc gchart data=sashelp.class
   title "Chart 1";
   pie age;
run;
quit;

ods tagsets.tableeditor close;
```




**Display 18. Displaying a Table and a Graph Side by Side**

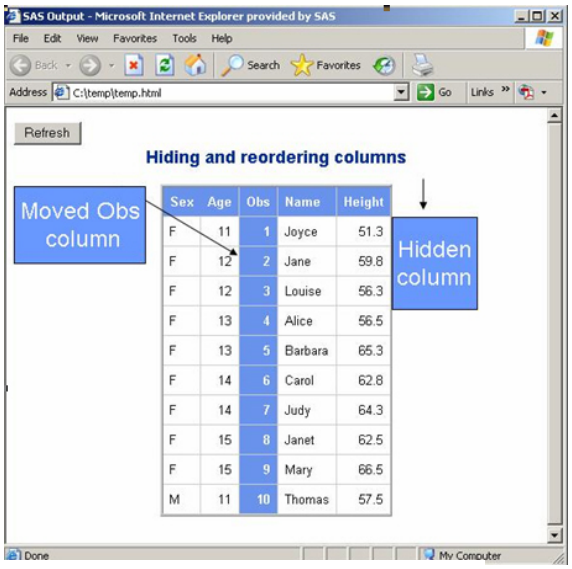
**MODIFYING COLUMNS**

Using the TableEditor tagset, you can remove (hide) columns, remove column headers, and reorder the columns. You can use the following tagset options to perform these tasks:

- **HIDE_COLS= option—**At times, a full view of columns in a table is not necessary. An effective data grid will provide the ability to dynamically remove columns from view. You can remove columns from a table with the HIDE_COLS= option. When you add the HIDE_COLS= option, you can then remove (hide) unwanted columns by double-clicking those columns. When this option is specified, a refresh button is automatically added to the page. When you click the refresh button, the hidden columns are restored.
- **HEADER_DISPLAY= option—**In cases where you do not want to use column headers to describe the data, you can use the HEADER_DISPLAY= option. Specifying HEADER_DISPLAY="No" removes column headers. Some SAS procedures, such as PROC REPORT, have the NOHEADER option. However, the HEADER_DISPLAY= option will work across procedures.
- **REORDER_COLS= option—**There may be times when you want to change the layout of columns. The REORDER_COLS= option gives you the flexibility to dynamically reorder columns. You can drag a column to other locations. Note, however, that when you save or refresh the page, the default column order is preserved.

The following example illustrates the HIDE_COLS = option and the REORDER_COLS= option:

```
ods tagsets.tableeditor
    file="temp.html"
    style=sasweb
    options(hide_cols="yes"
            reorder_cols="yes");

proc print data=sashelp.class(obs=10);
   title "Hiding and reordering columns";
run;

ods tagsets.tableeditor close;
```



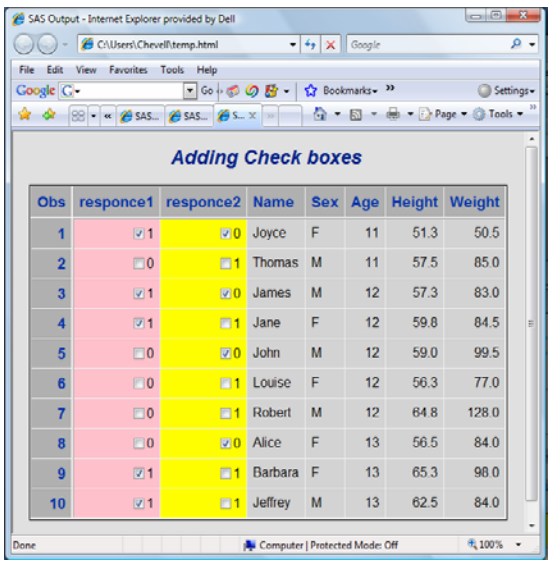

**Display 19. Hiding and Reordering Columns**

**CHECK BOXES**

Displaying data in various ways makes for a good data grid. Another useful feature you might want to incorporate is the check box. You can add check boxes to cells in your table. The reason why you might want to use check boxes for a column is that they can provide a good visual for difficult-to-read columns. For instance, adding a check box to a column that has a Boolean value will help to clarify the visual display of this column.

You use the RADIO= option to add check boxes to one or more columns. In the option, you need to separate each column number with a comma. Using the RADIO= option along with the RADIO_CHECKED= option enables you to check values in a cell based on the value. In the example supplied, the value of the column 2 of the cell should be checked when the value is 1, and column 3 should be checked when the value is 0.

```
data one;
   retain responce1 responce2;
   set sashelp.class;
   responce1=round(ranuni(0));;
   responce2=round(ranuni(0));;
run;

proc print;
run;

ods tagsets.tableeditor file="temp.html"
    options(radio="2,3"
            radio_checked="1,0"
            highlight_cols="2#pink,3#yellow");

proc print data=one(obs=10);
```



```
   title "Adding Check boxes";
run;

ods tagsets.tableeditor close;
```

**Display 20. Adding Check Boxes to a Data Grid**

## ADDING FEATURES FOR PRINTING, BROWSER WINDOW COMMUNICATION, AND NAVIGATION

After you enhance your data grid, you might want to customize the page for printing, modify elements of the actual browser window, and include a table of contents with a Tree view. This section explains the methods for performing these additional enhancements.

### PAGE SETUP AND PRINTING OPTIONS

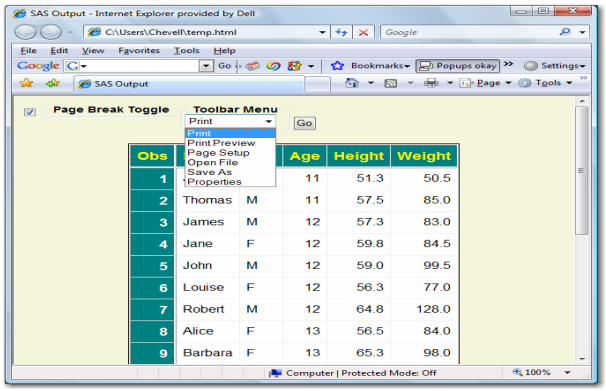
The perfect data grid should not only look good on the screen, but it should also look good when you print it. Typically, applications that have frozen headers will print only the material that can physically be viewed on the page and not the full data. However, regardless of the amount of data that is presented in the browser window, you can use the TableEditor destination to print all of your data, including column headers.

You can use the following TableEditor options to set up your data grid for printing:

- **PAGEBREAK= option—**When you print your data grid, you might want to remove page breaks from the Web pages to prevent unnecessary wasting of paper when the information is printed. You can do that using **PAGEBREAK="No"—**you can also use this same option to place multiple tables on a page. To do that, include the number of tables you want placed on a page. For example, if you want to include three tables, the option will be PAGEBREAK="3".
- **TOP_MARGIN=, BOTTOM_MARGIN=, LEFT_MARGIN=, and RIGHT_MARGIN= options—**These options give you the ability to modify page margins.
- **PAGEBREAK_TOGGLE= option—**You can dynamically modify page breaks directly from the Web page with this option by selecting or de-selecting the check box.
- **PRINT_DIALOG= option—**This option is an interactive option that adds a dialog box to your Web page with a list of functions you can select, including **`Print`**, **`Print Preview`**, **`Save As`**, **`Open File`**, and **`Properties`**.
- **PRINT_ZOOM= option—**This option enables you to scale your printed output.
- **FIT2PAGE= option—**This option enables you to fit your output to the page so it will not be truncated when you print the page.
- **BLACKANDWHITE= option—**With this option, you can specify that you want to print in black and white rather than color.

The following example illustrates the use of the two interactive options, PAGEBREAK_TOGGLE= and PRINT_DIALOG=, as well as the FIT2PAGE= option.

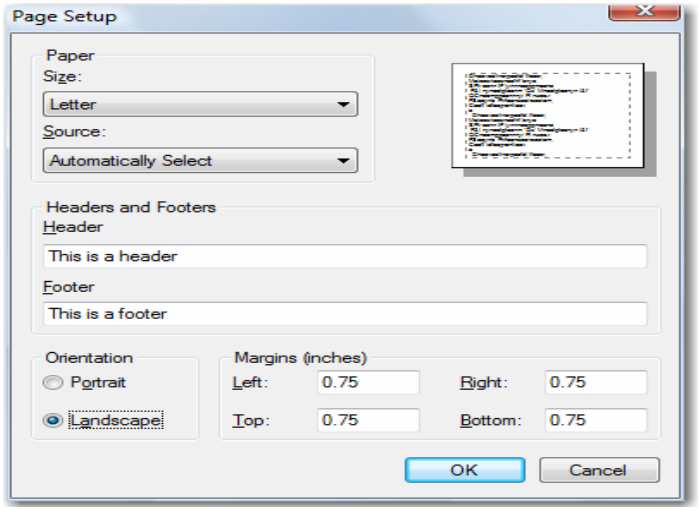
```
ods tagsets.tableeditor file="filename.html"
                        options(print_dialog="yes"
                                pagebreak_toggle="yes"
                                fit2page="yes"
                                rowheader_bgcolor="teal"
                                rowheader_fgcolor="white"
                                header_bgcolor="teal"
                                header_fgcolor="yellow"
                                background_color="beige"
                                data_bgcolor="white");

proc print data=sashelp.class;
run;
ods tagsets.tableeditor close;
```




You can also use an ActiveX control to manipulate headers, footers, and page orientation.. Typically, you cannot make these kinds of modifications programmatically, except within a limited subset of browsers. The TableEditor provides this capability by pointing to an ActiveX control. With earlier versions of the TableEditor, if the ActiveX control did not exist on your computer, TableEditor pointed to a hosted cabinet (.cab) file that downloaded the control over the Internet when you specified the ORIENTATION=, the PRINT_HEADER=, or the PRINT_FOOTER= options.

In the current version of TableEditor, the ActiveX control resides in a .zip file with the tagset installation. This file also includes an executable (.exe) file that you can use for a single installation of the ActiveX control. If you need to install the control for multiple users, you can use the .cab file to download the ActiveX control over the Internet. After you install this ActiveX control, which was created by MeadCo, you can use the ORIENTATION=, PRINT_HEADER=, and PRINT_FOOTER= options just as you can in Microsoft Excel.

```
ods tagsets.tableeditor file="filename.html"
                        options(print_header="This is a header"
                                print_footer="This is a footer"
                                orientation="landscape" );

proc print data=sashelp.class;
run;

ods tagsets.tableeditor close;
```




**Display 22. Modifying Headers, Footers, and Orientation**

**BROWSER WINDOW OPTIONS**

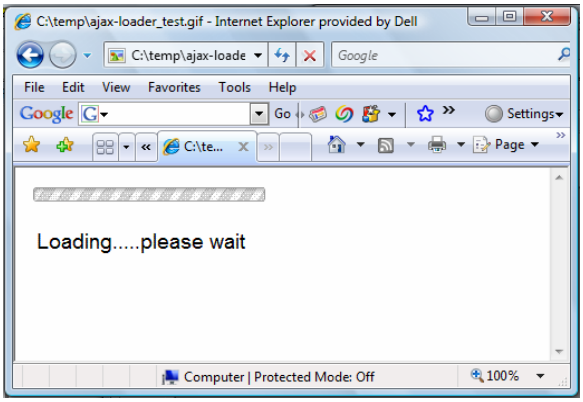
In addition to enhancing your data grid, you might also want to enhance the information in your browser window or modify the size of the window. You can perform these tasks using the following options:

- **WINDOW_TITLE= option—**You can use this option to display a title for the browser window.
- **WINDOW_STATUS= option—**This option enables you to add text or messages to the browser window status line.
- **ALERT_TEXT= option—**If you have alert information that you need to make visible, this option creates a dialog box with that information that displays when the page is opened. You will not be able to continue viewing the page until you actively close the dialog box. This behavior ensures that the alert will be read.
- **FIT2PAGE_MSG= option—**This option displays a dialog box that specifies how much the page is being scaled in order to fit the page.
- **LOAD_MSG= option—**With large tables, the browser typically does not indicate that the page is loading, so you might think that it is not loading. You can solve this problem with the LOAD_MSG= option, which displays a message specifying that the Web page is loading.
- **LOAD_IMG= option—**This option enables you to display an image when the page is loading. (For example, you might want to display an animated image until the page is completely loaded.). The .zip file for the TableEditor tagset download contains a couple of images you can use.

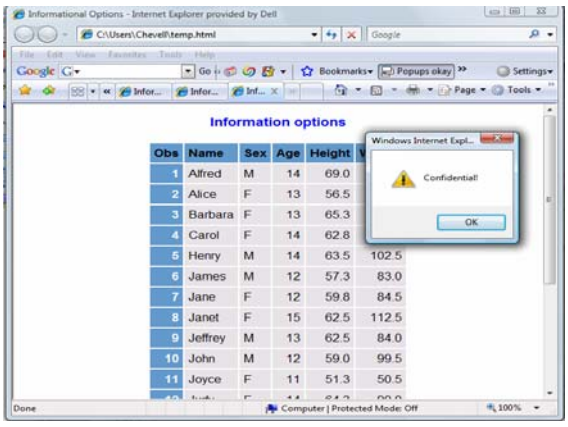
- **WINDOW_SIZE= option—**This option enables you to modify the size of the browser window when it is opened. You can specify X and Y coordinate values for a specific height and width size, or you can use the value MAX to open output in a maximized browser window. This option is not shown in the following example, but the option syntax is WINDOW_SIZE='x-value, y-value | MAX". Setting the size in this option guarantees how the window will be displayed, regardless of the environment.
- **INCLUDE= option—**You can use this option to include various types of files (HTML, PDF, Word, Excel) into the browser. You just include the name of the file (if it is relative to the generated file) or the file pathname that you want inserted into the browser. The file is inserted as an IFRAME file.

The following example illustrates the use of the WINDOW_TITLE=, WINDOW_STATUS=, FIT2PAGE=, ALERT_TEXT=, LOAD_MSG=, and the LOAD_IMG= options.

```
ods tagsets.tableeditor file="temp.html" style=barretsblue
                        options(window_title="Informational Options"
                                window_status="Status bar info"
                                fit2page_msg="yes"
                                alert_text="Confidential!"
                                load_msg="yes"
                                load_img="c:\temp\image_files\sas.gif");

proc print data=sashelp.class;
   title "Dynamic style switch";
run;

ods tagsets.tableeditor close;
```




**Display 23. Loading Message Created with the LOAD_MSG= Option**




**Display 24. Effects of the WINDOW_TITLE=, WINDOW_STATUS=, WINDOW_SIZE=, and ALERT_TEXT= Options**

**NAVIGATION AIDS**

Unless you only have one page of data, you probably need navigation tools, such as a table of contents and drill-down links, to help users navigate through your data. The section explains how to use various methods to create

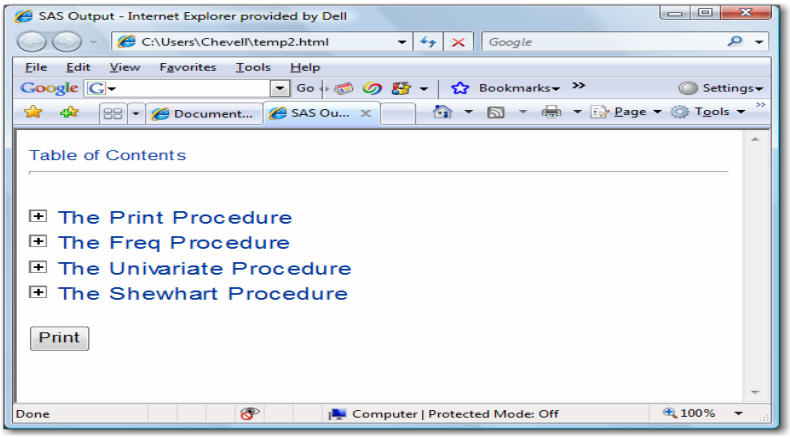
these navigation aids.

**Creating a Tree View with the Table of Contents**
You can create a tree view within the table of contents using the TableEditor. By default, TableEditor generates a collapsed table of contents. To expand the table of contents, you simply click the plus sign (+) next to each item. The TableEditor has several options you can use to modify this table of contents:

- **TOC_EXPAND= option—**displays the table of contents in expanded form
- **OPEN_IMAGE= option—**replaces the default “open” image ( “-“) in the table of contents with an image that you specify
- **CLOSE_IMAGE= option—**replaces the default “close” image ( “+“) in the table of contents with an image that you specify
- **LEAF_ IMAGE= option—**replaces the image that links to the table or graph with an image that you specify
- **TOC_PRINT= option—**provides the ability to print pages from the table of contents
- **TOC_BACKGROUND= option—**modifies the background color of the table of contents
- **TOC_PRINT= option—**adds a **`Print`** button to the table of contents

The next example shows the default table of contents generated with the TableEditor and adds a **`Print`** button.

```
ods tagsets.tableeditor  file="temp.html"
                         contents="temp2.html"
                         frame="frame2.html" style=sasweb
                         options(toc_print="yes");

proc print data=sashelp.class;
run;

proc freq data=sashelp.class;
run;

proc univariate data=sashelp.class;
run;

proc shewhart data=sashelp.class;
   cchart age*sex;
run;
quit;

ods tagsets.tableeditor close;
```




**Display 25. Adding a Tree View to the Table of Contents**

## EXPORTING YOUR DATA TO OTHER APPLICATIONS

For a more effective data grid, you need to be able to export data to other applications such as Excel. Exporting to other applications gives you the capability to further analyze and manipulate your data in the environment of your choice. The TableEditor has export options that enable you to move your data into other formats or applications. This section discusses how to use that feature to format and move your data from a Web page into Office application.

**EXPORTING DATA TO EXCEL**

After you export your data to Excel, you can use automation from Internet Explorer and JavaScript to modify Microsoft Office objects that enable you to access parts of applications, such as Excel. These objects let you modify such features as page setup, filters, frozen headers, zooms, naming and updating sheets, style creation, graphics, pivot tables and more.

Because this process uses JavaScript, which instantiates an ActiveX object, you can only perform these functions from Internet Explorer. Based on the options that are specified to the TableEditor, you can export data to various formats and applications. If you use the TableEditor FILE_FORMAT= option (or any of the export options), an **Export** button will be added to your page. If you click this button, the application begins the export process. The data, along with the formatting (depending on the options you choose), is copied to the target application (Excel). This process is all driven with ODS Markup Language. The next sections discuss how you can use options to manipulate the output for the exported data.

### Carrying Styles Forward into Excel

Styles are carried forward from the Web-page into Excel by default if no Excel style has been specified. You can specify styles that are not native Excel styles by applying any of the style methods that were discussed earlier in the section "Using Styles to Enhance Your Data Grid." This section discusses additional formatting options that you can use to include Excel's built-in formatting functionality. When you export to Excel (as well as to Word and PowerPoint), the style viewed on the Web page is passed along to the target application as long as the format supports styles. Therefore, you can specify Excel styles as well as the 25 styles that are shipped with Base SAS software, the CSS style overrides, or the styles that you specify using options to the tagset.

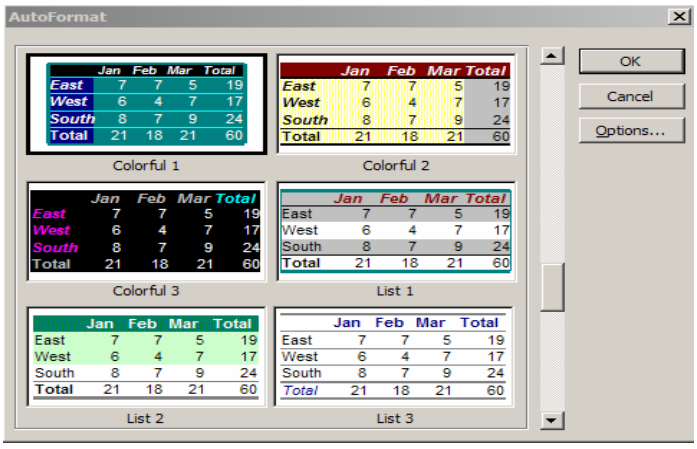
**Note:** The browser supports quite a few more colors than Excel; therefore, when colors are exported, they might not be exactly the same in Excel as they were on the Web page.

To use Excel styles, you need to specify the AUTO_FORMAT= option. These styles handle text wrapping pretty well. If you select **Formats ► AutoFormat** from the Excel menu, you can see a list of available styles.

**Note:** Styles that are specified in the AUTO_FORMAT= option will override any format that is specified using styles or options within SAS.

The following styles are available in Excel:

- Colorful1
- Colorful2
- Colorful3
- List1
- List2
- List3
- Classic1
- Classic2
- Classic3
- Accounting1
- Accounting2
- Accounting3
- Accounting4
- None
- Simple
- Report1
- Table1

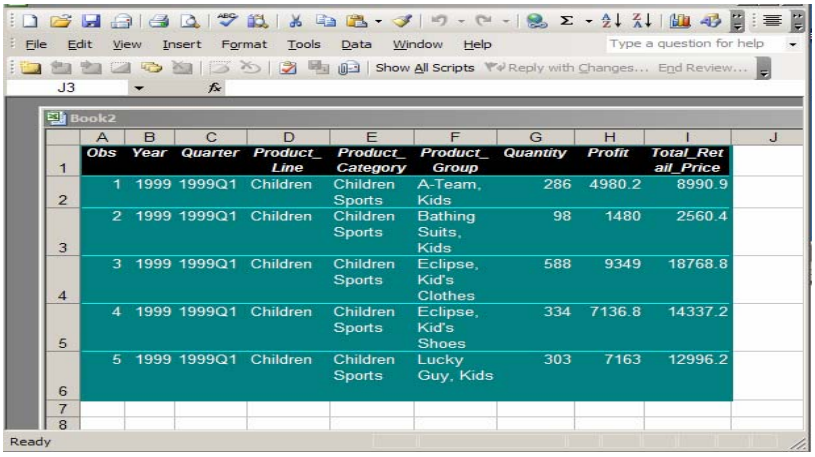



**Display 26. Examples of Several Excel Styles**

Using the AUTO_FORMAT= option, you can take advantage of any of the Excel styles. For example, you can use AUTO_FORMAT="None" to specify no formatting. Or you can specify AUTO_FORMAT="Simple", which incorporates minimal formatting by adding gridlines and putting the titles in boldface. Note, if you don't need the intermediate HTML file, use the AUTO_EXCEL= option which will start the export immediately without selecting the export button. Also, a path can be given to the work location which will not persist this file.

The following example uses the Excel style Colorful1:

```
ods tagsets.tableeditor file="%sysfunc(getoption(WORK))\temp.html"
options(auto_format="colorful" auto_excel="yes");

proc report data=sashelp.orsales(obs=5) nowd;
run;

ods tagsets.tableeditor close;
```

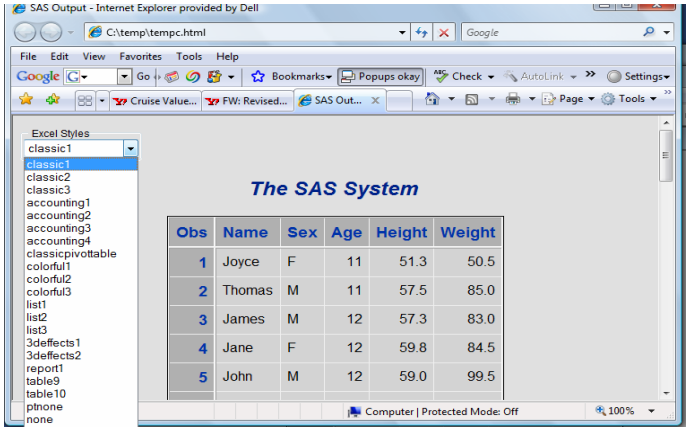



| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Obs | Year | Quarter | Product_ Line | Product_ Category | Product_ Group | Quantity | Profit | Total_Retail_Price | |
| 2 | 1 | 1999 | 1999Q1 | Children | Children Sports | A-Team, Kids | 286 | 4980.2 | 8990.9 | |
| 3 | 2 | 1999 | 1999Q1 | Children | Children Sports | Bathing Suits, Kids | 98 | 1480 | 2560.4 | |
| 4 | 3 | 1999 | 1999Q1 | Children | Children Sports | Eclipse, Kid's Clothes | 588 | 9349 | 18768.8 | |
| 5 | 4 | 1999 | 1999Q1 | Children | Children Sports | Eclipse, Kid's Shoes | 334 | 7136.8 | 14337.2 | |
| 6 | 5 | 1999 | 1999Q1 | Children | Children Sports | Lucky Guy, Kids | 303 | 7163 | 12996.2 | |
| 7 | | | | | | | | | | |
| 8 | | | | | | | | | | |

**Display 27. Data Grid Exported to Excel in the Colorful1 Style**

The Excel styles can also be selected dynamically by adding the AUTO_FORMAT_SELECT= option. This option generates a selection list with all of the available styles. Selecting a style exports the data to Excel with the with the selected style.

```
ods tagsets.tableeditor path="c:\temp" file="tempc.html"
options(auto_format_select="yes");

proc print data=sashelp.class;
run;

proc print data=sashelp.class;
run;

ods tagsets.tableeditor close;
```




**Display 27A. Data Grid Exported to Excel in the with selected Style**

**Exporting Formats and Saving Files in Those Formats**

You can export a number of file formats, including XML, HTML, SYLK, XLS, TXT, DOC, CVS, and others. As mentioned previously, if you specify the FILE_FORMAT= option with one of the file formats, an **`Export`** button for that format is added to the Web page. Selecting the **`Export`** button exports the file, in the specified format, to Excel.

To save one or more tables in a specified format, you need to specify the FILE_FORMAT= option in conjunction with the EXCEL_SAVE_FILE= option. If you omit the FILE_FORMAT= option, Excel will use the XLS format, which is Excel's default native format. Note that when you specify a file path in the EXCEL_SAVE_FILE= option, you need to use double backward slashes instead of single slashes; for example:

```
excel_save_file="c:\\temp\\test.csv"
```

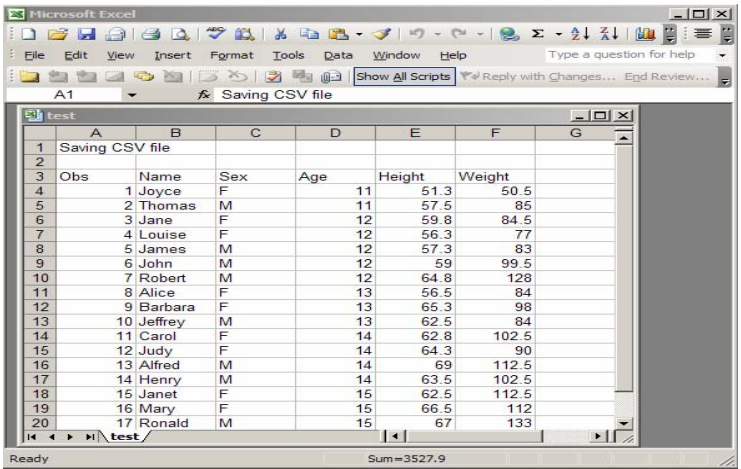
If you want to save this file in the background without opening Excel, specify the option VISIBLE="No".

Other methods for saving files are also available:

- **EXCEL_SAVE_DIALOG= option—**opens a dialog box in which you can name and save a file to a specified location. You can use this option along with the DEFAULT_FILENAME= option in order to specify a default filename in the dialog box.
- **EXCEL_SAVE_PROMPT= option—**generates a prompt window in which you can specify a location to which you want to save the file.
- **SAVEAS= option**—enables you to save a file, and it adds a **Save As** button on the Web page. You can also use this option in conjunction with the DEFAULT_FILENAME= option in order to specify a default filename.
- **QUIT= option—**closes a file automatically after the file is saved.

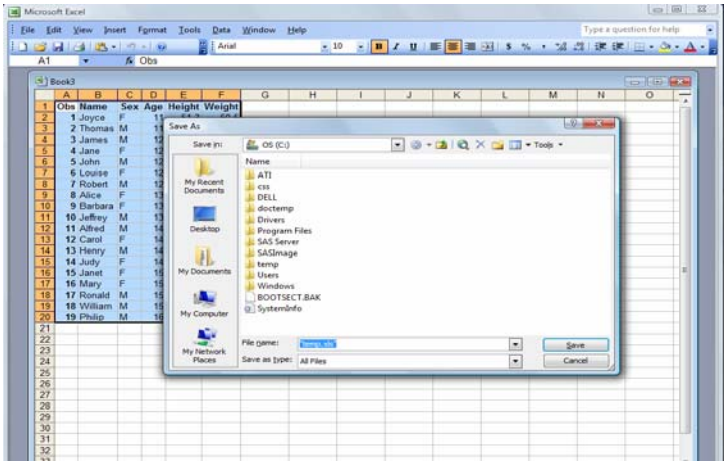
In the following example, the file TEMP.HTML is saved, in the background, as a CSV file.

```
ods tagsets.tableeditor file="temp.html"
                        options(sheet_name="Test"
                                excel_save_file="c:\\temp\\test.csv"
                                file_format="csv"
                                visible="no" quit="yes");

proc print
data=sashelp.class;
run;

ods tagsets.tableeditor
close;
```



**Display 28. Saving a File in the CSV Format**

```
ods tagsets.tableeditor file="temp.html" style=styles.mystyle
                        options(excel_save_dialog="yes"
                                default_file="c:\\temp.xls");

proc print data=sashelp.class;
run;

ods tagsets.tableeditor close;

/* This code segment uses the EXCEL_SAVE_DIALOG= option to create a
   dialog box in which you specify the location where you want the file
   to be saved.                                                        */
```



**Display 29. Creating a Dialog Box in Which You Specify a Filename and a Location in Which to Save a File**

## Enhancing Your Excel Output

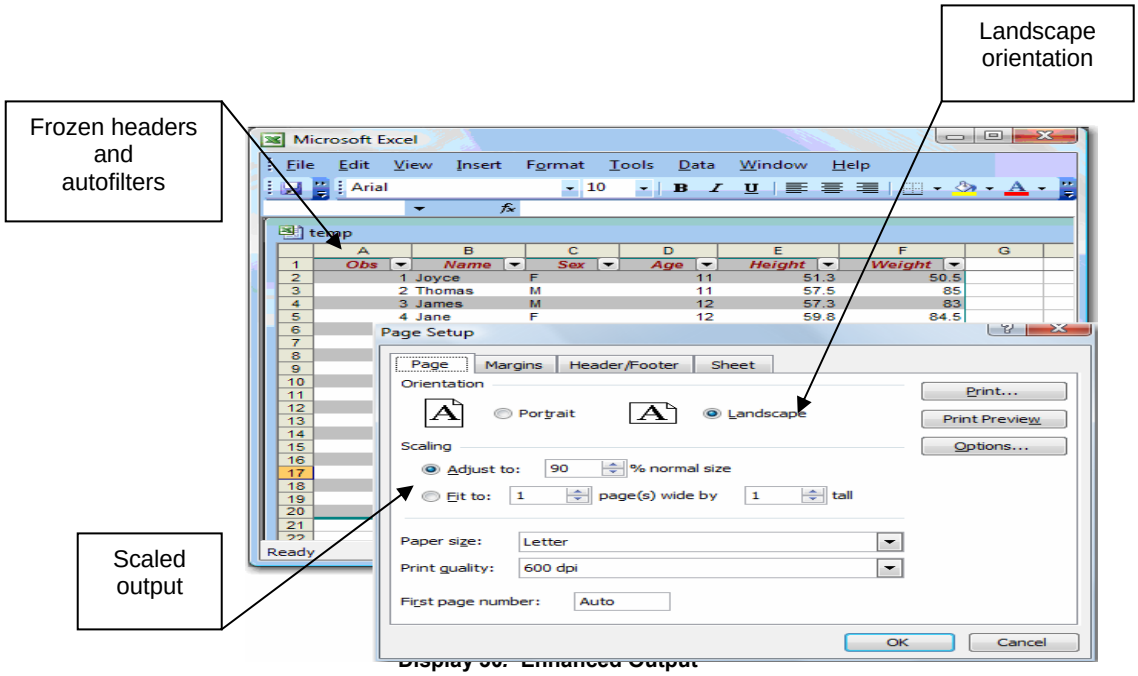
The TableEditor tagset provides a set of options that can generate new Excel worksheets and workbooks. You can use these options to display properties, define page setup, filter, and perform other tasks, creating high-quality reports at the end. All of these Excel options add an **`Export`** button to your Web page.

- **EXCEL_AUTOFILTER= option—**adds automatic filters to the headers.
- **EXCEL_ FROZEN_ HEADERS= option—**enables you to lock (freeze) column headers.
- **EXCEL_ZOOM= option—**enables you to zoom in on a worksheet.
- **EXCEL_DEFAULT_HEIGHT= option—**specifies a default height.
- **EXCEL_DEFAULT_WIDTH= option—**specifies a default width.
- **EXCEL_ORIENTATION**= option**—**specifies the orientation of a worksheet.
- **EXCEL_SCALE**= **option—**scales the printed output.

The following example uses a combination of these options to create presentational output:

```
ods tagsets.tableeditor file="temp.html"
                        options(excel_zoom="80"
```

```
                                excel_scale="90"
                                excel_autofilter="yes"
                                excel_frozen_headers="yes"
                                excel_orientation="landscape"
                                auto_format="list1"
                                excel_save_file="c:\\temp\\temp.xls");

proc print data=sashelp.class;
run;

ods tagsets.tableeditor close;
```




**Modifying Worksheets and Workbooks**

At times, you might need to modify your Excel worksheets and workbooks in order to add information or rename them, to move individual tables into separate worksheets, and so on. The TableEditor tagset provides options that help you accomplish these tasks:

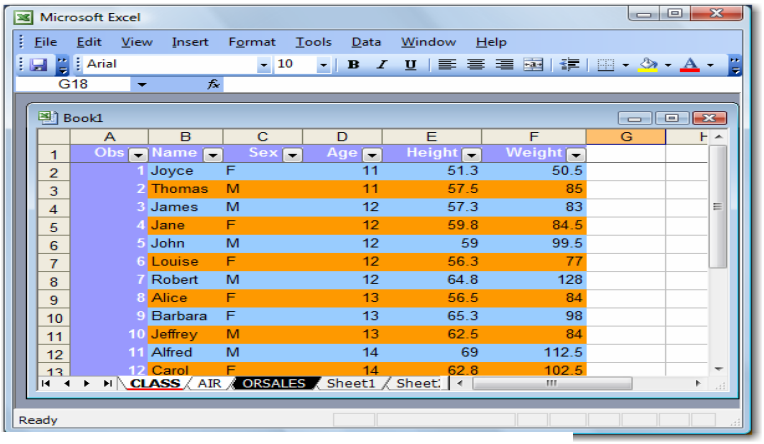
- **SHEET_NAME= option—**This option enables you to assign meaningful names to worksheets that are generated from tables. This allows the worksheet to have a name closely related to the meaning behind the data. This option provides the ability to name one or more tables from the Web page that are exported to the spreadsheet.
- **EXCEL_TABLE_MOVE= option—**When combined with the SHEET_NAME= option, this option enables you to place separate tables from your Web page into separate worksheets within the workbook. To specify tables on separate worksheets, you specify the sheet names, separated by commas, in the SHEET_NAME= option. Then, specify table numbers (from the page where you are moving the tables), also separated by commas, in the EXCEL_TABLE_MOVE= option in the order that corresponds to their respective sheet names in the SHEET_NAME= option. Each table will be on an individual sheet in the workbook, and the tables will be named based on the corresponding sheet name.

**Note:** If the EXCEL_TABLE_ MOVE= option is not specified, the first table is selected.

- **AUTO_FORMAT=** option**—**This option enables you to specify Excel styles for your generated worksheets. (You can also use styles from the Web page.) You can use this option in conjunction with other options for data presentation.

The following example uses the SHEET_NAME= option and the EXCEL_TABLE_MOVE= option to add worksheets and move tables to those worksheets:

```
ods tagsets.tableeditor file="temp.html" style=sasweb
                        options(sheet_name="CLASS,AIR,ORSALES"
                                excel_table_move="1,2,3"
                                excel_default_height="15"
                                excel_frozen_headers="yes"
                                excel_autofilter="yes"
                                banner_color_odd="lightblue"
                                banner_color_even="orange"
                                excel_save_file="c:\\temp\\temp.xls");

proc print data=sashelp.class;
run;

proc print data=sashelp.air(obs=20);
run;

proc print data=sashelp.orsales(obs=20);
run;

ods tagsets.tableeditor close;
```




| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | Obs | Name | Sex | Age | Height | Weight | |
| 2 | 1 | Joyce | F | 11 | 51.3 | 50.5 | |
| 3 | 2 | Thomas | M | 11 | 57.5 | 85 | |
| 4 | 3 | James | M | 12 | 57.3 | 83 | |
| 5 | 4 | Jane | F | 12 | 59.8 | 84.5 | |
| 6 | 5 | John | M | 12 | 59 | 99.5 | |
| 7 | 6 | Louise | F | 12 | 56.3 | 77 | |
| 8 | 7 | Robert | M | 12 | 64.8 | 128 | |
| 9 | 8 | Alice | F | 13 | 56.5 | 84 | |
| 10 | 9 | Barbara | F | 13 | 65.3 | 98 | |
| 11 | 10 | Jeffrey | M | 13 | 62.5 | 84 | |
| 12 | 11 | Alfred | M | 14 | 69 | 112.5 | |
| 13 | 12 | Carol | F | 14 | 62.8 | 102.5 | |

**Display 31. Modified Worksheet**

**Inserting Worksheets**

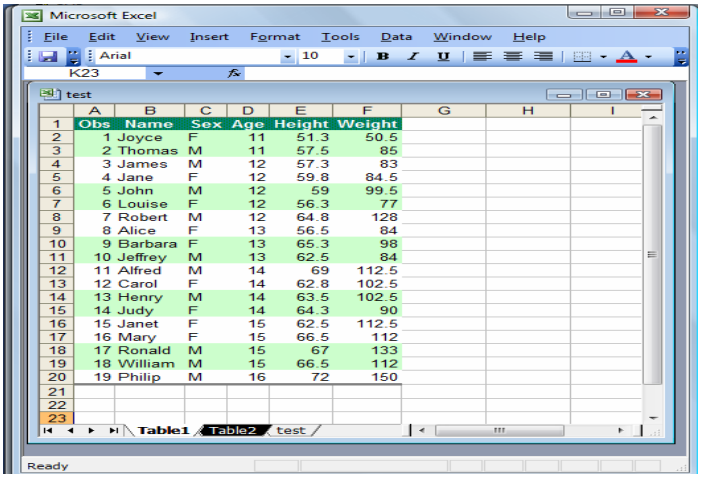
At times, you just need to add data to an existing Excel workbook. Using options with the TableEditor, you can insert tables from a Web page into an existing workbook. One of the benefits of using the TableEditor options is that you can insert the tables into file formats other than native Excel files. You can insert tables into any format supported by Excel. However, to save the file, you must use a format that supports multiple worksheets. For instance, if you create an XML file using the ExcelXP tagset, you can open the file and add a worksheet. Then, you can either leave the file in its current XML format or save the file as a native Excel formatted file (which reduces the size of the file). This capability to use different formats provides the full advantage in using other ODS destinations (such as the ExcelXP or the MSOffice2k destinations). You can use the following options to insert a new worksheet and move tables to that new sheet:

- **UPDATE_TARGET= option—**This option is used in conjunction with the SHEET_NAME= option and the EXCEL_TABLE_MOVE= option. The UPDATE_TARGET= option specifies the name of the workbook to which you want to add new worksheets; the SHEET_NAME= option specifies the name of the new sheet or sheets you want to insert; the EXCEL_TABLE_MOVE= option moves the tables to their specified sheets. Note that when you specify a file path in this option, you need to use double backward slashes instead of single forward slashes.

- **AUTO_ EXCEL=”Yes” option—**You can use this option to automatically export to Excel when the page is loaded. This does not require that you select the export button.

In the following example provides sample data that uses the ODS MSOffice2K destination to generate output that we read into Excel that is in HTML format. The example demonstrates inserting the first two tables from the Web page into this existing workbook that was generated with the MSOffice2K destination. You have the ability also to save this file in the format of your choice.

```
   /* Sample data provided to generate a default file */
   /* that is inserted into the workbook.             */

ods msoffice2k file="c:\temp\test.xls";

proc print data=sashelp.class;
run;

ods msoffice2k close;

ods tagsets.tableeditor file="filename.html"
                        options(update_target="c:\\temp\\test.xls"
                                sheet_name="Table1,Table2"
                                excel_table_move="1,2" auto_format=”list2”
                                excel_save_file=”c:\\temp\\temp1.xls”)style=sasweb2;
proc print data=sashelp.class;
run;

proc print data=sashelp.prdsale;
run;

ods tagsets.tableeditor close;
```




| Obs | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 1 | Joyce | F | 11 | 51.3 | 50.5 |
| 2 | Thomas | M | 11 | 57.5 | 85 |
| 3 | James | M | 12 | 57.3 | 83 |
| 4 | Jane | F | 12 | 59.8 | 84.5 |
| 5 | John | M | 12 | 59 | 99.5 |
| 6 | Louise | F | 12 | 56.3 | 77 |
| 7 | Robert | M | 12 | 64.8 | 128 |
| 8 | Alice | F | 13 | 56.5 | 84 |
| 9 | Barbara | F | 13 | 65.3 | 98 |
| 10 | Jeffrey | M | 13 | 62.5 | 84 |
| 11 | Alfred | M | 14 | 69 | 112.5 |
| 12 | Carol | F | 14 | 62.8 | 102.5 |
| 13 | Henry | M | 14 | 63.5 | 102.5 |
| 14 | Judy | F | 14 | 64.3 | 90 |
| 15 | Janet | F | 15 | 62.5 | 112.5 |
| 16 | Mary | F | 15 | 66.5 | 112 |
| 17 | Ronald | M | 15 | 67 | 133 |
| 18 | William | M | 15 | 66.5 | 112 |
| 19 | Philip | M | 16 | 72 | 150 |

**Display 32. Inserting Multiple Worksheets**

### Updating Worksheets

The TableEditor tagset also enables you to update existing worksheets with contents from Web-page tables. Use the following options to update worksheets:

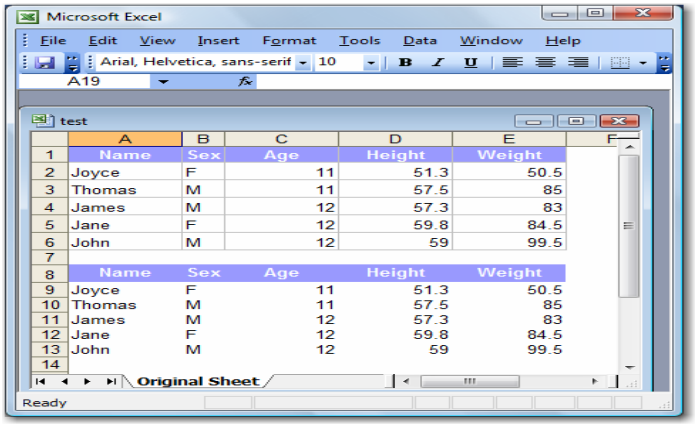
- **UPDATE_TARGET= option—**specifies the target file that you want to update.
- **UPDATE_SHEET**= **option—**names the specific sheet that you want to update.
- **UPDATE_ RANGE**= **option—**defines the range of columns and rows to update on the worksheet. The values for this option are specified in the order of column number and then row number.
- **EXCEL_TABLE_MOVE= option—**specifies which table you want to move from the Web page to the worksheet.
- **UPDATE_ FORMATTING= option—**enables you to keep the destinations style or use the one specified.

Web queries can also be generated to update worksheets. For a full list of available options, see the readme file that

is in the TableEditor download file. (The link to the download file is available in the Resources section of this paper.)

The following example uses the UPDATE_TARGET=, the UPDATE_SHEET=, the EXCEL_TABLE_MOVE=, and the UPDATE_RANGE= options. Note that this program does not use the UPDATE_FORMATTING= option, which only copied over the numbers.

```
    /* Create a sample workbook and worksheet to update using the
       ExcelXP destination                                        */

ods tagsets.excelxp file="c:\temp\test.xls" options(sheet_name="Original Sheet")
                    style=sasweb;

proc print data=sashelp.class(obs=5) noobs;
run;

ods tagsets.excelxp close;

ods tagsets.tableeditor file="temp.html" style=sasweb
                        options(update_target="c:\\temp\\test.xls"
                                update_sheet="Original sheet"
                                excel_table_move="1"
                                update_range="8,1");
proc print data=sashelp.class(obs=5) noobs;
run;

ods tagsets.tableeditor close;
```




| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Name | Sex | Age | Height | Weight |
| 2 | Joyce | F | 11 | 51.3 | 50.5 |
| 3 | Thomas | M | 11 | 57.5 | 85 |
| 4 | James | M | 12 | 57.3 | 83 |
| 5 | Jane | F | 12 | 59.8 | 84.5 |
| 6 | John | M | 12 | 59 | 99.5 |
| 7 | | | | | |
| 8 | Name | Sex | Age | Height | Weight |
| 9 | Joyce | F | 11 | 51.3 | 50.5 |
| 10 | Thomas | M | 11 | 57.5 | 85 |
| 11 | James | M | 12 | 57.3 | 83 |
| 12 | Jane | F | 12 | 59.8 | 84.5 |
| 13 | John | M | 12 | 59 | 99.5 |
| 14 | | | | | |

**Display 33. Updating a Worksheet**

**Pivot Tables**

Excel pivot tables are another powerful feature that you can use to help create the perfect grid. *Pivot tables* provide a dynamic way to look at summarized data in various views. When you use a pivot table, you have the ability to change your analysis view just by dragging the data.

Pivot-table output can also be filtered, expanded, collapsed, or simply rearranged, and you can quickly find answers to questions about the data by rotating or rearranging it. Pivot tables consist of any or all of the following objects: Page, Row, Column, and Data. In Excel, you can easily generate a pivot table by selecting **Data ►PivotTable** from the menu bar.

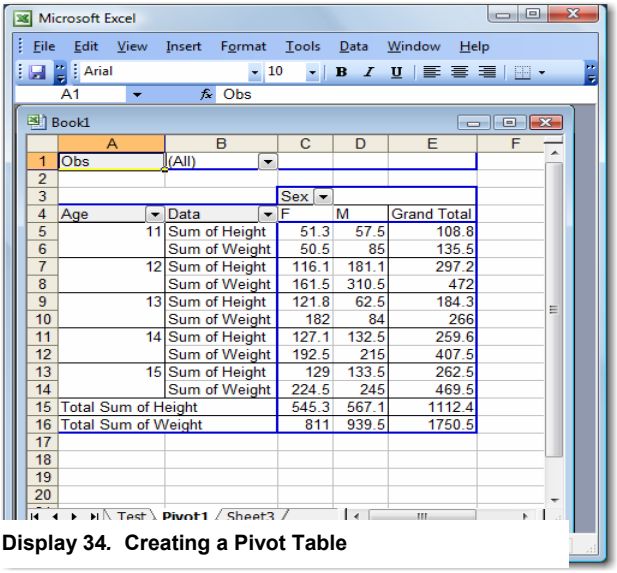
You can also generate pivot tables from the TableEditor using the following options:

- **PTSOURCE_RANGE= option—**specifies the range of the data to analyze.
- **PIVOTPAGE= option—**specifies a column page layout.
- **PIVOTROW**= **option—**specifies one or more columns of data for Pivot Row.
- **PIVOTCOL**= **option—**specifies one or more columns of data for Pivot Col.
- **PIVOTDATA**= **option—**specifies one or more columns of data for Pivot Data.
- **PTDEST_RANGE= option—**specifies a range on the existing worksheet to add the pivot table. By default it is added on a separate sheet when this option is not specified.

A pivot table will be generated on a new worksheet named Pivot1 unless you specify the PTDEST_RANGE= option. This option only changes the location of the worksheet in the current workbook to the range that you specify.

In the following example, column 3 (**Age**) is added to the column of the pivot table; columns 5 (**Height**) and 6 (**Weight**) are added to the data portion of the pivot table. The **Obs** column is also added to the page. A temporary HTML file is generated in the work area and Excel opened with the table automatically when the page is loaded.

```
ods tagsets.tableeditor file="%sysfunc(getoption(WORK))\temp.html"
     options(auto_excel="yes"
             excel_orientation="landscape"
             sheet_name="Test"
             ptsource_range="a1:f19"
             pivotrow="3"
             pivotcol="4"
             pivotdata="5,6"
             pivotpage="1" );


proc print data=sashelp.class;
   var name age sex height weight;
   title "Pivot tables";
run;
ods tagsets.tableeditor close;
```




| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | Obs | (All) | | | | |
| 2 | | | | | | |
| 3 | | | Sex | | | |
| 4 | Age | Data | F | M | Grand Total | |
| 5 | 11 | Sum of Height | 51.3 | 57.5 | 108.8 | |
| 6 | | Sum of Weight | 50.5 | 85 | 135.5 | |
| 7 | 12 | Sum of Height | 116.1 | 181.1 | 297.2 | |
| 8 | | Sum of Weight | 161.5 | 310.5 | 472 | |
| 9 | 13 | Sum of Height | 121.8 | 62.5 | 184.3 | |
| 10 | | Sum of Weight | 182 | 84 | 266 | |
| 11 | 14 | Sum of Height | 127.1 | 132.5 | 259.6 | |
| 12 | | Sum of Weight | 192.5 | 215 | 407.5 | |
| 13 | 15 | Sum of Height | 129 | 133.5 | 262.5 | |
| 14 | | Sum of Weight | 224.5 | 245 | 469.5 | |
| 15 | Total Sum of Height | | 545.3 | 567.1 | 1112.4 | |
| 16 | Total Sum of Weight | | 811 | 939.5 | 1750.5 | |
| 17 | | | | | | |
| 18 | | | | | | |
| 19 | | | | | | |
| 20 | | | | | | |



**Display 34. Creating a Pivot Table**

**EXCEL Graphs**

The ability to export and generate graphs is a must for any data grid. You can generate graphs in Excel using options of the TableEditor, which gives you access to all of the graphs that can be generated by Excel. Excel can generate a large number of graph types, so you should be able to find the type you want. The following is a partial list of graph types that are available in Excel:

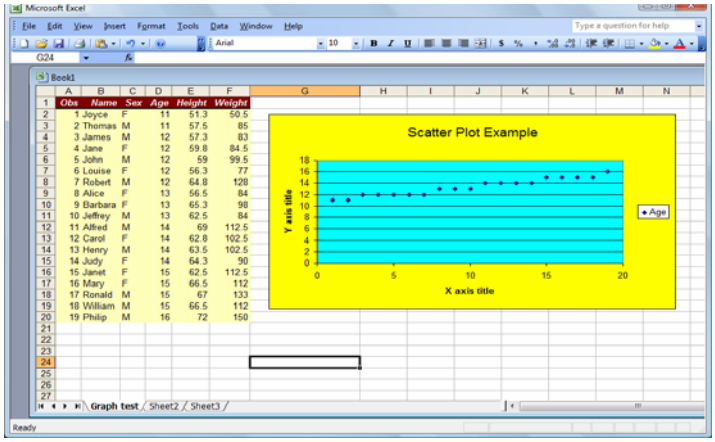
- Bar
- Stacked Bar
- Histogram
- Stacked histogram
- Pie chart
- 3d pie chart
- Cylinder BarClustered
- Cylinder BarStacked
- 3d area
- 3d area stacked
- Dubble 3deffect
- Column clustered
- 3dcolumn
- Cone col
- 3d pie exploded
- Cone col clustered
- Doughnut
- Pyramid bar clustered
- Xyscatter
- Xyscatterlines
- LIne

You generate graphs with the following required options:

- **CHART_TYPE= option—**specifies the type of Excel graph that you want to use
- **CHART_SOURCE= option—**is required in order to select the data range that you want to chart

Currently, only a few options are available for graphics, but additional functionality in this area is planned in the future.

```
ods tagsets.tableeditor file="temp.html"
 options(sheet_name="Graph test"
         chart_type="xyscatter"
         chart_title="Scatter Plot Example"
         chart_title_size="14"
         chart_xaxes_title="X axis title"
         chart_yaxes_title="Y axis title"
         chart_area_color="6"
         chart_plotarea_color="8"
         chart_source="d:d"
         excel_table_move="1"
         auto_format="color2");

proc print data=sashelp.class;
run;

ods tagsets.tableeditor close;
```




**Display 35. Generating a Graph of Your Data**

**EXCEL Macros**

One of Excel's most important features is the ability to create and use macros to modify the workbook or worksheet. Macros are generated using Visual Basic for Applications (VBA). Just as you can modify objects with the TableEditor, you can also modify them using Excel macros. You can use a macro for anything from adding a graph for the data and setting up common options to adding a pivot table or chart. If the options provided for the TableEditor do not offer the flexibility you need, you can use the TableEditor to execute the Excel macro of your choice. The macro is actually executed when a table is exported to Excel.

Using the MACRO= option, you can point to a file that contains the macro you want to execute when the data is exported. To keep from displaying the workbook or worksheet when the data is exported, you will want to hide the window or sheet.

The following example points to a macro file that creates a pivot table from the data.

```
ods tagsets.tableeditor file="temp.html"
                        options(macro="'c:\\temp\\samples\\filename.xls'!Pivot");

proc print data=sashelp.class;
   sum height weight;
run;

ods tagsets.tableeditor close;
```

**Number Formats**

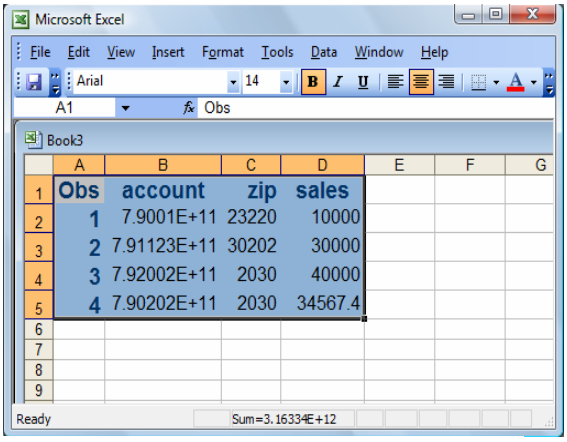
When output is exported to Excel, the General format is used by default, just as it is used for other ODS destinations. The General format is also used when you type numbers in an Excel cell if no other format is specified. You need to be careful when you export to Excel because when the General format is used, numbers containing leading zeros are removed. Other formatting issues, such as numbers being converted to scientific notation, can also occur.

To prevent numbers greater than 11 from being formatted in scientific notation, you can format a column as type String by using the TAGATTR=”x.str” attribute.

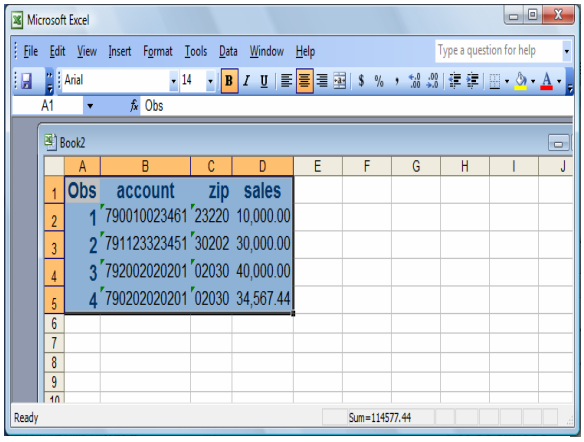
Specifying an Excel format will guarantee that your output will display exactly as you have requested. You can add Excel number formats by using the Microsoft specific CSS style property **`mso-number-format`**. You add this property with the HTMLSTYLE= attribute.

The following example uses the String attribute for both the variables ACCOUNT and ZIP so that their values display properly in Excel. The variable SALES uses an Excel custom format that causes the results to display to two decimal places.

```
data one;
  input account zip sales;
cards;
790010023461 23220 10000
791123323451 30202 30000
792002020201 02030 40000
790202020201 02030 34567.44
;
run;

ods tagsets.tableeditor file="temp6.html" options(file_format="xls");

proc print data=one;
   var account zip / style(data)={tagattr="x:str nowrap"};
   var sales / style(data)={htmlstyle="mso-number-format:'#,###.00'"};
   format zip z5.;
run;

ods tagsets.tableeditor close;
```




**Display 36. Output Before Applying Formatting**




**Display 37. Output After Applying Formatting**

**EXPORTING DATA TO POWERPOINT**

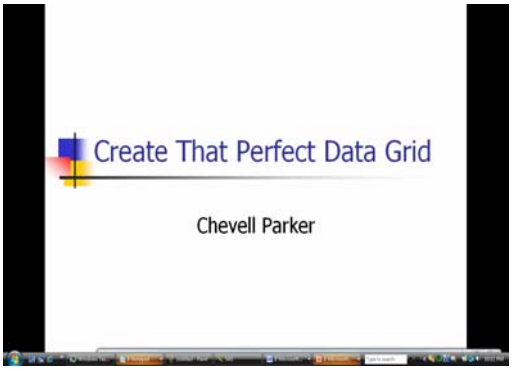
The TableEditor can also export Web pages into PowerPoint, just as it does for Excel. You can use the

following options to export data to PowerPoint:

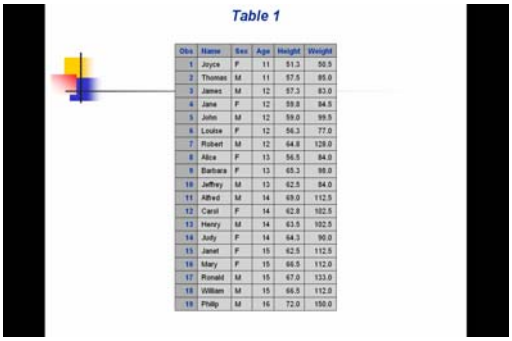
- **POWERPOINT_SLIDE= option—**specifies which Web pages, separated by commas, that you want to import into separate slides. This requires each Web page be a separate file.
- **POWERPOINT_TEMPLATE= option—**specifies a template to define the overall appearance of the PowerPoint slides.
- **POWERPOINT_MASTER= option—**enables you to add a title and subtitle. In the option, you must separate the two text strings with an octothorp (#) symbol.
- **POWERPOINT_SAVEAS**= **option—**enables you to save your file as a native PowerPoint (.ppt) file.
- **POWERPOINT_RUN= option—**automatically runs your PowerPoint slides.

The following example uses all of these PowerPoint options. For this example, the HTML files were generated using the MSOffice2K destination.

```
ods msoffice2k path="c:\temp" file="power.html" newfile=output;

proc print data=sashelp.class;
   title "Table 1";
run;

proc print data=sashelp.class;
   title "Table 2";
run;

ods msoffice2k close;

ods tagsets.tableeditor file="temp.html"
    options(powerpoint_master="Create That Perfect Data Grid#Chevell Parker"
            powerpoint_template="c:\\program files\\microsoft
                                 office\\templates\\presentation
                                 designs\\blends.pot"
            powerpoint_slides="c:\\temp\\power.html,
                               c:\\temp\\power1.html"
            powerpoint_saveas="c:\\temp\\power.ppt"
            powerpoint_run="yes");

proc print data=sashelp.orsales(obs=5);
 run;

ods tagsets.tableeditor close;
```




**Display 38. PowerPoint Master Page**




**Display 39. First PowerPoint Slide with the Table Data**

## ADDING EXAMPLES WITH STORED PROCESSES

Any of the examples within the document can be added while creating a stored process. All of the above examples wrote to physical files, however It can be used with dynamic files as well. To use any of the options with a stored

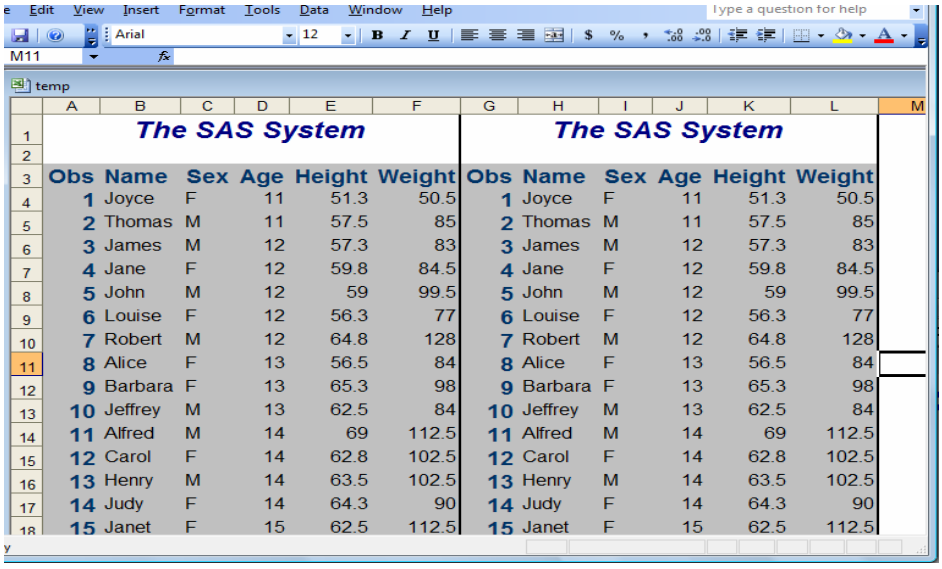
process, the stored process would need to be registered as streaming if you would like this to be dynamic and returned to the browser. If you are using the %STP macros, we will need to override the tagset name which will use the HTML4 tagset by default with the _ODSDEST= macro variable. We will also need to add the options via the _ODSOPTIONS macro variable.

```
%let _ODSDEST=tagsets.tableeditor;
%let _ODSOPTIONS=options(frozen_headers="yes" banner_color_even="teal");
%let _ODSSTYLE=SASWEB;

%stpbegin;

Proc print data=sashelp.orsales;
Run;

%stpend;
```

**EXPORTING PANELED OUTPUT**

We can create paneled output in Excel by using the combination of the PANELCOLS= options and the EMBEDDED_TABLES= option. Any number of tables can be paneled in Excel. Using the below example, only two tables are paneled, however, you can panel as many as you like.

```
ods tagsets.tableeditor path="c:\temp"  file="temp.xls"
options(panelcols="2"
        embedded_tables="yes"
        excel_zoom="80"
        excel_frozen_headers="3");

proc print data=sashelp.class;
run;

proc print data=sashelp.class;
run;

ods tagsets.tableeditor close;
```



The SAS System

| Obs | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 1 | Joyce | F | 11 | 51.3 | 50.5 |
| 2 | Thomas | M | 11 | 57.5 | 85 |
| 3 | James | M | 12 | 57.3 | 83 |
| 4 | Jane | F | 12 | 59.8 | 84.5 |
| 5 | John | M | 12 | 59 | 99.5 |
| 6 | Louise | F | 12 | 56.3 | 77 |
| 7 | Robert | M | 12 | 64.8 | 128 |
| 8 | Alice | F | 13 | 56.5 | 84 |
| 9 | Barbara | F | 13 | 65.3 | 98 |
| 10 | Jeffrey | M | 13 | 62.5 | 84 |
| 11 | Alfred | M | 14 | 69 | 112.5 |
| 12 | Carol | F | 14 | 62.8 | 102.5 |
| 13 | Henry | M | 14 | 63.5 | 102.5 |
| 14 | Judy | F | 14 | 64.3 | 90 |
| 15 | Janet | F | 15 | 62.5 | 112.5 |

The SAS System

| Obs | Name | Sex | Age | Height | Weight |
|---|---|---|---|---|---|
| 1 | Joyce | F | 11 | 51.3 | 50.5 |
| 2 | Thomas | M | 11 | 57.5 | 85 |
| 3 | James | M | 12 | 57.3 | 83 |
| 4 | Jane | F | 12 | 59.8 | 84.5 |
| 5 | John | M | 12 | 59 | 99.5 |
| 6 | Louise | F | 12 | 56.3 | 77 |
| 7 | Robert | M | 12 | 64.8 | 128 |
| 8 | Alice | F | 13 | 56.5 | 84 |
| 9 | Barbara | F | 13 | 65.3 | 98 |
| 10 | Jeffrey | M | 13 | 62.5 | 84 |
| 11 | Alfred | M | 14 | 69 | 112.5 |
| 12 | Carol | F | 14 | 62.8 | 102.5 |
| 13 | Henry | M | 14 | 63.5 | 102.5 |
| 14 | Judy | F | 14 | 64.3 | 90 |
| 15 | Janet | F | 15 | 62.5 | 112.5 |

**DISPLAY 40. PANELED EXCEL OUTPUT**

## CONCLUSION

This document demonstrates strategies that you can use to create an effective data grid using the SAS Output Delivery System. These strategies include modifying styles, highlighting data, and exporting data. Hopefully, you will be able to implement some of these features in your reports.

## RESOURCES

Mead & Co. Ltd. (MeadCo). 1998-2003. "ScriptX printing: technical manual". Available at www.meadroid.com/scriptx/docs/printdoc.htm.

Microsoft Corporation. 2008. "The Office XP Automation Help file is available for download". Available at support.microsoft.com/kb/302460.

Parker, Chevell. 2006. "Now—That's Your Style!!!!!" *Proceedings of the Thirty-first Annual SAS Users Group International Conference*. Cary, NC: SAS Institute Inc. Available at www2.sas.com/proceedings/sugi31/091-31.pdf.

Parker, Chevell. 2008. "Base SAS: Creating a Data Grid Like VB.NET". Available at support.sas.com/rnd/base/ods/odsmarkup/tableeditor/index.html. (This is the TableEditor download.)

SAS Institute Inc. 2008. "Base SAS: ODS Markup Resources". Available at support.sas.com/rnd/base/ods/odsmarkup/.

## ACKNOWLEDGEMENTS

I would like to thank Bari Lawhorn, Cynthia Zender, Susan Berry and Maggie Marcum for their contribution toward this paper. I would also like to thank Eric Gebhart and the rest of the ODS development team for writing great software that I used to create the TableEditor.

## CONTACT INFORMATION

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

Chevell Parker
SAS Institute Inc.
SAS Circle Dr.
Cary, NC 27513
Chevell.Parker@sas.com