SAS Global Forum 2008

March 16-19, 2008

Henry B. Gonzalez Convention Center

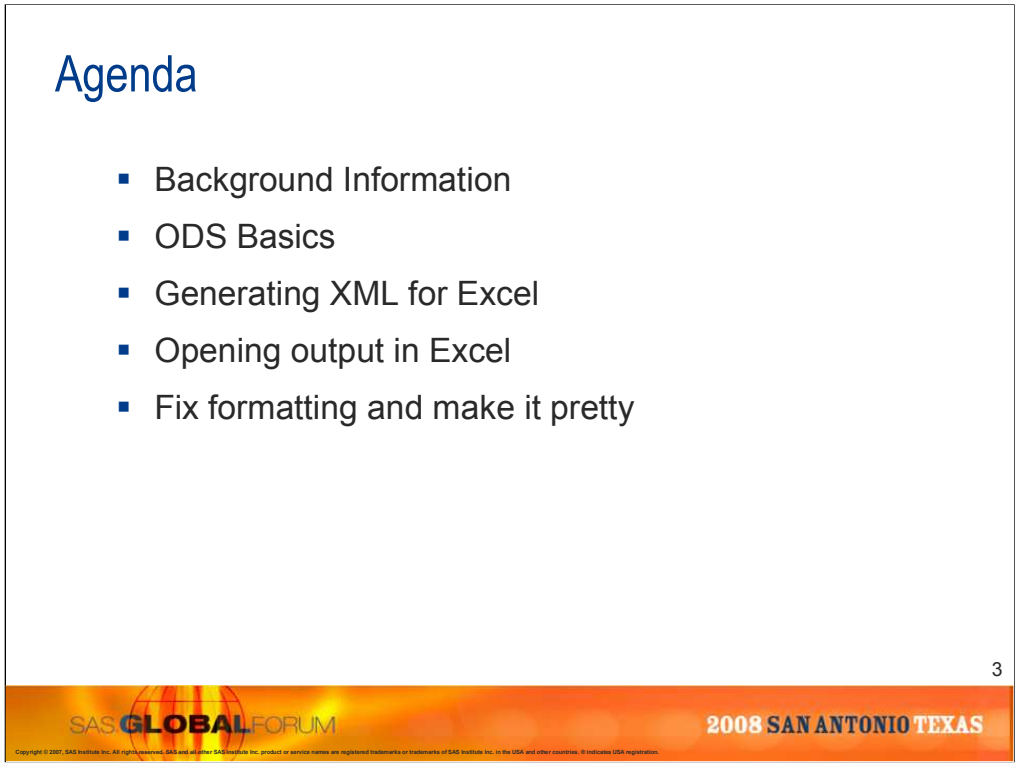
San Antonio, TX

A special *"thank you"* to Peter Eberhardt and Sue Douglass for inviting me to present this topic, and to Chris Barrett of SAS Institute Inc. for his valuable input on the accompanying paper.

Goals

- Give you something you can use TODAY
- Integrate SAS output w/ Excel

2



Software Requirements

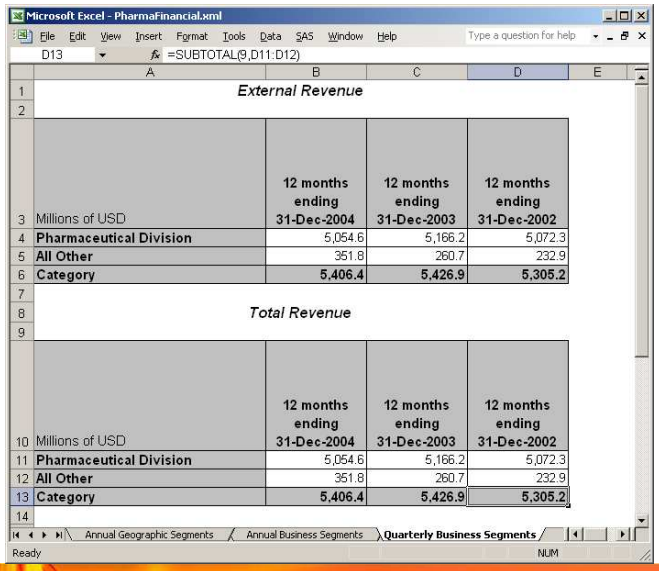
- Base SAS, ***any*** operating system.

- SAS 9.1.2 or later (9.1.3 provides greatly improved performance for the ExcelXP tagset).

- **Modified version** of the ExcelXP tagset (see paper for details).

- Microsoft Excel XP (a.k.a. Excel 2002) or later.

Sample SAS Output ⟶ Excel



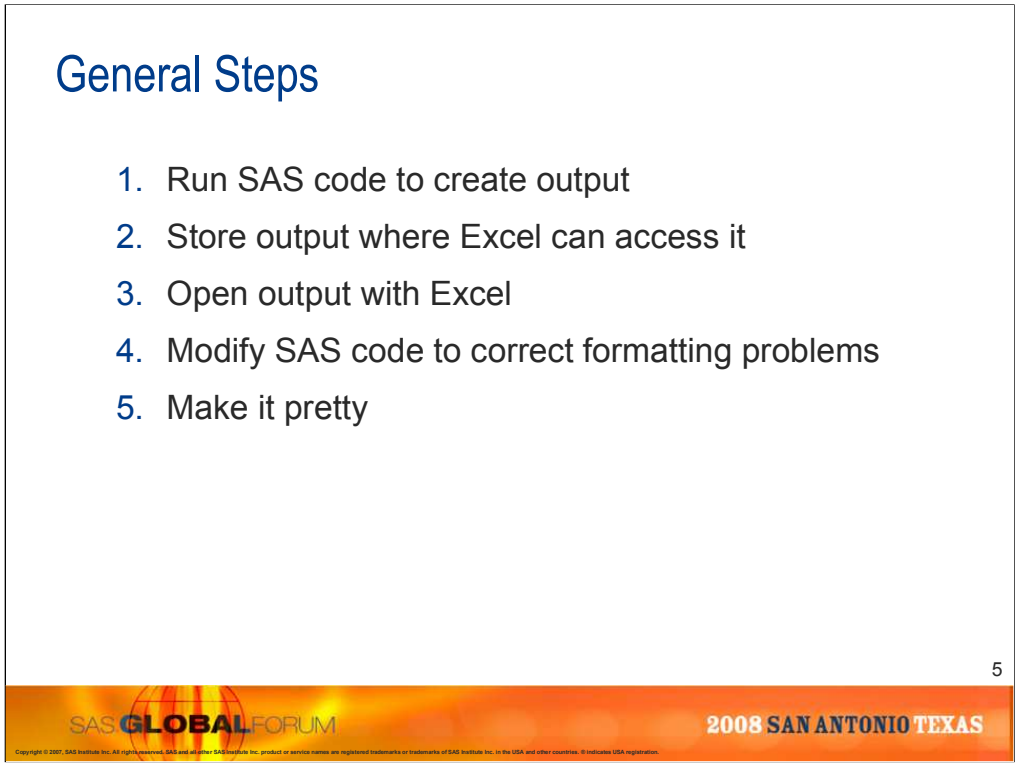
4

This illustrates the type of SAS output that we will be incorporating into Excel. It is financial data for a fictional pharmaceutical company.

This output was generated completely by ODS – there was no "hand-editing" of the Excel workbook. Furthermore, SAS can generate this type of output regardless of the platform – Windows, UNIX, OpenVMS, or z/OS.

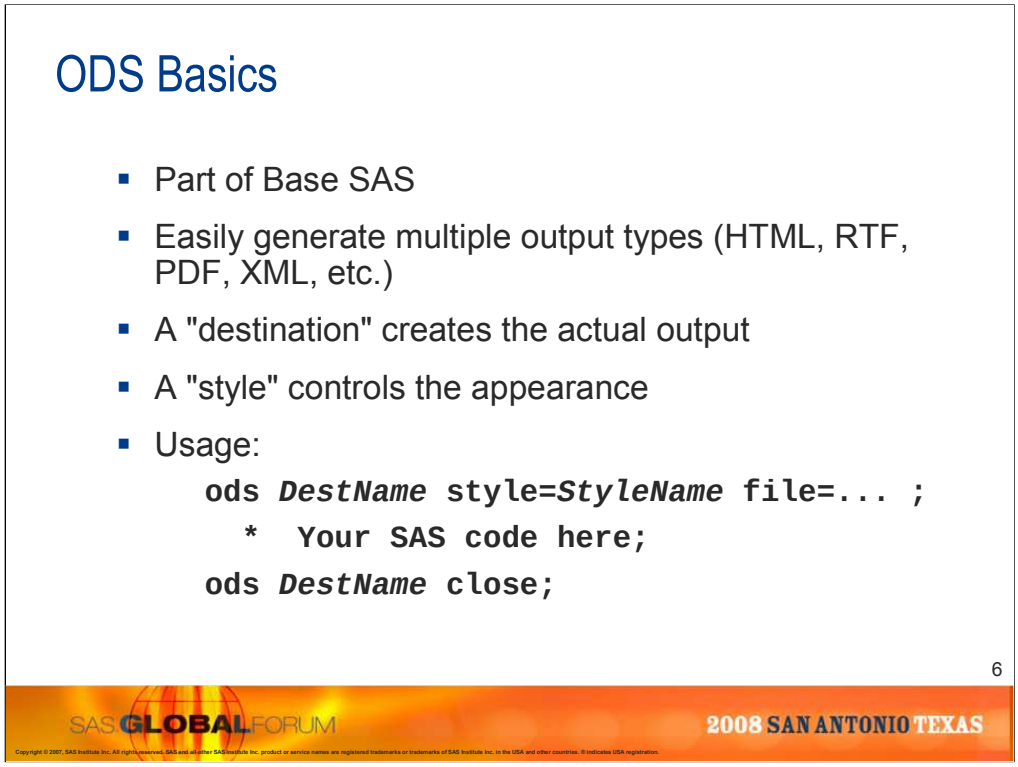
This Excel workbook has 3 worksheets, all created using the PRINT procedure.

Some things to note are the custom worksheet names, the presence of multiple Excel tables within a single worksheet and an Excel SUBTOTAL function used in the cells of the first summary row of each table.



We are going to use ODS to create XML files that are stored in a location where Excel can access them. In your production system, SAS and Excel may reside on two different machines. Thus, you may have to make use of network drives, FTP or some other means to move the SAS output to a location that Excel has access to.

Then the ODS output will be opened using Excel. If you have ever done this before, you have probably encountered formatting problems. We will fix those problems, and then explore techniques to instruct ODS to create output that Excel is happy with, and to make the output look pretty.



An ODS *destination* creates the actual output (HTML, RTF, PDF, XML, etc.) while an ODS *style* controls the appearance (colors, fonts, border lines, etc.).

Both a destination and a style are needed to generate output. If you do not specify a style, the style named "Default" will be used.

We will be using a special type of ODS destination called a "tagset". ODS tagsets can be modified to meet your specific needs using the TEMPLATE procedure. And you can use the TEMPLATE procedure to create your own tagsets.

ODS Basics – Output for Excel

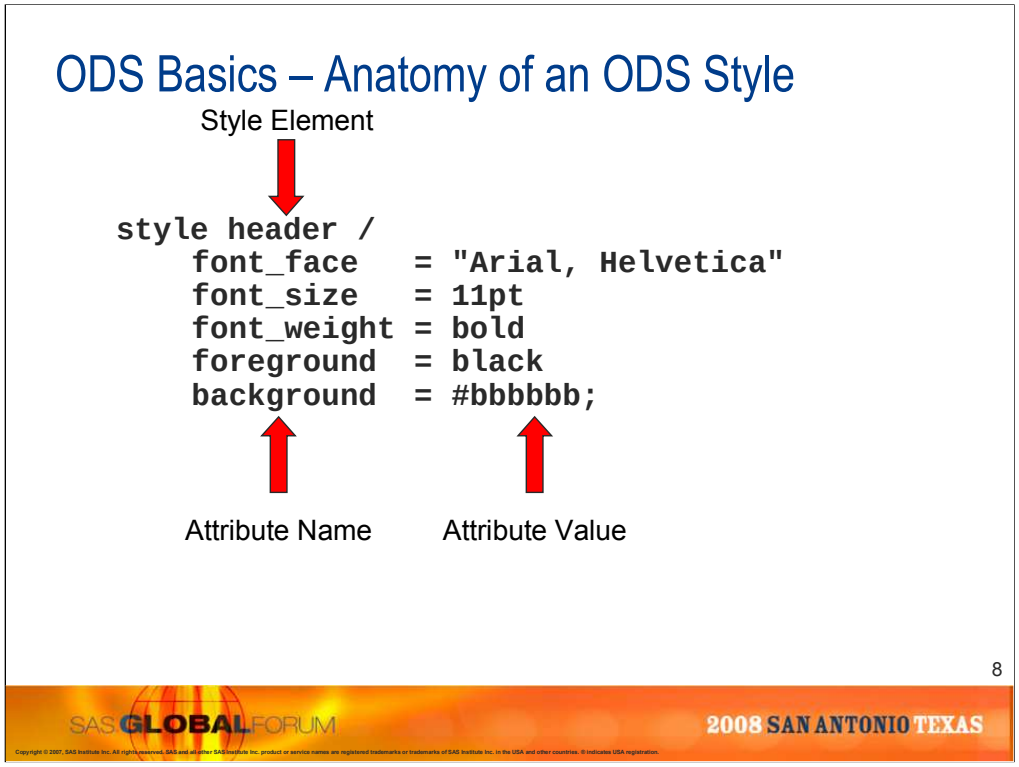
- Excel XP can open specially made XML files as multi-sheet workbooks (graphics not supported)
- Use the ExcelXP tagset and XLsansPrinter style:

```
ods listing close;
ods tagsets.ExcelXP style=XLsansPrinter
 file=... ;
   *  First PROC PRINT here;
   *  Second PROC PRINT here;
   *  Last PROC PRINT here;
ods tagsets.ExcelXP close;
```

7

Currently, the Microsoft XML Spreadsheet Specification does not support graphics. Thus, SAS/GRAPH procedures are not supported. This is a Microsoft Excel limitation, not a limitation of the ExcelXP tagset.

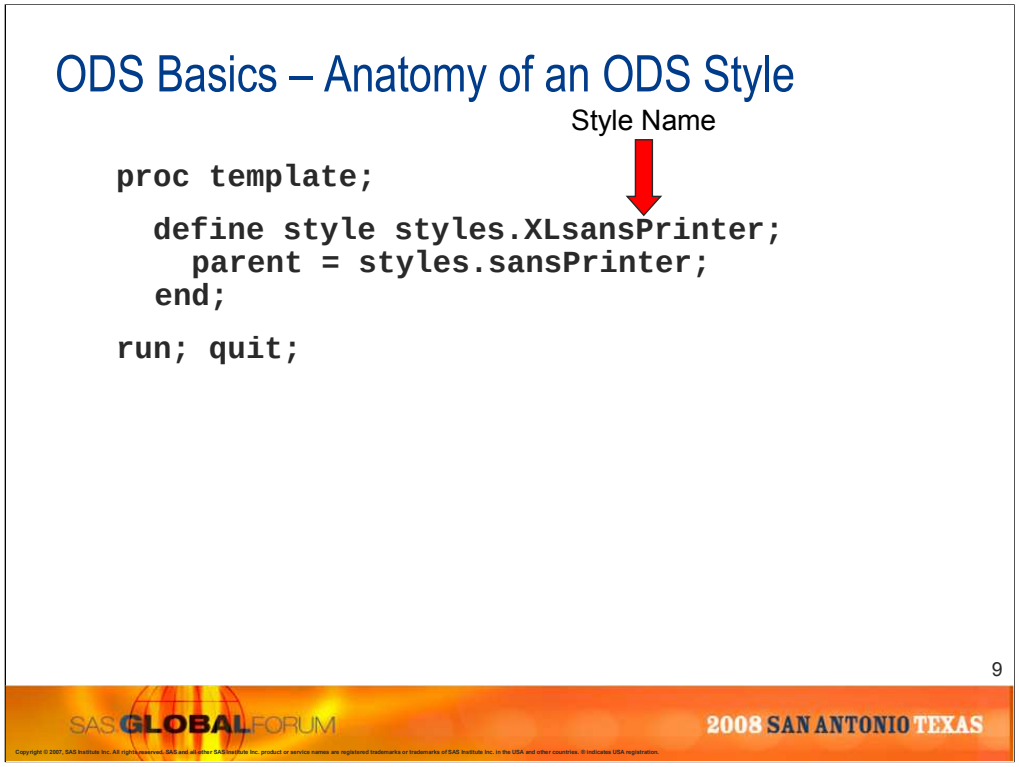
We will use the ODS tagset named ExcelXP to create XML output that can be opened using Microsoft Excel XP and later. When opened with Excel, the XML file will be rendered as a multi-sheet Excel workbook. Additionally, we will use an ODS style named XLsansPrinter to provide a look similar to the standard sansPrinter style that is shipped with Base SAS software.



A style is composed of *style elements,* each of which controls a particular part of the output. For example, all styles contain a style element named "header" that controls the appearance of column headings.

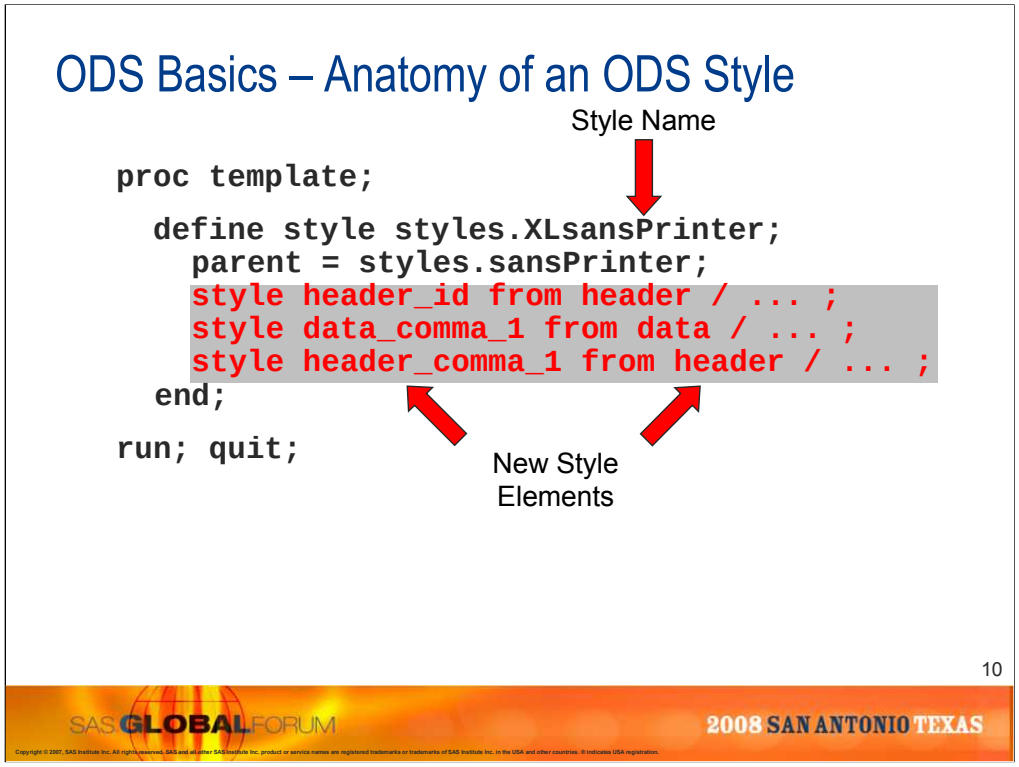
Style elements consist of collections of *style attributes*, such as the background color and font size. The definition of a style element named "header" might look what is show here.

You can use the TEMPLATE procedure, which is part of Base SAS, to change the attributes of style elements, create new style elements or even create completely new styles.



This is a partial listing of the TEMPLATE procedure code used to create the style named XLsansPrinter. As you can see, it is based on the sansPrinter style that is shipped with Base SAS.

At this point, the XLsansPrinter style is identical to the sansPrinter style. The XLsansPrinter style contains all the same style elements and style attributes as the sansPrinter style.

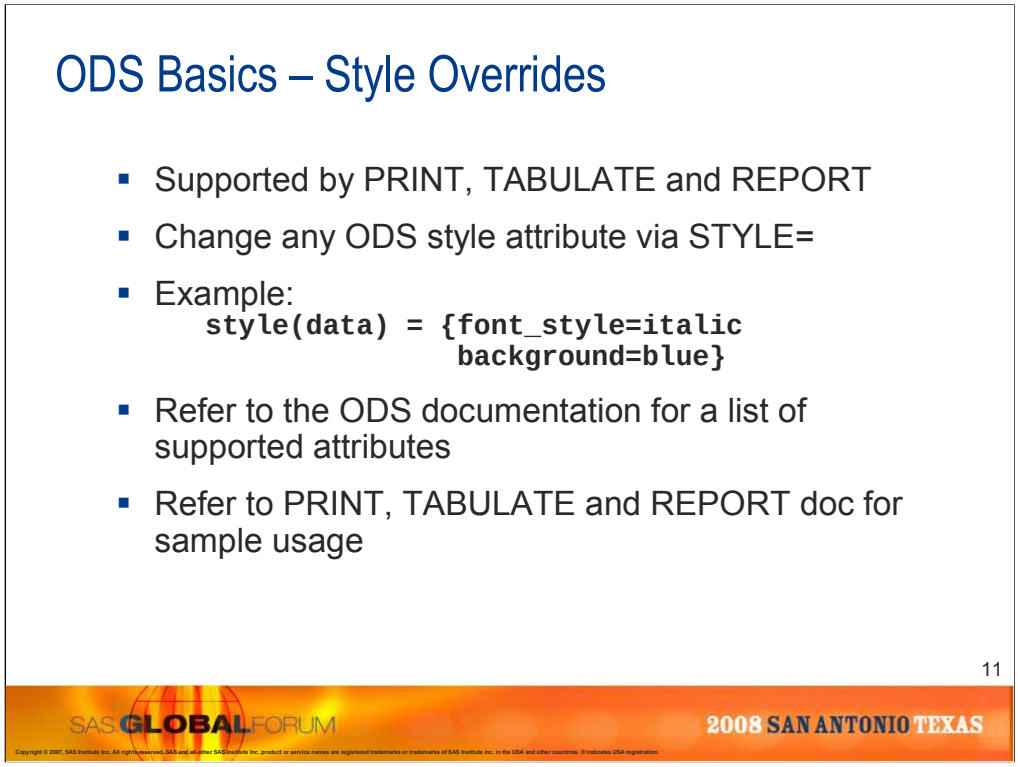


The XLsansPrinter style now contains 3 new style elements:

- header_id – based on the header style element
- data_comma_1 – based on the data style element
- header_comma_1 – based on the header style element

Because of their unique names, ODS does not use these style elements until they are specified by you via a style override.

Further information about the attributes associated with these style elements will be presented later.



ODS style overrides are used to override attributes of the ODS style you are using. They are intended to be used to make small changes to the appearance of your output, and should be used sparingly. You could use a style override to change the fonts or colors used by the XLsansPrinter style for the "data" location of PROC PRINT, as shown here. But there is a better way.

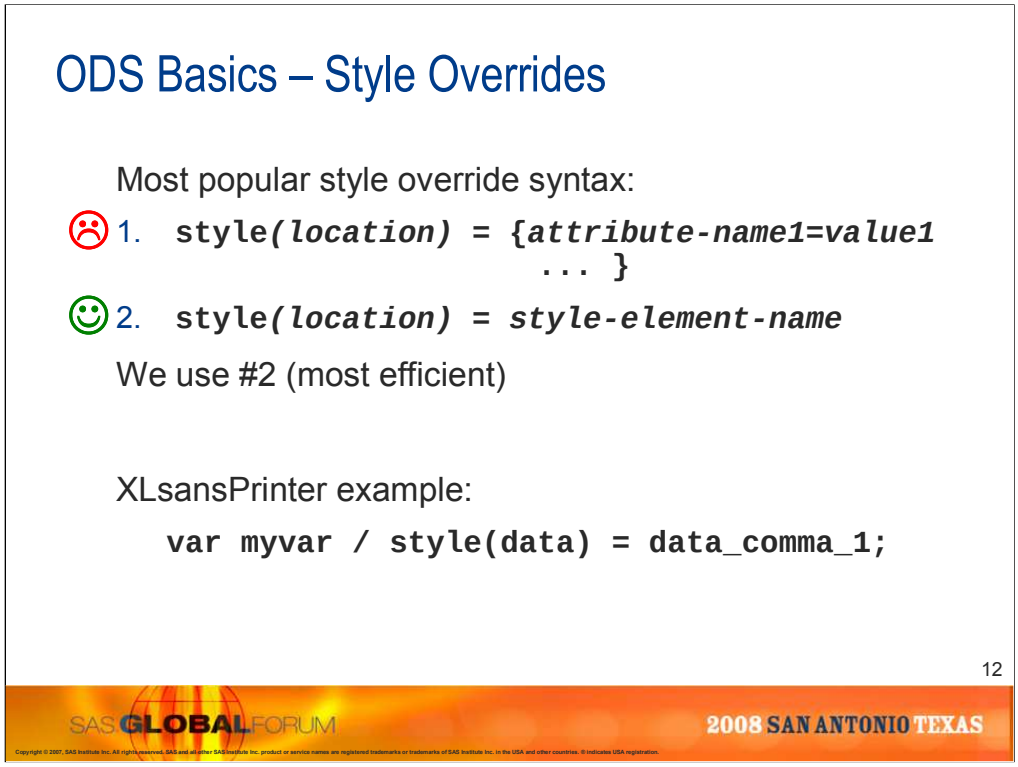
ODS Documentation:

"Chapter 9: TEMPLATE Procedure: Creating a Style Definition", *SAS 9.1 Output Delivery System, User's Guide* (www.sas.com/apps/pubscat/bookdetails.jsp?catid=1&pc=58966)

Also available at:

http://support.sas.com/onlinedoc/913/docMainpage.jsp?_topic=odsug.hlp/a002565239.htm

(ignore wrapping in URL above)



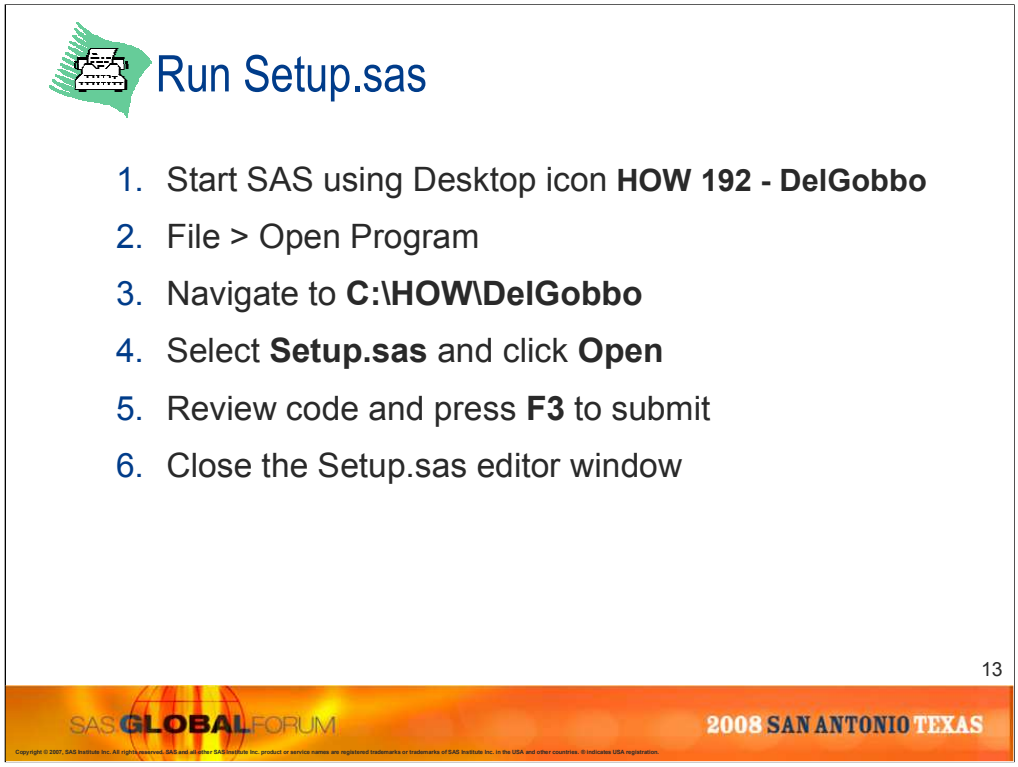
Above you see the syntax for using style overrides.

The first form is what you might use to change the fonts or colors used by a style element, as illustrated on the previous slide. While this form is the one that is most commonly used, it is also the least efficient. That is because, if overused, it can cause ODS to create files that are much larger than they would be if the style override was not used.

The second form is the most efficient, and is the one that we will use. It is used to associate a pre-defined ODS style element with a particular part of the output. You create new style elements with the desired attributes, then reference the style element names. This is what we will do using the XLsansPrinter style.

The XLsansPrinter style defined in the file "Setup.sas" contains a number of different style elements. Later, we will associate these style elements with discrete parts of the SAS output ("locations").

Locations will be discussed later.



The SAMPDIR global macro variable specifies the directory containing our sample code and data, as well as the ODS-generated XML output.

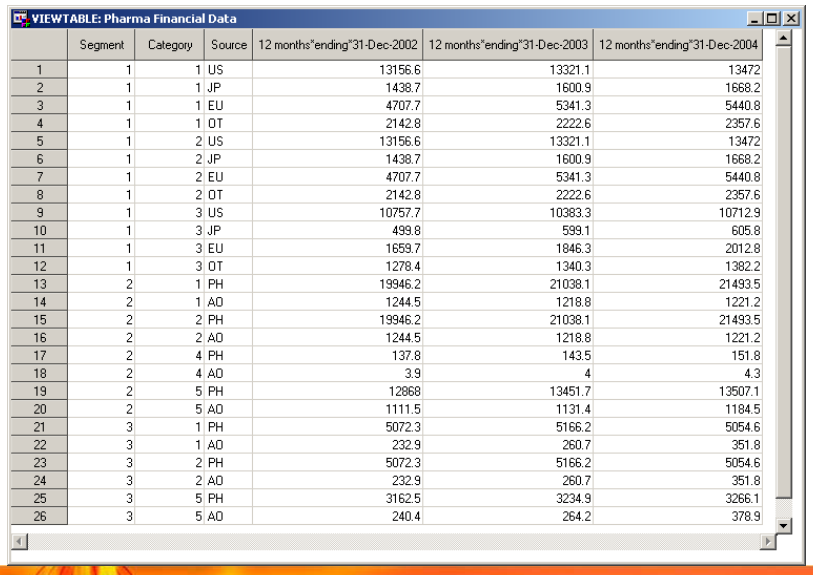
The program assigns a SAS library (SAMPLE) for the input data and one (MYLIB) for the output ODS tagsets and styles that we will be creating. While it is OK to use the WORK or SASUSER libraries to temporarily store custom tagsets and styles, it is a good idea use a different permanent library that is publicly accessible if you want to make them available to others.

The ODS PATH statement controls the search and storage locations for styles and tagsets. Thus, MYLIB will be searched/used first. The ODS tagset "ExcelXP" has undergone some important changes since SAS 9.1 was released, so we will import a newer copy and store it in the MYLIB library. Be sure to use a recent version of the ExcelXP tagset in your production code.

The FORMAT procedure is used to create formats for our sample data. In a production environment, you would store these formats in a format library, and simply reference them in your code using the FMTSEARCH system option.

Finally, the XLsansPrinter style, which produces output similar in appearance to the standard sansPrinter style, is created. Note the definition of new style elements "header_id", "data_comma_1" and "header_comma_1".

Preview Sample SAS Data



VIEWTABLE: Pharma Financial Data

	Segment	Category	Source	12 months*ending*31-Dec-2002	12 months*ending*31-Dec-2003	12 months*ending*31-Dec-2004
1	1	1	US	13156.6	13321.1	13472
2	1	1	JP	1438.7	1600.9	1668.2
3	1	1	EU	4707.7	5341.3	5440.8
4	1	1	OT	2142.8	2222.6	2357.6
5	1	2	US	13156.6	13321.1	13472
6	1	2	JP	1438.7	1600.9	1668.2
7	1	2	EU	4707.7	5341.3	5440.8
8	1	2	OT	2142.8	2222.6	2357.6
9	1	3	US	10757.7	10383.3	10712.9
10	1	3	JP	499.8	599.1	605.8
11	1	3	EU	1659.7	1846.3	2012.8
12	1	3	OT	1278.4	1340.3	1382.2
13	2	1	PH	19946.2	21038.1	21493.5
14	2	1	AO	1244.5	1218.8	1221.2
15	2	2	PH	19946.2	21038.1	21493.5
16	2	2	AO	1244.5	1218.8	1221.2
17	2	4	PH	137.8	143.5	151.8
18	2	4	AO	3.9	4	4.3
19	2	5	PH	12868	13451.7	13507.1
20	2	5	AO	1111.5	1131.4	1184.5
21	3	1	PH	5072.3	5166.2	5054.6
22	3	1	AO	232.9	260.7	351.8
23	3	2	PH	5072.3	5166.2	5054.6
24	3	2	AO	232.9	260.7	351.8
25	3	5	PH	3162.5	3234.9	3266.1
26	3	5	AO	240.4	264.2	378.9

14

This SAS table contains the raw data that we will be using.

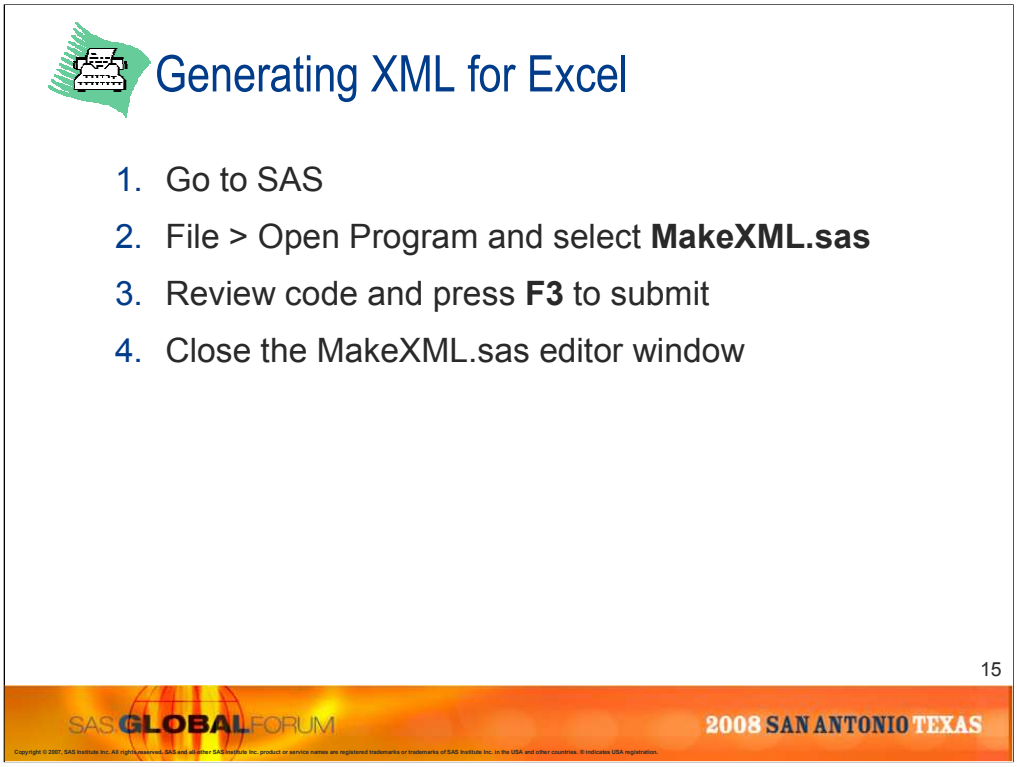
The column "Segment" represents the aggregation level for the various business segments, for example on a quarterly or annual basis. We will have one worksheet for each value of this column.

The column "Category" corresponds to the category of financial data, such as external revenue, long-lived assets or depreciation.

The "Source" column represents the country or division that the data pertains to.

The remaining 3 columns show the monetary values as of December 31, 2002, 2003 and 2004, respectively.

SAS formats will be applied to the first 3 columns to display more meaningful values.

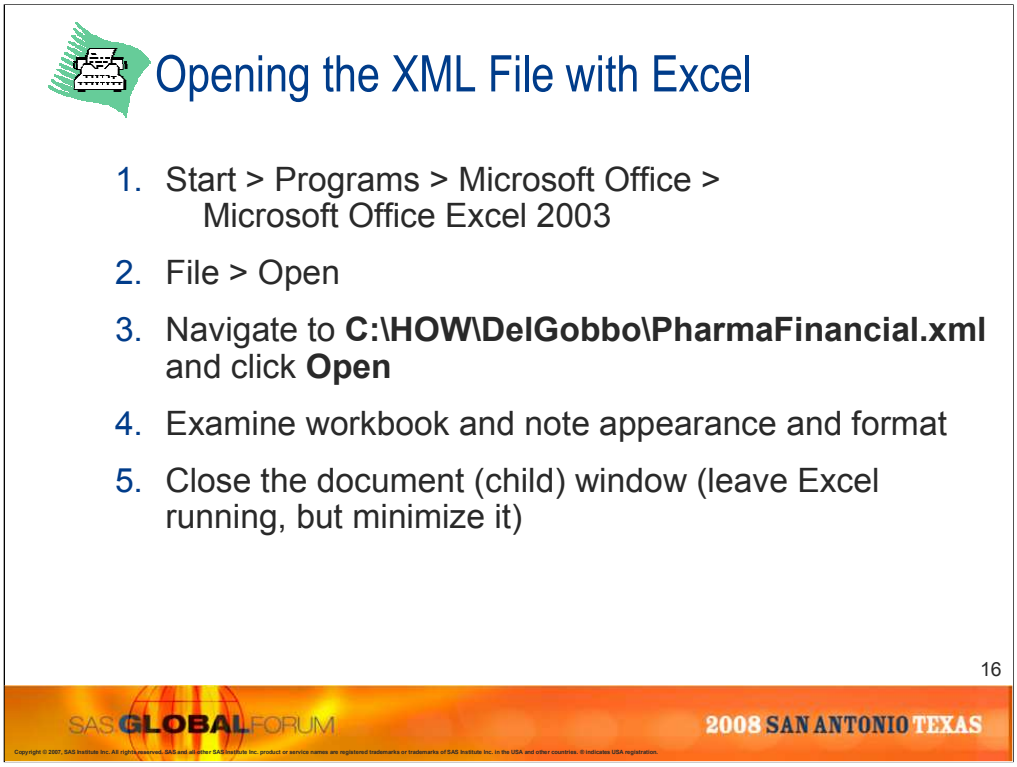


This is our first attempt to make an XML file that can be opened with Excel.

The ExcelXP tagset is used to generate XML output, and the XLsansPrinter style controls the appearance of the output.

The XML file created by ODS will be stored in a file named "PharmaFinancial.xml", residing in the directory specified by the SAMPDIR macro variable.

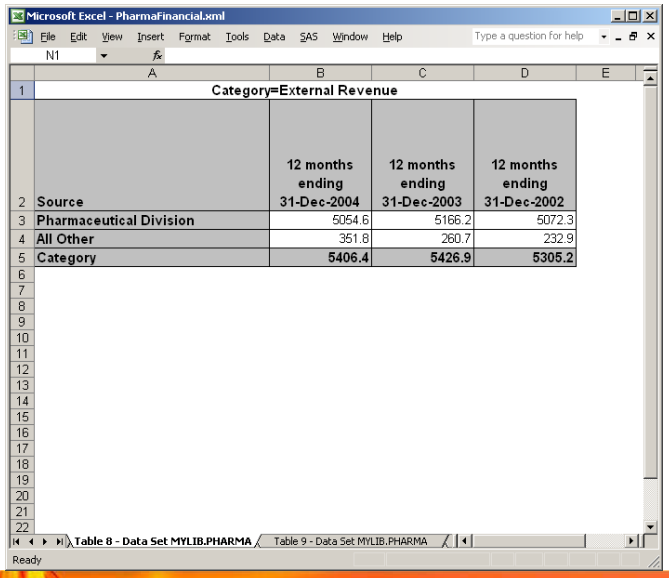
The PRINT procedure is run 3 times, once for each distinct value of "Segment", in order to produce 1 worksheet for each value of "Segment".



The XML file created by ODS can now be opened with Microsoft Excel.

When Excel reads the XML file, it renders it as a multi-sheet workbook. Thus you can use all the editing and formatting features of Excel to modify the output and save it as a native Excel workbook in binary format using Excel 2002 or 2003 (File > Save As and choose **Microsoft Excel Workbook (*.xls)**)

XML Output Viewed Using Excel



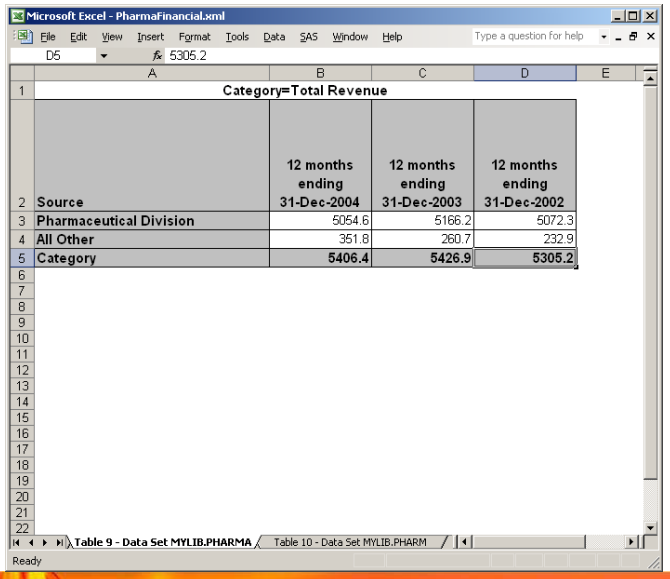
	A	B	C	D	E
1	Category=External Revenue				
2	Source	12 months ending 31-Dec-2004	12 months ending 31-Dec-2003	12 months ending 31-Dec-2002	
3	Pharmaceutical Division	5054.6	5166.2	5072.3	
4	All Other	351.8	260.7	232.9	
5	Category	5406.4	5426.9	5305.2	

17

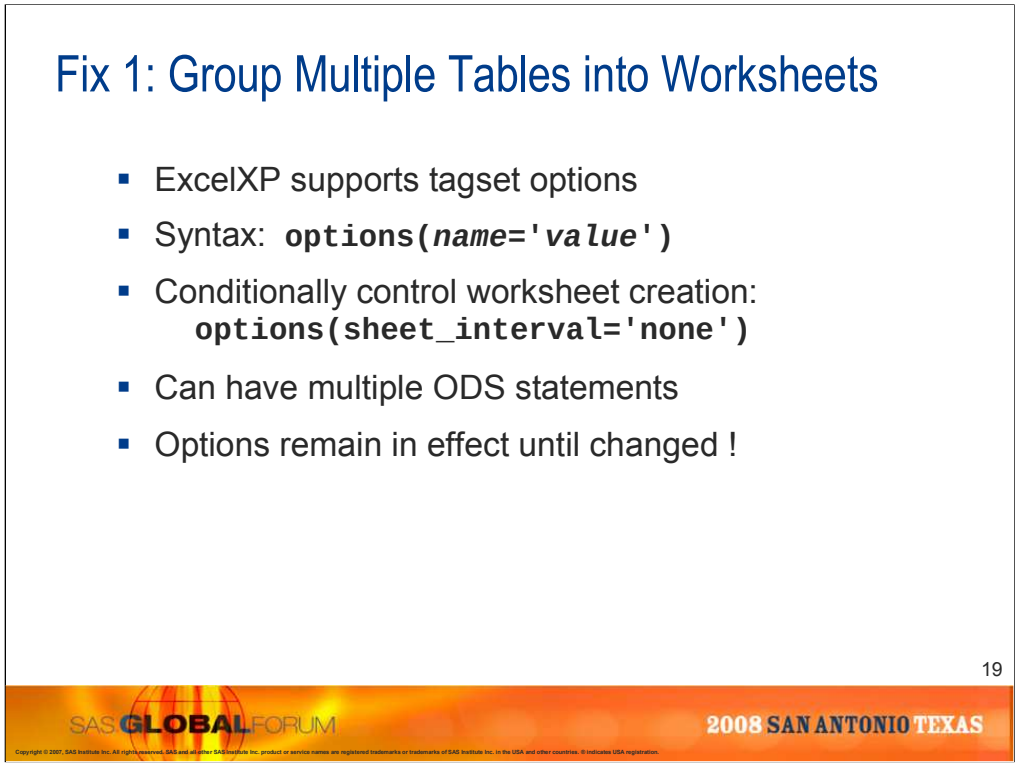
Problems with the output:

1. The data for each BY group is in a separate worksheet, resulting in a total of 10 worksheets, instead of 3.
2. Default worksheet names were used, instead of the name of the corresponding business segment.
3. Standard BY group text precedes the Excel tables, instead of custom titles.
4. An Excel SUBTOTAL function is not used in the summary cells. For example, cell D5 contains the static text "5305.2" instead of the appropriate Excel formula.
5. Numeric values are not formatted using a comma for the thousands separator.
6. The heading for the "Source" column should be "Millions of USD" in light weight text.

XML Output Viewed Using Excel



Another worksheet in the same workbook.



The ExcelXP tagset supports a number of different options that control both the appearance and functionality of the Excel workbook. You simply add an OPTIONS keyword to the ODS statement, as shown above.

The SHEET_INTERVAL option can be used to control when the tagset creates a new worksheet. Setting this option to "none" serves two purposes: (1) it suppresses the creation of new worksheets and (2) if a worksheet is currently being created, it "closed", and subsequent output is directed to a new worksheet.

You can have more than one ODS statement. This is important, because later we will use an ODS statement to set "global" options that affect all worksheets.

IMPORTANT NOTE: Tagset options remain in effect until they are changed !

Fix 1: Group Multiple Tables into Worksheets

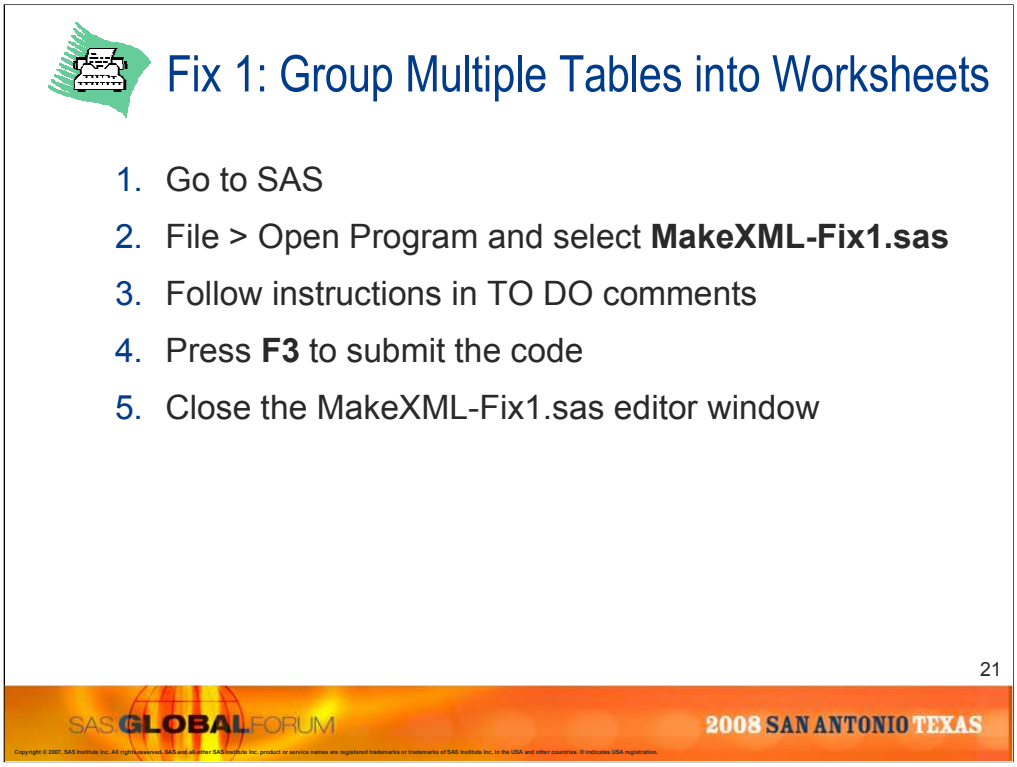
```
ods tagsets.ExcelXP style=XLsansPrinter file=... ;
  ods tagsets.ExcelXP
      options(sheet_interval='none');
  *  First PROC PRINT here;
  ods tagsets.ExcelXP
      options(sheet_interval='none');
  *  Second PROC PRINT here;
  ods tagsets.ExcelXP
      options(sheet_interval='none');
  *  Last PROC PRINT here;
ods tagsets.ExcelXP close;
```

20

The first instance of the SHEET_INTERVAL option begins the creation of the first worksheet, and the output from the first PRINT procedure code is directed to this worksheet.

When the option is encountered the second time, the first worksheet is closed and a new worksheet is created containing the output from the second PRINT procedure code.

Similarly, the third instance of the SHEET_INTERVAL option results in the creation of the third and final worksheet.



TO DO: Add `sheet_interval='none'` to the new ODS statements:

Line 22

Line 32

Line 42

Fix 1: Group Multiple Tables into Worksheets

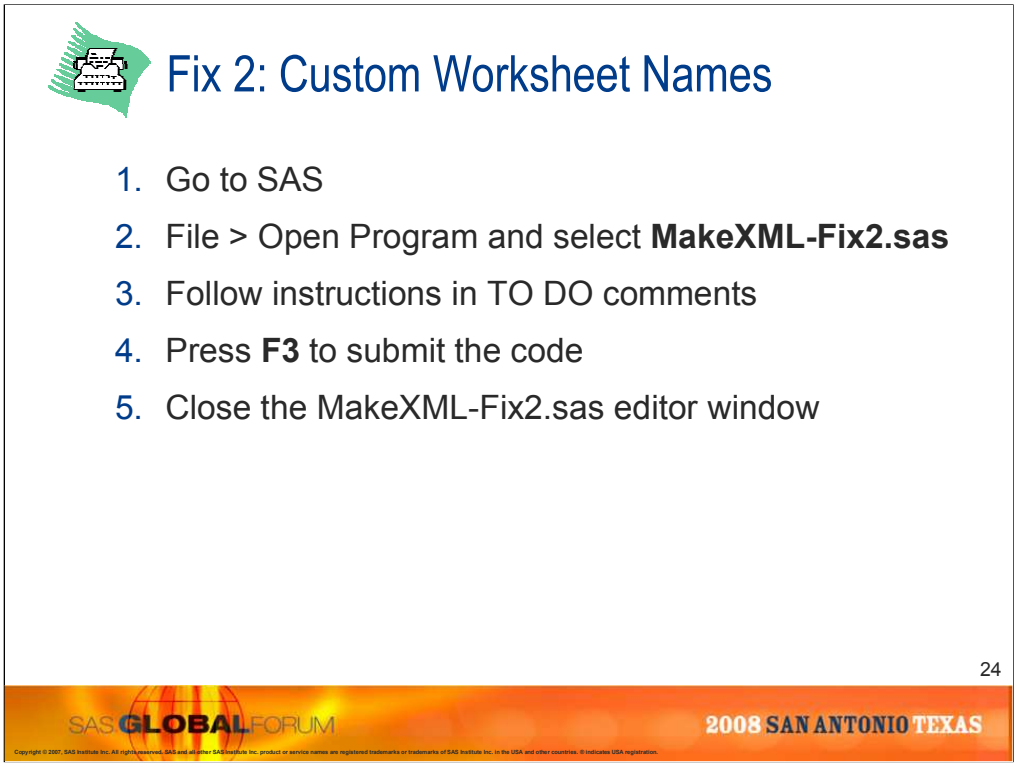
1. Go to Excel
2. File > **\HOW\DelGobbo\PharmaFinancial.xml**
   - or -
   PharmaFinancial.xml (from the recent file list)
3. Note only 3 worksheets now w/ new names
4. Close the document (child) window (leave Excel running, but minimize it)

Fix 2: Custom Worksheet Names

```
ods tagsets.ExcelXP
      options(sheet_interval='none'
              sheet_name='Annual Geographic Segments');

  *  First PROC PRINT here;

  ods tagsets.ExcelXP
       options(sheet_interval='none'
              sheet_name='Annual Business Segments');

  *  Second PROC PRINT here;

  ods tagsets.ExcelXP
       options(sheet_interval='none'
              sheet_name='Quarterly Business Segments');

  *  Last PROC PRINT here;
```

23

If you don’t like the default worksheet names that ODS chooses, use the SHEET_NAME option to change them.



TO DO: Add SHEET_NAME option to new ODS statements:

Line 24: `sheet_name='Annual Geographic Segments'`

Line 35: `sheet_name='Annual Business Segments'`

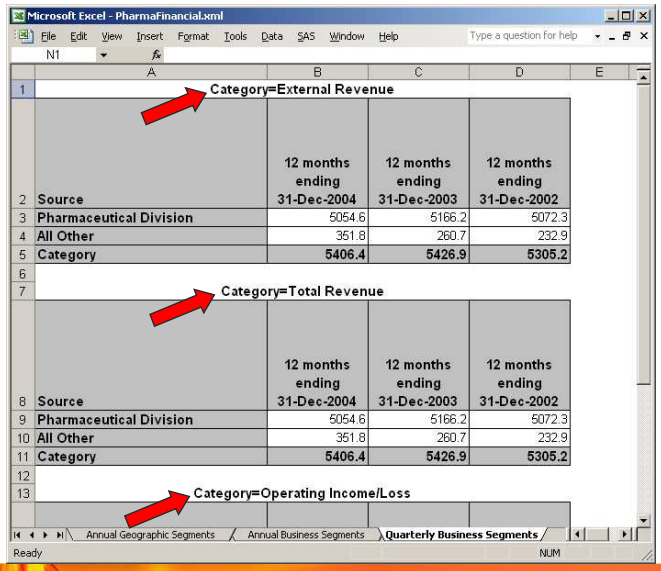
Line 46: `sheet_name='Quarterly Business Segments'`

Fix 2: Custom Worksheet Names

1. Go to Excel
2. File > **\HOW\DelGobbo\PharmaFinancial.xml**
   - or -
   PharmaFinancial.xml (from the recent file list)
3. Note custom worksheet names
4. Close the document (child) window (leave Excel running, but minimize it)

Fix 3: Titles Instead of BY Line Text

Have:



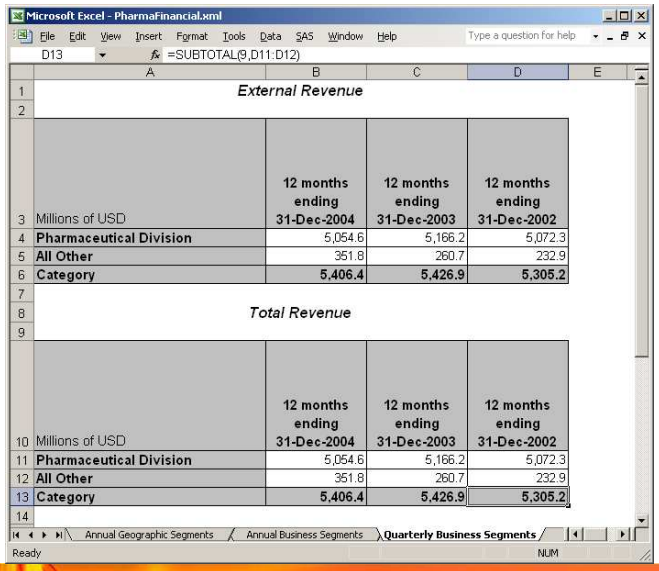
Source	12 months ending 31-Dec-2004	12 months ending 31-Dec-2003	12 months ending 31-Dec-2002
Pharmaceutical Division	5054.6	5166.2	5072.3
All Other	351.8	260.7	232.9
Category	5406.4	5426.9	5305.2

Source	12 months ending 31-Dec-2004	12 months ending 31-Dec-2003	12 months ending 31-Dec-2002
Pharmaceutical Division	5054.6	5166.2	5072.3
All Other	351.8	260.7	232.9
Category	5406.4	5426.9	5305.2

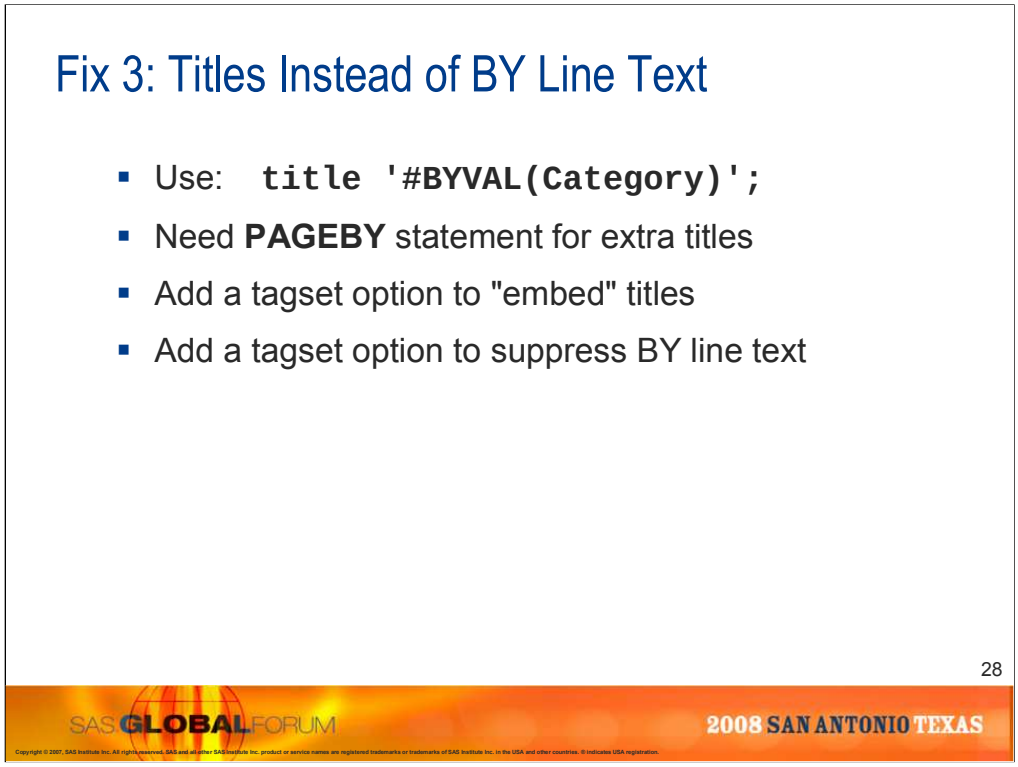
BY line text such as "Category=External Revenue" and "Category=Total Revenue" precedes each table in our workbook. We would like to get rid of the text "Category=".

Fix 3: Titles Instead of BY Line Text

Want:



Titles replace the BY line text.



To insert the current value of the "Category" BY variable into the title, specify `#BYVAL(Category)` in the text of the TITLE statement.

The PRINT procedure places a tile at the top of each page. To get a title before each BY group, add a PAGEBY statement to the PROC PRINT code.

By default, SAS titles and footnotes appear as Excel print headers and print footers, which are displayed only when the Excel document is printed. To include the titles in the worksheets, use the EMBEDDED_TITLES tagset option.

The SUPPRESS_BYLINES option can be used to prevent the BY line text from being included in the worksheet.

TITLE Statement documentation (see #BYVAL):

http://support.sas.com/onlinedoc/913/docMainpage.jsp?_topic=lrdict.hlp/a000220968.htm

(ignore wrapping in URL above)

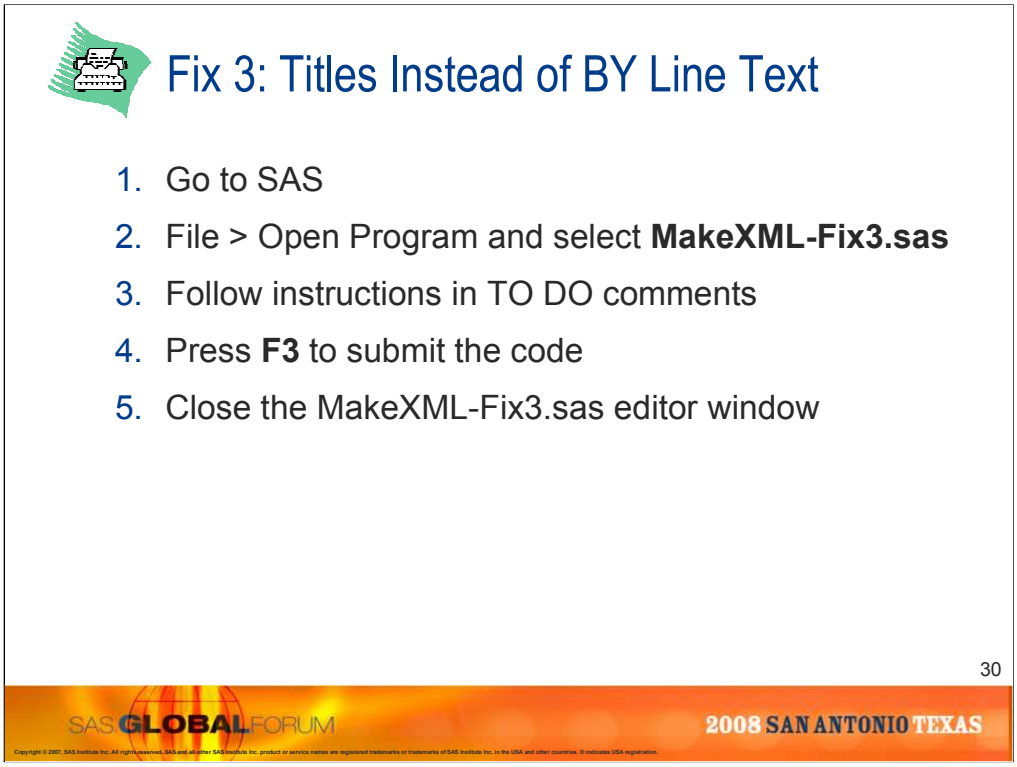
Fix 3: Titles Instead of BY Line Text

```
ods tagsets.ExcelXP style=XLsansPrinter file=... ;
  title '#BYVAL(Category)';
  footnote;
  *  Set global options;
  ods tagsets.ExcelXP options(embedded_titles='yes'
                              suppress_bylines='yes');
  ods tagsets.ExcelXP
      options(sheet_interval='none' sheet_name=... );
  *  Remainder of code with PAGEBY statements;
ods tagsets.ExcelXP close;
```

29

Note that an additional ODS statement was added to handle "global" options. These are options that will be set one time, and remain in effect until the destination is closed.

These options could have added to an existing ODS statement, but a new statement increases readability and also separates global options from options that pertain to a specific worksheet.



TO DO:

Line 22: Add `'#BYVAL(Category)'` to the TITLE statement

Line 27: Add `embedded_titles='yes' suppress_bylines='yes'` to the new ODS statement

Lines 37, 49 and 61: Note the addition of the PAGEBY statements to the PROC PRINT code.

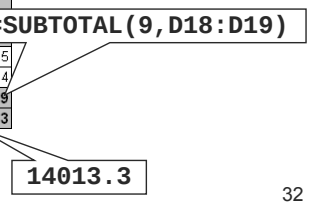
Fix 3: Titles Instead of BY Line Text

1. Go to Excel
2. File > **\HOW\DelGobbo\PharmaFinancial.xml**
   - or -
   PharmaFinancial.xml (from the recent file list)
3. Note titles instead of BY line text
4. Close the document (child) window (leave Excel running, but minimize it)

Fix 4: Subtotal Function for Sums

- SUM statement sums columns
- AUTO_SUBTOTALS option inserts function
- Only useful for first summary line
- Only supported by PROC PRINT

Source	12 months ending 31-Dec-2004	12 months ending 31-Dec-2003	12 months endin 31-Dec-2
Pharmaceutical Division	3266.1	3234.9	3162.5
All Other	378.9	264.2	240.4
Category	3645	3499.1	3402.9
	14457.8	14352.9	14013.3



32

The AUTO_SUBTOTALS option causes the ExcelXP tagset to insert an Excel SUBTOTAL function into the summary cell when the SUM statement is used with PROC PRINT.

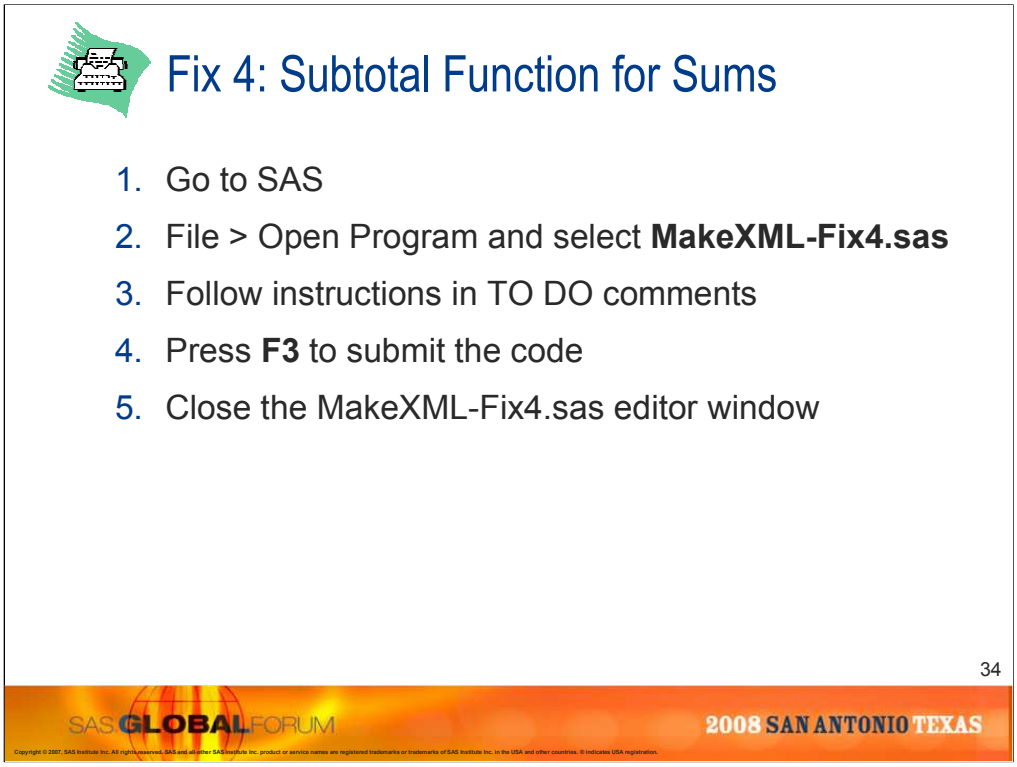
This option works best when a BY statement is not being used. When a BY statement is used, the SUBTOTAL function is correctly used in the first summary row, but not in subsequent summary rows that represent grand totals or multiple BY variables.

Fix 4: Subtotal Function for Sums

```
ods tagsets.ExcelXP style=XLsansPrinter file=... ;
  title '#BYVAL(Category)';
  footnote;
  *  Set global options;
  ods tagsets.ExcelXP options(embedded_titles='yes'
                              suppress_bylines='yes'
                              auto_subtotals='yes');
  *  Remainder of code;
ods tagsets.ExcelXP close;
```

33

Add the AUTO_SUBTOTALS option to the existing global options.



TO DO:

Line 24: Add `auto_subtotals='yes'`

Fix 4: Subtotal Function for Sums

1. Go to Excel
2. File > **\HOW\DelGobbo\PharmaFinancial.xml**
   - or -
   PharmaFinancial.xml (from the recent file list)
3. Note subtotals in first summary row of each table
4. Close the document (child) window (leave Excel running, but minimize it)

35

Fix 5: Tweak the Appearance

Have – Wrong column heading and no commas

Source	12 months ending 31-Dec-2004	12 months ending 31-Dec-2003	12 months ending 31-Dec-2002
Pharmaceutical Division	3266.1	3234.9	3162.5
All Other	378.9	264.2	240.4
Category	3645	3499.1	3402.9
	14457.8	14352.9	14013.3

36

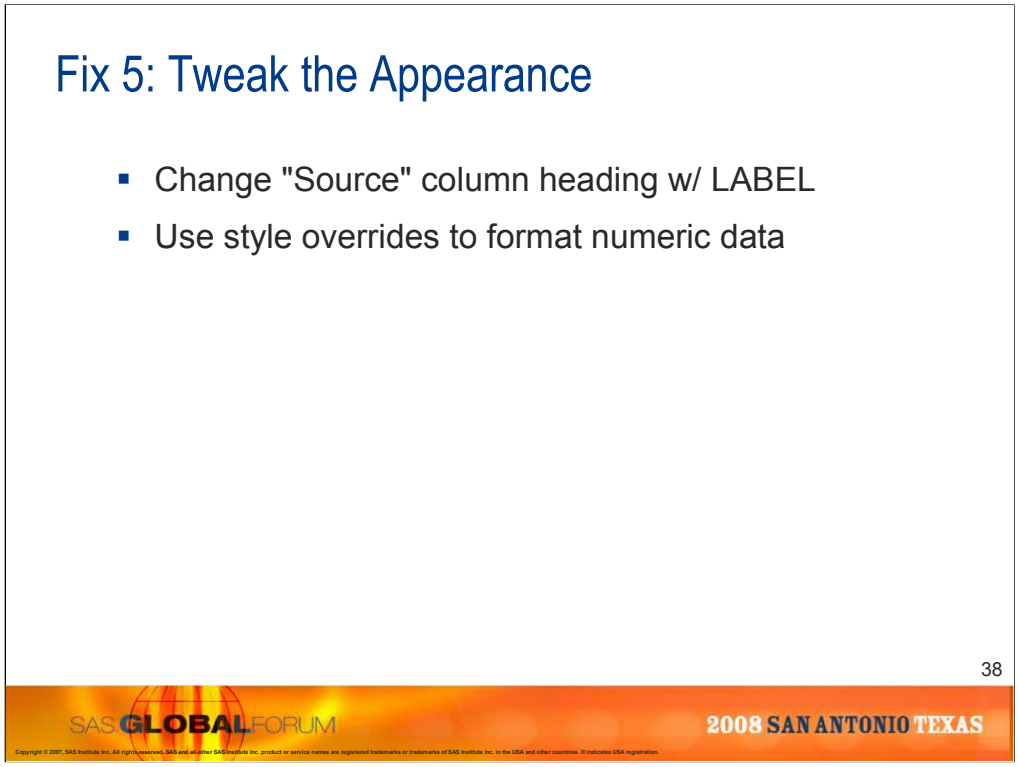
The heading for the "Source" column should be "Millions of USD" in light weight text.

All numeric values should be formatted with a comma for a thousands separator, and 1 decimal place, even if there is no decimal value in the raw data (similar to the SAS COMMAw.1 format).

Fix 5: Tweak the Appearance

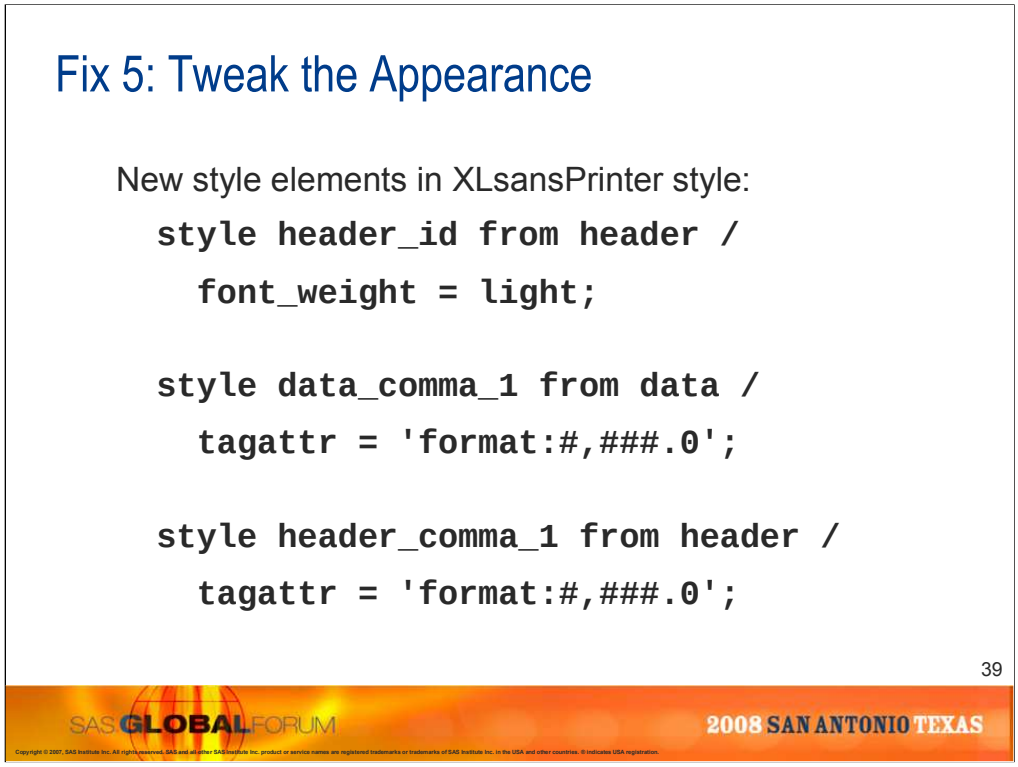
Want – Correct column heading and commas

Millions of USD	12 months ending 31-Dec-2004	12 months ending 31-Dec-2003	12 months ending 31-Dec-2002
Pharmaceutical Division	3,266.1	3,234.9	3,162.5
All Other	378.9	264.2	240.4
Category	**3,645.0**	**3,499.1**	**3,402.9**
	14,457.8	**14,352.9**	**14,013.3**



It is a simple matter to add a LABEL statement to the PRINT procedure code to change the label for the "Source" column to "Millions of USD". An ODS style override can be used to change the font weight of the text from bold to light.

ODS style overrides will be used to apply an Excel format that is similar to the SAS COMMAw.1 format. In general, it is best to rely on Excel formats, rather than SAS formats.



As mentioned earlier, ODS styles control the appearance of your output. Here you see a subset of the code used to create the XLsansPrinter style. Refer to the file "Setup.sas" for a complete listing of the code.

The style element "header_id" was created based on the standard style element named "header". The "header_id" style element has all the same style attributes as its parent style element ("header"), except the font weight will not be bold. This style element will be used to control the appearance of the column header for the ID variable ("Source").

The style elements "data_comma_1" and "header_commma_1" were created based on the standard style elements named "data" and "header", respectively. These style elements contain all the same style attributes as their parents, and have an additional style attribute named "tagattr". This attribute is used to apply an Excel format that causes data to be presented similarly to the SAS COMMAw.1 format.

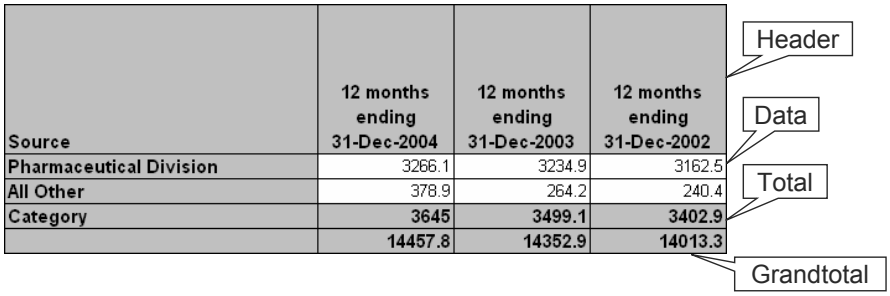
The Excel format is similar to the SAS COMMAw.1 format. Numeric values will always contain 1 decimal place, and will padded with a zero if there is no decimal value in the raw data. If you want to display decimal places only when they are non-zero, change the "0" to "#". To omit decimal places, remove the "0". These style elements will be used to control the appearance of all numeric data.

To find out more about Excel formats, refer to a book on Excel or type "create a custom number format" into the Excel help system.

Fix 5: Tweak the Appearance

Style override syntax:
`style(location)=style-element-name`

Style locations for PROC PRINT



Source	12 months ending 31-Dec-2004	12 months ending 31-Dec-2003	12 months ending 31-Dec-2002
Pharmaceutical Division	3266.1	3234.9	3162.5
All Other	378.9	264.2	240.4
Category	3645	3499.1	3402.9
	14457.8	14352.9	14013.3

40

When you apply style overrides, you must tell ODS what part of the output you want to apply the style element to. This part of the output is referred to as the "location".

Here you can see some of the locations for PRINT procedure output.

To change the appearance of the text in the column header for the "Source" column, we will apply a style override to the "Header" location for this column.

Similarly, we will apply style overrides to the "Data", "Total" and "Grandtotal" locations to assign Excel formats that will cause the numeric data to be displayed with a comma for a thousands separator and 1 decimal place.

Fix 5: Tweak the Appearance

Style Locations and Their Style Elements

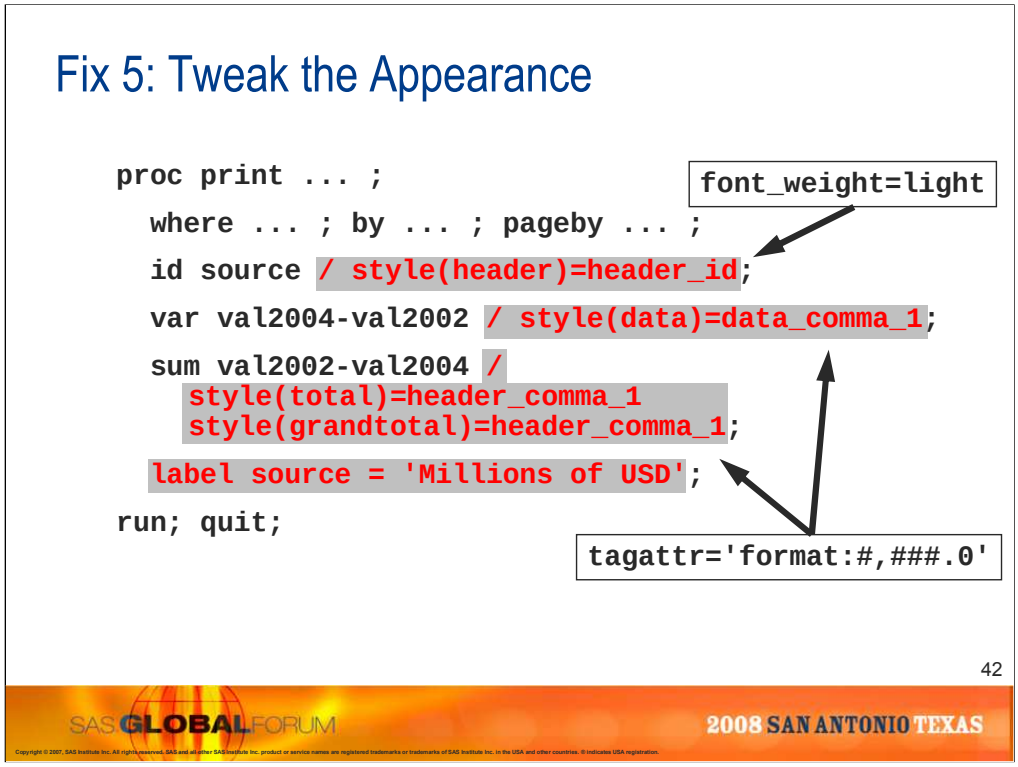
Location Name	Default Style Element	Overridden Style Element
Header	header	header_id
Data	data	data_comma_1
Total	header	header_comma_1
Grandtotal	header	header_comma_1

41

This table shows the default style elements that are applied to various locations of PRINT procedure output.

By default, the "header" style element controls the appearance the "Header", "Total" and "Grandtotal" locations, and the "data" style element applies to the "Data" location.

We will use style overrides to change the style elements used in these locations.



Adding these style overrides will cause the column heading for the "Source" column to be displayed with a light weight font, instead of bold.

Additionally, all numeric values will be displayed with 1 decimal place, and a comma for the thousands separator.

TO DO:

Line 40: Add style element name header_id

Line 41: Add style element name data_comma_1

Line 42: Add style element name header_comma_1 (two times)

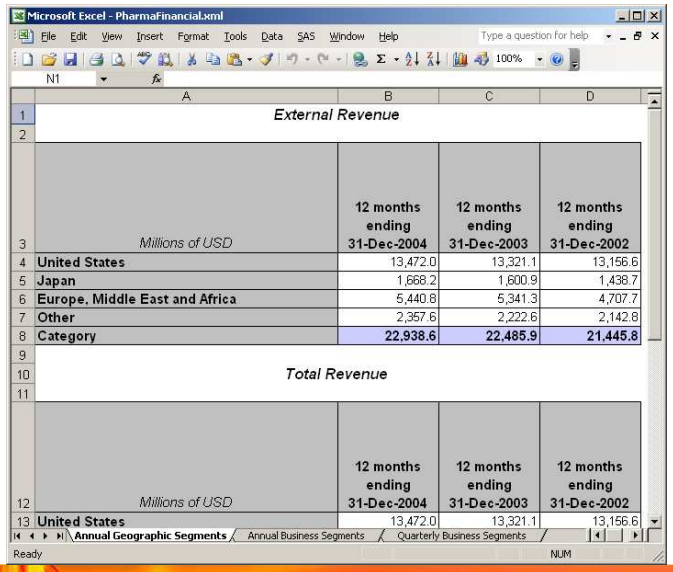
Line 43: Add text to LABEL statement: Millions of USD

Note: All of these changes are already done for the remaining PROC PRINT code.

Fix 5: Tweak the Appearance

1. Go to Excel
2. File > **\HOW\DelGobbo\PharmaFinancial.xml**
   - or -
   PharmaFinancial.xml (from the recent file list)
3. Note header text and formatted numeric values
4. Close the document (child) window (leave Excel running, but minimize it)

Homework – First Worksheet



45

These exercises are intended to give you a little more experience with creating new style elements and using style overrides.

All worksheets:

- Header cell text for the "Source" column is italic and centered horizontally
- Sum and grandtotal cells have #ccccff for a background color

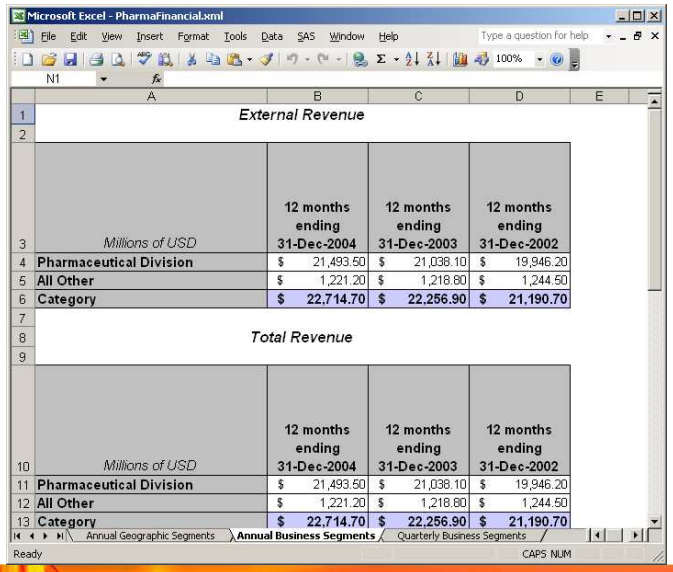
This worksheet:

- No other changes

Hints:

1. Add the "font_style" and "just" style attributes to "header_id" style element.
2. Add the "background" style attribute to the "header_comma_1" style element.
3. Last resort: Look at file "Homework.sas" in download package.

Homework – Second Worksheet



46

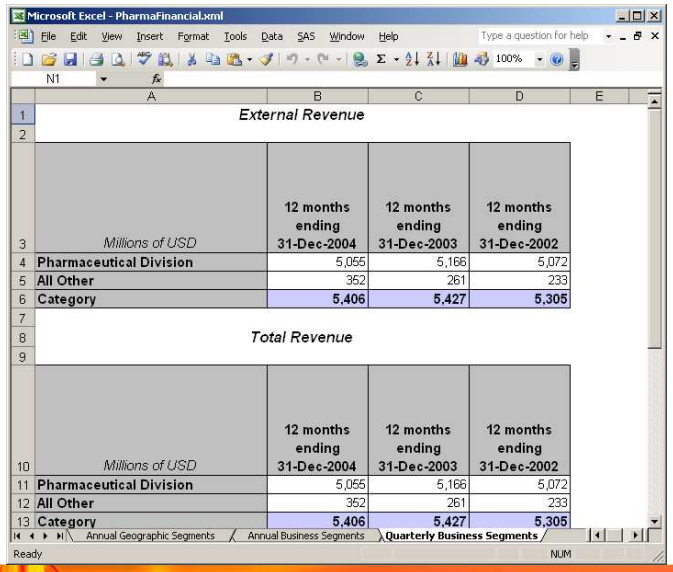
This worksheet:

- Numeric values are formatted using the "Currency" Excel format

Hints:

1. Create new style elements similar to "data_comma_1" and "header_comma_1", but use the Excel format "Currency" instead of "#,###.0"
2. Apply the new style elements using style overrides.
3. Last resort: Look at file "Homework.sas" in download package.

Homework – Third Worksheet



Microsoft Excel - PharmaFinancial.xml

External Revenue

Millions of USD	12 months ending 31-Dec-2004	12 months ending 31-Dec-2003	12 months ending 31-Dec-2002
Pharmaceutical Division	5,055	5,166	5,072
All Other	352	261	233
Category	5,406	5,427	5,305

Total Revenue

Millions of USD	12 months ending 31-Dec-2004	12 months ending 31-Dec-2003	12 months ending 31-Dec-2002
Pharmaceutical Division	5,055	5,166	5,072
All Other	352	261	233
Category	5,406	5,427	5,305

Annual Geographic Segments | Annual Business Segments | Quarterly Business Segments

Ready NUM

47

This worksheet:

- Numeric values are formatted as integers using an Excel format

Hints:

1. Create new style elements similar to "data_comma_1" and "header_comma_1", but use an Excel format that does not have a decimal point and does not have a decimal place.
2. Apply the new style elements using style overrides.
3. Last resort: Look at file "Homework.sas" in download package.

Advanced Topic: Omitting the Extra Summaries

- Problem: Extra summary rows w/o subtotals
- A solution: Get rid of the extra summary rows

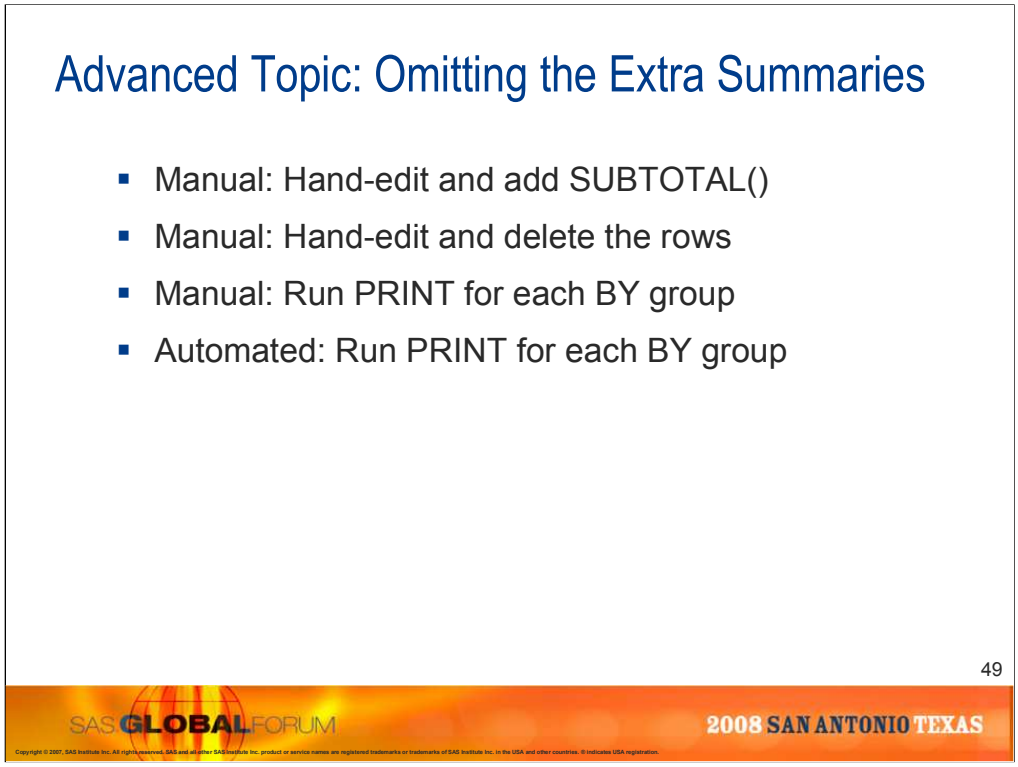
Source	12 months ending 31-Dec-2004	12 months ending 31-Dec-2003	12 months ending 31-Dec-2002
Pharmaceutical Division	3266.1	3234.9	3162.5
All Other	378.9	264.2	240.4
Category	3645	3499.1	3402.9
	14457.8	14352.9	14013.3

=SUBTOTAL(9,D2:D5)

57087.2

48

The AUTO_SUBTOTALS tagset option is used to insert an Excel SUBTOTAL function into summary cells created by the PRINT procedure. However, the SUBTOTAL function is only inserted into the cells corresponding to the first summary line, not the grand total summary line, or any additional summary lines resulting from multiple BY groups.



This behavior may be changed in a future version of the tagset, but currently you must hand edit the Excel workbook to add the SUBTOTAL function to the additional summary lines.

If you do not want the additional summary lines, you can manually delete them, or you can run PROC PRINT repeatedly without a BY statement to create the 10 Excel tables.

Rather than duplicating the code for each BY group, a better solution is to create a SAS macro that examines the input data, determines all the BY group values, and then executes PROC PRINT automatically for each BY group.

Refer to the Appendix of the accompanying paper for a description of this macro, and also to the file named "MakeXML-Macro.sas", which is available in the download package.

Summary: Creating Multi-Sheet Workbooks

- Use ExcelXP tagset to create XML file
- Resulting XML file can be viewed with Excel
- Can have multiple tables in a worksheet
- Apply ODS style overrides carefully
- Make use of tagset options
- Use Excel formats instead of SAS formats

Using SAS/IntrNet

and

SAS Stored Processes

51

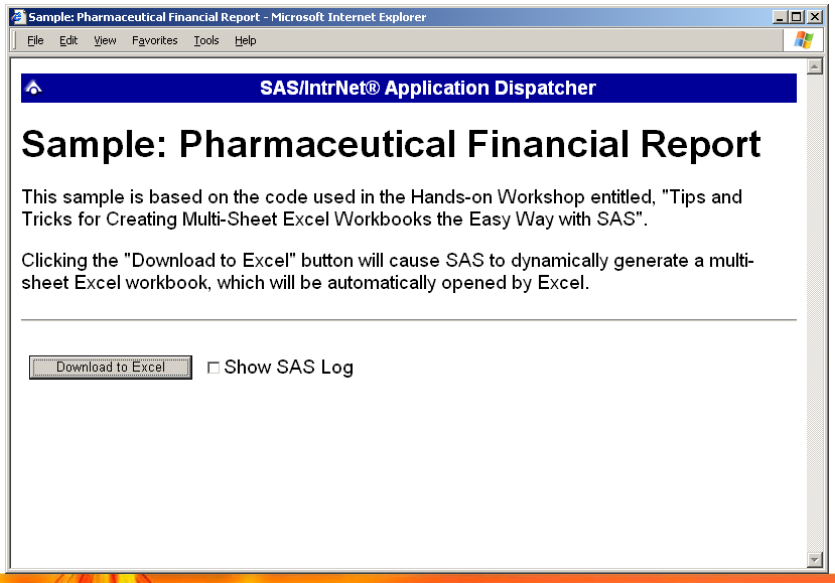
The purpose of the SAS/IntrNet Application Dispatcher or SAS Stored Processes are to allow you to execute SAS programs from a machine that does not have SAS installed. The machine *may* have SAS installed, but that is not required.

A typical client-server model is followed. The SAS server can reside on any hardware platform (Windows, UNIX, z/OS, etc.) and is standing by, waiting to execute a SAS program. The most common client is a Web browser, again, running on any platform.

When the "OK" button of the Web page is clicked, input parameters, if any, are sent to the SAS server. Your SAS code executes, and the output is delivered in "real time" to the Web browser.

The following slides illustrate this process, using a "Web-enabled" version of the SAS code we have been working with.

Dynamically-Generated XML



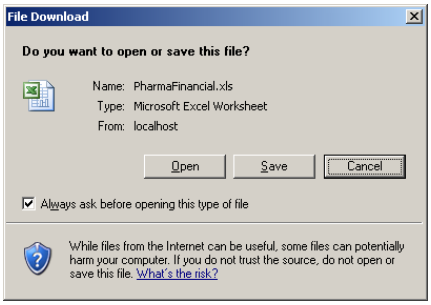
53

Here is a simple Web page that can be used to execute SAS code stored on a server using the SAS/IntrNet product.

The code that will be executed is substantially similar to the final versions of the code that we used to generate XML file, with a few changes to "Web-enable" it.

Clicking "Download to Excel" causes the SAS program to execute on the server.

Dynamically-Generated XML



54

Once the SAS program executes, the results are sent back to the Web browser.

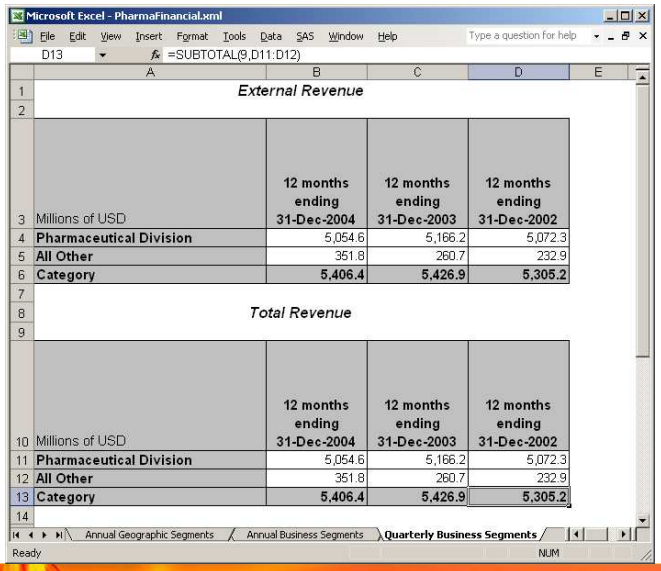
Note that you are presented with a File Download dialog, instead of the results being displayed in the Web browser. Clicking "Open" will cause the SAS output to be displayed in Excel, provided that Excel is installed on the machine.

This SAS/IntrNet-specific code, which must be specified before any ODS statements, was used to cause the SAS output to be displayed in Excel:

```
%let RV =%sysfunc(appsrv_header(Content-type,application/vnd.ms-excel));
```

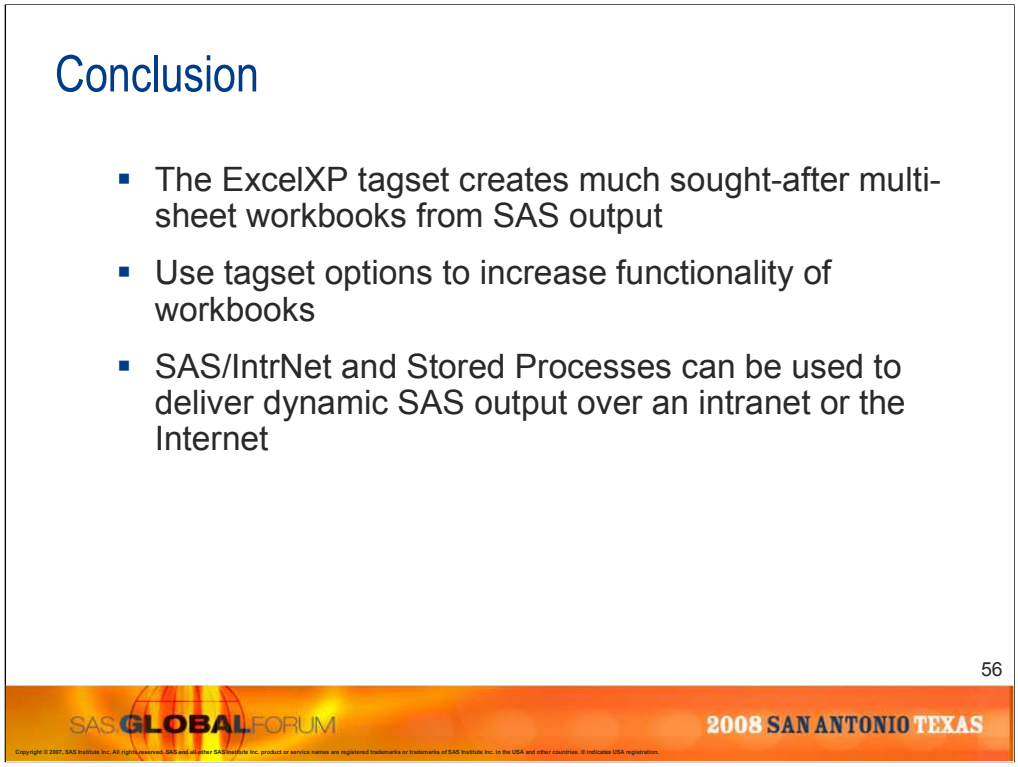
This code will also work with SAS Stored Processes.

Dynamically-Generated XML



55

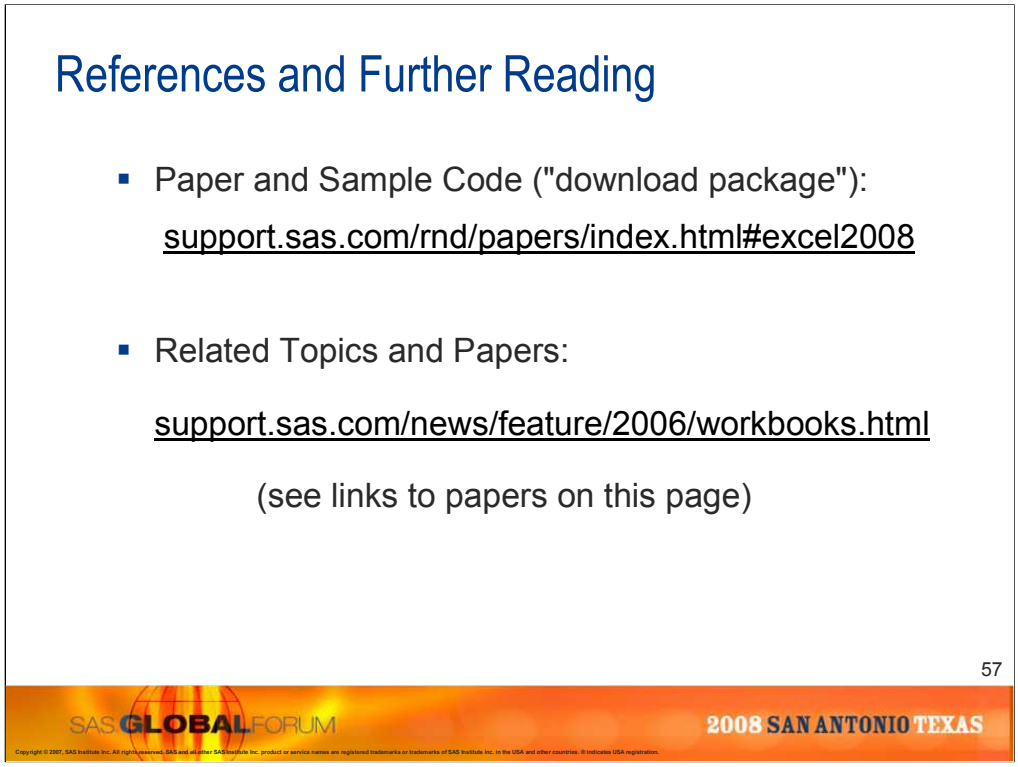
Here is the SAS output, created in "real time", and delivered to the client. Note that Excel is used to view the SAS output, not the Web browser.



Using ODS to generate specially-formatted XML output is an effective method of incorporating SAS output in Excel documents.

The SAS 9.1 ExcelXP tagset complies with the Microsoft XML Spreadsheet Specification and provides an easy way to export your data to Excel workbooks that contain multiple worksheets. The tagset supports a great number of options that can be used provide many Excel functions.

SAS Institute continues to work toward better Microsoft Office integration, and future releases of SAS software will provide even more robust means of using SAS content with Microsoft Office applications.

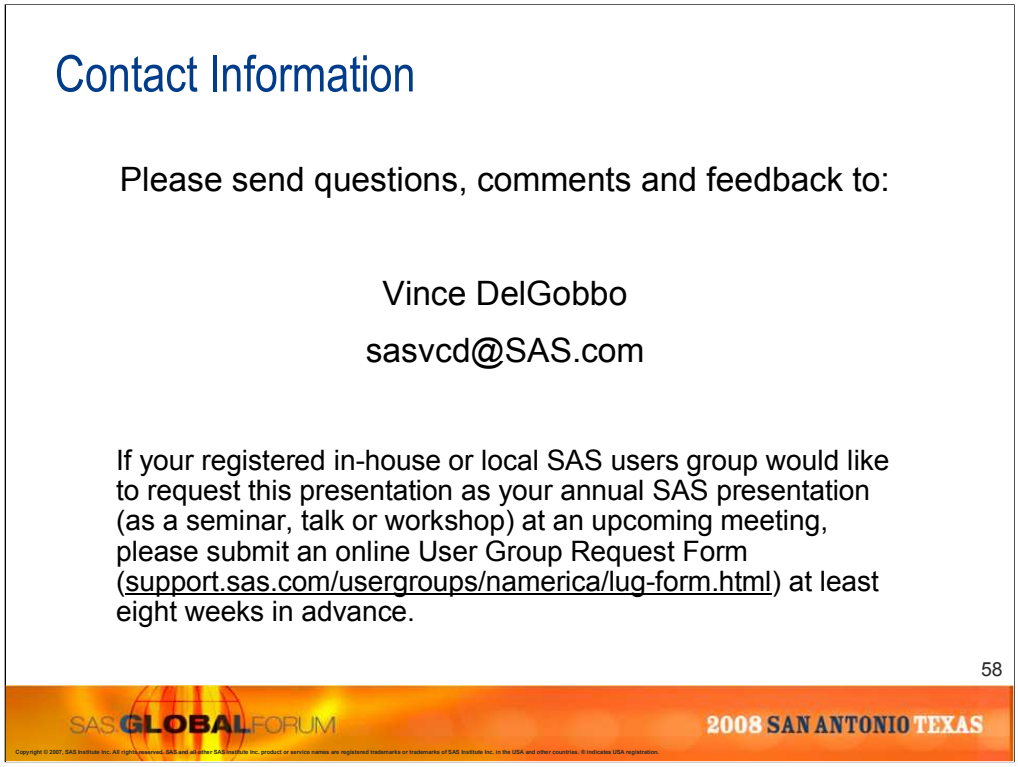


The sample programs and data used in this workshop as well as a copy of the accompanying paper are available at the SAS Presents Web site. Go to http://support.sas.com/rnd/papers/index.html#excel2008 and find the entry "Tips and Tricks for Creating Multi-Sheet Excel Workbooks the Easy Way with SAS".

IMPORTANT NOTE: Be sure to install a recent version of the ExcelXP tagset on your system. See the accompanying paper for details and instructions.

Review the "Installation" and "Usage" sections of the file named "ReadMe.txt" in the ZIP archive for important information on using the sample files.

Refer to the ODS documentation to find out more about styles, style attributes and the TEMPLATE procedure.



About the author:

Vince DelGobbo is a Senior Systems Developer in the Web Tools group at SAS. This group is responsible for developing the SAS/IntrNet Application Dispatcher and SAS Stored Processes. He is the developer of the HTML Formatting Tools and the SAS Design-Time Controls, and is developing other new Web- and server-based technologies, as well as integrating SAS output with Microsoft Office. He is also involved in the development of the ExcelXP ODS tagset. Vince has been a SAS Software user since 1982, and joined SAS in 1992.