Paper SAS710-2016

More Than Just a Pretty Face: Using the SAS® Output Delivery System to Create Microsoft Excel Worksheets That Answer Those Difficult Questions

Chevell Parker, SAS Institute Inc.

ABSTRACT

Microsoft Excel worksheets enable you to explore data that answers the difficult questions that you face daily in your work. When you combine the SAS® Output Deliver System (ODS) with the capabilities of Excel, you have a powerful toolset you can use to manipulate data in various ways, including highlighting data, using formulas to answer questions, and adding a pivot table or graph. In addition, ODS and Excel give you many methods for enhancing the appearance of your tables and graphs.

This paper, written for the beginning analyst to the most advanced programmer, illustrates first how to manipulate styles and presentation elements in your worksheets by controlling text wrapping, highlighting and exploring data, and specifying Excel templates for data. Then, the paper explains how to use the TableEditor tagset and other tools to build and manipulate both basic and complex pivot tables that can help you answer all of the questions about your data. You will also learn techniques for sorting, filtering, and summarizing pivot-table data.

INTRODUCTION

This paper demonstrates how to generate useful and effective Microsoft Excel worksheets from the SAS Output Delivery System. The worksheets are created using a variety of techniques and ODS destinations. First, the paper illustrates how to "add a pretty face" to your worksheets (that is, to visually enhance the worksheets) by using the ODS Excel destination. Using this method, you can create attractive, yet highly functional worksheets that answer the difficult questions you encounter. Then, the paper explores how to create and manipulate basic and advanced pivot tables using the TableEditor tagset. Finally, the discussion explains how to build a GUI interface that enables you to generate pivot tables.

CREATING PRETTY AND FUNCTIONAL REPORTS USING THE ODS EXCEL DESTINATION

Everyone loves a pretty face! Or, in the case of Excel, everyone likes to see visually appealing worksheets. But just because a worksheet is visually attractive does not mean that it can't still be functional, as you will discover in this paper.

You can create visually appealing worksheets in a number of ways using ODS. For example, you can enhance your presentation by using text wrapping effectively. You can also use the ODSTABLE procedure to modify styles. Modifying styles is one of the easiest methods for enhancing the appearance and readability of a worksheet. By modifying the style (through fonts, colors, and borders), you also add visual focus to parts of your data. You can also use dynamic styles (that is, with dynamic components) to help visually highlight the information that can answer your business questions. All of these methods are available by using the ODS Excel destination. This section covers the following topics:

- controlling text wrapping
- using the ODSTABLE procedure to add styles
- using the ODS Excel destination and cascading style sheets to generate dynamic styles

CONTROLLING TEXT WRAPPING WITHIN A MICROSOFT EXCEL WORKSHEET

The ODS Excel destination is a *measured destination* that uses an algorithm to determine when text should wrap within a cell. This wrapping algorithm creates a best fit for columns so that they are not overly wide. When text does wrap within a cell, a carriage-return/line-feed character (CRLF) is added where the line break occurs. However, when a CRLF character is added to the cell in which text wraps, negative consequences such as the following can occur:

- Excel formulas or functions might become invalid when the CRLF character is added to the cell.
- Text cannot be extended fully within the cell without first deleting the CRLF character.
- Some applications that read worksheets might have various types of problems when they encounter the CRLF character.

One method for preventing the addition of the CRLF character to a cell is to use the WIDTH= attribute with a value large enough so that the text does not wrap. However, the most dynamic method is to use the FLOW= tagset option.

The fourth maintenance release for SAS® 9.4 (TS1M4) introduces the ODS Excel destination's FLOW= suboption. When this option is specified, the Excel destination does not insert newline characters to force the text to wrap in the part of the output that is specified as an argument in the option. The FLOW= option also turns on the **Wrap Text** feature in Excel so that Excel will wrap the text to the column width. Table 1 shows values for ODS Excel suboption FLOW= and describes the part of the output that is affected by the option.

Value	Area Effected
TABLES	Enables flow of column headings, row headings, and data cells.
HEADERS	Enables flow for headings only.
ROWHEADERS	Enables flow for row headings.
DATA	Enables flow for data cells only.
TEXT	Enables flow for the ODSTEXT procedure, the ODSLIST procedure, the ODS TEXT= option, the procedure title, and the Report Writing Interface.
RANGE	Enables flow for a cell or range of cells (for example, A:E).

Table 1. Values and Descriptions for the Excel Destination's FLOW= Suboption

You can also control text wrapping using the TITLE_FOOTNOTE_NOBREAK= ODS suboption. (From this point on, the suboptions are just called options.) The TITLE_FOOTNOTE_NOBREAK= option controls how text wraps for titles, footnotes, and BY lines within your worksheet. Another option, ABSOLUTE_COLUMN_WIDTH=, also has new functionality in SAS 9.4 TS1M4. This option, which enables you to control the width of one or more columns, now allows you to add a unit of measure such as points (pt), millimeters (mm), inches (in), and pixels (px).

The following example demonstrates the use of the ODS Excel destination along with the TITLE_FOONTNOTE_BREAK=, ABSOLUTE_COLUMN_WIDTH=, and FLOW= options.

Example 1.

```
ods excel file="c:\temp1.xlsx" options(embedded_titles="Yes"
                                        title_footnote_nobreak="yes"
                                        flow="header,data"
                                        absolute_column_width="75px,50px,
                                        70px,50px,100px,125px,300px");


ods text="Confidential Report";
```

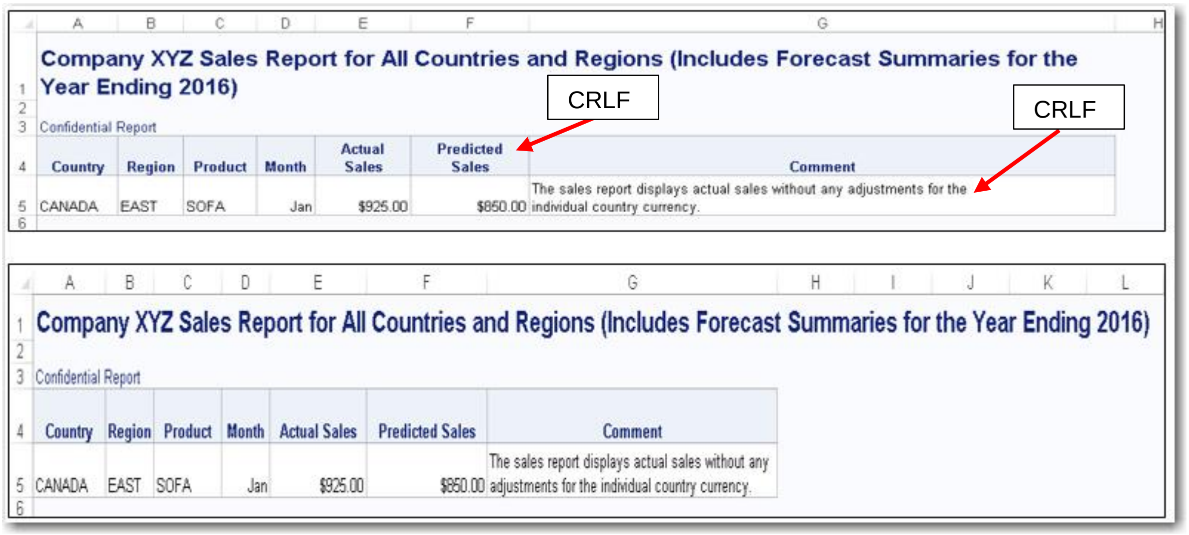
(code continued)

```
      proc report data=sashelp.prdsale(obs=1);
         title j=l h=16pt "Company XYZ Sales Report for All Countries and
                           Regions (Includes Forecast Summaries for the Year
                           Ending 2016)";
         column Country Region Product Month Actual Predict Comment;
         define comment / computed;
         compute comment / length=100;
            comment='The sales report displays actual sales without any
                     adjustments for the individual country currency.';
         endcomp;
      run;

   ods excel close;
```

In This Example

- The FLOW= option specifies the arguments HEADER and DATA. Respectively, these arguments apply to only the column headers and the data cells in the table.
- The TITLE_FOOTNOTE_NOBREAK= option prevents the title from wrapping.
- The ABSOLUTE_COLUMN_WIDTH= option supplies a width for each column. The width is specified in pixels (px) as the measurement in this example.

Output



Company XYZ Sales Report for All Countries and Regions (Includes Forecast Summaries for the Year Ending 2016)

Confidential Report

Country	Region	Product	Month	Actual Sales	Predicted Sales	Comment
CANADA	EAST	SOFA	Jan	$925.00	$850.00	The sales report displays actual sales without any adjustments for the individual country currency.

Company XYZ Sales Report for All Countries and Regions (Includes Forecast Summaries for the Year Ending 2016)

Confidential Report

Country	Region	Product	Month	Actual Sales	Predicted Sales	Comment
CANADA	EAST	SOFA	Jan	$925.00	$850.00	The sales report displays actual sales without any adjustments for the individual country currency.

Output 1. Report That Shows the Results of the FLOW=, TITLE_FOOTNOTE_NOBREAK=, and ABSOLUTE_COLUMN_WIDTH Options

USING THE ODSTABLE PROCEDURE TO ADD STYLES

SAS 9.4 introduced three new ODS procedures: ODSTABLE, ODSTEXT, and ODSLIST. The ODSTABLE procedure is an easy method of using table templates. Some of the important features in PROC ODSTABLE include the ability to create custom table headers, dynamic styles, calculated columns, concatenated columns, substituted text, and formulas.

The CELLSTYLE statement within PROC ODSTABLE enables you to highlight or to add style to individual cells or rows based on a condition. Such highlighting might entail anything from adding background or foreground colors to a cell or row in order to identify outliers to adding a computed column that identifies the percentage of returns. With PROC ODSTABLE, you can also generate a source data sheet that enables you to create pivot tables because PROC ODSTABLE does not allow summaries. In addition, the STORE= option in PROC ODSTABLE enables you to save your style information for future use.

Example 2 uses PROC ODSTEXT, PROC ODSTABLE, the CELLSTYLE statement, the START_AT= option (in the ODS EXCEL option), and the _ROW_ automatic variable to add styles and formulas to output.

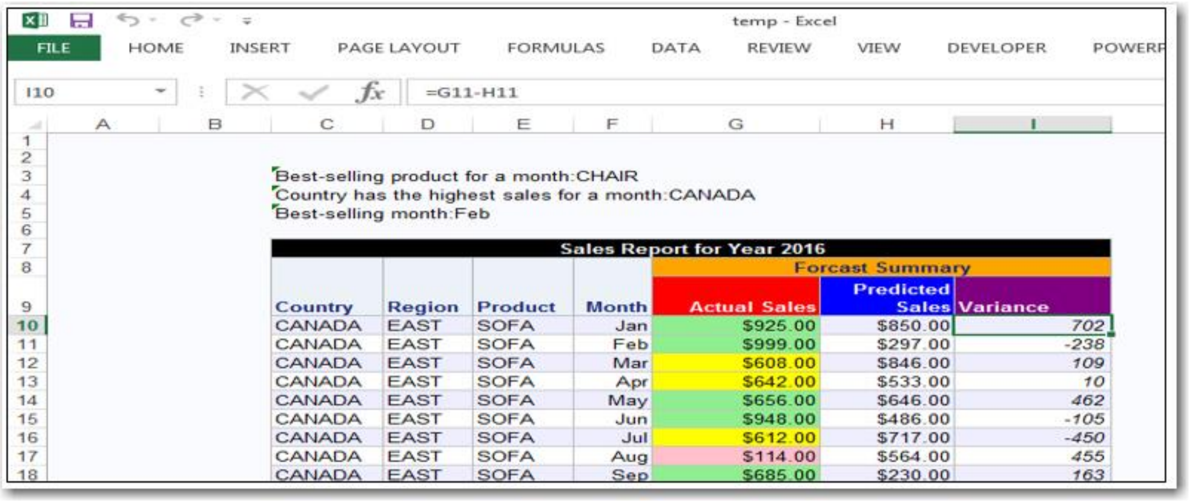
Example 2.

```
ods excel file="c:\temp.xlsx" options(sheet_interval="none"
                                      start_at="3,3");
proc odstext;
   p '=concatenate("Best-selling product for a month:
                 ",index(E13:E100,match(max($G$13:$G100),$G$13:G$100,
                 0)))';
   p '=concatenate("Country has the highest sales for a month:
                 ",index(C13:C100,match(max($G$13:$G100),$G$13:G$100,0)))';
   p '=concatenate("Best-selling month:
                 ",index(F13:F100,match(max($G$13:$G100),$G$13:G$100,0)))';
run;

proc odstable data=sashelp.prdsale(obs=100);
   column country region product month actual predict variance;
   cellstyle mod(_row_,2) as data{background=cxeeeeff};
   define header h1;
      start=country;
      text "Sales Report for Year 2016";
      style={background=black foreground=white};
   end;
   define header h2;
      start=Actual;text " Forecast Summary";
      style={background=orange};
   end;
   define column actual;
   define header h3;
      style={background=red foreground=white};
   end;
   header=h3;
   cellstyle  _val_ > 650 as data {background=lightgreen},
             450 < _val_ < 650 as data {background=yellow},
             0 < _val_ < 450 as data {background=pink},
             1 as data;
   end;
   define column predict;
   define header h4;
      style={background=blue foreground=white};
   end;
   header=h4;
   end;
   define variance;
   define header h5;
      style={background=purple foreground=white};
      text "Variance";
   end;
   header=h5;
   compute as cat("=G",_row_ +10,"-",cat("H",_row_ +10));
      style={just=right fontstyle=italic  };
   end;
run;

ods excel close;
```

In This Example

- PROC ODSTEXT procedure uses P statements to add Excel CONCATENATE functions. The CONCATENATE functions contain nested INDEX, MAX, and MATCH functions that perform a table lookup to determine the highest-selling product along with its associated country and month.
- The CELLSTYLE statement uses the MOD() function with the _ROW_ automatic variable to apply a background color to every other row for readability purposes.
- The DEFINE HEADER statements create custom headers (for example, H1 and H2 in the code) that span all of the columns. The START= attribute in the ODS EXCEL statement indicates that these headers should span three columns.
- The CELLSTYLE statement adds background color, based on values, for the ACTUAL column in the output.
- The VARIANCE column values are computed based on the difference between the ACTUAL and PREDICT values. The values are calculated with a formula that is derived from the column name and the automatic variable _ROW_.

Output



=G11-H11

Best-selling product for a month:CHAIR
Country has the highest sales for a month:CANADA
Best-selling month:Feb

Sales Report for Year 2016						
				Forcast Summary		
Country	Region	Product	Month	Actual Sales	Predicted Sales	Variance
CANADA	EAST	SOFA	Jan	$925.00	$850.00	702
CANADA	EAST	SOFA	Feb	$999.00	$297.00	-238
CANADA	EAST	SOFA	Mar	$608.00	$846.00	109
CANADA	EAST	SOFA	Apr	$642.00	$533.00	10
CANADA	EAST	SOFA	May	$656.00	$646.00	462
CANADA	EAST	SOFA	Jun	$948.00	$486.00	-105
CANADA	EAST	SOFA	Jul	$612.00	$717.00	-450
CANADA	EAST	SOFA	Aug	$114.00	$564.00	455
CANADA	EAST	SOFA	Sep	$685.00	$230.00	163

Output 2. Adding Styles and Formulas Using PROC ODSTEXT and PROC ODSTABLE

USING THE ODS EXCEL DESTINATION AND CASCADING STYLE SHEETS TO GENERATE DYNAMIC STYLES

You can also enhance your worksheet by using global styles. To generate global styles, you use both PROC TEMPLATE and cascading style sheets. Cascading style sheets (CSS) enable more elaborate formatting that cannot be done with PROC TEMPLATE. There are diverse methods for applying styles using a cascading style sheet. For example, some available methods include pseudo-class selectors for identifying which part of the output should be formatted, based on the position of the parent element. The use of attributes with the selectors is a powerful method for adding style to your output. In addition, you have a lot of flexibility when you specify the plain-class selector.

When you use a cascading style sheet, it is easy to create data-driven CSS styles. With these styles, you can identify trends in data by using SAS macros, macro variables, or SAS formats. For example, the next code example uses a SAS format to specify a range of values, and it includes a hook in the CSS file to use the format in a style property. The hook is shown here. In Example 3, the CSSSTYLE= option imports the CSS file.

Example 3.

```
proc format;
   value back 0-<2000='#e6f5f'      2000 -< 2500='#ccebff'
              2500-<3000='#b3e0ff'  3000-<4000='#99d6ff'
              4000-high='#80ccff';
run;

ods excel file="c:\temp3.xlsx" cssstyle="c:\test\background.css";
options(sheet_interval="none");
goptions iback="c:\saslogo.jpg" imagestyle=fit vsize=1in hsize=2in;

   proc gslide;
   run;
   quit;

   proc tabulate data=sashelp.prdsale format=dollar.;
      class country product month;
      var actual;
      table country*product all='Total',month=' '*actual=" ";
      keylabel sum=" ";
   run;
   proc odstext;
      p '=if(sum(C18:N18) >700000,"The Year was Profitable","The Year was
                                 Unprofitable")' / style={font_weight=bold};
   run;

ods excel close;
```

In This Example

- The CSS file that is used with the CSSSTYLE= option consists of the following two style properties.

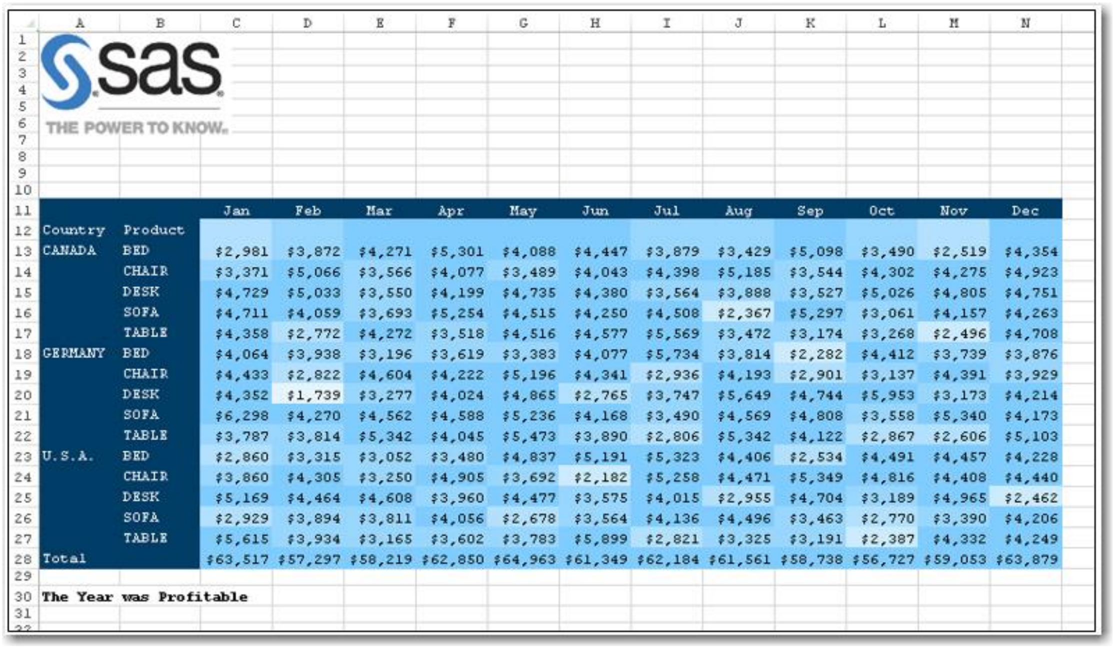
```
background.css

   .header, .rowheader {background-color:#003d66;color:white}

   .data {background-color:format('back.')};
```

  The first style property uses the `.header` and `.rowheader` class selectors, and it modifies the background and foreground colors.

  The second style property uses the `.data` class selector, and it dynamically assigns the background color to the style based on the `back.` format. This format is applied in to the actual sales of over-all months in the TABULATE procedure.

- The nested =IF formula displays text that specifies whether the year was profitable or not, based on a mark that is selected against a range of cells that contain totals.

Output



Country	Product	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
CANADA	BED	$2,981	$3,872	$4,271	$5,301	$4,088	$4,447	$3,879	$3,429	$5,098	$3,490	$2,519	$4,354
	CHAIR	$3,371	$5,066	$3,566	$4,077	$3,489	$4,043	$4,398	$5,185	$3,544	$4,302	$4,275	$4,923
	DESK	$4,729	$5,033	$3,550	$4,199	$4,735	$4,380	$3,564	$3,888	$3,527	$5,026	$4,805	$4,751
	SOFA	$4,711	$4,059	$3,693	$5,254	$4,515	$4,250	$4,508	$2,367	$5,297	$3,061	$4,157	$4,263
	TABLE	$4,358	$2,772	$4,272	$3,518	$4,516	$4,577	$5,569	$3,472	$3,174	$3,268	$2,496	$4,708
GERMANY	BED	$4,064	$3,938	$3,196	$3,619	$3,383	$4,077	$5,734	$3,814	$2,282	$4,412	$3,739	$3,876
	CHAIR	$4,433	$2,822	$4,604	$4,222	$5,196	$4,341	$2,936	$4,193	$2,901	$3,137	$4,391	$3,929
	DESK	$4,352	$1,739	$3,277	$4,024	$4,865	$2,765	$3,747	$5,649	$4,744	$5,953	$3,173	$4,214
	SOFA	$6,298	$4,270	$4,562	$4,588	$5,236	$4,168	$3,490	$4,569	$4,808	$3,558	$5,340	$4,173
	TABLE	$3,787	$3,814	$5,342	$4,045	$5,473	$3,890	$2,806	$5,342	$4,122	$2,867	$2,606	$5,103
U.S.A.	BED	$2,860	$3,315	$3,052	$3,480	$4,837	$5,191	$5,323	$4,406	$2,534	$4,491	$4,457	$4,228
	CHAIR	$3,860	$4,305	$3,250	$4,905	$3,692	$2,182	$5,258	$4,471	$5,349	$4,816	$4,408	$4,440
	DESK	$5,169	$4,464	$4,608	$3,960	$4,477	$3,575	$4,015	$2,955	$4,704	$3,189	$4,965	$2,462
	SOFA	$2,929	$3,894	$3,811	$4,056	$2,678	$3,564	$4,136	$4,496	$3,463	$2,770	$3,390	$4,206
	TABLE	$5,615	$3,934	$3,165	$3,602	$3,783	$5,899	$2,821	$3,325	$3,191	$2,387	$4,332	$4,249
Total		$63,517	$57,297	$58,219	$62,850	$64,963	$61,349	$62,184	$61,561	$58,738	$56,727	$59,053	$63,879

The Year was Profitable

Output 3. Generating Dynamic Styles Using a SAS Format

CREATING PIVOT-TABLE REPORTS

One of the most important features in Excel, *pivot tables,* enables you to extract the significance of large, detailed data so that you can make better business decisions. Excel contains various methods for you to create a pivot table. However, using the TableEditor tagset, you can automate those methods through a SAS program and generate the pivot table with the Excel, ExcelXP, MSOffice2K, and CSV destinations. (The TableEditor tagset is available for downloading from **`support.sas.com/rnd/base/ods/odsmarkup/tableeditor/tableeditor.tpl`**.

You can also include the TableEditor tagset in your code by submitting the following code:

```
filename output url "http://support.sas.com/rnd/base/ods/odsmarkup/
                     tableeditor/tableeditor.tpl";

%include output;
```

This section covers the following topics:

- generating pivot-table reports
- adding fields to pivot-table drop zones
- selecting ranges for the input and output of a pivot table
- modifying the report layout, grand totals, and styles
- selecting statistics and basic options for a pivot table
- adding a visual enhancement to the output by using conditional formatting
- creating calculated columns from a pivot-table column
- sorting and filtering pivot-table reports
- adding dashboards

GENERATING PIVOT-TABLE REPORTS

You can use either of three methods with the TableEditor tagset to generate a pivot table:

- Create a web page with detailed data that is used as the source data for pivot table when you include any pivot-table options in your code. This method places an interactive button on the web browser, as shown in the following code example. This button opens the pivot table. The following code creates a web page in the temporary `Work` location. This code creates a button named **Pivot Table** on the web page. This method requires the browser to be Microsoft Internet Explorer.

```
ods tagsets.tableeditor file="%sysfunc(getoption(WORK))\test.html"
    options(button_text="Pivot Table" pivotpage="year"
            pivotrow="Country" pivotdata="actual");
   proc print data=sashelp.prdsale;
   run;
ods tagsets.tableeditor close;
```

- Update an existing workbook with a pivot table from the worksheet's source data by selecting the **Pivot Table** button on the web page. Then, modify the previous code example by add the UPDATE_TARGET= option. This option enables you to add a pivot table to an existing workbook. This method also requires that you use Internet Explorer.

```
ods tagsets.tableeditor file="c:\temp\PivotTable.html"
       /* Remember to escape the backslashes in the path */
    options(update_target="c:\\existingfile\\temp.xlsx"
            pivotrow="region" pivotdata="actual");

   data _null_;
      file print;
      put _all_;
   run;

ods tagsets.tableeditor close;
```

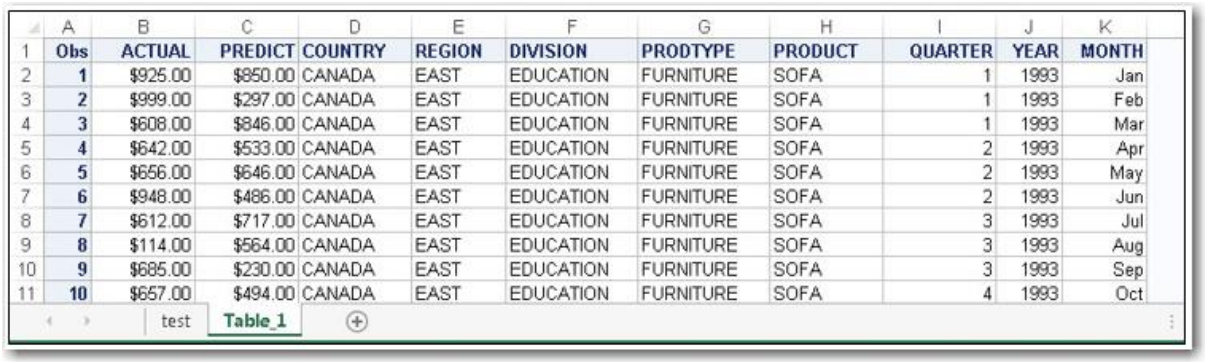
- Create a JavaScript file by using the option OUTPUT_TYPE= option. Like in the previous example, this method updates a previously created Excel file. Unlike the two previous examples, this method does not require human intervention to select the **Pivot Table** button, and this method is not limited with regard to which browser you use. After the script file is generated, you can manually execute it by double-clicking the JS (JavaScript) file or by using the X statement in your code, The X statement passes the JavaScript to the operating system for execution. The rest of the examples in this section use this method. The following example illustrates execution of the code only.

```
ods noxwait noxsync;
ods noresults;
ods excel file="c:\temp\temp.xlsx" options(sheet_name="Table_1");

proc print data=sashelp.class;
run;

ods excel close;

   ods tagsets.tableeditor file="c:\test.js"
       options(update_target='c:\\temp\\temp.xlsx'
               output_type='script'             sheet_name="Table_1"
               pivot_sheet_name="test"          pivotrow="Country"
               pivotdata="actual");

   data _null_;
      file print;
      put "Create Pivot Table";
   run;

ods tagsets.tableeditor close;
x 'c:\test.js';
```

In This Example

- The NOXWAIT and NOXSYNC options are Microsoft Windows system options that enable the script to execute without waiting for a command. The ODS NORESULTS statement prevents the browser from opening when the script file is being generated.
- The ODS Excel destination creates a worksheet (which is to be updated with the tagset) in the temporary Work library location to be updated with the tagset.
- The UPDATE_TARGET= tagset option adds the path to the source data's location.
- The OUTPUT_TYPE_= tagset option specifies that the output will be a script file.
- The SHEET_NAME= tagset option specifies the worksheet name for the source data. The UPDATE_TARGET= option specifies the name and path of the workbook.
- The PIVOT_SHEET_NAME= tagset option names the pivot table. If this option is not used, the pivot table that is created is given a default name.
- The DATA _NULL_ statement forces the tagset to run.
- The X statement executes the JavaScript file and creates the pivot table.

Output



	A	B	C	D	E	F	G	H	I	J	K
1	Obs	ACTUAL	PREDICT	COUNTRY	REGION	DIVISION	PRODTYPE	PRODUCT	QUARTER	YEAR	MONTH
2	1	$925.00	$850.00	CANADA	EAST	EDUCATION	FURNITURE	SOFA	1	1993	Jan
3	2	$999.00	$297.00	CANADA	EAST	EDUCATION	FURNITURE	SOFA	1	1993	Feb
4	3	$608.00	$846.00	CANADA	EAST	EDUCATION	FURNITURE	SOFA	1	1993	Mar
5	4	$642.00	$533.00	CANADA	EAST	EDUCATION	FURNITURE	SOFA	2	1993	Apr
6	5	$656.00	$646.00	CANADA	EAST	EDUCATION	FURNITURE	SOFA	2	1993	May
7	6	$948.00	$486.00	CANADA	EAST	EDUCATION	FURNITURE	SOFA	2	1993	Jun
8	7	$612.00	$717.00	CANADA	EAST	EDUCATION	FURNITURE	SOFA	3	1993	Jul
9	8	$114.00	$564.00	CANADA	EAST	EDUCATION	FURNITURE	SOFA	3	1993	Aug
10	9	$685.00	$230.00	CANADA	EAST	EDUCATION	FURNITURE	SOFA	3	1993	Sep
11	10	$657.00	$494.00	CANADA	EAST	EDUCATION	FURNITURE	SOFA	4	1993	Oct

Output 4. Excel Spreadsheet That Contains a Pivot Table Created from a JavaScript File

This worksheet is used as the data source for all sample code in the entire Creating Pivot-Table Reports section.

ADDING FIELDS TO DROP ZONES IN A PIVOT TABLE

To display fields in a pivot table in Excel, columns in the data source that are listed in the field list can be dragged to one of four areas (called *drop zones*) on the page. The four drop-zone areas are **Report Filters**, **Row**, **Column**, and **Values** (as shown in Output 5). You can add fields to these drop zones by using TableEditor tagset options. The tagset option name corresponds directly to the actual drop zone.

The following code is a simple example of using tagset options to add fields to the various drop zones in a basic pivot table.

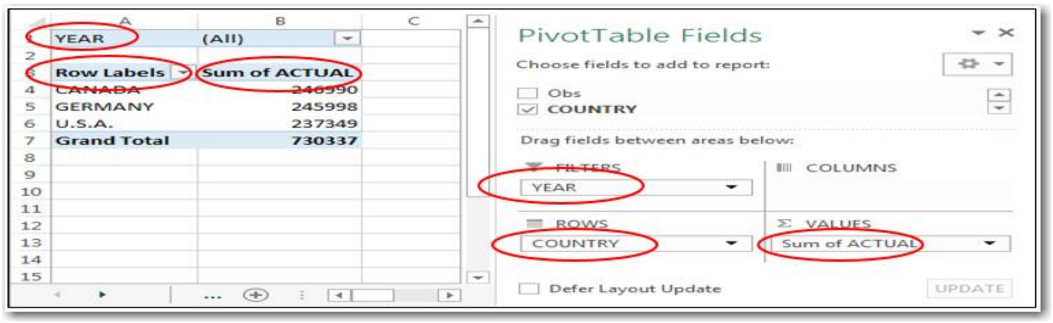
Example 5

```
ODS tagsets.tableeditor file="c:\temp\test.js"
    Options (update_target="c:\\temp\\temp.xlsx"
             output_type="script"
             sheet_name="Table_1"
             pivot_sheet_name="Table_1_pivot"   pivotpage="year"
             pivotrow="country"                 pivotdata="actual");

data _null_;
   file print;
   put _all_;
run;

ods tagsets.tableeditor close;
```

In This Example

- The PIVOT_SHEET_NAME= option names the worksheet that is created from the pivot table.
- The PIVOTPAGE= option names the page or report filter that adds the field **YEAR**.
- The PIVOTROW= option adds the field Country to the row label.
- The PIVOTDATA= option adds the column ACTUAL (**Sum of Actual)** to the **VALUES** section.

Output



	A	B	C
1	YEAR	(All)	
2			
3	Row Labels	Sum of ACTUAL	
4	CANADA	246990	
5	GERMANY	245998	
6	U.S.A.	237349	
7	Grand Total	730337	

PivotTable Fields

Choose fields to add to report:

☐ Obs
☑ COUNTRY

Drag fields between areas below:

FILTERS: YEAR
COLUMNS:
ROWS: COUNTRY
VALUES: Sum of ACTUAL

☐ Defer Layout Update   UPDATE

Output 5. A Basic Pivot Table with a Report Filter, a Row, and a Value Added

Suppose that you want to modify the table that was created in the previous section. For example, you might want to change the view so that it is a two-dimensional table (that is, a table with rows and columns). You can make that modification by adding the PIVOTCOL= option with the value **`Product`** to the previous example, which creates Example 6 below. This additional view, shown in Output 6, enables you to see the totals, per country, for all of the products. That information then enables you to answer the question as to which products performed better within the individual countries. Such information can be helpful, for example, if you are planning for inventory.

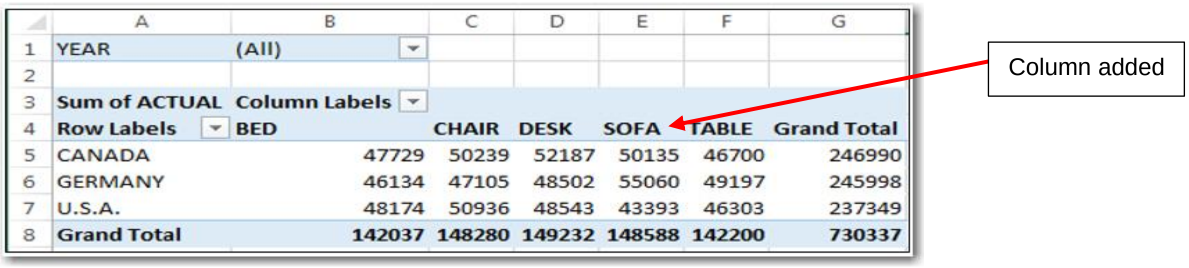
Example 6

```
ods tagsets.tableeditor file="c:\temp\test.js"
                        options(update_target="c:\\temp\\temp.xlsx"
                        output_type="script"     sheet_name="Table_1"
                        pivot_sheet_name="Table_1_pivot"
                        pivotpage="year"         pivotrow="country"
                        pivotcol="product"       pivotdata="actual");
   data _null_;
      file print;
      put _all_;
   run;
ods tagsets.tableeditor close;
```

In This Example

The PIVOTCOL= option generates a two-dimensional table by adding a column.

Output



	A	B	C	D	E	F	G
1	YEAR	(All)					
2							
3	Sum of ACTUAL	Column Labels					
4	Row Labels	BED	CHAIR	DESK	SOFA	TABLE	Grand Total
5	CANADA	47729	50239	52187	50135	46700	246990
6	GERMANY	46134	47105	48502	55060	49197	245998
7	U.S.A.	48174	50936	48543	43393	46303	237349
8	Grand Total	142037	148280	149232	148588	142200	730337

Output 6. Changing the Output View by Adding a Column Dimension

SELECTING RANGES FOR THE INPUT AND OUTPUT OF THE PIVOT TABLE

By default, a pivot table begins reading the source worksheet at position A1, and Excel begins writing the pivot table on a new worksheet at positon A1. However, at times, you might want to run analysis only on a specific portion of your data. You can do that by using the PTSOURCE_RANGE= and PTDEST_RANGE= tagset options, respectively, to specify where data can be read from and written to. For example, you might want to alter the location of where data is written for readability purposes or for better visual presentation, as demonstrated in this next example.

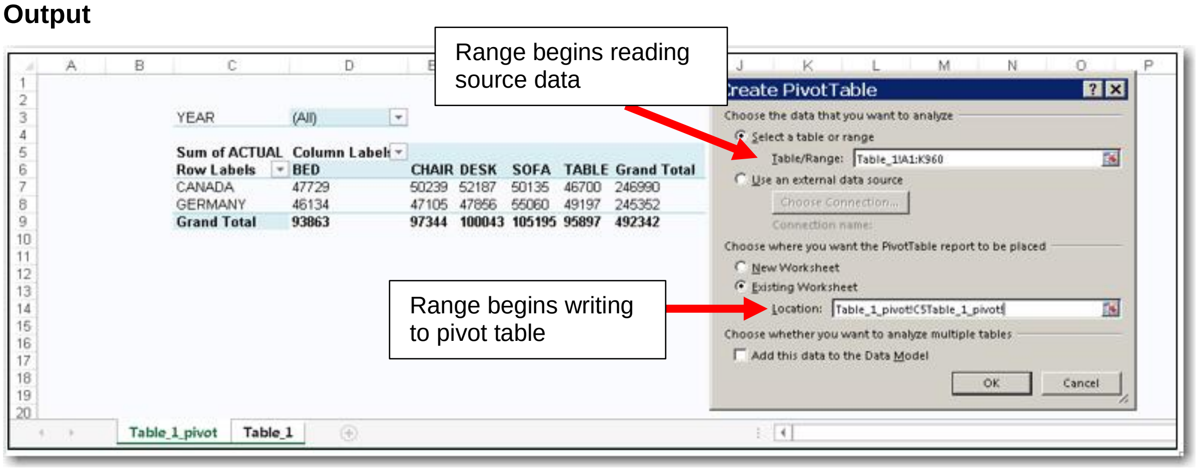
Example 7

```
ods tagsets.tableeditor file="c:\temp\test.js"
    options(update_target="c:\\temp\\temp.xlsx" output_type="script"
            sheet_name="Table_1"
            ptsource_range="Table_1!A1:K960"
            ptdest_range="Table_1_pivot!C5"
            pivotpage="year"
            pivotrow="country"
            pivotcol="product"
            pivotdata="actual");

data _null_;
   file print;
   put _all_;
run;

ods tagsets.tableeditor close;
```

In This Example

- The PTSOURCE_RANGE= option reads a data range that excludes the USA.
- The PTDEST_RANGE= option writes data beginning in cell C5 in the output.

Output


Range begins reading source data
Range begins writing to pivot table
Table_1_pivot
Table_1
OK
Cancel

YEAR (All)

Sum of ACTUAL	Column Labels					
Row Labels	BED	CHAIR	DESK	SOFA	TABLE	Grand Total
CANADA	47729	50239	52187	50135	46700	246990
GERMANY	46134	47105	47856	55060	49197	245352
Grand Total	93863	97344	100043	105195	95897	492342

Create PivotTable

Choose the data that you want to analyze
[✓ Select a table or range]
Table/Range: Table_1!A1:K960
Use an external data source
Choose Connection...
Connection name:

Choose where you want the PivotTable report to be placed
New Worksheet
[✓ Existing Worksheet]
Location: Table_1_pivot!C5Table_1_pivot!

Choose whether you want to analyze multiple tables
Add this data to the Data Model

Output 7. Using Tagset Options to Read and Write Data

MODIFYING THE REPORT LAYOUT, GRAND TOTALS, AND STYLES

Under **PivotTable Tools** in an Excel worksheet, there is a **Design** tab that enables you to modify the report layout, grand totals and subtotals, and the style and format of the pivot table. Depending on your preference, you might want to modify the default behaviors for these options. For example, you might want to modify the report layout because beginning with Excel 2007 the default layout changed to the compact layout. In Excel 2007 and later, the default layout is compact rather than tabular. Available layout types include outline, tabular, and compact. Each of these types has a slightly different presentation.

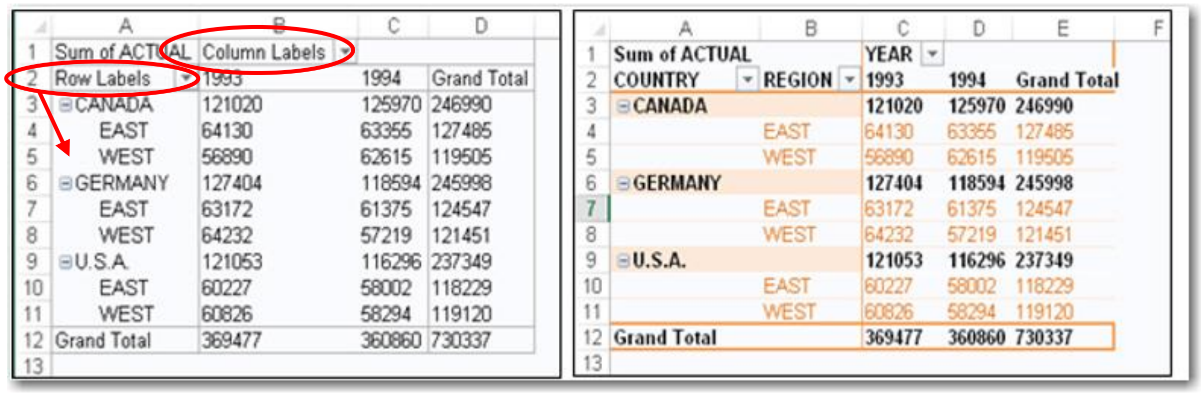
Example 8

```
ods tagsets.tableeditor file=" c:\temp\temp.js"
    options (update_target="c:\\temp\\temp.xlsx"
              output_type="script"         sheet_name="Table_1"
              pivot_layout="tabular"       pivot_format="light23"
              pivotdata_grandtotal="yes"   pivotrow="country,product"
              pivotcol="year"              pivotdata="actual");
data _null_;
   file print;
   put _all_;
run;

ods tagsets.tableeditor close;
```

In This Example

- The PIVOT_LAYOUT= option specifies the tabular format.
- The PIVOT_FORMAT= option modifies the overall style, changing it to Light23.
- The PIVOTDATA_GRANDTOTAL= option displays the grand total. PIVOTDATA_GRANDTOTAL=YES is the default setting.
- In Output 8 below, the tabular output on the right displays both the row and column names as well as the subtotals. This tabular report is created with the PIVOT_LAYOUT= option. The grand totals for this report are generated with the PIVOT_GRANDTOTAL= option, and the style is controlled by using the PIVOT_FORMAT= option.
- In Output 8 below, the table on the left is displayed in the compact layout while the table on the right is shown in the tabulate layout.

Output



	A	B	C	D
1	Sum of ACTUAL	Column Labels		
2	Row Labels	1993	1994	Grand Total
3	⊟CANADA	121020	125970	246990
4	EAST	64130	63355	127485
5	WEST	56890	62615	119505
6	⊟GERMANY	127404	118594	245998
7	EAST	63172	61375	124547
8	WEST	64232	57219	121451
9	⊟U.S.A.	121053	116296	237349
10	EAST	60227	58002	118229
11	WEST	60826	58294	119120
12	Grand Total	369477	360860	730337
13				

	A	B	C	D	E	F
1	Sum of ACTUAL		YEAR			
2	COUNTRY	REGION	1993	1994	Grand Total	
3	⊟CANADA		121020	125970	246990	
4		EAST	64130	63355	127485	
5		WEST	56890	62615	119505	
6	⊟GERMANY		127404	118594	245998	
7		EAST	63172	61375	124547	
8		WEST	64232	57219	121451	
9	⊟U.S.A.		121053	116296	237349	
10		EAST	60227	58002	118229	
11		WEST	60826	58294	119120	
12	Grand Total		369477	360860	730337	
13						

Output 8. Changing the Report Layout, the Grand Total, and the Style

SELECTING STATISTICS AND BASIC OPTIONS FOR A PIVOT TABLE

The field-setting options in Excel enable you to modify how a report is summarized. By default, summarization is a count for character values and a sum for numerical values. The Excel field-setting options are responsible for, but not limited to, the following tasks:

- modifying functions or statistics
- adding captions to summary fields
- changing how values are displayed
- modifying number formats
- computing columns

As mentioned earlier, the TableEditor tagset contains options that automate the field-setting option in Excel. The next example uses several of those options:

- PIVOTDATA_STATS=
- PIVOT_CALC=
- PIVOTDATA_CAPTION=
- PIVOTDATA_TOCOLUMNS=
- PIVOTDATA_FMT=

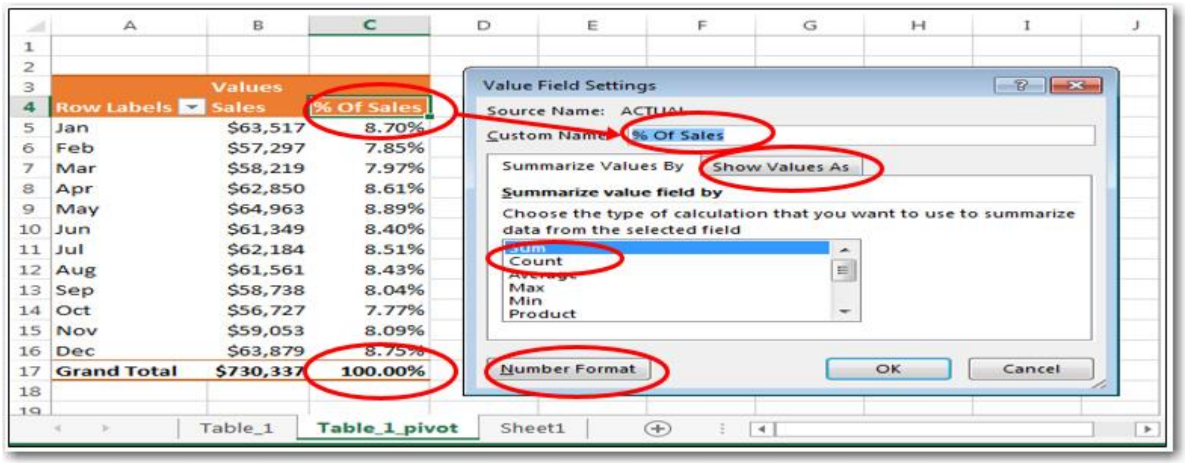
Example 9

```
ods tagsets.tableeditor file="c:\temp\temp.js"
    options(update_target="c:\\temp\\temp.xlsx"   output_type="script"
            sheet_name="Table_1"                    pivot_format="light7"
            pivotrow="month"                        pivotdata="actual,actual"
            pivotdata_stats="sum,sum"               pivotcalc="none,total"
            pivotdata_caption="Sales,% Of Sales"
            pivotdata_fmt="$#,###~###.00%"          pivotdata_tocolumns="yes"
            );

   data _null_;
     file print;
     put _all_;
   run;

ods tagsets.tableeditor close;
```

In This Example

- The PIVOTDATA_STATS= option specifies which statistic Excel should use for summarizing data values.
- The PIVOT_CALC= option displays a second column for ACTUAL that shows the value for that column as a percentage of TOTAL.
- The PIVOTDATA_CAPTION= option specifies a label for the summary fields.
- The PIVOTDATA_TOCOLUMNS= option splits the summary field ACTUAL from one vertical row (which is the default behavior) into two columns (SALES and % OF SALES).
- The PIVOTDATA_FMT= option adds two number formats: one for the summary (SALES column) and another for the summary field (% of SALES column) displayed as a percentage. Notice that the formats in this option are separated with the tilde (~) character.

Output



Row Labels	Values Sales	% Of Sales
Jan	$63,517	8.70%
Feb	$57,297	7.85%
Mar	$58,219	7.97%
Apr	$62,850	8.61%
May	$64,963	8.89%
Jun	$61,349	8.40%
Jul	$62,184	8.51%
Aug	$61,561	8.43%
Sep	$58,738	8.04%
Oct	$56,727	7.77%
Nov	$59,053	8.09%
Dec	$63,879	8.75%
Grand Total	$730,337	100.00%

Value Field Settings

Source Name: ACTUAL

Custom Name: % Of Sales

Summarize Values By | Show Values As

Summarize value field by

Choose the type of calculation that you want to use to summarize data from the selected field

Sum
Count
Average
Max
Min
Product

Number Format | OK | Cancel

Output 9. Using TableEditor Tagset Options to Select Statistics and Set Formatting

USING CONDITIONAL FORMATTING TO ADD VISUAL ELEMENTS TO OUTPUT

Adding conditional formatting enables you to highlight data based on a condition, an expression, or values. In Excel, some common methods of applying conditional formats includes using the **Data Bars**, **Icon Sets**, and **Color Scales** options as well as by using expressions and cell-value rules. With the FORMAT_CONDITION= tagset option, you can apply the same conditional formats in a programmatically automated way. You can use this option to specify one or more of the format conditions that are listed in Table 3. Be aware that each format request in the option must be separated by the tilde (~) character.

. Format Type	Function	Syntax for Format Type Parameters
DATABAR	Adds data bars to columns or ranges.	`type, range`
ICONSET	Adds an icon set to columns or ranges.	`type, range, iconset`
COLORSCALE	Adds a color scale to columns or ranges.	`type, range`
CELLVALUE	Adds formatting to columns or ranges based on cell values.	`type, range, operator, formula1, formula2`
EXPRESSION	Adds formatting to columns or ranges based on an expression.	(`type, range, operator, formula1, formula2`)

Table 3. Parameters for the FORMAT_CONDITON= Tagset Option

Table 3 lists only a subset of the format types and their parameters that are available in the tagset. For a full list of parameters, see the TableEditor tagset's help, available at in this ZIP file: **support.sas.com/rnd/base/ods/odsmarkup/tableeditor/tableeditor.zip**

The following example illustrates how to use the FORMAT_CONDITION= option to add conditional formatting to columns in a pivot report. This example shows all five conditions: DATABAR, ICONSET, COLORSCALE, CELLVALUE, and EXPRESSION

Example 10

```
  ods tagsets.tableeditor file="c:\temp\test.js"
      options(update_target="c:\\temp\\temp.xlsx" output_type="script"
              sheet_name="Table_1"
              format_condition="
              databar,b3:c16       ~   iconsets,d3:d16,6 ~
              colorscale,e3:e16    ~   cellvalue,f3:f16,6,6000 ~
              expression,g3:g16,0,=Mod(row(),2)"
              pivotrow="month" pivotcol="product" pivotdata="actual");
  data _null_;
     file print;
     put _all_;
  run;

ods tagsets.tableeditor close;
```

In This Example

The FORMAT_CONDITION= option specifies the following conditional formatting:

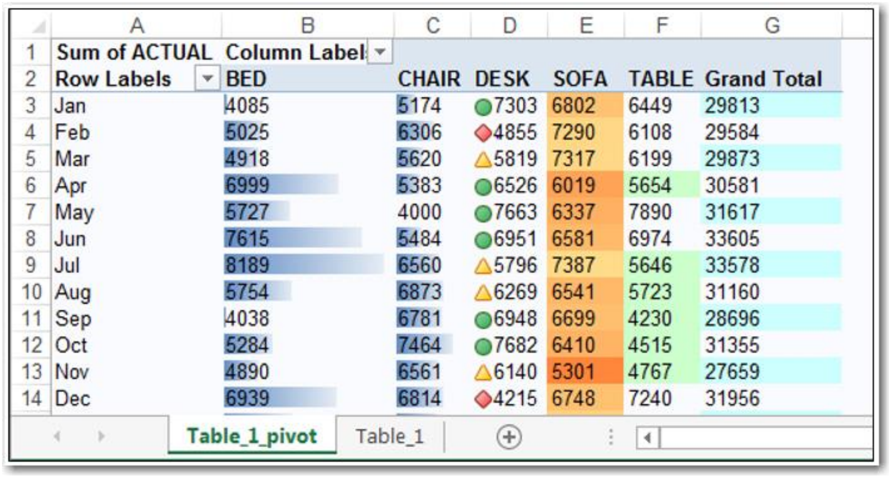
- Data bars are added to columns 2 and 3 by specifying a range for the DATABARS parameter to add data bars to columns 2 and 3.
- An icon sets (traffic-sign icons) are added to column 4. The traffic-sign icon set is available in Excel's icon-set list.

(list continued)

- A colors-scale condition is added to column 5. This condition generates a scale of color gradations based on a rule.
- In Excel, the condition shown in Example 10 for column 6 is evaluated as `cell value is "<6000"`. This condition modifies the background color of values in column 6 when those values are less than 6000.
- The 6 shown in the condition in Example 10 (`cellvalue,f3:f16,6,6000`) is the operator. The operators that are available for this formula are **0** (none),**1** (between), **2** (not between), **3** (equal), **4** (not equal), **5** (greater) , **6** (less) , **7** (greater equal), and **8** (less equal).
- An expression format type is used for column 7. That expression, shown below, modifies the background color on every other row.

```
=Mod(row(),2)
```

Output



	A	B	C	D	E	F	G
1	Sum of ACTUAL	Column Label					
2	Row Labels	BED	CHAIR	DESK	SOFA	TABLE	Grand Total
3	Jan	4085	5174	●7303	6802	6449	29813
4	Feb	5025	6306	◆4855	7290	6108	29584
5	Mar	4918	5620	△5819	7317	6199	29873
6	Apr	6999	5383	●6526	6019	5654	30581
7	May	5727	4000	●7663	6337	7890	31617
8	Jun	7615	5484	●6951	6581	6974	33605
9	Jul	8189	6560	△5796	7387	5646	33578
10	Aug	5754	6873	△6269	6541	5723	31160
11	Sep	4038	6781	●6948	6699	4230	28696
12	Oct	5284	7464	●7682	6410	4515	31355
13	Nov	4890	6561	△6140	5301	4767	27659
14	Dec	6939	6814	◆4215	6748	7240	31956

Output 10. Using the FORMAT_CONDITION= Tagset Option to Add Conditional Formatting for Columns

In a quick glance at Output 10, you can answer several questions based on the formatting, even before you look at the numbers in detail. For example, based on the data bars, you can easily see that July is the best sales month for the product **BED** because that column has the longest data bar.

CREATING CALCULATED COLUMNS FROM A PIVOT-TABLE COLUMN

As part of your business, you might need to determine how well your finance department forecasted sales. You can make that determination by creating calculated (or, computed) columns in your pivot table. These columns show you a different view of the data by calculating a new field that is not in the source data. A *calculated column* can be created from a formula as simple as the difference between two fields, or it can be a more complicated calculation (for example, a calculation that dynamically projects forecasted returns based on a standard rate of .05%).

To create a calculated column using the tagset, you use the ADDFIELD= option. This option contains an expression that is required to generate the new column. You then add the name of this calculated in the PIVOTDATA= option. This option selects the new column as one of the columns that should be displayed. To add multiple calculated columns, you need to separate the values by a comma.

The following example illustrates how to use the ADDFIELD= and PIVOTDATA= options to add a calculated column to a pivot table:

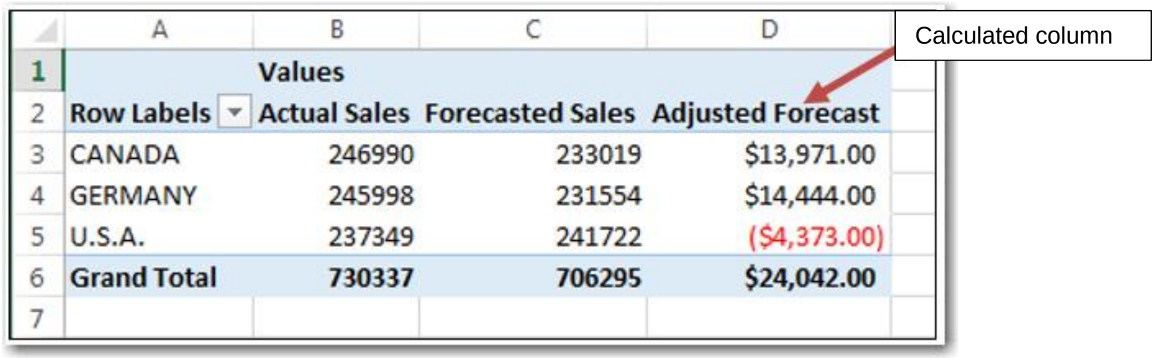
Example 11

```
ods tagsets.tableeditor file="c:\temp\temp.js"
    options(update_target="c:\\temp\\temp.xlsx" output_type="script"
             sheet_name="Table_1"
             addfield="Adjusted =actual - predict"
             pivotdata_caption="Actual Sales,Forecasted Sales,Adjusted
                                Forecast"
             pivotdata="actual,predict,adjusted" pivotrow="country");

data _null_;
   file print;
   put _all_;
run;

ods tagsets.tableeditor close;
```

In This Example

- The ADDFIELD= option, along with a calculation expression, creates the calculated variable (column) ADJUSTED.
- The new column (ADJUSTED) is included in the PIVOTDATA= option, which displays the column on the worksheet.
- The PIVOTDATA_CAPTION= option creates a label for all three summary variables (ACTUAL, PREDICT, and ADJUSTED).

Output

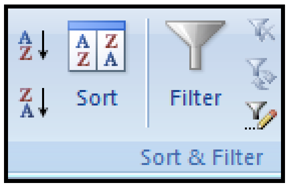


	A	B	C	D
1		Values		
2	Row Labels	Actual Sales	Forecasted Sales	Adjusted Forecast
3	CANADA	246990	233019	$13,971.00
4	GERMANY	245998	231554	$14,444.00
5	U.S.A.	237349	241722	($4,373.00)
6	Grand Total	730337	706295	$24,042.00
7				

Output 11. Using the ADDFIELD= and PIVOTDATA= Options to Add a Calculated Column

SORTING AND FILTERING IN PIVOT-TABLE REPORTS

Sorting and filtering on values within a table allows you to have yet another view of your data. When you filter data, only rows that meet the filter criteria are displayed; the other values are hidden. When you add a filter, a filter icon (shown in the image below) is added to the field.



Excel also provides sorting capability for pivot tables, which enables you to view data based on a preselected order that you want. You can sort the report filter, the rows, or the columns either in ascending or descending order. Using a combination of filtering and sorting, you can view only the subset of data that you want and sort that data based on the order that you choose. This section discusses several options for filtering and sorting.

Another important feature for Excel users is the ability to use pre-selected filters and a sorting order in a pivot table. With the TableEditor tagset, you can add filters to the page dimension by including the

PIVOTPAGE= and PIVOTPAGE_FILTER= options in your SAS code. You can also pass in the preselected value by which you want to filter your pivot table. The PIVOTPAGE= option specifies the column while the PIVOTPAGE_FILTER= option specifies the value within that column by which the page should be filtered.

You add sorting capability by using the PIVOT_SORT= option. This option enables you to specify one or more columns on which to sort. For example, you might want to see the output presented in descending order for the rows and columns. Using the PIVOT_SORT= option , you can specify the exact column or columns that you want sorted as well as how the columns should be sorted. The PIVOT_SORT= option has two parameters: a parameter that specifies the name of the column that is to be sorted and another parameter that specifies how the column is sorted.

PIVOT_SORT="*var*-1,*sort-type* ~ *var*-2,*sort-type*"

A *sort-type* value of **1** specifies that the variable is sorted in ascending order. A *sort-type* value of **2** that the variable is sorted in descending order.

The following example includes both filters and sorting order:

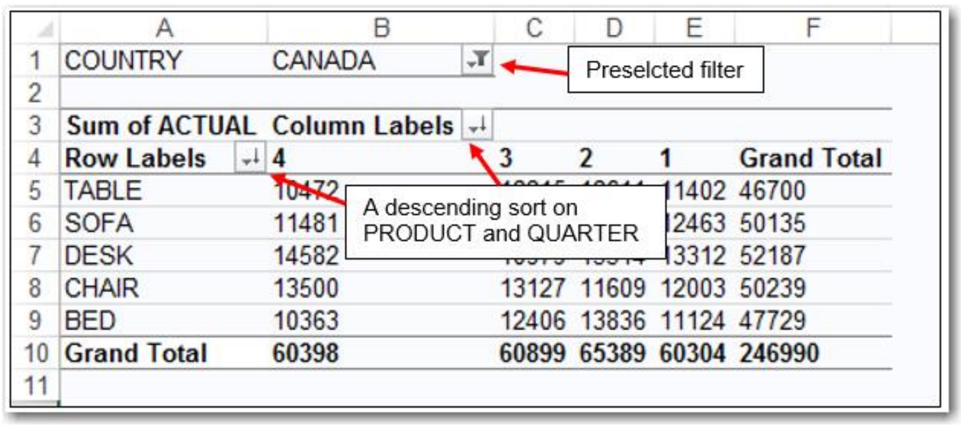
Example 12

```
ods tagsets.tableeditor file="c:\temp\test.js"
    options(update_target="c:\\temp\\temp.xlsx" output_type="script"
            sheet_name="Table_1"
            pivot_sort="product,2 ~ quarter,2"
            pivotpage_filter="Canada"          pivotpage="Country"
            pivotcol="quarter"                 pivotrow="product"
            pivot_format="light1" pivotdata="actual");
data _null_;
   file print;
   put _all_;
run;

ods tagsets.tableeditor close;
```

In This Example

- The PIVOTPAGE= option specifies the column COUNTRY, while the PIVOTPAGE_FILTER= option specifies **Canada** as the value by which to filter the page.
- The PIVOT_SORT= option specifies that both the PRODUCT and QUARTER fields should be sorted in descending order. The fields are sorted for the row and the column dimensions. Notice that each column is separated by the tilde (~) character.

Output



	A	B	C	D	E	F
1	COUNTRY	CANADA				
2						
3	Sum of ACTUAL	Column Labels				
4	Row Labels	4	3	2	1	Grand Total
5	TABLE	10472	[illegible]	[illegible]	11402	46700
6	SOFA	11481	[illegible]	[illegible]	12463	50135
7	DESK	14582	[illegible]	[illegible]	13312	52187
8	CHAIR	13500	13127	11609	12003	50239
9	BED	10363	12406	13836	11124	47729
10	Grand Total	60398	60899	65389	60304	246990
11						

Output 12. Using the PIVOTPAGE= and PIVOT_SORT= Tagset Options to Add a Pre-Selected Filter and a Sort Order

Filtering Reports Using Slicers

Slicers are filters that enable you to select data visually by using buttons, which control how a report is filtered. Slicers (available since Excel 2010) are much like the filters that are covered in the previous example, but they are more interactive and accessible. Slicers enable you to create interactive pivot tables and charts by changing the table and chart based on a selected value.

With the TableEditor tagset options, you can add a slicer the report filter (page), a row, or a column. For example, the PIVOT_SLICER= option enables you to add one or more slicers to the report filter or to row and column filters. If you include multiple variables in the option, you need to separate each field by a comma.

The following examples use the report that is generated in Example 11 and adds a slicer to the report. The slicer filter creates a report with interactive buttons that enable you to determine the best-selling product for each country.

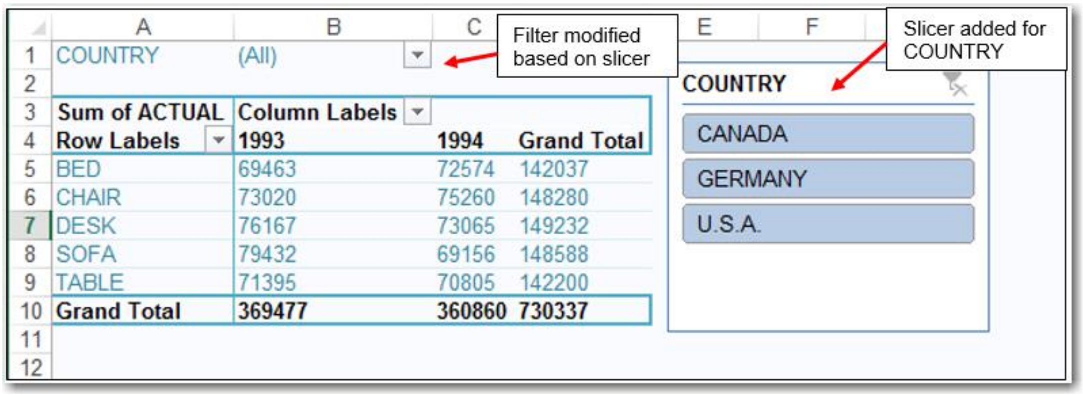
Example 13.

```
  ods tagsets.tableeditor file=" c:\temp\test.js"
      options(update_target="c:\\temp\\temp.xlsx" output_type="script"
              sheet_name="Table_1"                pivot_slicer="country"
              pivot_slicer_position="200,10"      pivotpage="country"
              pivot_format="light13"              pivotrow="product"
              pivotcol="year"                     pivotdata="actual");

     data _null_;
        file print;
        put _all_;
     run;

ods tagsets.tableeditor close;
```

In This Example

- Using the PIVOT_SLICER= and PIVOTPAGE= options together creates a slicer for the variable COUNTRY.
- The PIVOT_SLICER_POSITION= option specifies where the slicer should appear. In this case, the slicer starts in the top (200), left (10) position. This option has four, possible parameters: *top, left, width,* and *height.*

Output



	A	B	C	D
1	COUNTRY	(All)		
2				
3	Sum of ACTUAL	Column Labels		
4	Row Labels	1993	1994	Grand Total
5	BED	69463	72574	142037
6	CHAIR	73020	75260	148280
7	DESK	76167	73065	149232
8	SOFA	79432	69156	148588
9	TABLE	71395	70805	142200
10	Grand Total	369477	360860	730337
11				
12				

Output 13. Using the PIVOT_SLICER and PIVOT_SLICER_POSITION Options to Add Slicers to a Pivot Table

Generating Filters That Are Attached to Both Tables and Charts

In the last section, you learned how to create a slicer that dynamically changes the data in a pivot table based on a selected value. The next example illustrates how you can filter data further by creating a slicer that is attached to both a table and a chart.

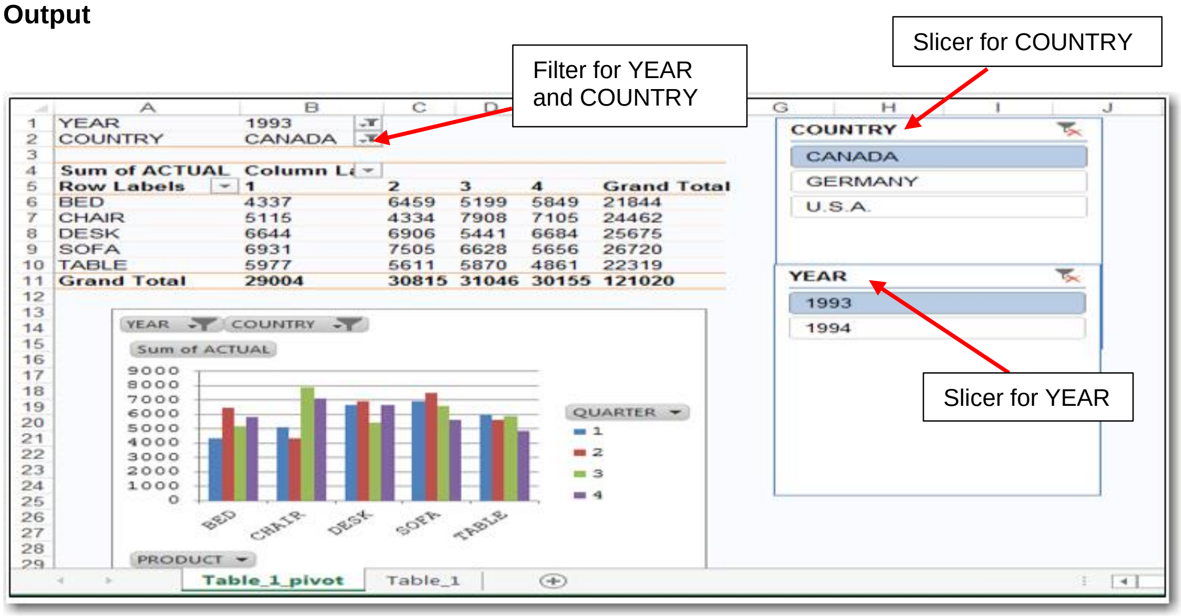
Example 14

```
ods tagsets.tableeditor file=" c:\temp\temp.js"
    options(update_target="c:\\temp\\temp.xlsx" output_type="script"
           sheet_name="Table_1" pivot_format="light7"
           pivotpage="country,year" pivotrow="product"  pivotcol="quarter"
           pivotdata="actual" pivot_slicer="country,year"
           pivot_slicer_position="200,10 ~ 200,130" pivotcharts="yes"
           chart_type="bar" chart_location="Table_1_Pivot"
           chart_position="100,100,200,200");
   data _null_;
      file print;
      put _all_;
   run;

ods tagsets.tableeditor close;
```

In This Example

- The PIVOT_SLICER= option adds a slicer for both the COUNTRY and YEAR variables.
- The PIVOTCHARTS= option specifies that you want to add a chart, and the CHART_TYPE='BAR' option specifies that you want a bar chart.
- The CHART_LOCATION= option specifies to which worksheet the slicer should be attached, and the CHART_POSITION= option specifies where the slicer image is located, based on coordinates. (Those coordinates are specified from the parameters in the CHART_POSITION= option syntax, which is CHART_LOCATION=(*x*, *y*, *width*, *height*)

Output



```
YEAR               1993
COUNTRY            CANADA

Sum of ACTUAL      Column La
Row Labels         1          2      3      4      Grand Total
BED                4337       6459   5199   5849   21844
CHAIR              5115       4334   7908   7105   24462
DESK               6644       6906   5441   6684   25675
SOFA               6931       7505   6628   5656   26720
TABLE              5977       5611   5870   4861   22319
Grand Total        29004      30815  31046  30155  121020
```

Output 14. Using the PIVOTCHARTS=, CHART_TYPE=, CHART_LOCATION=, and CHART_POSITION= Options to Add Slicers to a Table and a Chart

ADDING DASHBOARDS FOR VISUAL DISPLAY

A dashboard simplifies complex data and offers, at a glance, the information you need to make good decisions quickly. Typically, dashboards consist of tables, graphs, gauges, and numbers that enable you to make quick decisions. To ensure that the dashboard is useful, though, it takes some thought as to exactly which types of elements to include your dashboard. The following example uses the DASHBOARD=, PIVOT_SERIES=, PIVOTCHART=, PTDEST_RANGE=, and CHART_TYPE= tagset options to create a dashboard:

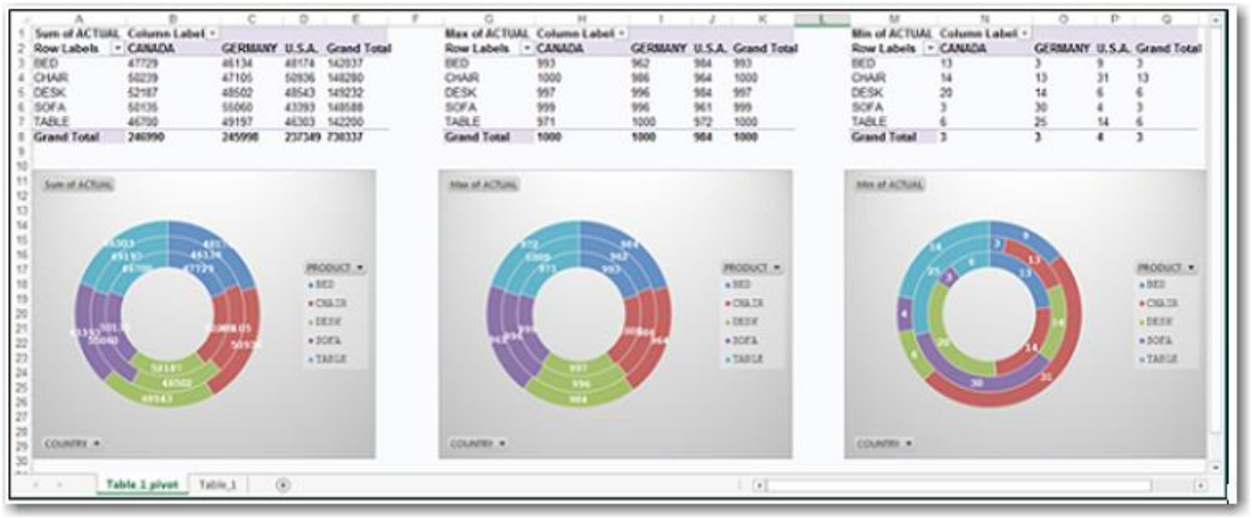
Example 15

```
ods tagsets.tableeditor file="c:\temp\temp.js"
    options (update_target="c:\\temp\\temp.xlsx" output_type="script"
              sheet_name="Table_1" pivot_series="yes"
              dashboard="yes" pivotrow= "product|product|product"
              pivotcol=" country|country|country"
              pivotdata="actual |actual |actual"
              pivotdata_stats="sum| max |min"
              ptdest_range="a3,g3,m3" pivotcharts="yes"
              chart_type="doughnut" chart_style="300"
              chart_location="Table_1_Pivot"
              chart_position="150,0,250,250|
                               150,350,250,250|
                               150,700,250,250");
   data _null_;
      file print;
      put _all_;
   run;

ods tagsets.tableeditor close;
```

In This Example

- Multiple pivot tables are added, per worksheet, with the PIVOT_SERIES= option. This option generates a new analysis, with each analysis field separated with a vertical bar (|).
- The DASHBOARD= option specifies that the tables are to be generated within the same worksheet.
- The PIVOTROW=, PIVOTCOL=, PIVOTDATA=, and PIVOTDATA_STATS= options generate separate analyses for each column, which are separated by a vertical bar (|) in the options.
- The PTDEST_RANGE= option specifies where pivot tables should be placed in the worksheet.

Output



Sum of ACTUAL	Column Label			
Row Labels	CANADA	GERMANY	U.S.A.	Grand Total
BED	47729	46134	48174	142037
CHAIR	50239	47105	50936	148280
DESK	52187	48502	48543	149232
SOFA	50135	55060	43393	148588
TABLE	46700	49197	46303	142200
Grand Total	246990	245998	237349	730337

Max of ACTUAL	Column Label			
Row Labels	CANADA	GERMANY	U.S.A.	Grand Total
BED	993	962	984	993
CHAIR	1000	986	964	1000
DESK	997	996	984	997
SOFA	999	996	961	999
TABLE	971	1000	972	1000
Grand Total	1000	1000	984	1000

Min of ACTUAL	Column Label			
Row Labels	CANADA	GERMANY	U.S.A.	Grand Total
BED	13	3	9	3
CHAIR	14	13	31	13
DESK	20	14	6	6
SOFA	3	30	4	3
TABLE	6	25	14	6
Grand Total	3	3	4	3

Output 15. Adding Multiples Dashboard Tables to Output

ADDING A CUSTOM TASK IN THE GUI TO GENERATE PIVOT TABLES

Including options in code is a great way to generate custom pivot tables. However, if you use a GUI-based application such as SAS® Enterprise Guide® or SAS® Studio, you might prefer a point-and-click approach. The SAS® Add-In for Microsoft Office offers this capability. If you use SAS Enterprise Guide, you can generate a custom task using either the Visual Basic .NET or C# programming languages. If you use SAS Studio, you can generate a custom task using the common task model. This section discusses how to generate a basic pivot table using the Common Task Model in SAS Studio. You can then take that basic report and build on it as you want to later.

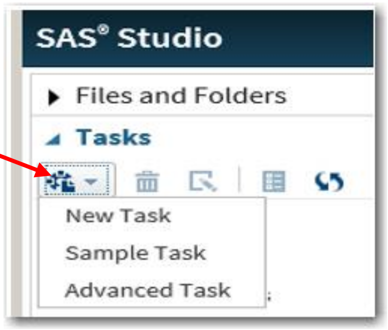
The common task model is an XML file that defines how a task should appear in SAS Studio, and it defines how to run that task. The CTM contains **`<Task>`** elements, which have the following children elements:

- The **`<Registration>`** element registers and identifies the task and adds information such as the task name and description.
- The **`<Metadata>`** element specifies any input data source and role assignments for variables.
- The **`<UI>`** element determines how the user interface is presented to the user.
- The **`<CodeTemplates>`** element determines what type of output to use for the task. Typically, the result or output is SAS code.
- The **`<Dependencies>`** element describes the dependencies that options might have on one another.
- The **`<Requirements>`** element sets up requirements that must be met in order to execute the code.

Not all of these children elements are necessary in order to generate a custom task. For a detailed discussion about the common task model, see "Chapter 1: Introduction to the Common Task Model" in the *SAS® Studio 3.5: Developer's Guide to Writing Custom Tasks* for your release of the software. (**`support.sas.com/documentation/cdl/en/webeditordg/68829/PDF/default/webeditordg.pdf`**)

To begin creating a new pivot-table task, perform the following steps:

1. Open SAS Studio. In the left navigation pane and select **Tasks**.
2. Select **New Task**, as shown below:



These steps generate a simple, blank task that contains a limited set of elements that are needed to execute the task. If you click the icon that is indicated in window above by the arrow, you receive a menu of choices to create either a new task, a sample task, or an advanced task that you can modify. You can also copy an existing task into the **My Tasks** folder by selecting the task and right-clicking it. When you right-click, an option is available to move the task to the **My Tasks** folder.

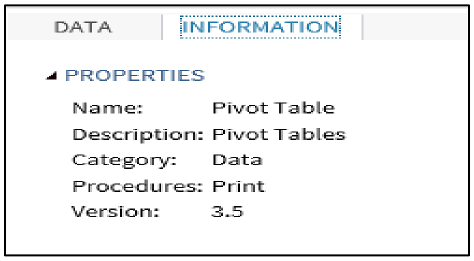
For this discussion, a simple, new task is created SAS Studio (Single-User Edition). Each code segment that is discussed is part of the larger code example that generates the new task.

After the XML prologue statement in the New Task window, the task begins with the <**`Task>`** element, which is displayed below along with the **`schemaVersion=`** attribute.

```
<?xml version="1.0" encoding="UTF-8" ?>
   <Task schemaVersion="4.0">
```

The **`<Registration>`** element is the first child element of the task that is needed in order to identify the type of task .In the example below for the new pivot-table task, a unique name is added with the <**`Name>`** subelement. A unique identifier is added using the **`<GUID>`** subelement. The **`<Version>`** subelement is included to denote the release version of SAS Studio. Finally, the **`<Category>`** subelement specifies the category type for this task. All of the information in this **`<Registration>`** element appears on the **Information** tab in the left pane of SAS Studio.

```
<Registration>
   <Name>Pivot Table</Name>
   <GUID>9D3F4022-79A0-4D47-A7B1
         -A83579EB6504</GUID>
   <Version>3.5</Version>
   <Category>Data</Category>
</Registration>
```



Name:	Pivot Table
Description:	Pivot Tables
Category:	Data
Procedures:	Print
Version:	3.5

The element, **`<Metadata>`**, shown in the next part of the code example, comprises two subordinate elements: **`<DataSources>`** and **`<Roles>`**. In the example, you can see that the **`<DataSource>`** element is a child element of the **`<DataSources>`** element, and it specifies the input data source for the pivot table. The **`name=`** attribute specifies the name DATASET for this role.

```
  <Metadata>
     <DataSources>
        <DataSource name="dataset">
           <Roles>
              <Role name="page" type="C">Row Filter</Role>
              <Role name="row">Row Labels</Role>
              <Role name="column">Column Labels</Role>
              <Role name="values" type="N">Values</Role>
           </Roles>
        </DataSource>
     </DataSources>
```

The **`<Roles>`** element in this example identifies the variables that are required in order to execute the pivot table. These variables are added to specific drop zones when the Excel pivot table is generated.

The <**`Role>`** element, a child of the **`<Roles>`** element, describes each role type for the task. The **`name=`** attribute specifies a name for the role. The **`type`**= attribute specifies the type of column to use. For example, **`type='n'`** restricts the actual role to numeric variables or fields only.

To include the metadata from the data source and the roles into the user interface, you have to define that metadata by using multiple **`<Option>`** elements, as shown in the example below. The **`<Option>`** element uses the **`name=`** attribute to add the metadata to the UI. The metadata includes the labels for the tab and the input source, roles, options, and options group.

```
     <Options>
        <Option name="dataTab" inputType="string">DATA</Option>
        <Option name="dataGroup" inputType="string">DATA</Option>
        <Option name="rolesGroup" inputType="string"></Option>
        <Option name="optionsGroup" inputType="string"/>
        <Option name="charts" inputType="checkbox">Pivot Chart</Option>
        <Option name="grandtot" inputType="checkbox">Grand Total</Option>
        <Option name="data2cols" inputType="checkbox">Data2Columns</Option>
     </Options>
  </Metadata>
```

After you define all of the metadata, you can then define how you want to metadata to be displayed in the UI, as shown in the next segment of the code example:

```
  <UI>
      <Container option="dataTab">
          <Group option="dataGroup" open="true">
             <DataItem data="dataset" />
          </Group>
          <Group option="rolesGroup" open="true">
             <RoleItem role="page" />
             <RoleItem role="row" />
             <RoleItem role="column" />
             <RoleItem role="values" />
          </Group>
          <Group option="optionsGroup" open="true">
             <OptionItem option="charts"/>
             <OptionItem option="grandtot"/>
             <OptionItem option="data2cols"/>
          </Group>
      </Container>
     </UI>
```

In This Example

- The **`<Container>`** element adds a tab labeled **DATA** to the UI. This label attribute is defined in the **`<Options name='dataTab' inputType="string">`** attribute on the previous page.
- The **`<Group>`** elements indicate for the metadata (created in the **`<Option>`** elements on the previous page) to be displayed in the UI. The **`<Group option="dataGroup">`** element adds the label **DATA** to the UI. This **`<Group>`** group element refers back to the **`<Option name="dataGroup" inputType="string">`** element shown on page XXXXX. The **`<DataItem>`** element creates a field for the input data source, which is a data set.
- The **`<Group option='rolesGroup'>`** element refer back to the **`<Option name="rolesGroup" inputType="string">`** element in the list of **`<Option>`** elements. The label for this section is **Drop Zone**.
- The **`<RoleItem>`** element creates a field for each item that is defined in the roles that are specified in the metadata.

In this next example, this **`<CodeTemplate>`** segment generates SAS code as a result of items that you select in the UI. This element uses the Apache Velocity template language to generate the SAS code. The generated SAS code is shown below. Items in the code preceded by a hash sign (#) are directives that manipulate the output, and strings that begin with a dollar sign ($) are velocity variables.

```
<CodeTemplate>

   <![CDATA[
       options noxwait noxsync;
       ods noresults;
       %let workdir=%sysfunc(getoption(WORK));
       %let re=%sysfunc(prxparse(s/\\/\\\\/));
       %let worksheet_path=%sysfunc(prxchange(&re,-1,"&workdir"));
       %let target_path=%sysfunc(dequote(&worksheet_path));

       ods excel file="&workdir\temp.xlsx" options(sheet_name="$dataset");

          proc print data=sashelp.prdsale;
          run;

       ods excel close;
       ods tagsets.tableeditor file="&workdir\temp.js"
           options(update_target="&target_path\\temp.xlsx"
                   output_type="script"
                   sheet_name="$dataset"
                   #if ($page.size() > 0)
                       pivotpage="#foreach( $item in $page)
                                  $item #if( $foreach.hasNext), #end
                                  #end"
                   #end

                   #if ($row.size() >0)
                       pivotrow=" #foreach( $item in $row)
                                  $item #if( $foreach.hasNext), #end
                                  #end"
                   #end
                   #if ($column.size() > 0)
                       pivotcol=" #foreach( $item in $column)
                                  $item #if( $foreach.hasNext ), #end
                                  #end"
                   #end
```

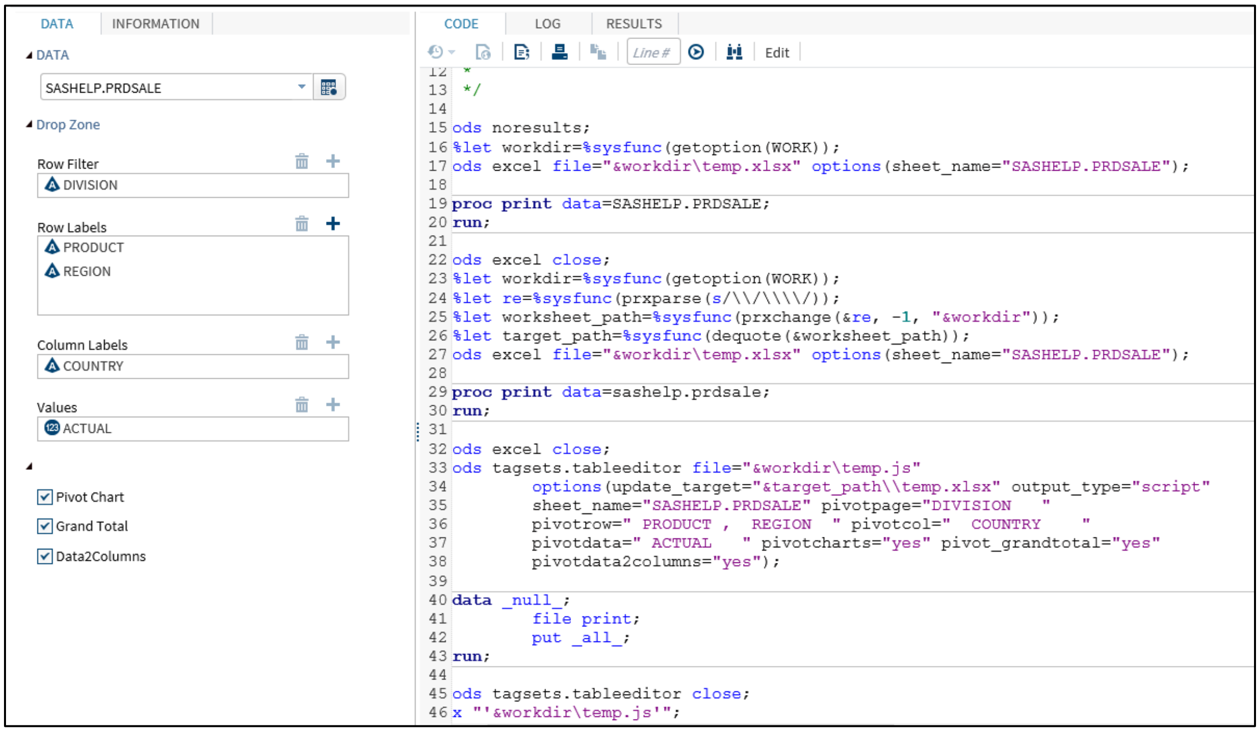
(code continued)

```
                #if ($values.size() > 0)
                    pivotdata="#foreach( $item in $values)
                               $item #if( $foreach.hasNext ), #end
                               #end"
                #end

                #if ($charts==1)  pivotcharts="yes" #end
                #if (grandtot==1) pivot_grandtotal="yes" #end
                #if ($data2cols==1)pivotdata2columns="yes" #end

        data _null_;
           file print;
           put _all_;
        run;

      ods tagsets.tableeditor close;
      x "'&workdir\temp.js'";
    ]]>

  </CodeTemplate>

</Task>
```

In This Example

- The macro variable WORKDIR is created with the GETOPTION() function to store the current **`Work`** library, which is temporary.
- The PRXPARSE() and PRXCHANGE() functions create an escaped path that is used with the UPDATE_TARGET= option to update the workbook with the pivot table.
- The ODS Excel destination (ODS EXCEL statement) creates a workbook with the input source data, and the PRINT procedure prints the output.
- The TableEditor tagset (ODS TAGSETS.TABLEEDITOR statement) updates the temporary worksheet that is created by generating a JavaScript file. The JavaScript file, created with the OUTPUT_TYPE='SCRIPT' option, reads the worksheet that is created with the Excel destination and generates a pivot table from that worksheet.
- The Velocity variables PAGE, ROW, COLUMN, and VALUES that are created in the UI are added to the PIVOTPAGE=, the PIVOTROW=, the PIVOTCOL= and the PIVOTPAGE= option. These options add the variables that are selected in the UI to the comparable Excel drop zone.
- The #IF directive is used with the SIZE() method to determine whether variables are selected for one of the roles in the UI.
- Each of the tagset options PIVOTPAGE=, PIVOTROW=, PIVOTCOL=, and PIVOTDATA= are assigned one or more variables by using the FOREACH directive to loop through the velocity variables. Each variable name is separated by a comma. The check boxes have a value of **`0`** if they are not selected; they have a value of **`1`** if they are selected for the remaining options.
- To save the task, click and enter a unique name for the task.

Output



DATA: SASHELP.PRDSALE

Drop Zone

Row Filter: DIVISION

Row Labels: PRODUCT, REGION

Column Labels: COUNTRY

Values: ACTUAL

☑ Pivot Chart

☑ Grand Total

☑ Data2Columns

```
 *
 */

ods noresults;
%let workdir=%sysfunc(getoption(WORK));
ods excel file="&workdir\temp.xlsx" options(sheet_name="SASHELP.PRDSALE");

proc print data=SASHELP.PRDSALE;
run;

ods excel close;
%let workdir=%sysfunc(getoption(WORK));
%let re=%sysfunc(prxparse(s/\\/\\\\/));
%let worksheet_path=%sysfunc(prxchange(&re, -1, "&workdir"));
%let target_path=%sysfunc(dequote(&worksheet_path));
ods excel file="&workdir\temp.xlsx" options(sheet_name="SASHELP.PRDSALE");

proc print data=sashelp.prdsale;
run;

ods excel close;
ods tagsets.tableeditor file="&workdir\temp.js"
        options(update_target="&target_path\\temp.xlsx" output_type="script"
        sheet_name="SASHELP.PRDSALE" pivotpage="DIVISION    "
        pivotrow=" PRODUCT ,  REGION  " pivotcol="  COUNTRY    "
        pivotdata=" ACTUAL   " pivotcharts="yes" pivot_grandtotal="yes"
        pivotdata2columns="yes");

data _null_;
        file print;
        put _all_;
run;

ods tagsets.tableeditor close;
x "'&workdir\temp.js'";
```

Output 16. Pivot-Table Task That Is Created in SAS Studio

CONCLUSION

As demonstrated in this paper, Microsoft Excel does not simply generates fancy worksheets with a "pretty face." When it is used in conjunction with the SAS ODS Excel destination, the ODS TableEditor tagset, formulas, and cascading style sheets, Excel is a powerful tool that can help you answer your organization's most difficult business questions dynamically, and in a timely and efficient manner.

RESOURCES

Downloadable files that are discussed in this paper:
support.sas.com/rnd/base/ods/odsmarkup/tableeditor/tableeditor.zip

Eberhart, Peter and Louanna Kong. 2012 “The Armchair Quarterback: Writing SAS® Code for the Perfect Pivot (Table, That Is)." *Proceedings of the SAS Global Forum 2010 Conference*. Cary, NC: SAS Institute Inc. Available at **support.sas.com/resources/papers/proceedings12/146-2012.pdf**.

Parker, Chevell. 2010. “Using SAS® Output Delivery System (ODS) Markup to Generate Custom PivotTable and PivotChart Reports.” *Proceedings of the SAS Global Forum 2010 Conference*. Cary, NC: SAS Institute Inc. Available at **support.sas.com/resources/papers/proceedings10/003-2010.pdf**.

SAS Institute Inc. 2016. *SAS® Studio 3.5: Developer's Guide to Writing Custom Tasks*. Cary, NC: SAS Institute Inc. Available at **support.sas.com/documentation/cdl/en/webeditordg/68829/PDF/default/webeditordg.pdf**.

ACKNOWLEDGMENTS

I would like to thank Susan Berry for her help on this paper.

CONTACT INFORMATION

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

Chevell Parker
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Email: `support@sas.com`
Web: `support.sas.com`