The Power to Know.

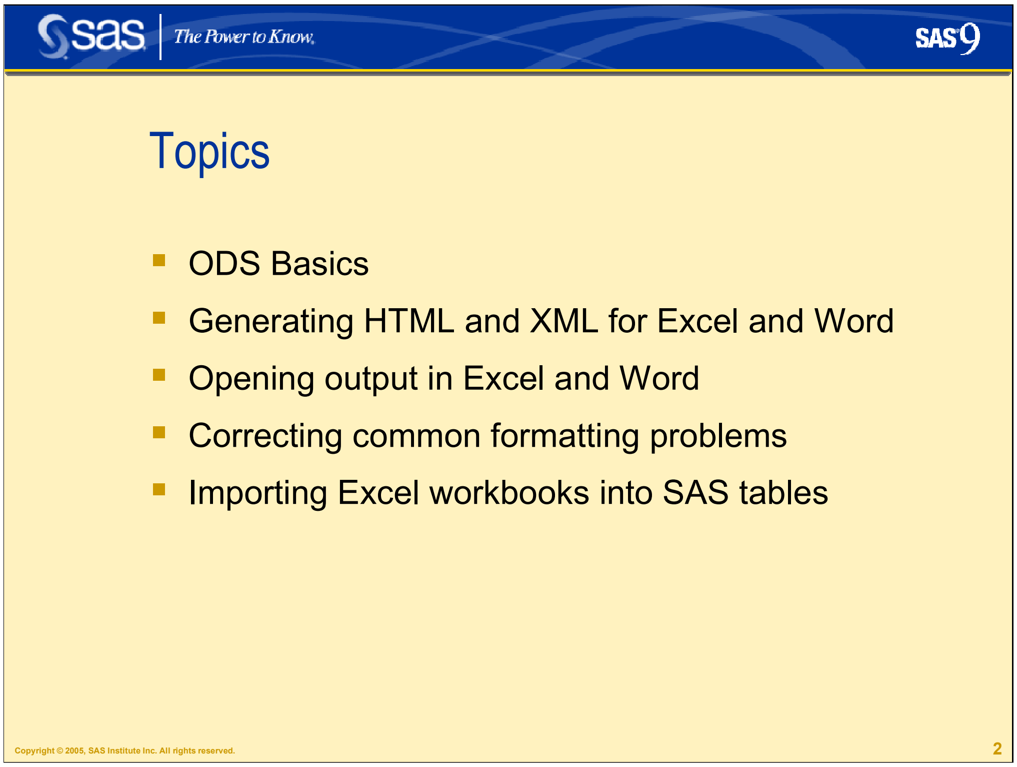
SUGI 30
PHILADELPHIA
SAS® USERS GROUP INTERNATIONAL
APRIL 10–13, 2005

Moving Data and Analytical Results between SAS® and Microsoft Office

Vince DelGobbo
Web Tools Group

Presented at the 2005 SAS Users Group International Conference (SUGI 30)
Pennsylvania Convention Center
Philadelphia, PA
April 10-13, 2005

A special *"thank you"* to the SUGI Hands-on Workshop section chairs Jenine Eason and Michael Mace for inviting me to present this topic, and to Chris Barrett of SAS Institute Inc. for his valuable input on the accompanying paper.



sas The Power to Know.
SAS 9

Topics

- ODS Basics
- Generating HTML and XML for Excel and Word
- Opening output in Excel and Word
- Correcting common formatting problems
- Importing Excel workbooks into SAS tables

Software Requirements:

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

- SAS 8.2 for generating HTML.

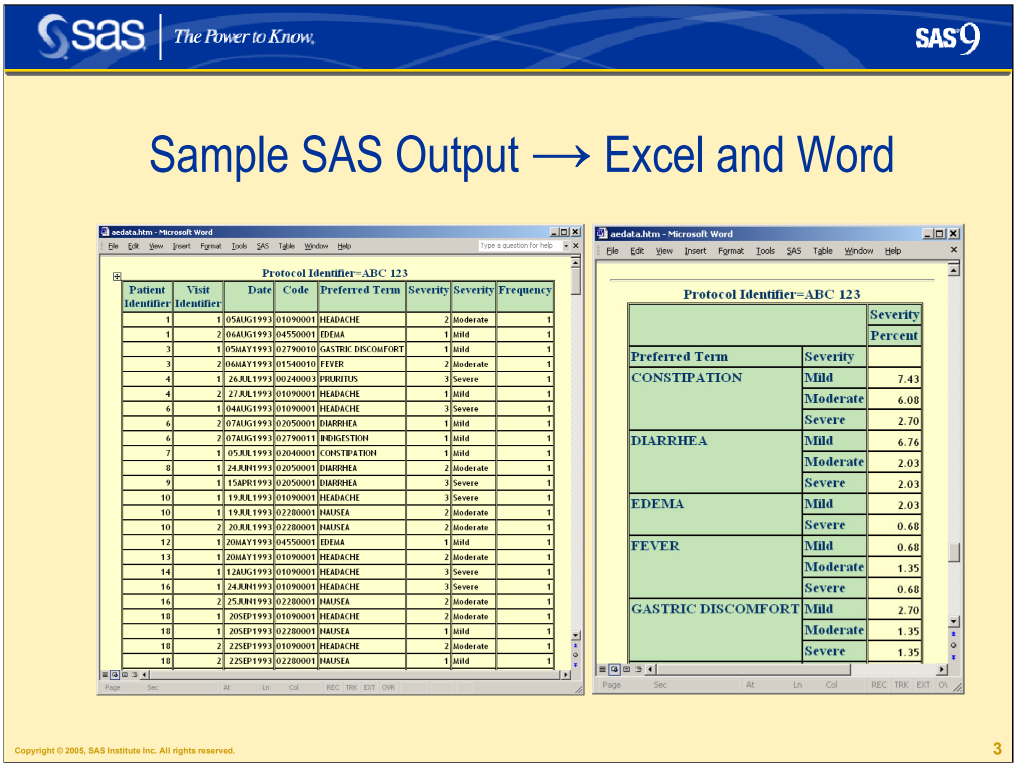
- SAS 9.1.2 or later for generating XML (9.1.3 provides greatly improved performance for the "ExcelXP" tagset).

- SAS 9.1.2 or later for importing Excel XML files into SAS.

- For this workshop, we are using SAS 9.1.3.

- Microsoft Office 2000 or later for accessing SAS HTML output.

- Microsoft Office XP (a.k.a. Office 2002) or later for importing SAS XML output or creating XML for SAS to read.


sas The Power to Know.
SAS9
aedata.htm - Microsoft Word
File Edit View Insert Format Tools SAS Table Window Help
Type a question for help
Page Sec At Ln Col REC TRK EXT OVR

Sample SAS Output ⟶ Excel and Word

Protocol Identifier=ABC 123

Patient Identifier	Visit Identifier	Date	Code	Preferred Term	Severity	Severity	Frequency
1	1	05AUG1993	01090001	HEADACHE	2	Moderate	1
1	2	06AUG1993	04550001	EDEMA	1	Mild	1
3	1	05MAY1993	02790010	GASTRIC DISCOMFORT	1	Mild	1
3	2	06MAY1993	01540010	FEVER	2	Moderate	1
4	1	26JUL1993	00240003	PRURITUS	3	Severe	1
4	2	27JUL1993	01090001	HEADACHE	1	Mild	1
6	1	04AUG1993	01090001	HEADACHE	3	Severe	1
6	2	07AUG1993	02050001	DIARRHEA	1	Mild	1
6	2	07AUG1993	02790011	INDIGESTION	1	Mild	1
7	1	05JUL1993	02040001	CONSTIPATION	1	Mild	1
8	1	24JUN1993	02050001	DIARRHEA	2	Moderate	1
9	1	15APR1993	02050001	DIARRHEA	3	Severe	1
10	1	19JUL1993	01090001	HEADACHE	3	Severe	1
10	1	19JUL1993	02280001	NAUSEA	2	Moderate	1
10	2	20JUL1993	02280001	NAUSEA	2	Moderate	1
12	1	20MAY1993	04550001	EDEMA	1	Mild	1
13	1	20MAY1993	01090001	HEADACHE	2	Moderate	1
14	1	12AUG1993	01090001	HEADACHE	3	Severe	1
16	1	24JUN1993	01090001	HEADACHE	3	Severe	1
16	2	25JUN1993	02280001	NAUSEA	2	Moderate	1
18	1	20SEP1993	01090001	HEADACHE	2	Moderate	1
18	1	20SEP1993	02280001	NAUSEA	1	Mild	1
18	2	22SEP1993	01090001	HEADACHE	2	Moderate	1
18	2	22SEP1993	02280001	NAUSEA	1	Mild	1

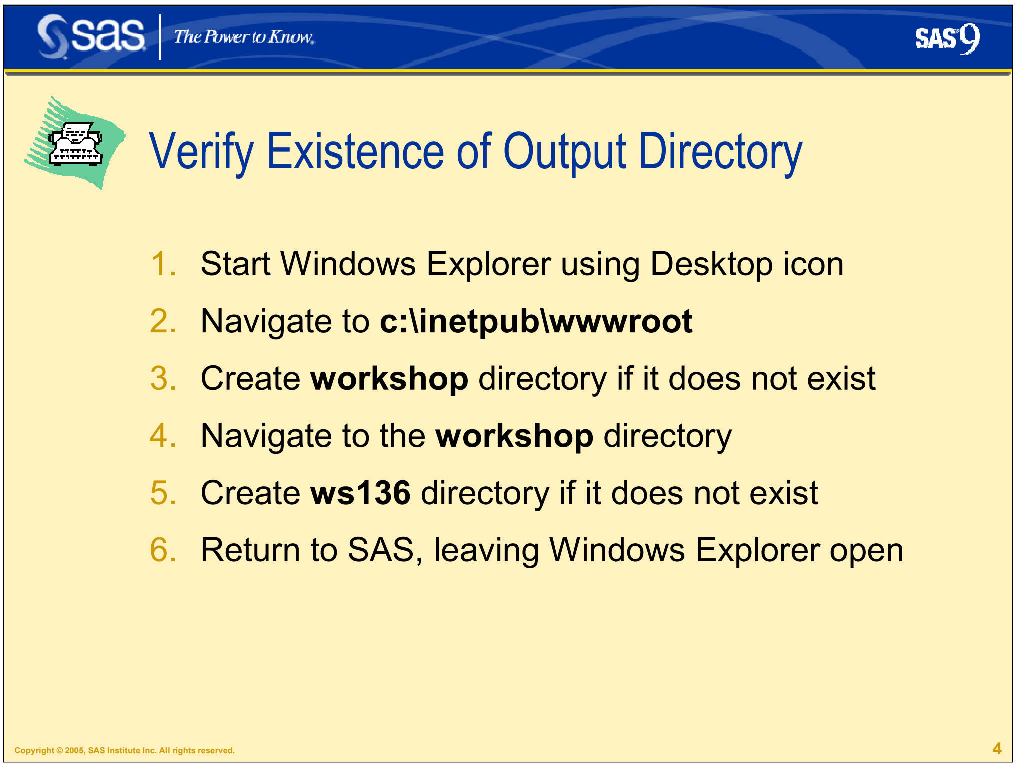
Protocol Identifier=ABC 123

		Severity
		Percent
Preferred Term	Severity	
CONSTIPATION	Mild	7.43
	Moderate	6.08
	Severe	2.70
DIARRHEA	Mild	6.76
	Moderate	2.03
	Severe	2.03
EDEMA	Mild	2.03
	Severe	0.68
FEVER	Mild	0.68
	Moderate	1.35
	Severe	0.68
GASTRIC DISCOMFORT	Mild	2.70
	Moderate	1.35
	Severe	1.35

Here is HTML output generated by ODS and viewed using Microsoft Word. You are looking at two different pages of a single Word document. The image on the left represents output from the PRINT procedure while PROC TABULATE output is shown on the right.

This illustrates the type of SAS output that we will be incorporating into Excel and Word.

A few important things to note are that we have a colored page background as well as cell backgrounds, all the cells have border lines, and there are some columns that contain data, such as dates and leading zeroes, that may cause problems when importing into to Excel or Word.



Verify Existence of Output Directory

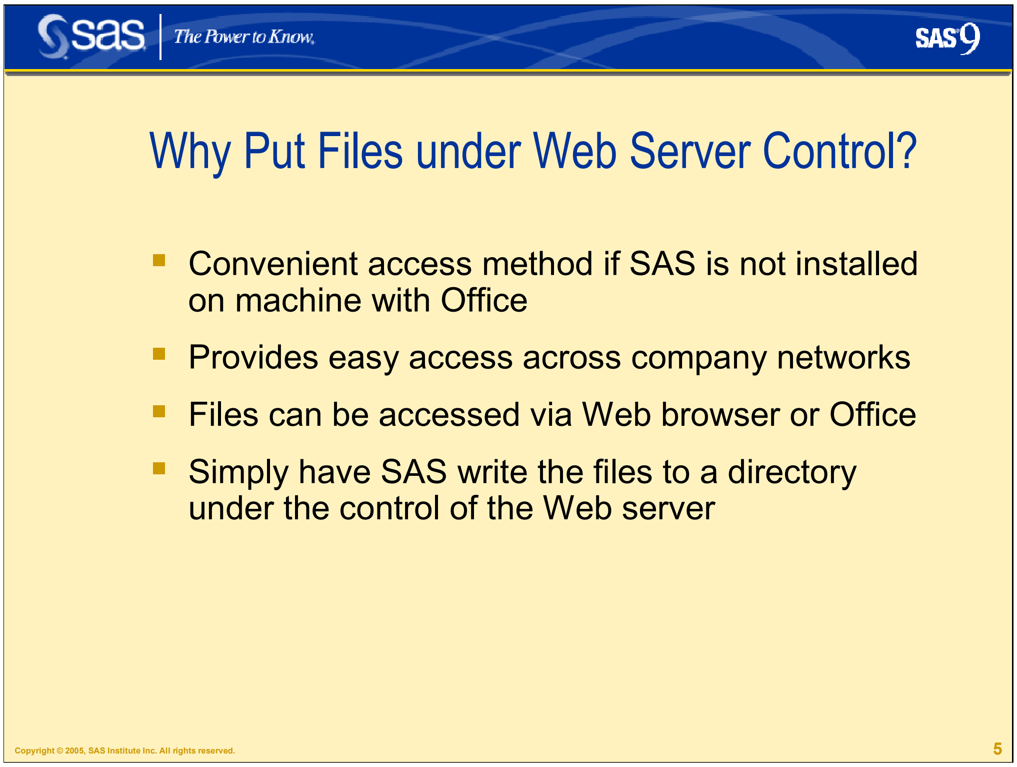
1. Start Windows Explorer using Desktop icon
2. Navigate to **c:\inetpub\wwwroot**
3. Create **workshop** directory if it does not exist
4. Navigate to the **workshop** directory
5. Create **ws136** directory if it does not exist
6. Return to SAS, leaving Windows Explorer open

The typewriter logo in the upper left corner indicates a hands-on activity.

We are going to use ODS to write files to a directory that is under the control of a Web server, and this directory must exist (ODS will not create it for you).

We are using the Microsoft Internet Information Services (IIS) Web server, which is part of the Windows 2000 and Windows XP operating systems. The root directory for this Web server is "c:\inetpub\wwwroot". ODS will write files to the "workshop\ws136" subdirectory.

For this workshop, the directory should have been set up for you.

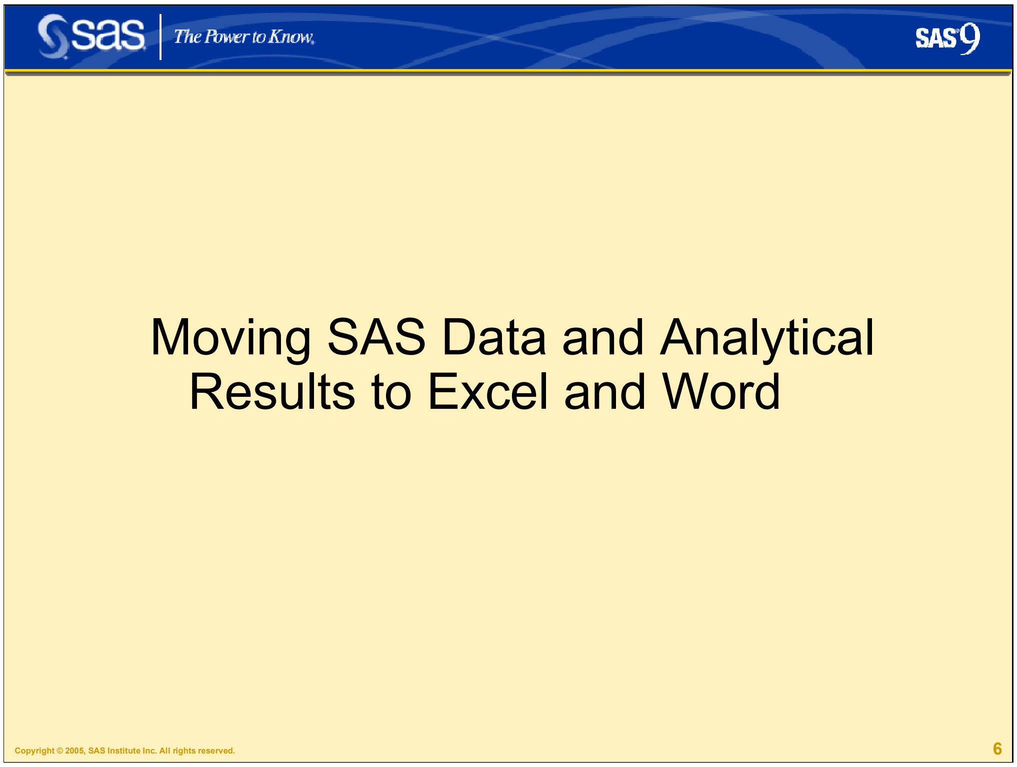


Why Put Files under Web Server Control?

- Convenient access method if SAS is not installed on machine with Office
- Provides easy access across company networks
- Files can be accessed via Web browser or Office
- Simply have SAS write the files to a directory under the control of the Web server

You can use ODS to write output to any directory of your choosing. If you have a Web server available, as we do in this workshop, it is convenient to make use of it for the reasons noted above.

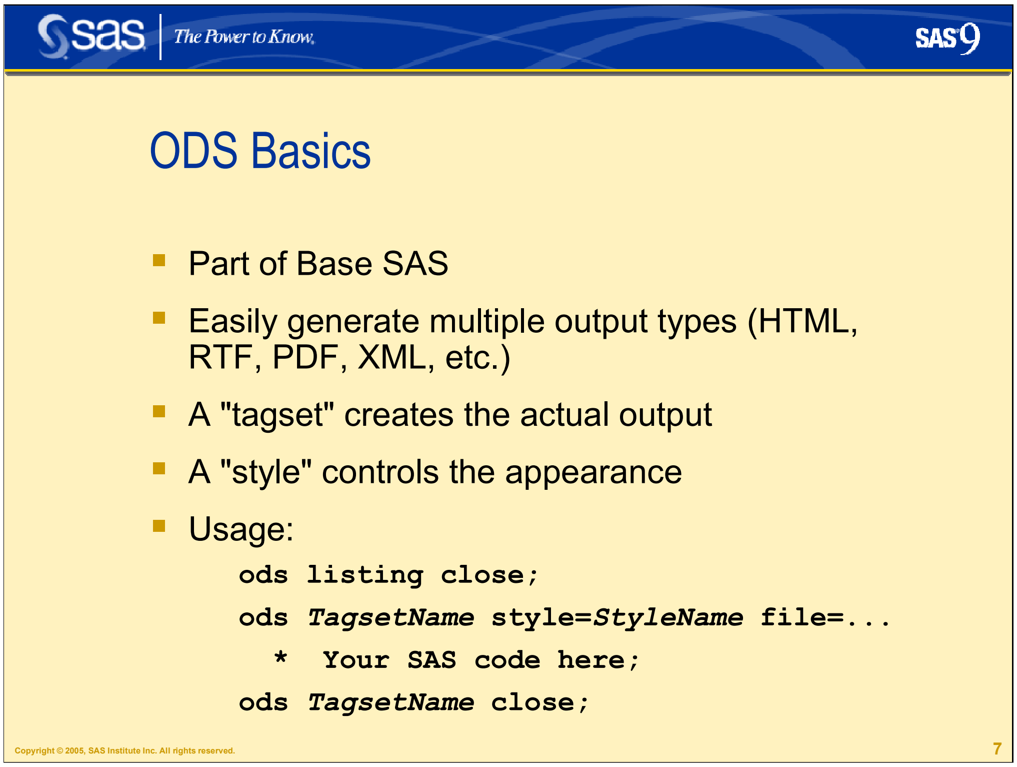
In summary, you can make a single HTML file that is available to anyone with network access, and it can be viewed by a Web browser, Excel or Word.



Moving SAS Data and Analytical Results to Excel and Word

This section focuses on using ODS to create HTML and XML files that can be imported into Excel and Word (only Excel supports HTML and XML).

A later section will focus on importing Excel workbooks into SAS tables.



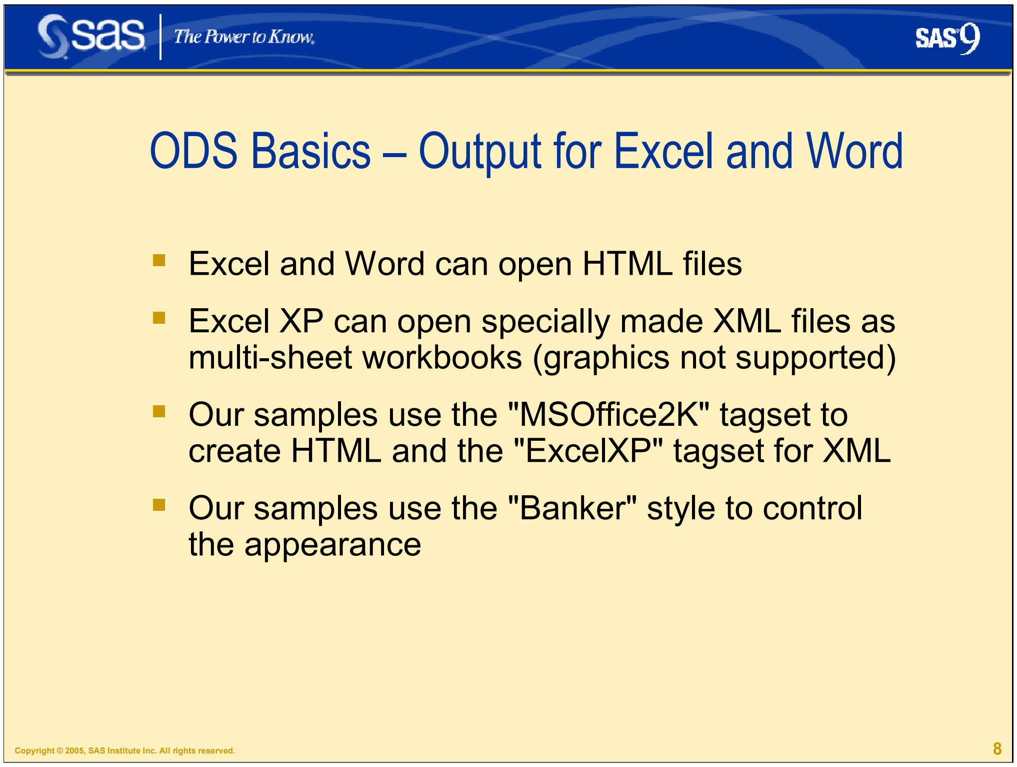
ODS Basics

- Part of Base SAS
- Easily generate multiple output types (HTML, RTF, PDF, XML, etc.)
- A "tagset" creates the actual output
- A "style" controls the appearance
- Usage:

```
ods listing close;
ods TagsetName style=StyleName file=...
  *  Your SAS code here;
ods TagsetName close;
```

ODS tagsets create the actual output (HTML, RTF, PDF, XML, etc.) while an ODS style controls the appearance (colors, fonts, border lines, etc.).

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



ODS Basics – Output for Excel and Word

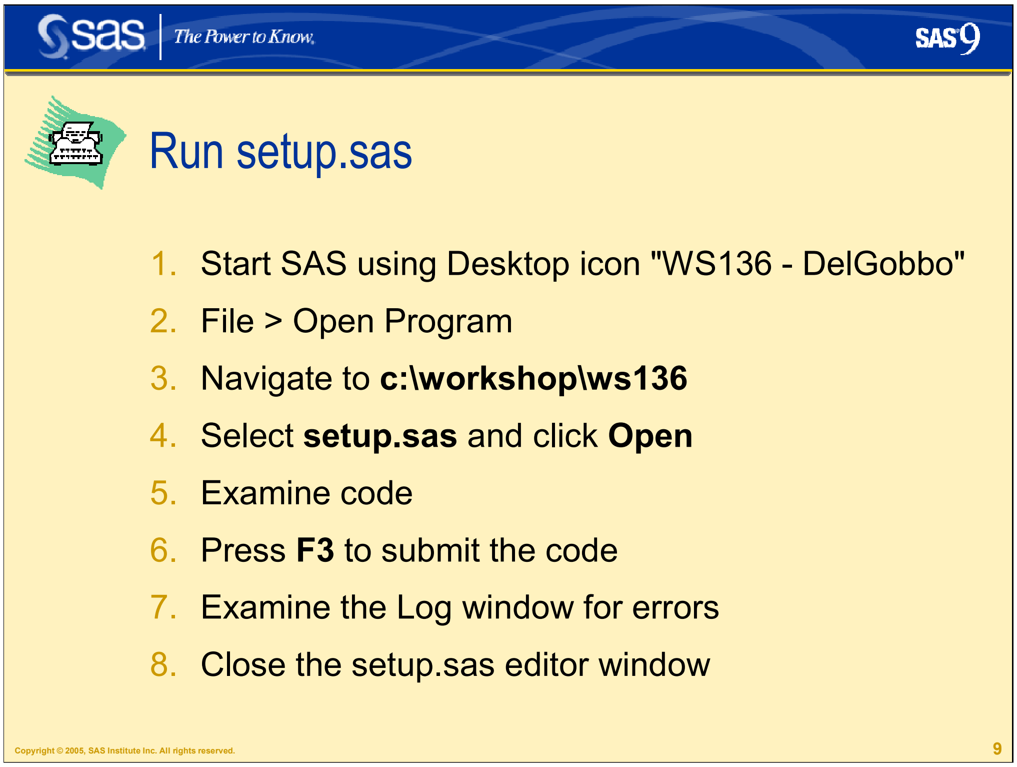
- Excel and Word can open HTML files
- Excel XP can open specially made XML files as multi-sheet workbooks (graphics not supported)
- Our samples use the "MSOffice2K" tagset to create HTML and the "ExcelXP" tagset for XML
- Our samples use the "Banker" style to control the appearance

Since Excel and Word can open HTML files, you can use the "HTML" or "HTML3" ODS destinations to generate HTML for Excel and Word as follows:

```
ods HTML3 style=StyleName file=...
  *  Your SAS code here;
ods HTML3 close;
```

While you can use the "HTML" destination, some versions of Excel and Word are not capable of consuming the HTML 4.0-compliant HTML that the tagset generates. If you have excessive display problems using the "HTML" destination, try using "HTML3" or "MSOffice2K" instead.

IMPORTANT NOTE: The "MSOffice2K" tagset has been optimized for use with Office 2000 and later.


sas The Power to Know.
SAS9

Run setup.sas

1. Start SAS using Desktop icon "WS136 - DelGobbo"
2. File > Open Program
3. Navigate to **c:\workshop\ws136**
4. Select **setup.sas** and click **Open**
5. Examine code
6. Press **F3** to submit the code
7. Examine the Log window for errors
8. Close the setup.sas editor window

SAS macro variables

OUTDIR – directory for ODS HTML and XML output for Excel and Word

INDIR – directory containing input files and data

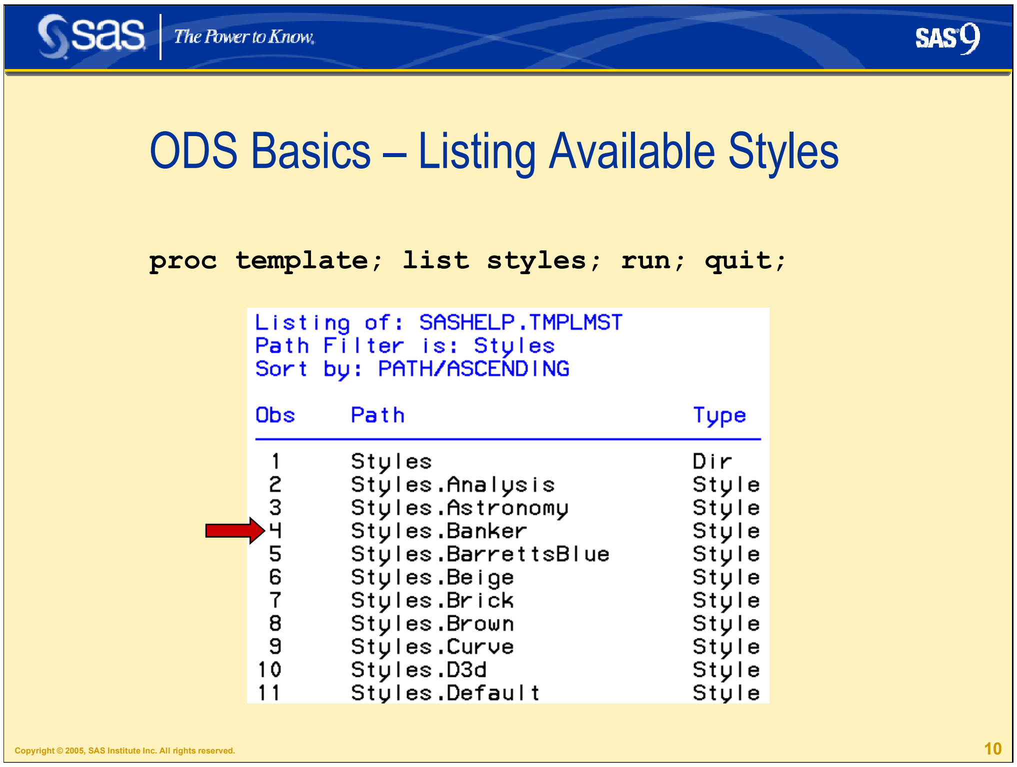
WEBDIR – URL for the Web server directory containing the ODS output

The program assigns a SAS library (PHARMA) 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 SASUSER library to 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 root directory of the IIS Web server is c:\inetpub\wwwroot. We will keep our files in a subdirectory under this root named "workshop\ws136", and the OUTDIR variable points to this subdirectory. Files in this subdirectory can be accessed via this URL: http://localhost/workshop/ws136/*file-name*

The ODS PATH statement controls the search and storage locations for styles and tagsets. Thus, MYLIB will be searched/used first. The ODS tagsets "MSOffice2K" and "ExcelXP" have undergone some important changes, so we will import new copies and store them in the MYLIB library.

Finally, load the SAS macro (XLXP2SAS) that imports Excel workbooks into SAS tables.



ODS Basics – Listing Available Styles

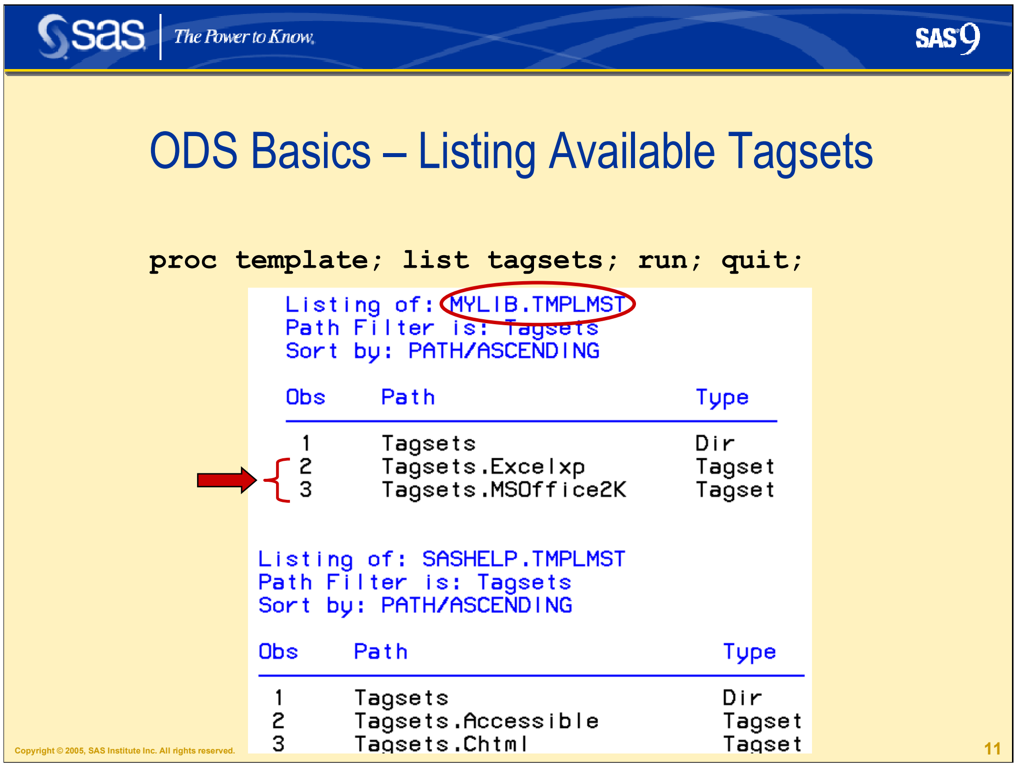
```
proc template; list styles; run; quit;
```

```
Listing of: SASHELP.TMPLMST
Path Filter is: Styles
Sort by: PATH/ASCENDING

Obs     Path                          Type
------------------------------------------
  1     Styles                        Dir
  2     Styles.Analysis               Style
  3     Styles.Astronomy              Style
  4     Styles.Banker                 Style
  5     Styles.BarrettsBlue           Style
  6     Styles.Beige                  Style
  7     Styles.Brick                  Style
  8     Styles.Brown                  Style
  9     Styles.Curve                  Style
 10     Styles.D3d                    Style
 11     Styles.Default                Style
```

This code is used to list the ODS styles available on your system. The image above shows a partial listing of the available styles.

We will be using the built-in style named "Banker".

You can use the TEMPLATE procedure to create your own custom styles.



ODS Basics – Listing Available Tagsets

```
proc template; list tagsets; run; quit;
```

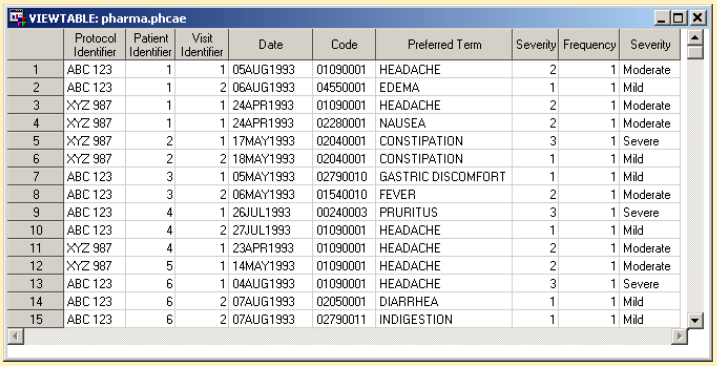
```
 Listing of: MYLIB.TMPLMST
 Path Filter is: Tagsets
 Sort by: PATH/ASCENDING

 Obs     Path                      Type
 ___________________________________________

   1     Tagsets                   Dir
   2     Tagsets.Excelxp           Tagset
   3     Tagsets.MSOffice2K        Tagset


Listing of: SASHELP.TMPLMST
Path Filter is: Tagsets
Sort by: PATH/ASCENDING

Obs     Path                           Type
___________________________________________

  1     Tagsets                        Dir
  2     Tagsets.Accessible             Tagset
  3     Tagsets.Chtml                  Tagset
```

This code is used to list the ODS tagsets available on your system. The image above shows a partial listing of the available tagsets.

Copies of the tagsets "MSOffice2K" and "ExcelXP" exist in both the MYLIB and SASHELP libraries. Since the MYLIB library appears first, the copies of the two tagsets located there will be used when referenced by your SAS code. These tagsets appear first due to the ODS PATH statement that was issued by the program "setup.sas".

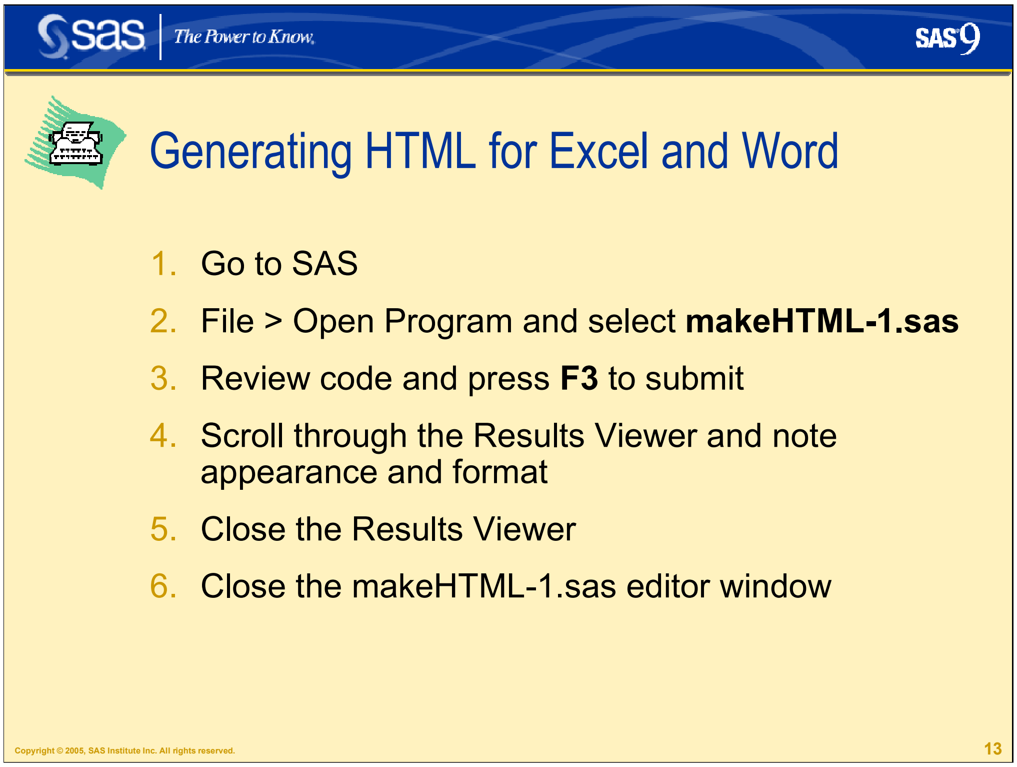
You can use the TEMPLATE procedure to create your own custom tagsets.

Preview Sample SAS Data



	Protocol Identifier	Patient Identifier	Visit Identifier	Date	Code	Preferred Term	Severity	Frequency	Severity
1	ABC 123	1	1	05AUG1993	01090001	HEADACHE	2	1	Moderate
2	ABC 123	1	2	06AUG1993	04550001	EDEMA	1	1	Mild
3	XYZ 987	1	1	24APR1993	01090001	HEADACHE	2	1	Moderate
4	XYZ 987	1	1	24APR1993	02280001	NAUSEA	2	1	Moderate
5	XYZ 987	2	1	17MAY1993	02040001	CONSTIPATION	3	1	Severe
6	XYZ 987	2	2	18MAY1993	02040001	CONSTIPATION	1	1	Mild
7	ABC 123	3	1	05MAY1993	02790010	GASTRIC DISCOMFORT	1	1	Mild
8	ABC 123	3	2	06MAY1993	01540010	FEVER	2	1	Moderate
9	ABC 123	4	1	26JUL1993	00240003	PRURITUS	3	1	Severe
10	ABC 123	4	2	27JUL1993	01090001	HEADACHE	1	1	Mild
11	XYZ 987	4	1	23APR1993	01090001	HEADACHE	2	1	Moderate
12	XYZ 987	5	1	14MAY1993	01090001	HEADACHE	2	1	Moderate
13	ABC 123	6	1	04AUG1993	01090001	HEADACHE	3	1	Severe
14	ABC 123	6	2	07AUG1993	02050001	DIARRHEA	1	1	Mild
15	ABC 123	6	2	07AUG1993	02790011	INDIGESTION	1	1	Mild

This SAS table contains data pertaining to adverse events of a fictitious drug.



The Power to Know.

Generating HTML for Excel and Word

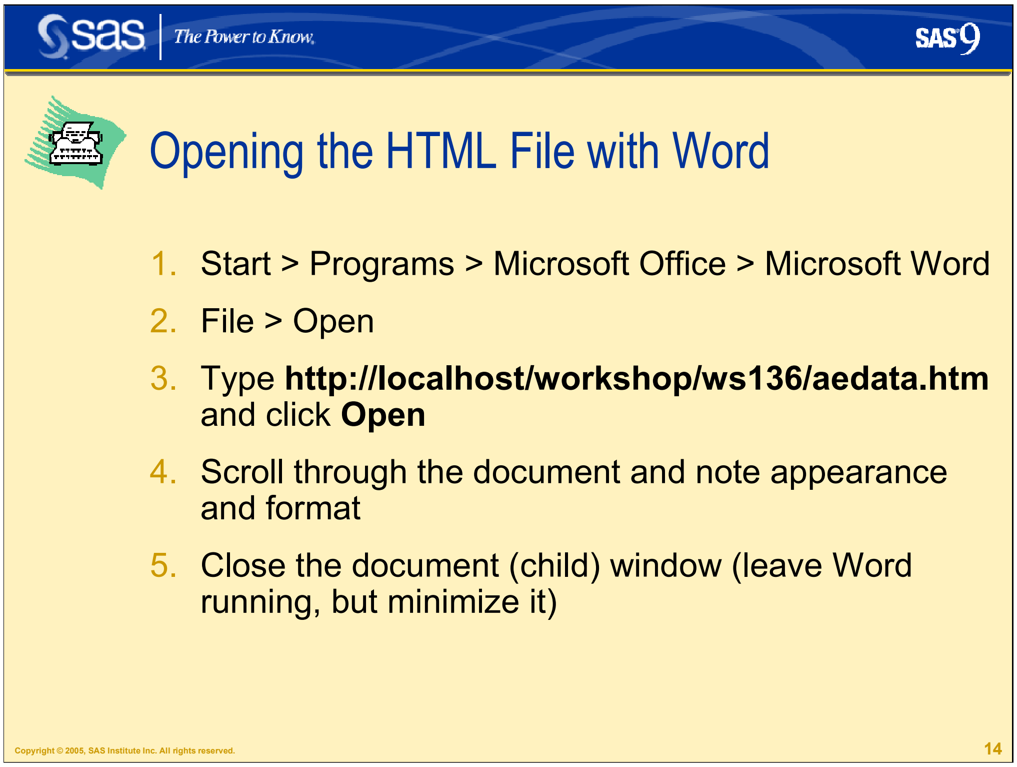
1. Go to SAS
2. File > Open Program and select **makeHTML-1.sas**
3. Review code and press **F3** to submit
4. Scroll through the Results Viewer and note appearance and format
5. Close the Results Viewer
6. Close the makeHTML-1.sas editor window

This is our first attempt to make an HTML file that can be used with Excel and Word.

The "MSOffice2K" tagset is used to generate HTML output, and the "Banker" style controls the appearance of the HTML.

The HTML file will be stored in a directory controlled by our Web server (as specified in the OUTDIR macro variable) and will be named "aedata.htm".

We use PROC PRINT to list the data, and perform an analysis using the TABULATE procedure. In both cases, the results are displayed by protocol.



sas
The Power to Know.
SAS 9

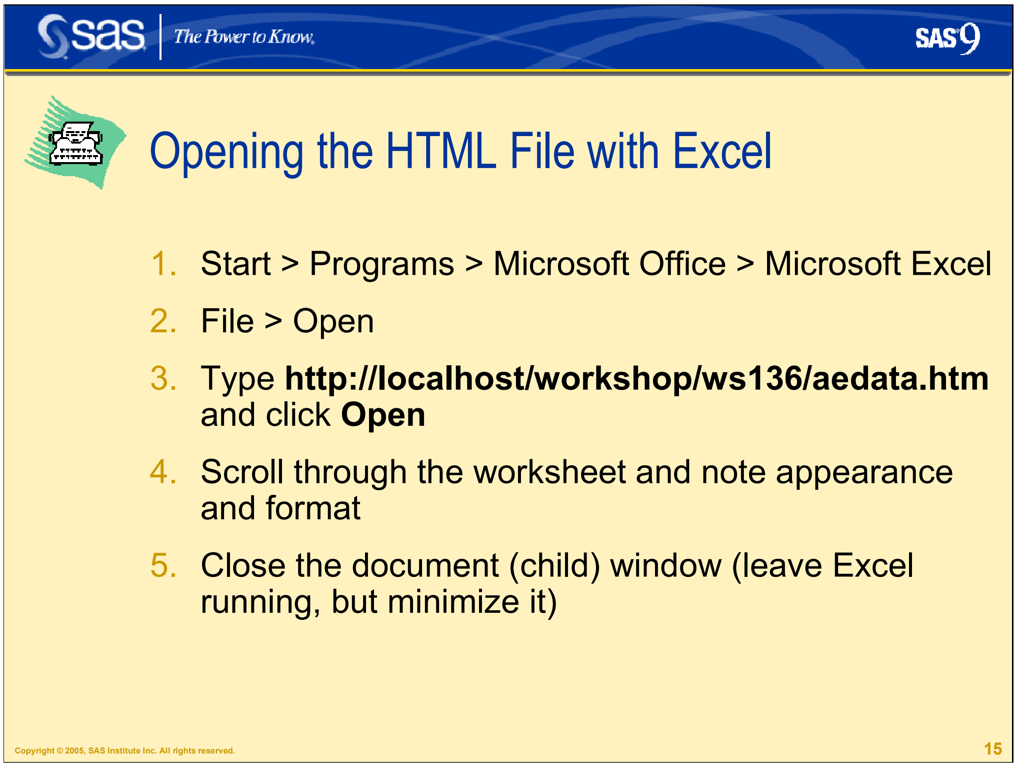
Opening the HTML File with Word

1. Start > Programs > Microsoft Office > Microsoft Word
2. File > Open
3. Type **http://localhost/workshop/ws136/aedata.htm** and click **Open**
4. Scroll through the document and note appearance and format
5. Close the document (child) window (leave Word running, but minimize it)

The HTML file created by ODS can now be opened with Microsoft Word.

When Word reads the HTML file, the HTML tables are automatically converted to native Word tables. Thus you can use all the editing and formatting features of Word to modify the output and save it as a native Word document (File > Save As and choose "Word Document (*.doc)")

NOTE: In a production environment, you would replace "localhost" with the address of your Web server.



Opening the HTML File with Excel

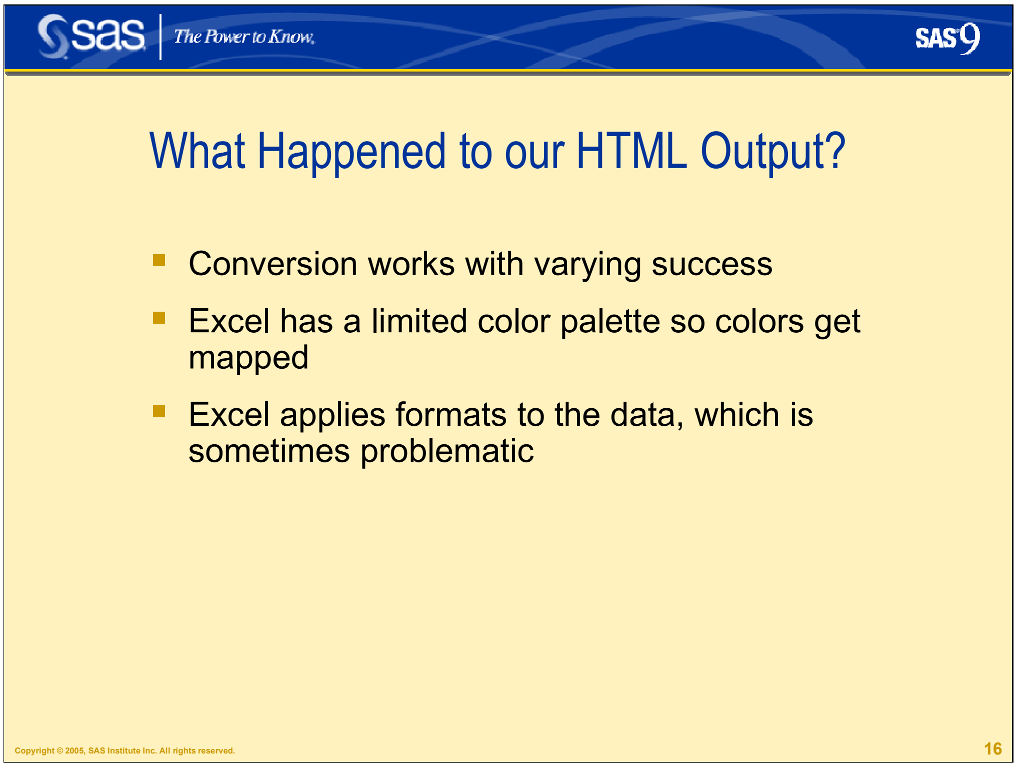
1. Start > Programs > Microsoft Office > Microsoft Excel
2. File > Open
3. Type **http://localhost/workshop/ws136/aedata.htm** and click **Open**
4. Scroll through the worksheet and note appearance and format
5. Close the document (child) window (leave Excel running, but minimize it)

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

When Excel reads the HTML file, the HTML tables are automatically converted to native Excel tables. Thus you can use all the editing and formatting features of Excel to modify the output and save it as a native Excel workbook
(File > Save As and choose "Microsoft Excel Workbook (*.xls)")

All the output tables appear in a single worksheet. Later, we will use ODS to create an XML file that, when opened by Excel, will contain multiple worksheets.

NOTE: In a production environment, you would replace "localhost" with the address of your Web server.



sas The Power to Know.
SAS 9

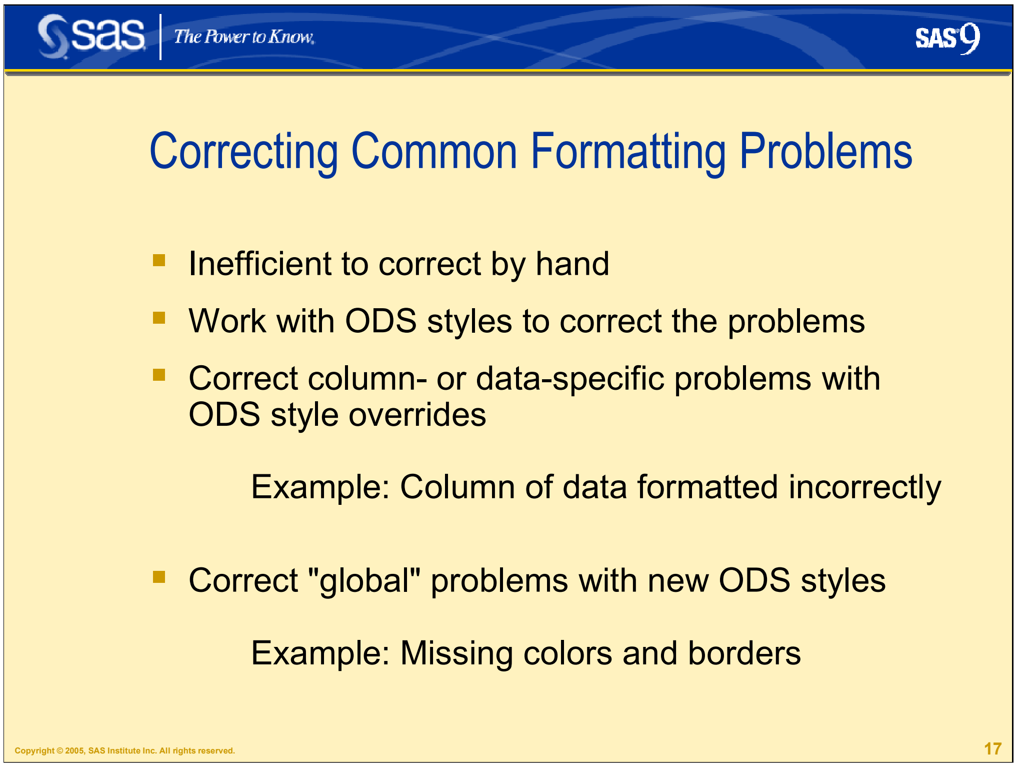
What Happened to our HTML Output?

- Conversion works with varying success
- Excel has a limited color palette so colors get mapped
- Excel applies formats to the data, which is sometimes problematic

Word generally handles the HTML conversion better than Excel. What you see in Word is very close to what you sill see if you opened the HTML file with a Web browser.

Excel has a limited color palette, and maps the unsupported green to gray.

Excel also makes some assumptions about the input data and applies Excel formats. Sometimes these assumptions are not correct, as is the case for the "Date" and "Code" columns.



Correcting Common Formatting Problems

- Inefficient to correct by hand
- Work with ODS styles to correct the problems
- Correct column- or data-specific problems with ODS style overrides

  Example: Column of data formatted incorrectly

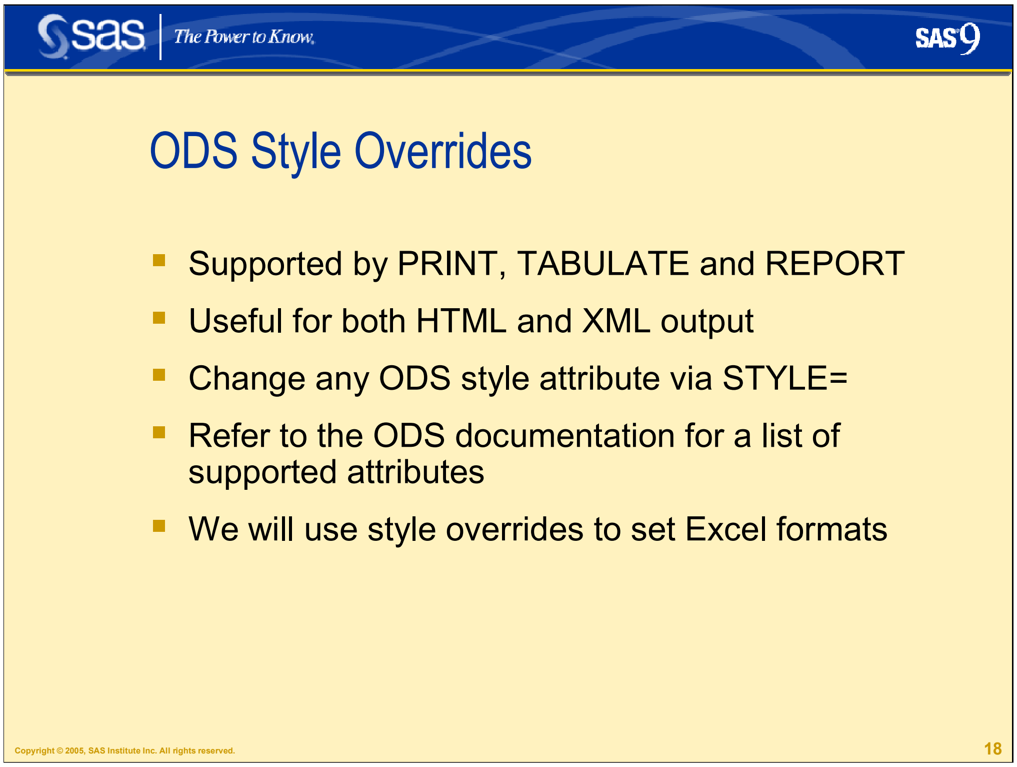
- Correct "global" problems with new ODS styles

  Example: Missing colors and borders

You could correct each individual Excel file by hand, but that would be cumbersome. It is far better to have ODS create the correct file in the first place. Fortunately, we can use ODS styles and style overrides to fix these problems.

For the best efficiency, create an ODS style to correct problems that are related to appearance and/or affect a wide portion of the document. Use ODS style overrides sparingly as extra HTML (or XML) is added to affected tags, making for a larger file.

Style overrides are best used to correct formats or other characteristics of a specific column of data, as there is no way to do this with a style. That is because a style has no way to single out a specific column at run time.



The Power to Know.

ODS Style Overrides

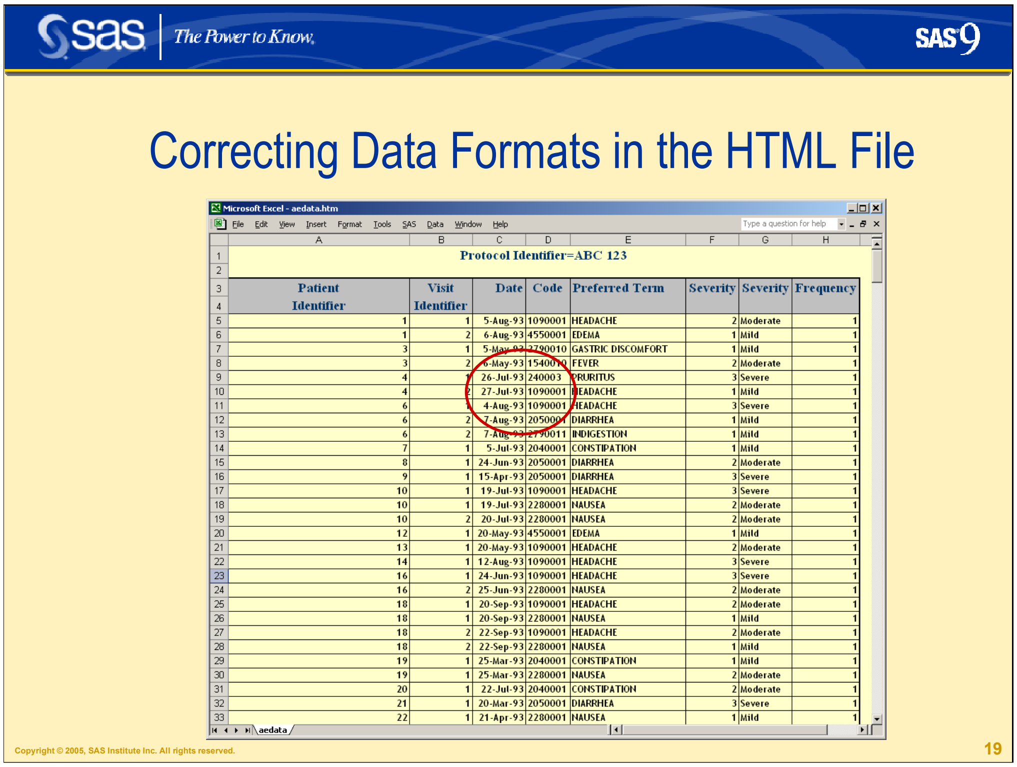
- Supported by PRINT, TABULATE and REPORT
- Useful for both HTML and XML output
- Change any ODS style attribute via STYLE=
- Refer to the ODS documentation for a list of supported attributes
- We will use style overrides to set Excel formats

We can use style overrides to instruct ODS to associate an Excel format with a column. This will cause Excel to use this format instead of automatically assigning one.

ODS Documentation:

"Chapter 5: The TEMPLATE Procedure", *The Complete Guide to the SAS Output Delivery System, Version 8* (see "The STYLE Statement")

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


sas The Power to Know.
SAS 9

Correcting Data Formats in the HTML File

Microsoft Excel - aedata.htm
File Edit View Insert Format Tools SAS Data Window Help
Type a question for help
aedata

Protocol Identifier=ABC 123

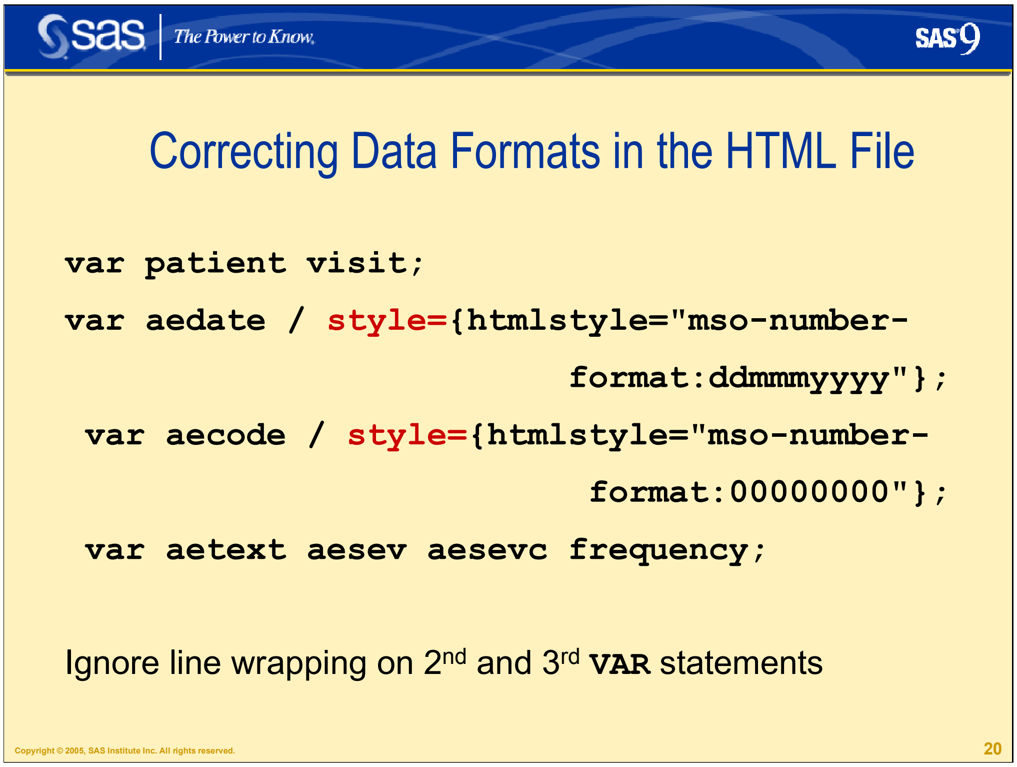
Patient Identifier	Visit Identifier	Date	Code	Preferred Term	Severity	Severity	Frequency
1	1	5-Aug-93	1090001	HEADACHE	2	Moderate	1
1	2	6-Aug-93	4550001	EDEMA	1	Mild	1
3	1	5-May-93	2790010	GASTRIC DISCOMFORT	1	Mild	1
3	2	6-May-93	1540010	FEVER	2	Moderate	1
4	1	26-Jul-93	240003	PRURITUS	3	Severe	1
4	2	27-Jul-93	1090001	HEADACHE	1	Mild	1
6	1	4-Aug-93	1090001	HEADACHE	3	Severe	1
6	2	7-Aug-93	2050001	DIARRHEA	1	Mild	1
6	2	7-Aug-93	2790011	INDIGESTION	1	Mild	1
7	1	5-Jul-93	2040001	CONSTIPATION	1	Mild	1
8	1	24-Jun-93	2050001	DIARRHEA	2	Moderate	1
9	1	15-Apr-93	2050001	DIARRHEA	3	Severe	1
10	1	19-Jul-93	1090001	HEADACHE	3	Severe	1
10	1	19-Jul-93	2280001	NAUSEA	2	Moderate	1
10	2	20-Jul-93	2280001	NAUSEA	2	Moderate	1
12	1	20-May-93	4550001	EDEMA	1	Mild	1
13	1	20-May-93	1090001	HEADACHE	2	Moderate	1
14	1	12-Aug-93	1090001	HEADACHE	3	Severe	1
16	1	24-Jun-93	1090001	HEADACHE	3	Severe	1
16	2	25-Jun-93	2280001	NAUSEA	2	Moderate	1
18	1	20-Sep-93	1090001	HEADACHE	2	Moderate	1
18	1	20-Sep-93	2280001	NAUSEA	1	Mild	1
18	2	22-Sep-93	1090001	HEADACHE	2	Moderate	1
18	2	22-Sep-93	2280001	NAUSEA	1	Mild	1
19	1	25-Mar-93	2040001	CONSTIPATION	1	Mild	1
19	1	25-Mar-93	2280001	NAUSEA	2	Moderate	1
20	1	22-Jul-93	2040001	CONSTIPATION	2	Moderate	1
21	1	20-Mar-93	2050001	DIARRHEA	3	Severe	1
22	1	21-Apr-93	2280001	NAUSEA	1	Mild	1

Here is the ODS-generated HTML file viewed using Excel.

Excel applied an improper date format to the "Date" column. While this does not result in a grossly incorrect display of the data, it does differ from the Web and Word views.

The "General" format was automatically applied to the "Code" column, resulting in a loss of the leading zeroes.

Use the VAR statements on the following slide to apply style overrides to correct both of these problems.



Correcting Data Formats in the HTML File

```
var patient visit;
var aedate / style={htmlstyle="mso-number-
                              format:ddmmmyyyy"};
 var aecode / style={htmlstyle="mso-number-
                              format:00000000"};
 var aetext aesev aesevc frequency;
```

Ignore line wrapping on 2nd and 3rd **VAR** statements

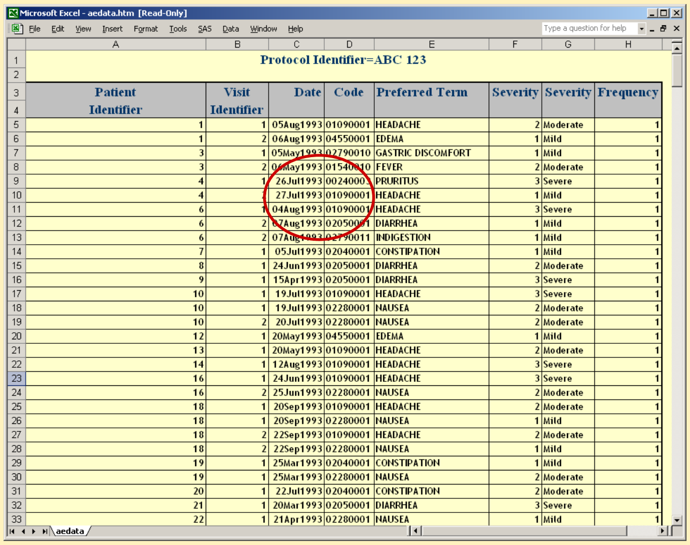
HTMLSTYLE is used to add arbitrary Cascading Style Sheet (CSS) attributes to the HTML generated by ODS. You must specify the complete text of the CSS attribute. Refer to the ODS documentation for more information about HTMLSTYLE; a general HTML reference book or online resource can be consulted to learn more about CSS.

CSS attributes starting with "mso-" are proprietary Microsoft attributes intended for use with Microsoft Office. Although these attributes appear in the HTML file, Web browsers gracefully ignore them. Thus, you can view these HTML pages with a Web browser, Excel or Word.

The "mso-number-format" attribute is used to assign Excel formats to a column of data. In this case, we are using ODS to instruct Excel to use its "ddmmmyyyy" format for the column that contains the date. This format is comparable to the SAS DATE9. format.

Similarly, Excel will use the "00000000" format for the column that contains the adverse event code. This format is comparable to the SAS Z8. format.

Correcting Data Formats in the HTML File


Microsoft Excel - aedata.htm [Read-Only]
File Edit View Insert Format Tools SAS Data Window Help
Type a question for help
aedata

Protocol Identifier=ABC 123

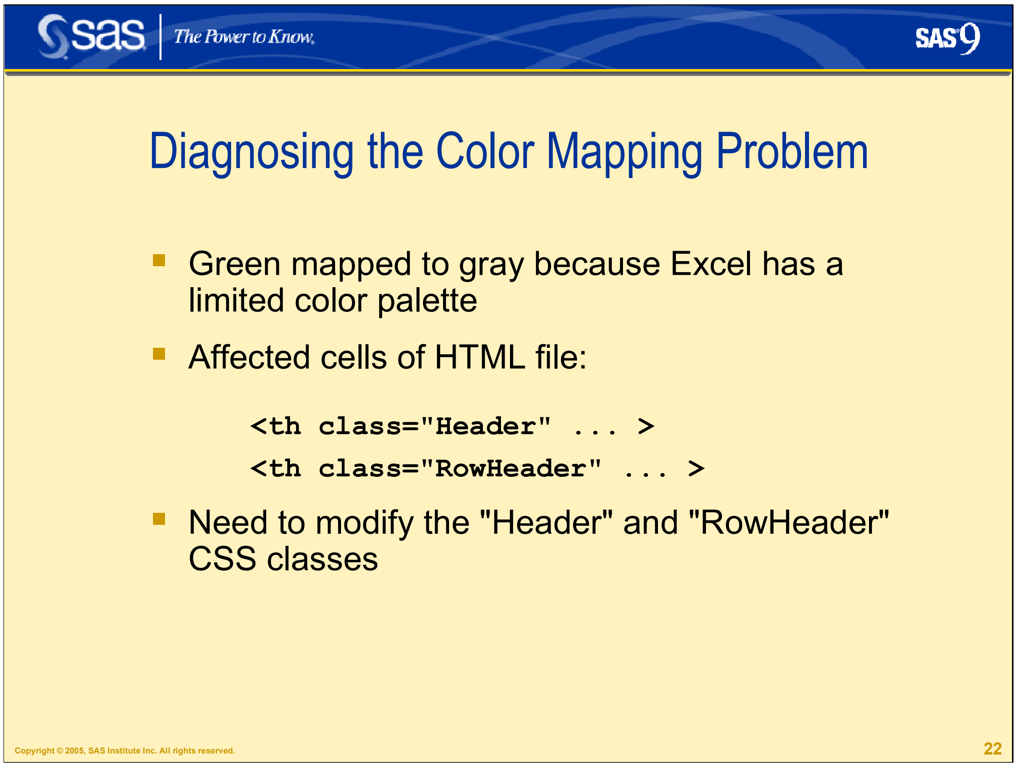
Patient Identifier	Visit Identifier	Date	Code	Preferred Term	Severity	Severity	Frequency
1	1	05Aug1993	01090001	HEADACHE	2	Moderate	1
1	2	06Aug1993	04550001	EDEMA	1	Mild	1
3	1	05May1993	02790010	GASTRIC DISCOMFORT	1	Mild	1
3	2	06May1993	01540010	FEVER	2	Moderate	1
4	1	26Jul1993	00240003	PRURITUS	3	Severe	1
4	2	27Jul1993	01090001	HEADACHE	1	Mild	1
6	1	04Aug1993	01090001	HEADACHE	3	Severe	1
6	2	07Aug1993	02050001	DIARRHEA	1	Mild	1
6	2	07Aug1993	02790011	INDIGESTION	1	Mild	1
7	1	05Jul1993	02040001	CONSTIPATION	1	Mild	1
8	1	24Jun1993	02050001	DIARRHEA	2	Moderate	1
9	1	15Apr1993	02050001	DIARRHEA	3	Severe	1
10	1	19Jul1993	01090001	HEADACHE	3	Severe	1
10	1	19Jul1993	02280001	NAUSEA	2	Moderate	1
10	2	20Jul1993	02280001	NAUSEA	2	Moderate	1
12	1	20May1993	04550001	EDEMA	1	Mild	1
13	1	20May1993	01090001	HEADACHE	2	Moderate	1
14	1	12Aug1993	01090001	HEADACHE	3	Severe	1
16	1	24Jun1993	01090001	HEADACHE	3	Severe	1
16	2	25Jun1993	02280001	NAUSEA	2	Moderate	1
18	1	20Sep1993	01090001	HEADACHE	2	Moderate	1
18	1	20Sep1993	02280001	NAUSEA	1	Mild	1
18	2	22Sep1993	01090001	HEADACHE	2	Moderate	1
18	2	22Sep1993	02280001	NAUSEA	1	Mild	1
19	1	25Mar1993	02040001	CONSTIPATION	1	Mild	1
19	1	25Mar1993	02280001	NAUSEA	2	Moderate	1
20	1	22Jul1993	02040001	CONSTIPATION	2	Moderate	1
21	1	20Mar1993	02050001	DIARRHEA	3	Severe	1
22	1	21Apr1993	02280001	NAUSEA	1	Mild	1

The image above shows the result of opening the partially corrected HTML file using Excel. The HTML file was created by applying the style overrides shown on the previous slide, which correct the appearance of the Date and Code columns.

The SAS code corresponding to this change can be found in the file "makeHTML-2.sas".

In the interest of saving time, we will not view or submit this code now. The next code we submit will incorporate this and other changes.

If you view this HTML file with Word, it will look the same as the original HTML file created earlier. Word also ignores the "mso-number-format" CSS attribute.



sas The Power to Know.
SAS 9

Diagnosing the Color Mapping Problem

- Green mapped to gray because Excel has a limited color palette
- Affected cells of HTML file:

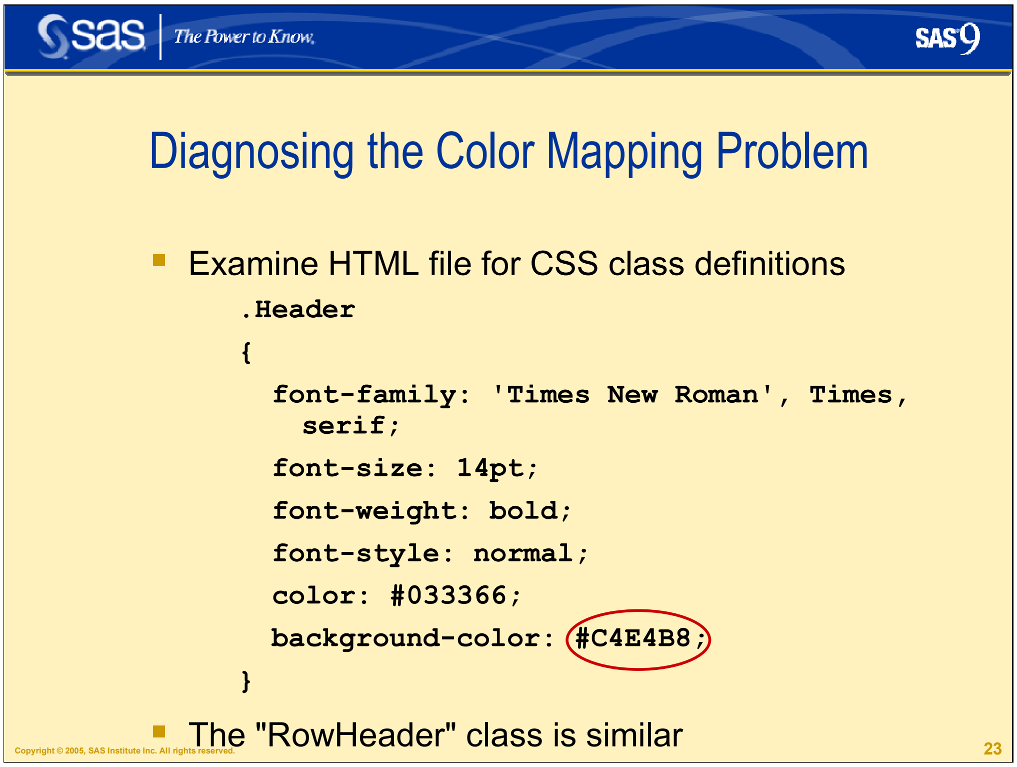
```
<th class="Header" ... >
<th class="RowHeader" ... >
```

- Need to modify the "Header" and "RowHeader" CSS classes

The final formatting problem that needs to be corrected is the case where Excel maps the green cell background color to gray. This happens because by default, Excel does not support the shade of green that ODS used when it created the HTML file.

If you examine the HTML file using a text editor, you can determine that the cells being affected have one thing in common: they are using the "Header" or "RowHeader" CSS classes.

CSS classes are used to control the appearance of HTML elements. The ODS style is responsible for controlling these CSS class definitions, which are also placed inside the HTML file.



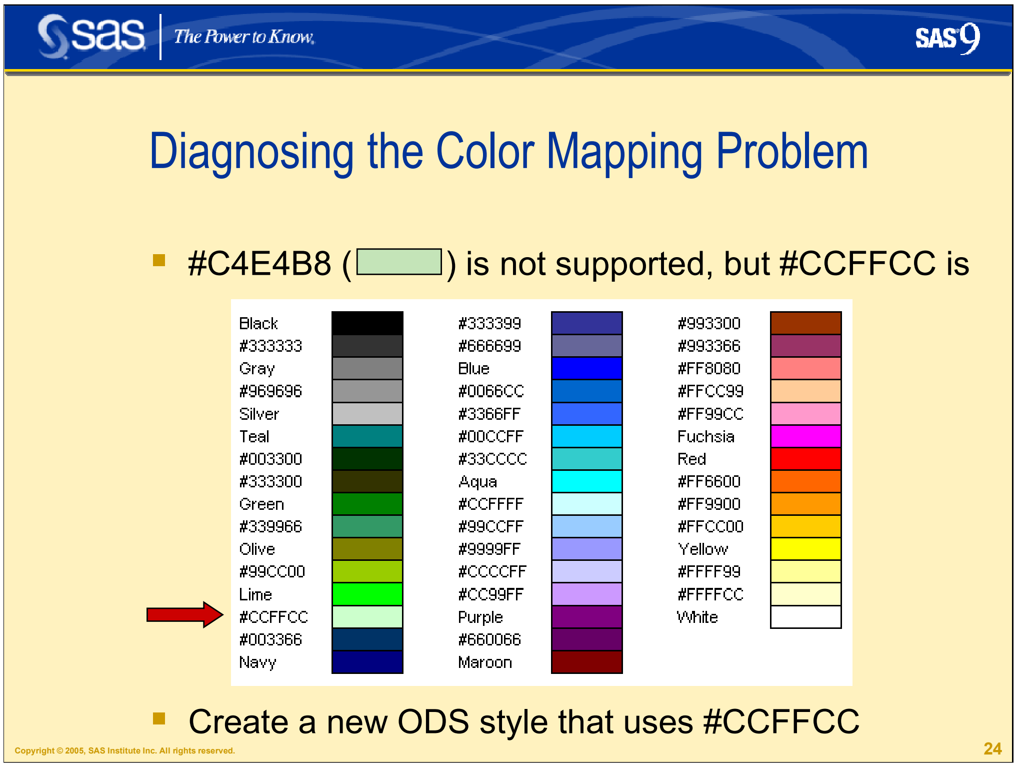
Diagnosing the Color Mapping Problem

- Examine HTML file for CSS class definitions

```
.Header
{
  font-family: 'Times New Roman', Times,
    serif;
  font-size: 14pt;
  font-weight: bold;
  font-style: normal;
  color: #033366;
  background-color: #C4E4B8;
}
```

- The "RowHeader" class is similar

Examination of the "Header" and "RowHeader" CSS classes (in the HTML file) reveals that the color "#C4E4B8" (a shade of green) is being used as the background color.



Diagnosing the Color Mapping Problem

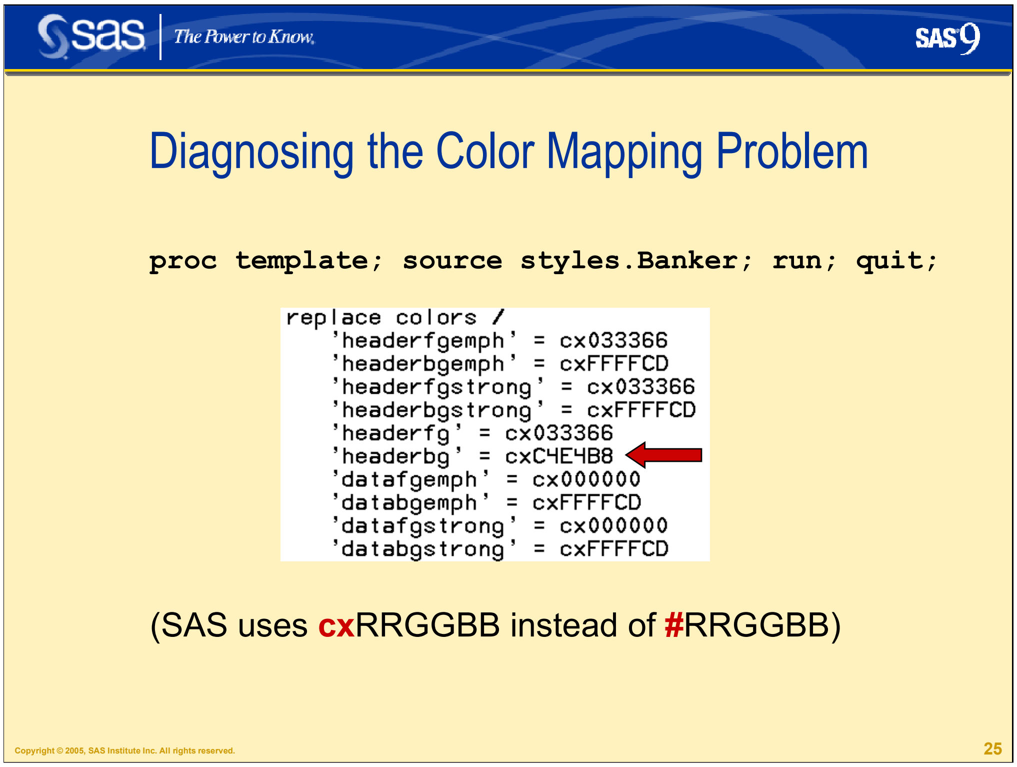
- #C4E4B8 (🟩) is not supported, but #CCFFCC is

Black
#333333
Gray
#969696
Silver
Teal
#003300
#333300
Green
#339966
Olive
#99CC00
Lime
#CCFFCC
#003366
Navy
#333399
#666699
Blue
#0066CC
#3366FF
#00CCFF
#33CCCC
Aqua
#CCFFFF
#99CCFF
#9999FF
#CCCCFF
#CC99FF
Purple
#660066
Maroon
#993300
#993366
#FF8080
#FFCC99
#FF99CC
Fuchsia
Red
#FF6600
#FF9900
#FFCC00
Yellow
#FFFF99
#FFFFCC
White

- Create a new ODS style that uses #CCFFCC

The graphic above shows the colors that Excel supports by default. It is clear that "#C4E4B8" is not supported, but a similar color, "#CCFFCC" is. We will make a new ODS style that uses the supported color.

NOTE: The graphic above shows the supported colors for an unaltered version of Excel. It is possible for each individual Excel client to have a different, customized color template. This is accomplished via the menu selection Tools > Options and then selecting the "Color" tab. Thus the technique of picking a color that appears in the graphic above is not foolproof.



Diagnosing the Color Mapping Problem

```
proc template; source styles.Banker; run; quit;
```

```
replace colors /
    'headerfgemph' = cx033366
    'headerbgemph' = cxFFFFCD
    'headerfgstrong' = cx033366
    'headerbgstrong' = cxFFFFCD
    'headerfg' = cx033366
    'headerbg' = cxC4E4B8
    'datafgemph' = cx000000
    'databgemph' = cxFFFFCD
    'datafgstrong' = cx000000
    'databgstrong' = cxFFFFCD
```

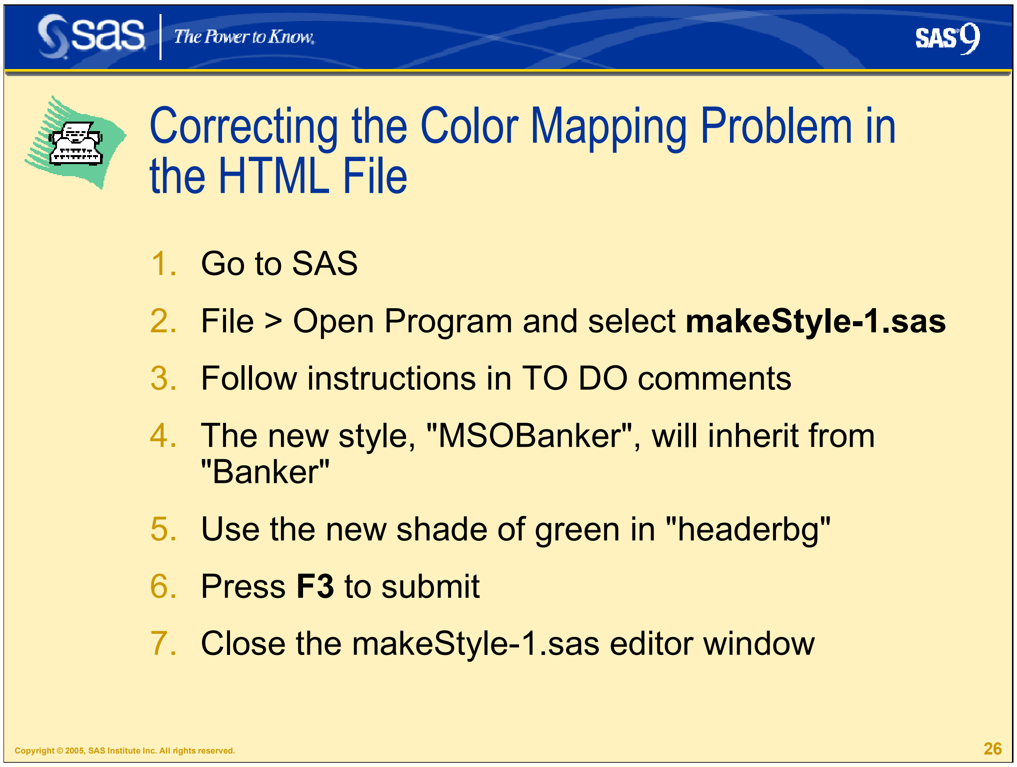
(SAS uses **cx**RRGGBB instead of **#**RRGGBB)

Instead of making a new ODS style from scratch, we will make one that is based on the "Banker" style. Since the only problem with the "Banker" style is that it has a "bad" color, it is much easier to start with that style and change only the color.

We must examine the source code of the "Banker" style to determine where the color is set, and how to change it. Use the TEMPLATE procedure to list the source code for the style, and look in the Log window for the offending color.

The graphic above shows a partial listing of the source code for the "Banker" style.

By reviewing the source code, we can see that the "replace colors" block of code is where the header background color ("headerbg") is set to "cxC4E4B8". Note that SAS uses the "cx" notation, while HTML uses the "#" notation when referencing RGB colors.



The Power to Know.

Correcting the Color Mapping Problem in the HTML File

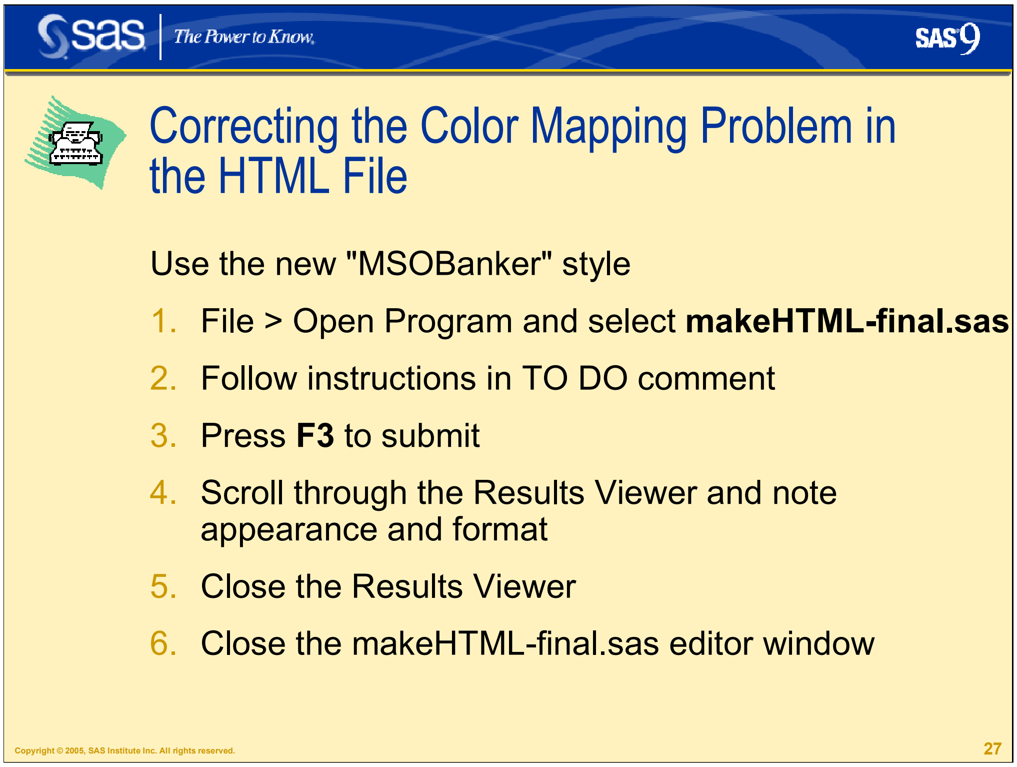
1. Go to SAS
2. File > Open Program and select **makeStyle-1.sas**
3. Follow instructions in TO DO comments
4. The new style, "MSOBanker", will inherit from "Banker"
5. Use the new shade of green in "headerbg"
6. Press **F3** to submit
7. Close the makeStyle-1.sas editor window

1. We will call our new style "MSOBanker". On the "define style" statement, change "newStyleName" to "**MSOBanker**".

2. Since we want this style to inherit all the properties of the Banker style, change "oldStyleName" to "**Banker**" on the "parent" statement.

3. Finally, change the header background color ("headerbg") from "cxC4E4B8" to "**cxCCFFCC**".

The new style will be stored in the permanent library MYLIB due to the ODS PATH settings specified earlier (in "setup.sas"). Thus, the new style can be made available to everyone at your site.



sas The Power to Know.
SAS 9

Correcting the Color Mapping Problem in the HTML File

Use the new "MSOBanker" style

1. File > Open Program and select **makeHTML-final.sas**
2. Follow instructions in TO DO comment
3. Press **F3** to submit
4. Scroll through the Results Viewer and note appearance and format
5. Close the Results Viewer
6. Close the makeHTML-final.sas editor window

The SAS code you are now viewing contains the ODS style overrides to correct the date and leading zero problem discussed earlier.

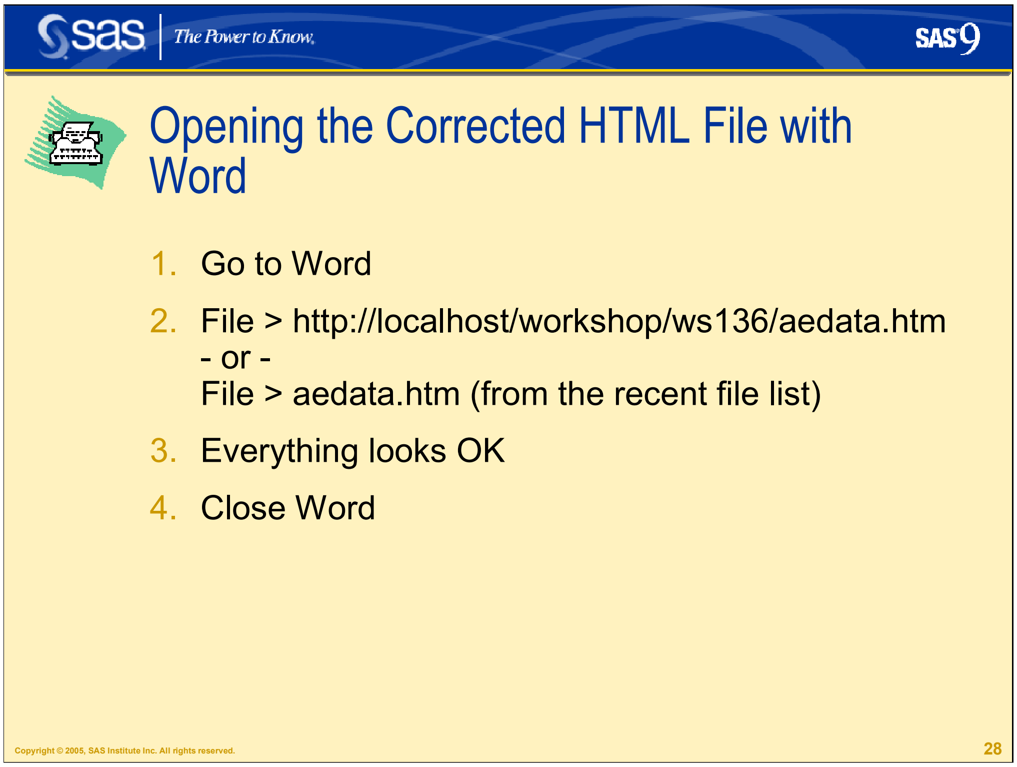
The only change that is needed is to use the "MSOBanker" style instead of the "Banker" style.

Change `style=Banker` to `style=`**`MSOBanker`**

Final Code:

```
  ods tagsets.MSOffice2K style=MSOBanker ...
```

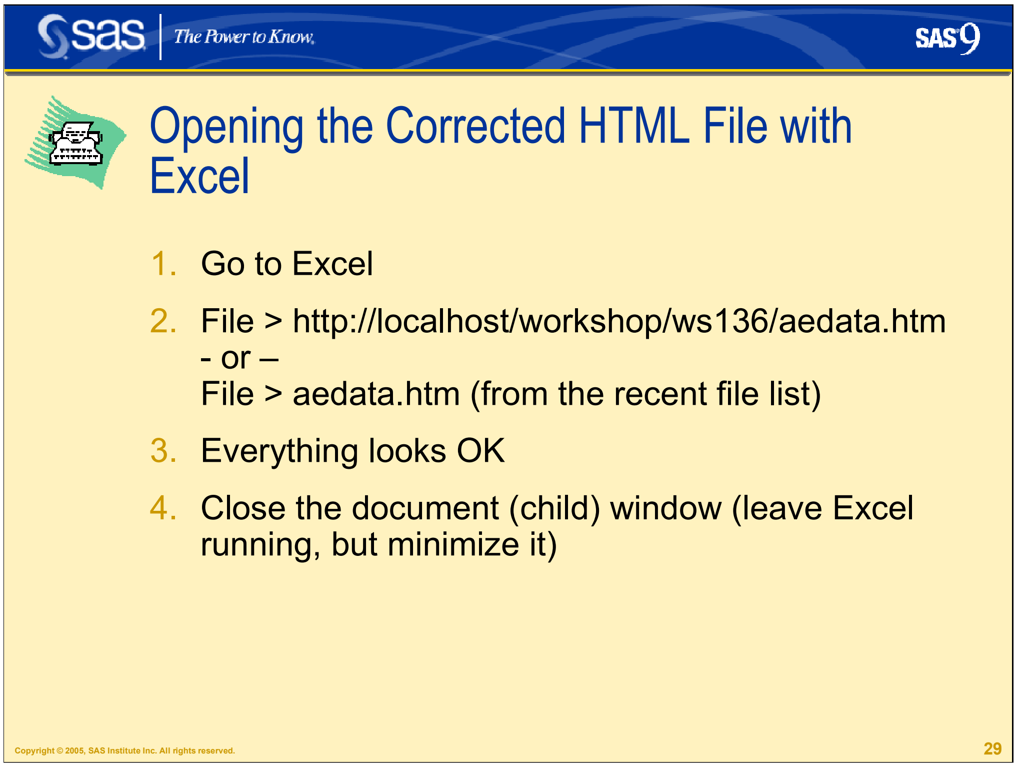
Scroll through the Results Viewer to verify that the new color of green is used (this color is brighter than the original).



Opening the Corrected HTML File with Word

1. Go to Word
2. File > http://localhost/workshop/ws136/aedata.htm
   - or -
   File > aedata.htm (from the recent file list)
3. Everything looks OK
4. Close Word

The new shade of green is used for background color of the header and row header cells.



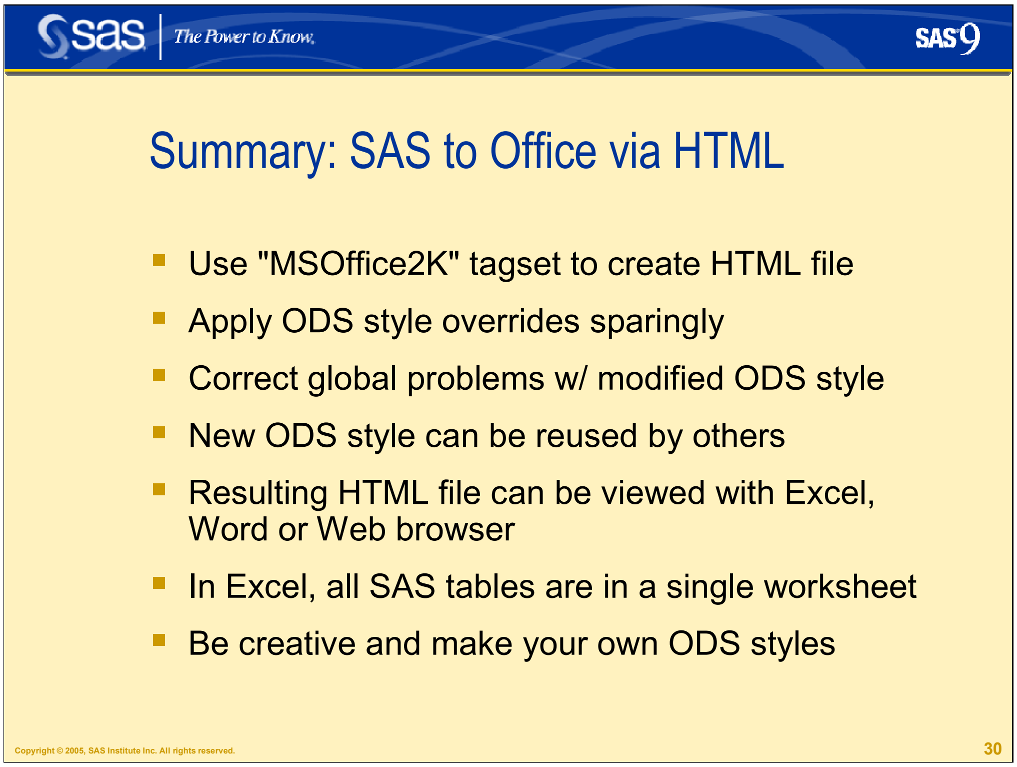
Opening the Corrected HTML File with Excel

1. Go to Excel
2. File > http://localhost/workshop/ws136/aedata.htm
   - or –
   File > aedata.htm (from the recent file list)
3. Everything looks OK
4. Close the document (child) window (leave Excel running, but minimize it)

The new shade of green is used for background color of the header and row header cells.

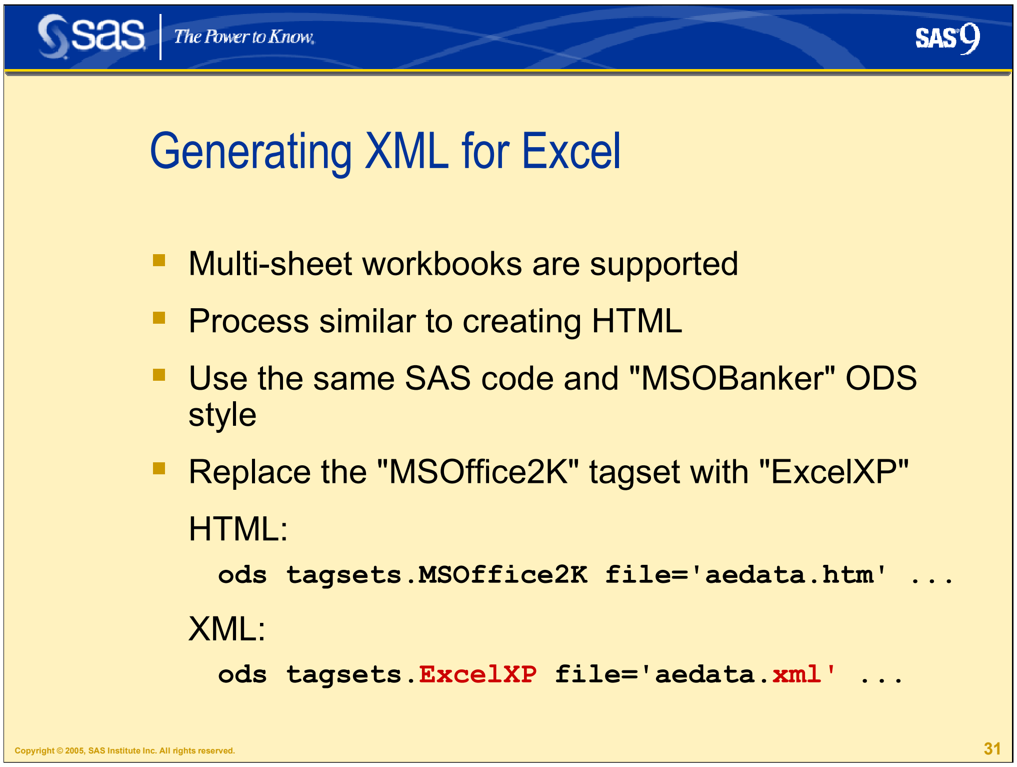
Dates are formatted correctly.

Leading zeroes are retained in the adverse event code.



Summary: SAS to Office via HTML

- Use "MSOffice2K" tagset to create HTML file
- Apply ODS style overrides sparingly
- Correct global problems w/ modified ODS style
- New ODS style can be reused by others
- Resulting HTML file can be viewed with Excel, Word or Web browser
- In Excel, all SAS tables are in a single worksheet
- Be creative and make your own ODS styles



Generating XML for Excel

- Multi-sheet workbooks are supported
- Process similar to creating HTML
- Use the same SAS code and "MSOBanker" ODS style
- Replace the "MSOffice2K" tagset with "ExcelXP"

HTML:

```
ods tagsets.MSOffice2K file='aedata.htm' ...
```

XML:

```
ods tagsets.ExcelXP file='aedata.xml' ...
```

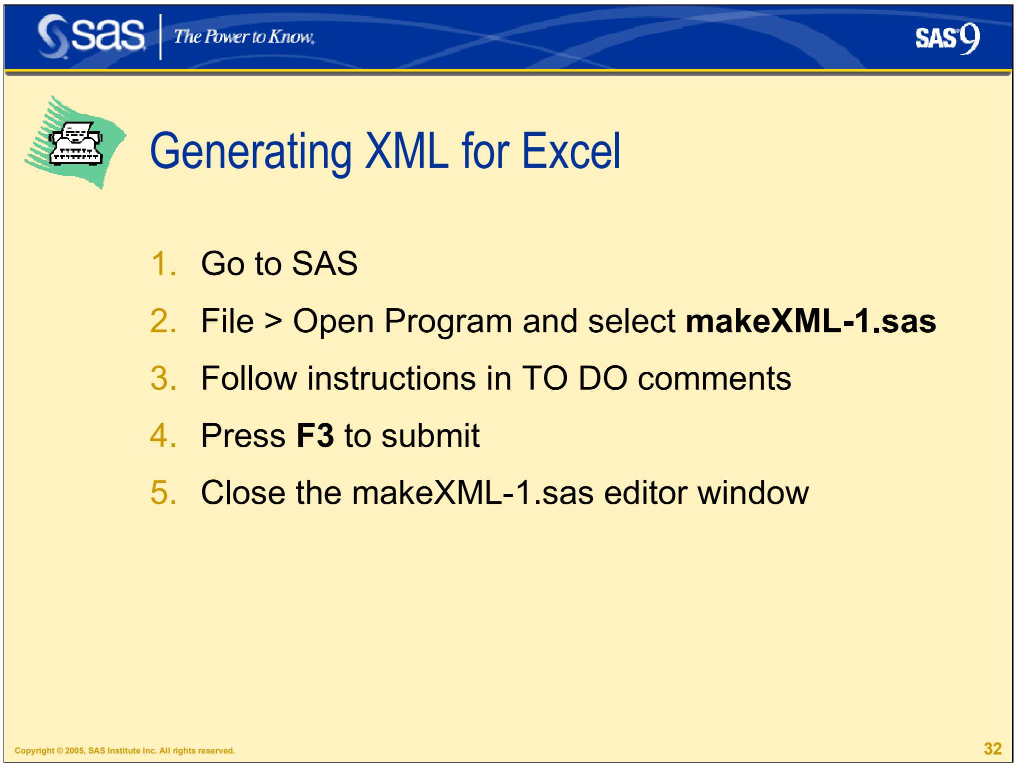
One advantage of using ODS to generate XML instead of HTML is that with XML, your SAS output appears in separate sheets of a workbook.

Also, the ExcelXP tagset supports a number of useful features, such as autofilters and frozen panes. To see the features supported by the ExcelXP tagset, submit this code:

```
ods tagsets.ExcelXP file='c:\temp\temp.xml'
  options(doc='help');
ods tagsets.ExcelXP close;
```

Information will be printed to the SAS Log.

Note, however, Microsoft Excel does not support graphics in their XML specification.



Generating XML for Excel

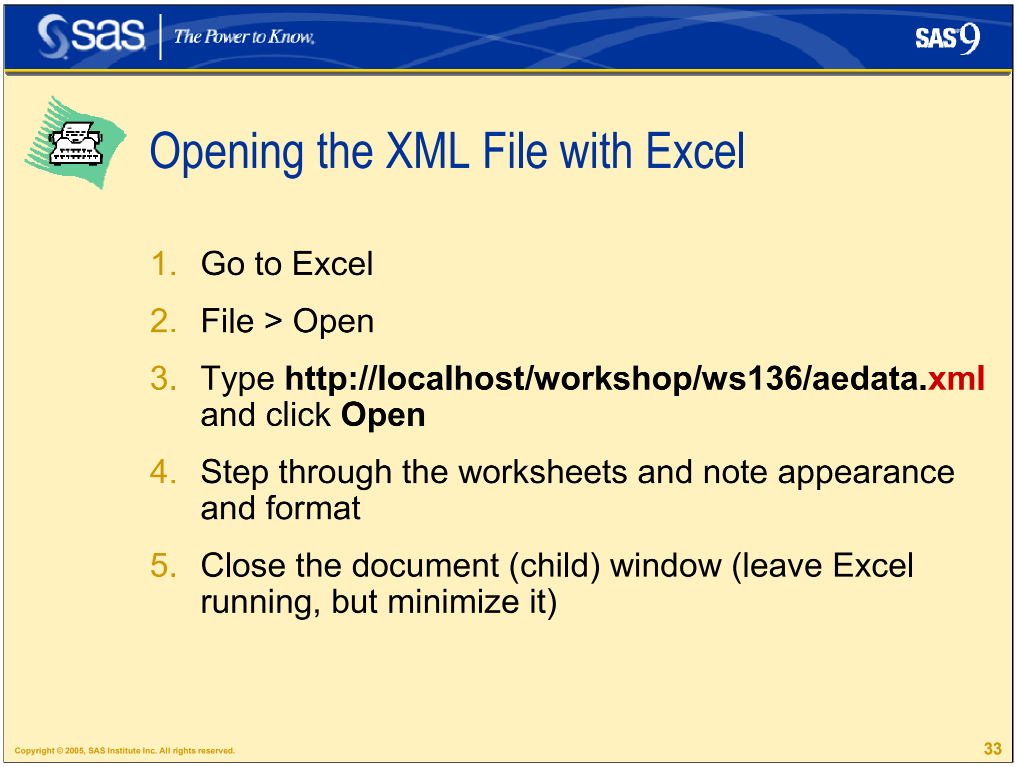
1. Go to SAS
2. File > Open Program and select **makeXML-1.sas**
3. Follow instructions in TO DO comments
4. Press **F3** to submit
5. Close the makeXML-1.sas editor window

This is the same code that was used to make the very first HTML file, except the new "MSOBanker" style is used instead of "Banker". Using the "MSOBanker" style insures that our background colors will be correct.

We need to use the "ExcelXP" tagset, which creates XML, instead of the "MSOffice2K" tagset. We also want the file extension to indicate that the file contains XML content, not HTML.

1. On the opening ODS statement, change "MSOffice2K" to "**ExcelXP**".
2. On the opening ODS statement, also change "aedata.htm" to "aedata.**xml"**.
3. Change "MSOffice2K" to **"ExcelXP"** in the closing ODS statement.

NOTE: Performance is much better in SAS 9.1.3 than in SAS 9.1.2.



sas The Power to Know.
SAS9

Opening the XML File with Excel

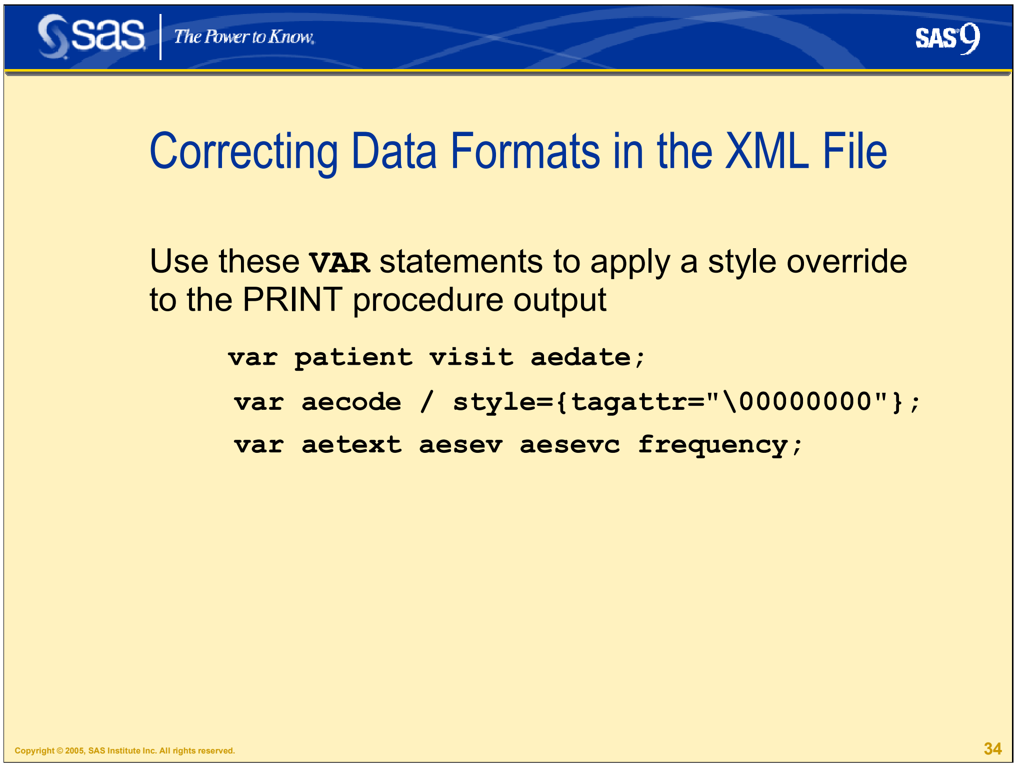
1. Go to Excel
2. File > Open
3. Type **http://localhost/workshop/ws136/aedata.xml** and click **Open**
4. Step through the worksheets and note appearance and format
5. Close the document (child) window (leave Excel running, but minimize it)

Note that a separate worksheet was generated for each SAS table.

The "MSOBanker" style insures that the header and row header background color is green. However, we once again have the problem with the leading zeroes being dropped from the adverse event code column because Excel automatically applied the "General" format. Additionally, there are no cell border lines because Excel interprets XML cell border line specifications differently from HTML. We will use ODS to correct both of these issues.

Some of the columns are not autosized because according to the Microsoft XML Spreadsheet Reference, Excel will only autofit columns that contain date or numeric data. So unfortunately, you will have to make this correction yourself. Refer to the Microsoft Excel Help system for details (search for "change column width"). A future release of the tagset will attempt to correct this problem.

NOTE: In a production environment, you would replace "localhost" with the address of your Web server.



Correcting Data Formats in the XML File

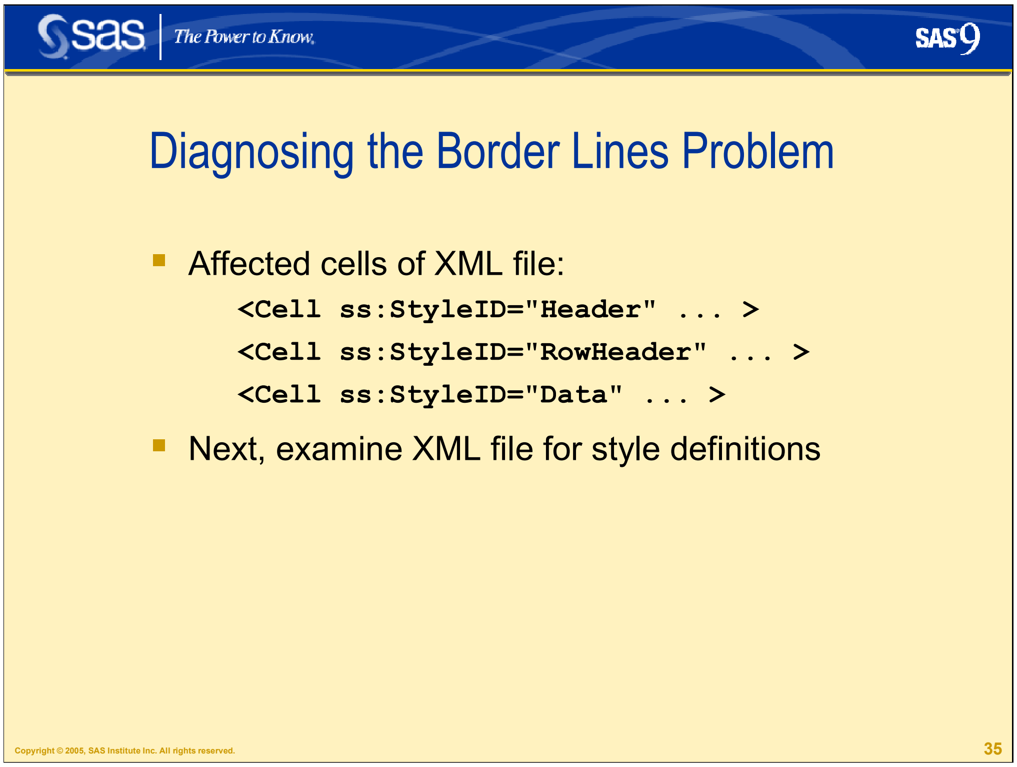
Use these **`VAR`** statements to apply a style override to the PRINT procedure output

```
var patient visit aedate;
var aecode / style={tagattr="\00000000"};
var aetext aesev aesevc frequency;
```

As was the case with the HTML file, we can use an ODS style override to correct the missing leading zero problem. The syntax for the "ExcelXP" tagset is slightly different from HTML, but the concept is the same.

The SAS code corresponding to this change can be found in the file "makeXML-2.sas".

In the interest of saving time, we will not view or submit this code now. The next code we submit will incorporate this and other changes.



Diagnosing the Border Lines Problem

- Affected cells of XML file:

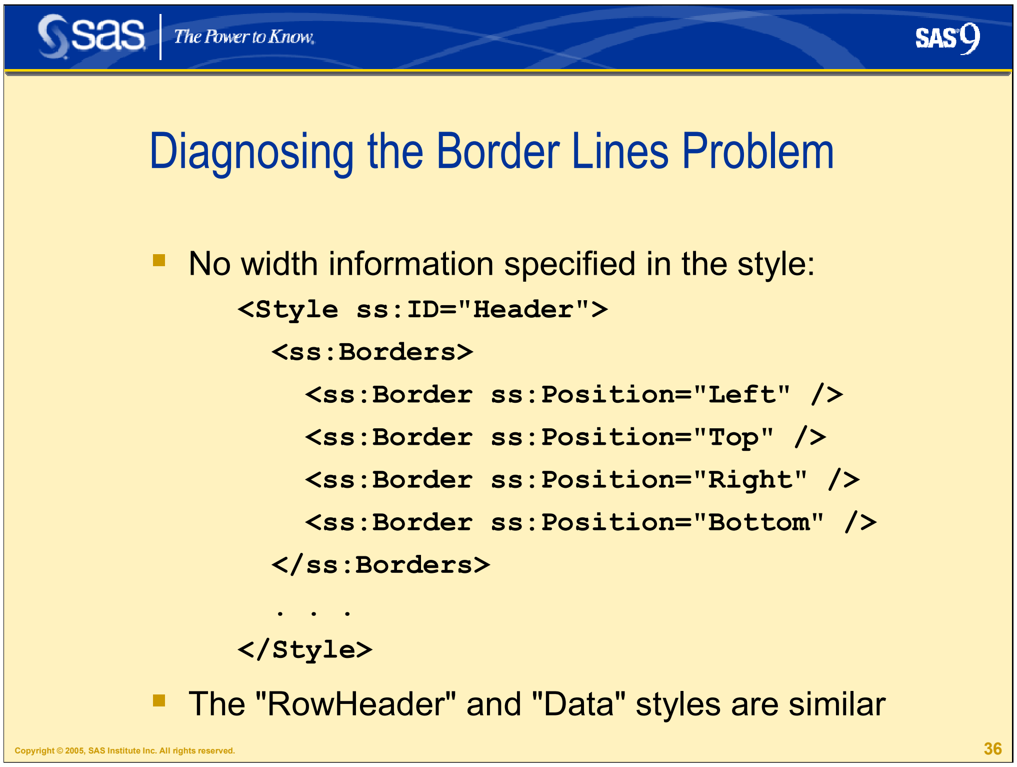
```
<Cell ss:StyleID="Header" ... >
<Cell ss:StyleID="RowHeader" ... >
<Cell ss:StyleID="Data" ... >
```

- Next, examine XML file for style definitions

The final formatting problem that needs to be corrected is the missing cell border lines. This happens because Excel interprets XML cell border line specifications differently from HTML.

If you examine the XML file using a text editor, you can determine that the cells being affected have one thing in common: they have "Header", "RowHeader" or "Data" as a value in the "StyleID" attribute.

These XML style definitions are similar in concept to the CSS style definitions we saw when analyzing the HTML file. The XML style definitions are responsible for the appearance characteristics of the data. As was the case with HTML, the ODS style is responsible for controlling these XML style definitions.



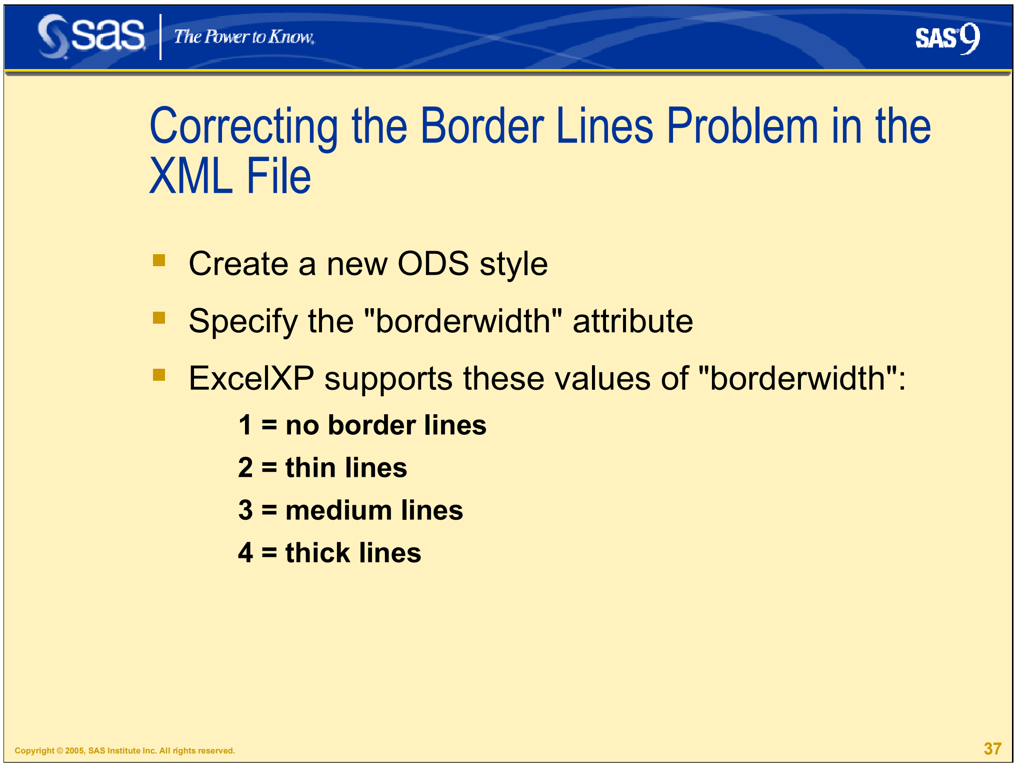
Diagnosing the Border Lines Problem

- No width information specified in the style:

```
<Style ss:ID="Header">
  <ss:Borders>
    <ss:Border ss:Position="Left" />
    <ss:Border ss:Position="Top" />
    <ss:Border ss:Position="Right" />
    <ss:Border ss:Position="Bottom" />
  </ss:Borders>
  . . .
</Style>
```

- The "RowHeader" and "Data" styles are similar

If you further examine the source of the XML file, you will see that the "Header", "RowHeader" and "Data" style definitions do not specify width information for the border element. Thus, Excel is assuming that you do not want border lines.



Correcting the Border Lines Problem in the XML File

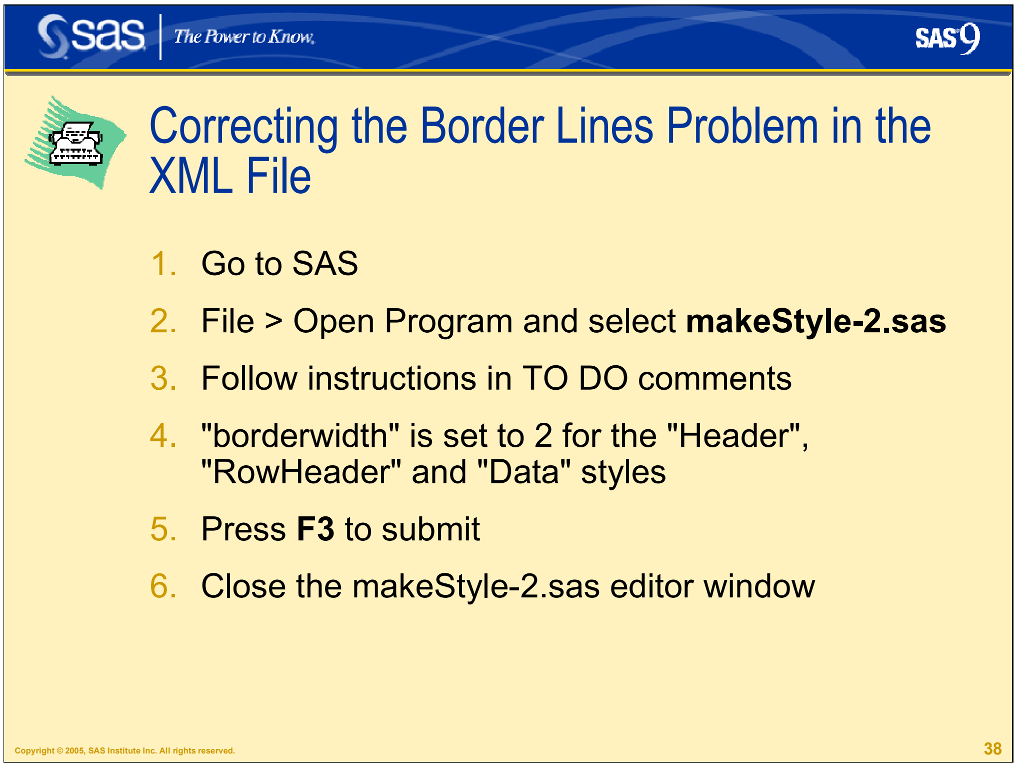
- Create a new ODS style
- Specify the "borderwidth" attribute
- ExcelXP supports these values of "borderwidth":

1 = no border lines
2 = thin lines
3 = medium lines
4 = thick lines

Since the "MSOBanker" style has served us well up to this point, we will correct the missing border lines by making a new ODS style based on it.

Because the missing border lines are only a problem with Excel XML, it is best to use the "MSOBanker" style for generating HTML output, and to create a new style for creating XML output.

A review of the ODS documentation revealed that the "borderwidth" style attribute needs to be modified in order to change the width of border lines.



Correcting the Border Lines Problem in the XML File

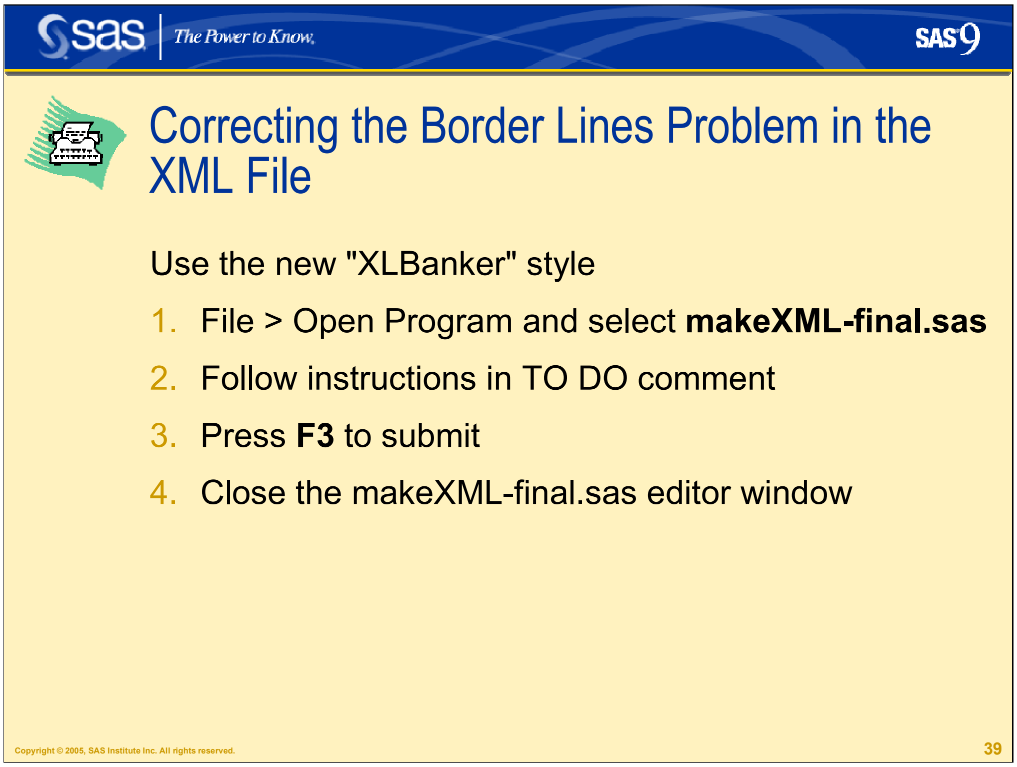
1. Go to SAS
2. File > Open Program and select **makeStyle-2.sas**
3. Follow instructions in TO DO comments
4. "borderwidth" is set to 2 for the "Header", "RowHeader" and "Data" styles
5. Press **F3** to submit
6. Close the makeStyle-2.sas editor window

1. We will call our new style "XLBanker". On the "define style" statement, change "newStyleName" to "**XLBanker**".

2. Since we want this style to inherit all the properties of the "MSOBanker" style, change "oldStyleName" to "**MSOBanker**" on the "parent" statement.

3. Note the "borderwidth" value of "2" specifies thin border lines.

The new style will be stored in the permanent library MYLIB due to the ODS PATH settings specified earlier (in "setup.sas"). Thus, the new style can be made available to everyone at your site.



Correcting the Border Lines Problem in the XML File

Use the new "XLBanker" style

1. File > Open Program and select **makeXML-final.sas**
2. Follow instructions in TO DO comment
3. Press **F3** to submit
4. Close the makeXML-final.sas editor window

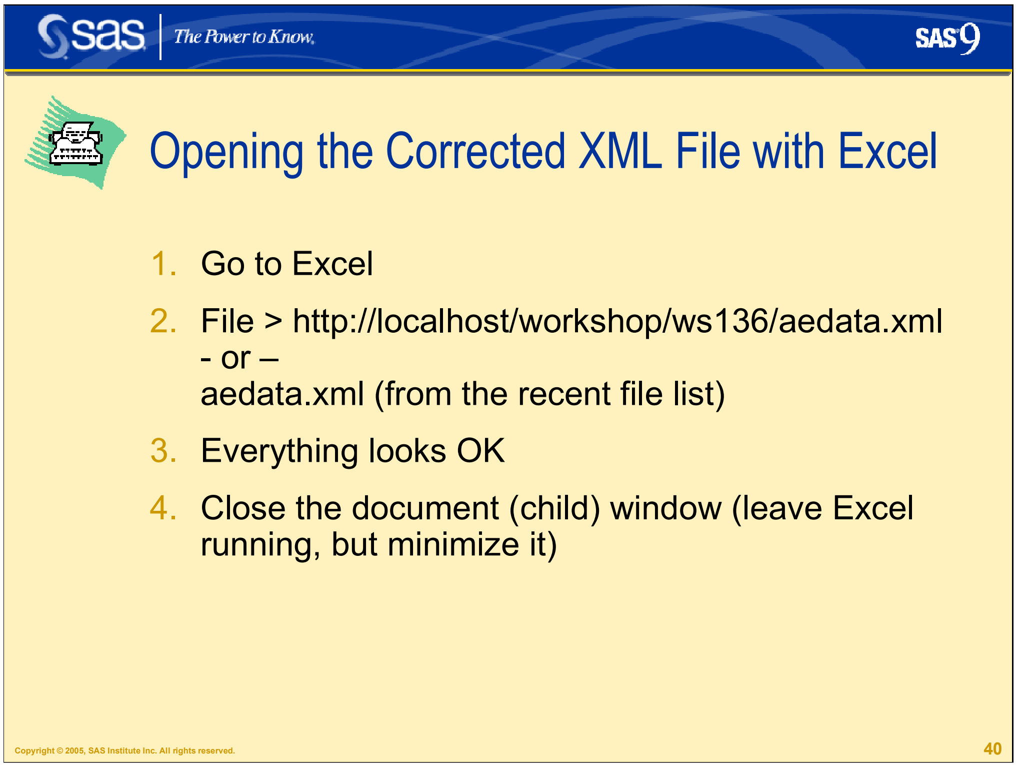
The SAS code you are now viewing contains the ODS style override to correct the leading zero problem discussed earlier.

The only change that is needed is to use the "XLBanker" style instead of the "MSOBanker" style. This will correct the missing border lines problem.

Change `style=MSOBanker` to `style=XLBanker`

Final Code:

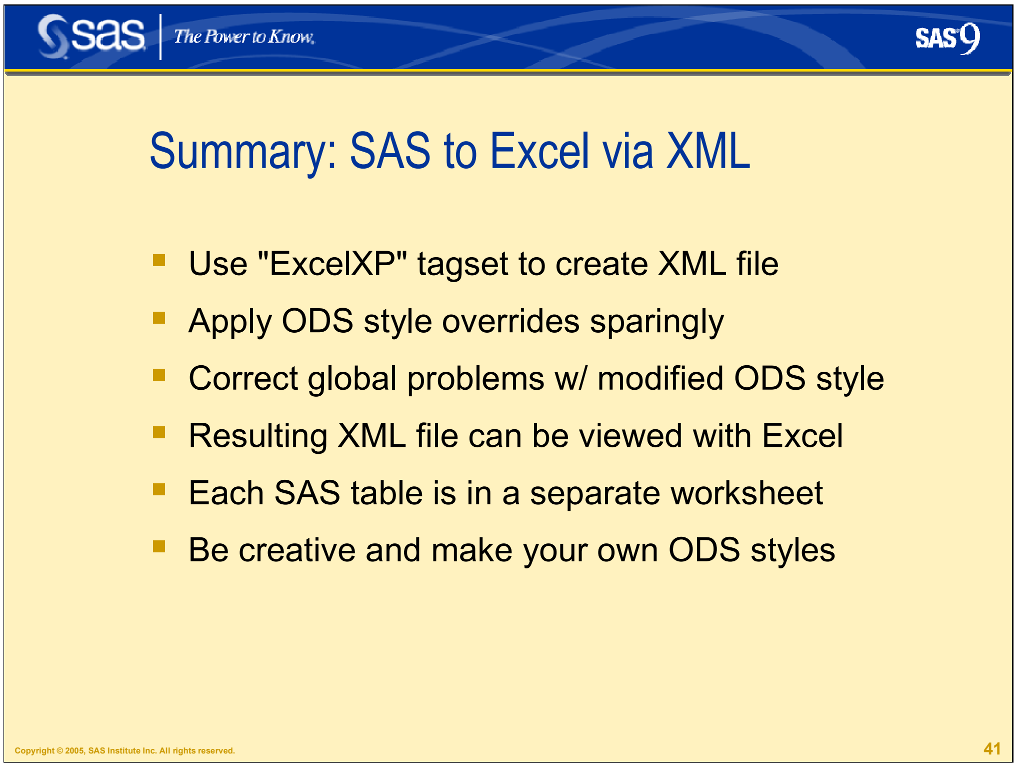
```
ods tagsets.ExcelXP style=XLBanker ...
```



Opening the Corrected XML File with Excel

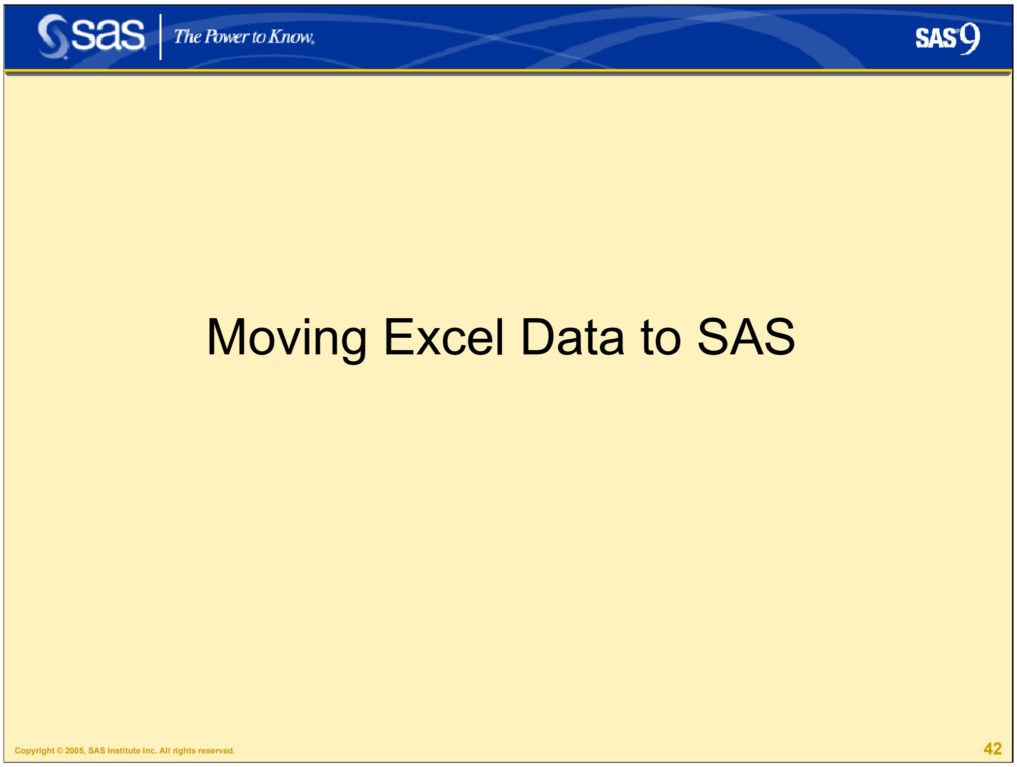
1. Go to Excel
2. File > http://localhost/workshop/ws136/aedata.xml
   - or –
   aedata.xml (from the recent file list)
3. Everything looks OK
4. Close the document (child) window (leave Excel running, but minimize it)

All cells now have border lines.



Summary: SAS to Excel via XML

- Use "ExcelXP" tagset to create XML file
- Apply ODS style overrides sparingly
- Correct global problems w/ modified ODS style
- Resulting XML file can be viewed with Excel
- Each SAS table is in a separate worksheet
- Be creative and make your own ODS styles

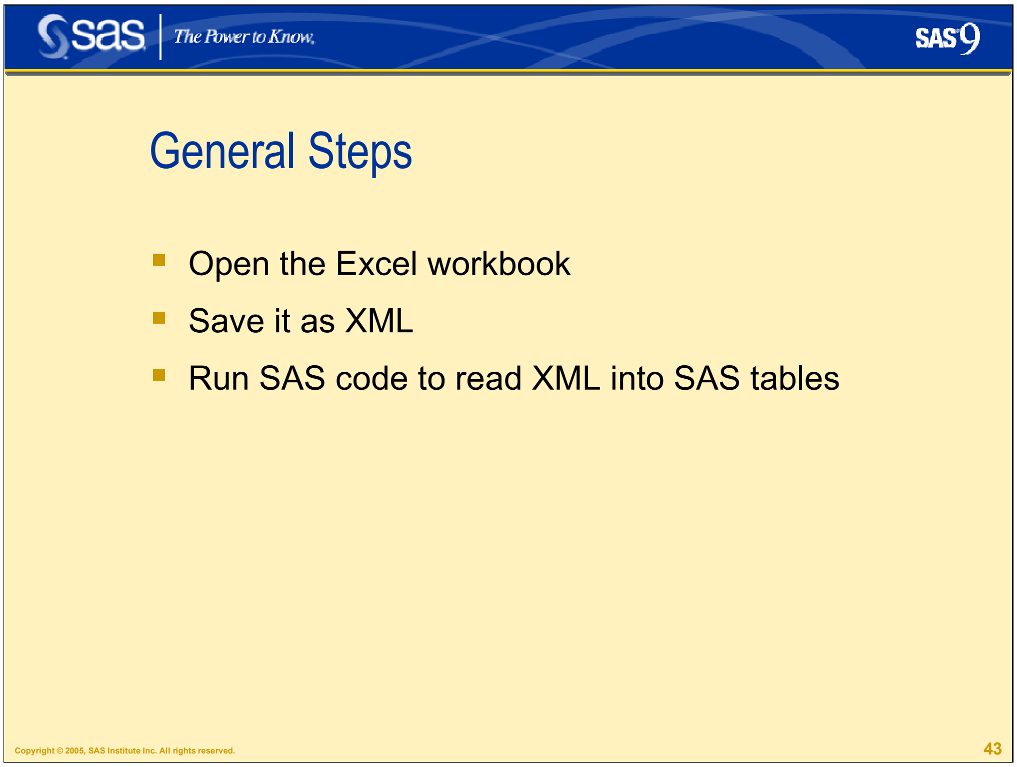


The Power to Know.

Moving Excel Data to SAS

This section focuses on importing Excel workbooks into SAS tables.

Excel XP (a.k.a. Excel 2002) or later is required for this operation because your Excel workbooks must first be saved as XML files. Only Excel XP and later support saving workbooks in XML format.

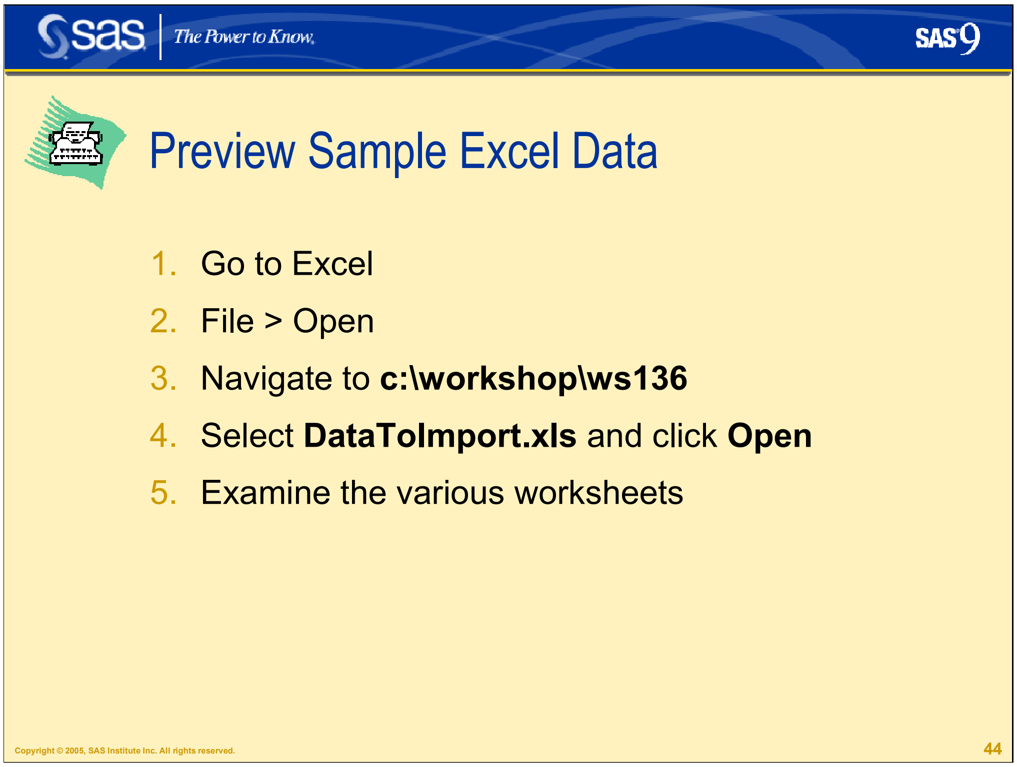


The Power to Know.
SAS 9

General Steps

- Open the Excel workbook
- Save it as XML
- Run SAS code to read XML into SAS tables

We will use the new XML support features of SAS 9 to read in the Excel XML-format workbook into SAS tables. Specifically, we will use the SAS Libname Engine (SXLE) and a SAS XMLMap that was designed specifically for reading Excel XML data.



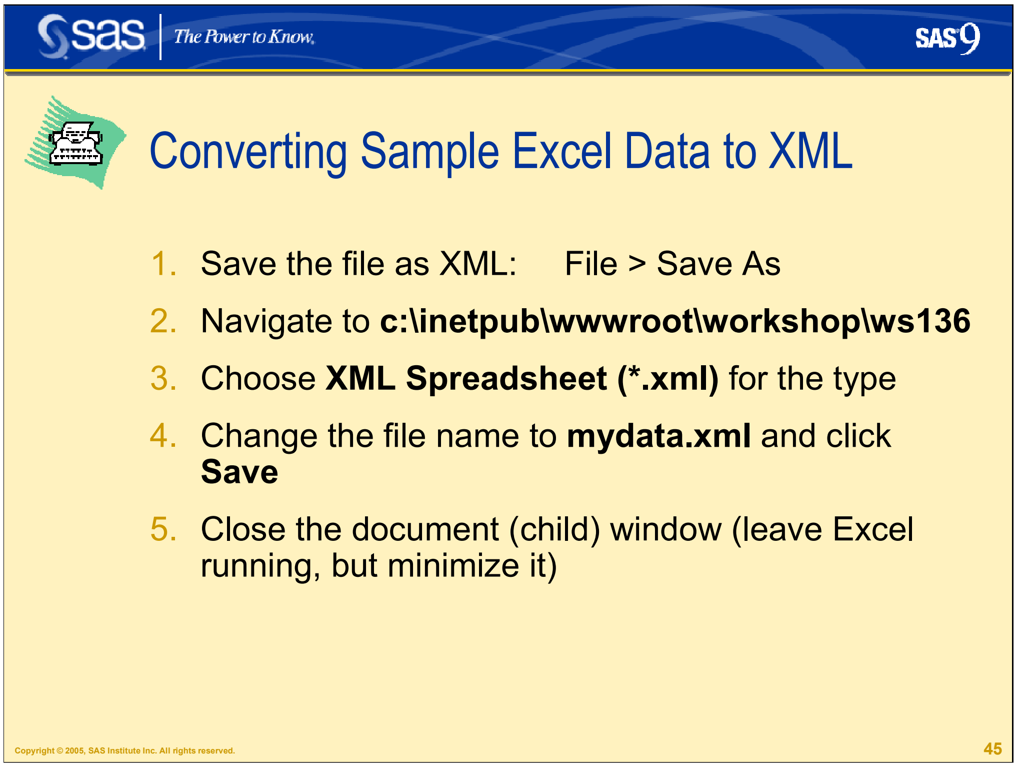
The Power to Know.

Preview Sample Excel Data

1. Go to Excel
2. File > Open
3. Navigate to **c:\workshop\ws136**
4. Select **DataToImport.xls** and click **Open**
5. Examine the various worksheets

This Excel workbook consists of 4 worksheets, each containing diverse data.

Each worksheet will be imported into a separate SAS table.



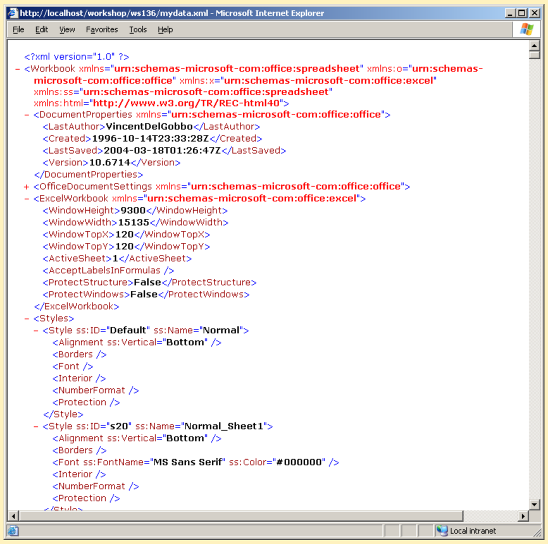
Converting Sample Excel Data to XML

1. Save the file as XML: File > Save As
2. Navigate to **c:\inetpub\wwwroot\workshop\ws136**
3. Choose **XML Spreadsheet (*.xml)** for the type
4. Change the file name to **mydata.xml** and click **Save**
5. Close the document (child) window (leave Excel running, but minimize it)

The first step is to save the Excel Workbook as an XML file.

Once again we are going to save the XML file in a directory that is under the control of our Web server.

Preview Sample Excel XML Data


http://localhost/workshop/ws136/mydata.xml - Microsoft Internet Explorer
File
Edit
View
Favorites
Tools
Help
Local intranet

```
  <?xml version="1.0" ?>
- <Workbook xmlns="urn:schemas-microsoft-com:office:spreadsheet" xmlns:o="urn:schemas-
    microsoft-com:office:office" xmlns:x="urn:schemas-microsoft-com:office:excel"
    xmlns:ss="urn:schemas-microsoft-com:office:spreadsheet"
    xmlns:html="http://www.w3.org/TR/REC-html40">
  - <DocumentProperties xmlns="urn:schemas-microsoft-com:office:office">
      <LastAuthor>VincentDelGobbo</LastAuthor>
      <Created>1996-10-14T23:33:28Z</Created>
      <LastSaved>2004-03-18T01:26:47Z</LastSaved>
      <Version>10.6714</Version>
    </DocumentProperties>
  + <OfficeDocumentSettings xmlns="urn:schemas-microsoft-com:office:office">
  - <ExcelWorkbook xmlns="urn:schemas-microsoft-com:office:excel">
      <WindowHeight>9300</WindowHeight>
      <WindowWidth>15135</WindowWidth>
      <WindowTopX>120</WindowTopX>
      <WindowTopY>120</WindowTopY>
      <ActiveSheet>1</ActiveSheet>
      <AcceptLabelsInFormulas />
      <ProtectStructure>False</ProtectStructure>
      <ProtectWindows>False</ProtectWindows>
    </ExcelWorkbook>
  - <Styles>
    - <Style ss:ID="Default" ss:Name="Normal">
        <Alignment ss:Vertical="Bottom" />
        <Borders />
        <Font />
        <Interior />
        <NumberFormat />
        <Protection />
      </Style>
    - <Style ss:ID="s20" ss:Name="Normal_Sheet1">
        <Alignment ss:Vertical="Bottom" />
        <Borders />
        <Font ss:FontName="MS Sans Serif" ss:Color="#000000" />
        <Interior />
        <NumberFormat />
        <Protection />
      </Style>
```

XLS: 0.4 MB

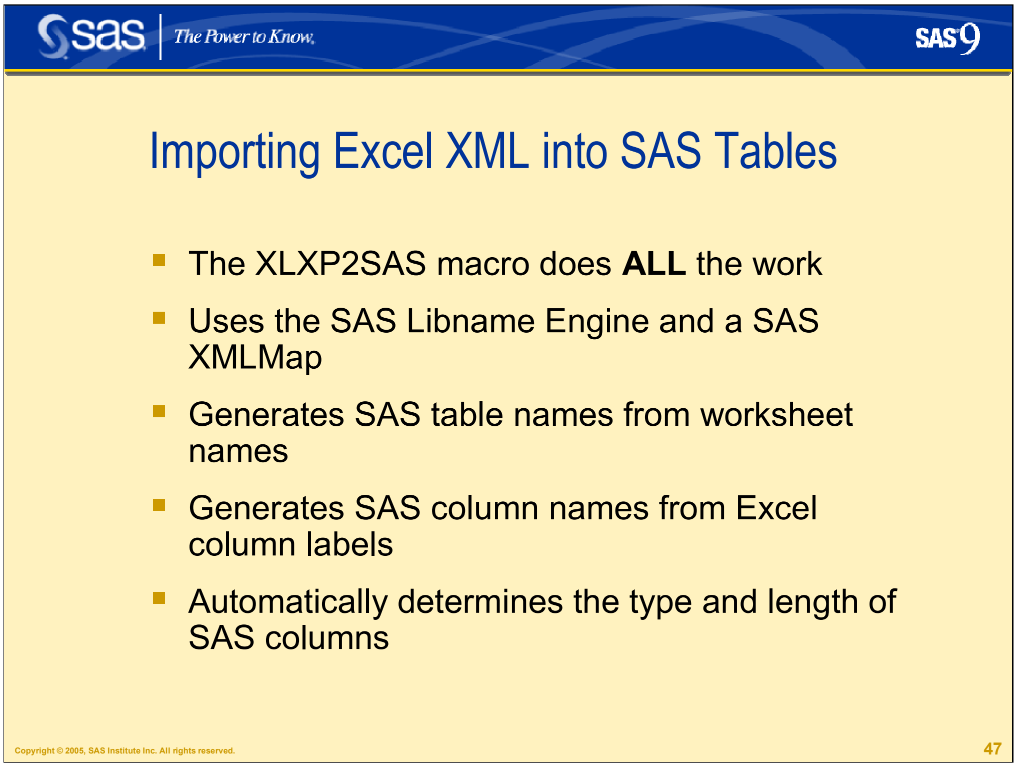
XML: 1.2 MB

Next we will spend just a moment looking at the structure and size of the XML file that represents the Excel workbook.

If you wish, you can open Windows Explorer and navigate to the directory "c:\inetpub\wwwroot\workshop\ws136" and double-click the file "mydata.xml". This will cause the file to open with Internet Explorer (by default) or whatever application you have defined to open XML files.

As you can see, this file is quite large, and writing DATA Step code to read it would be very cumbersome.

Fortunately, you are provided with a SAS XMLMap and macro (XLXP2SAS) that greatly simplifies the task of reading this XML data, *and any other Excel XML data*, into separate SAS tables.



Importing Excel XML into SAS Tables

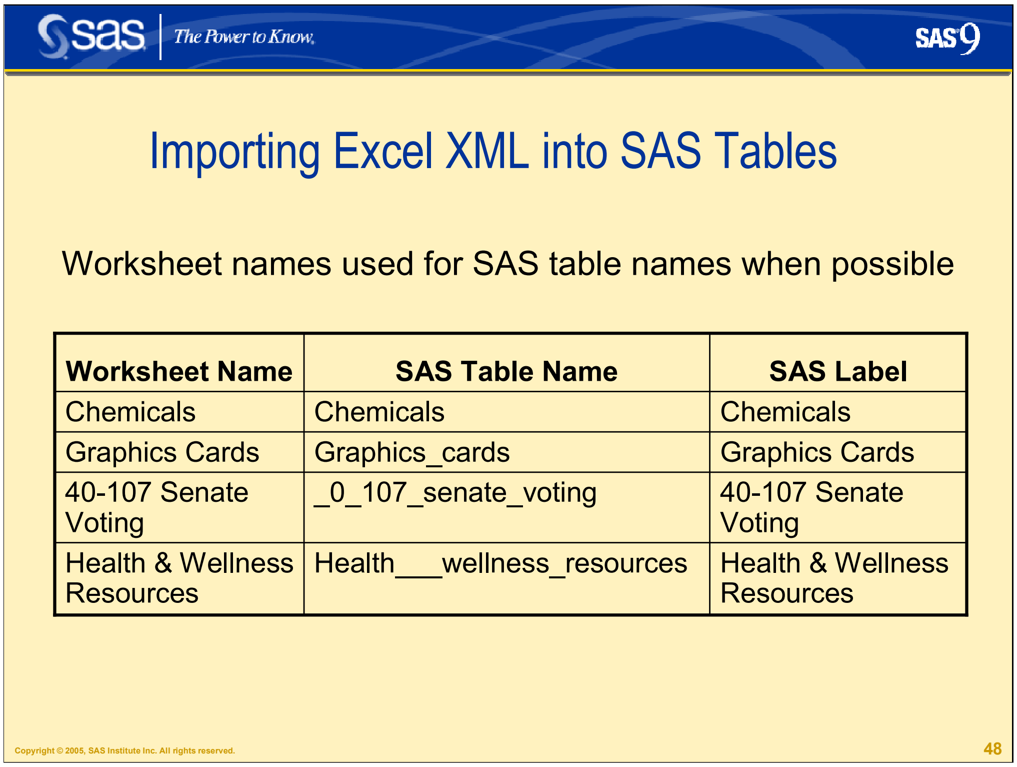
- The XLXP2SAS macro does **ALL** the work
- Uses the SAS Libname Engine and a SAS XMLMap
- Generates SAS table names from worksheet names
- Generates SAS column names from Excel column labels
- Automatically determines the type and length of SAS columns

The XLXP2SAS macro is provided for you, and does all the work of importing the Excel XML file into a SAS table. The macro uses the SAS Libname Engine and a SAS XMLMap, both of which are part of Base SAS.

The purpose of a SAS XMLMap is to map generic XML elements to SAS columns. It is used in conjunction with the SAS Libname Engine to import an XML files into a SAS table. An Excel-specific XMLMap has been developed and provided for you to use with the XLXP2SAS macro – you do not have to write any code.

The worksheet names will be used for the SAS table names and similarly, SAS column names will be generated from the Excel column labels. If no Excel column labels are available, then the SAS column names will be "COLUMN1", "COLUMN2", etc.

Only Base SAS is required to take advantage of this code, and it works for any XML file generated by Excel.



Importing Excel XML into SAS Tables

Worksheet names used for SAS table names when possible

Worksheet Name	SAS Table Name	SAS Label
Chemicals	Chemicals	Chemicals
Graphics Cards	Graphics_cards	Graphics Cards
40-107 Senate Voting	_0_107_senate_voting	40-107 Senate Voting
Health & Wellness Resources	Health___wellness_resources	Health & Wellness Resources

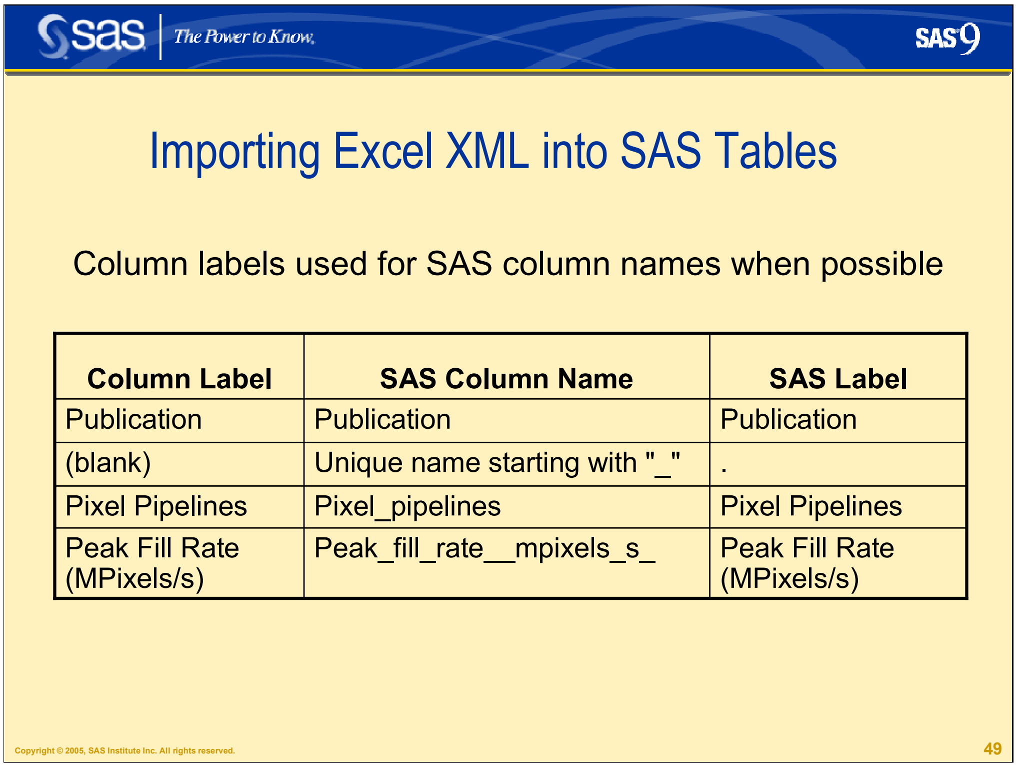
The table above shows how table names and labels will be derived from our sample XML file.

When the worksheet name does not contain any spaces or invalid characters, as is the case with "Chemicals", that name will be used as the SAS table name.

Otherwise, all invalid characters will be converted to underscores, and the resulting value will be used for the SAS table name.

Finally, the original worksheet name will be used as the label for the SAS table.

In all cases, the SAS table name will be truncated to 32 characters, and the SAS label will be truncated to 256 characters.



Importing Excel XML into SAS Tables

Column labels used for SAS column names when possible

Column Label	SAS Column Name	SAS Label
Publication	Publication	Publication
(blank)	Unique name starting with "_"	.
Pixel Pipelines	Pixel_pipelines	Pixel Pipelines
Peak Fill Rate (MPixels/s)	Peak_fill_rate__mpixels_s_	Peak Fill Rate (MPixels/s)

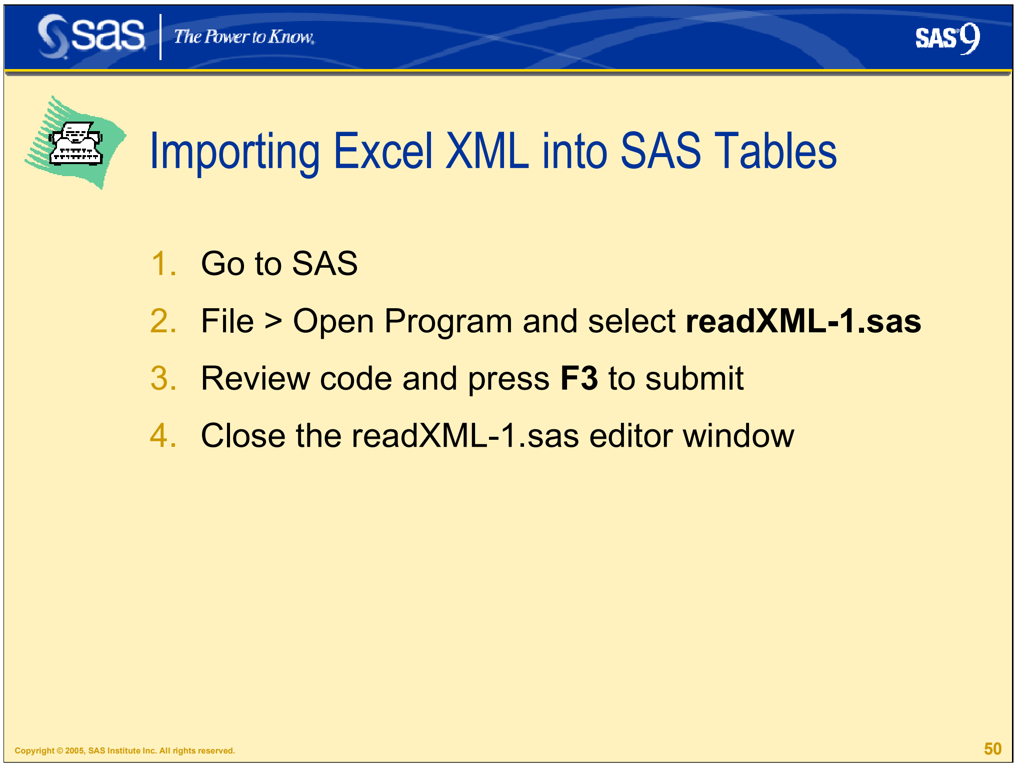
The table above shows an example of how SAS column names and labels will be derived from our sample XML file.

When the worksheet column label does not contain any spaces or invalid characters, as is the case with "Publication", that label will be used as the SAS column name.

Otherwise, all invalid characters will be converted to underscores, and the resulting value will be used for the SAS column name.

Finally, the original worksheet column label will be used as the label for the SAS table column.

In all cases, the SAS column name will be truncated to 32 characters, and the column label will be truncated to 256 characters.



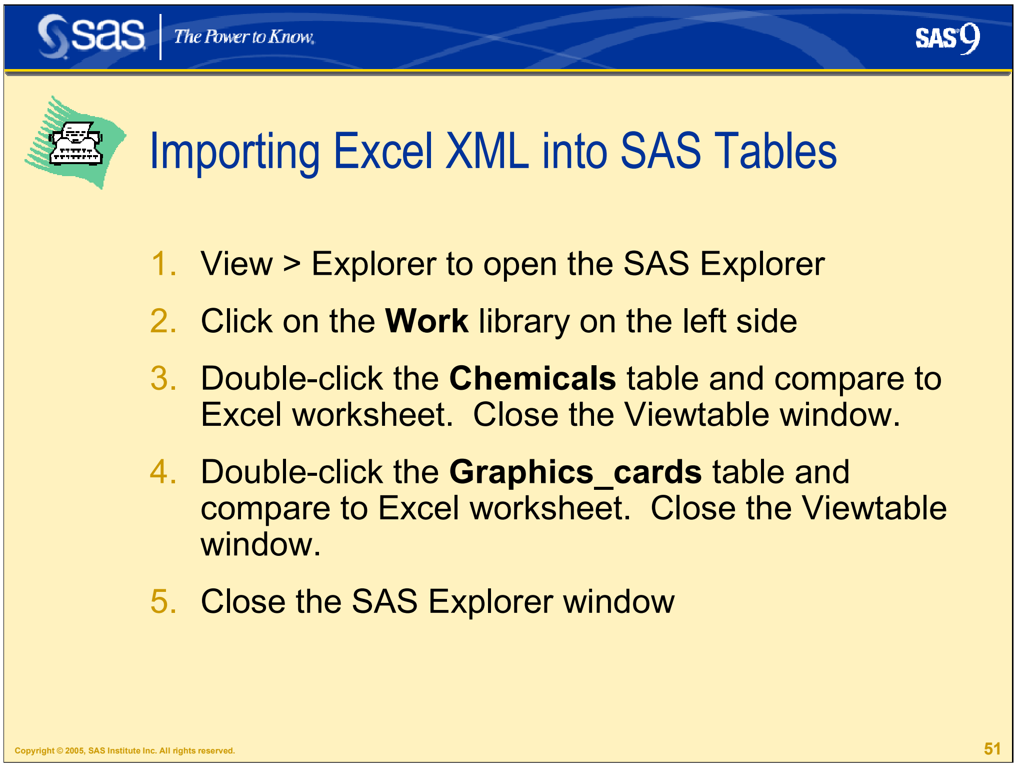
Importing Excel XML into SAS Tables

1. Go to SAS
2. File > Open Program and select **readXML-1.sas**
3. Review code and press **F3** to submit
4. Close the readXML-1.sas editor window

The XLXP2SAS macro was made available to your SAS session using a %INCLUDE statement in the program "setup.sas". In a production environment, you may wish to add the XLXP2SAS macro to your autocall macro library.

Run the XLXP2SAS macro via the program "readXML-1.sas". You only need to specify two arguments: the location of the XML file you wish to import, and the location of the SAS XMLMap (which is provided for you).

The macro reads the huge XML file with the help of the SAS Libname Engine and the SAS XMLMap. In our case, the result is 4 tables stored in the WORK library.



Importing Excel XML into SAS Tables

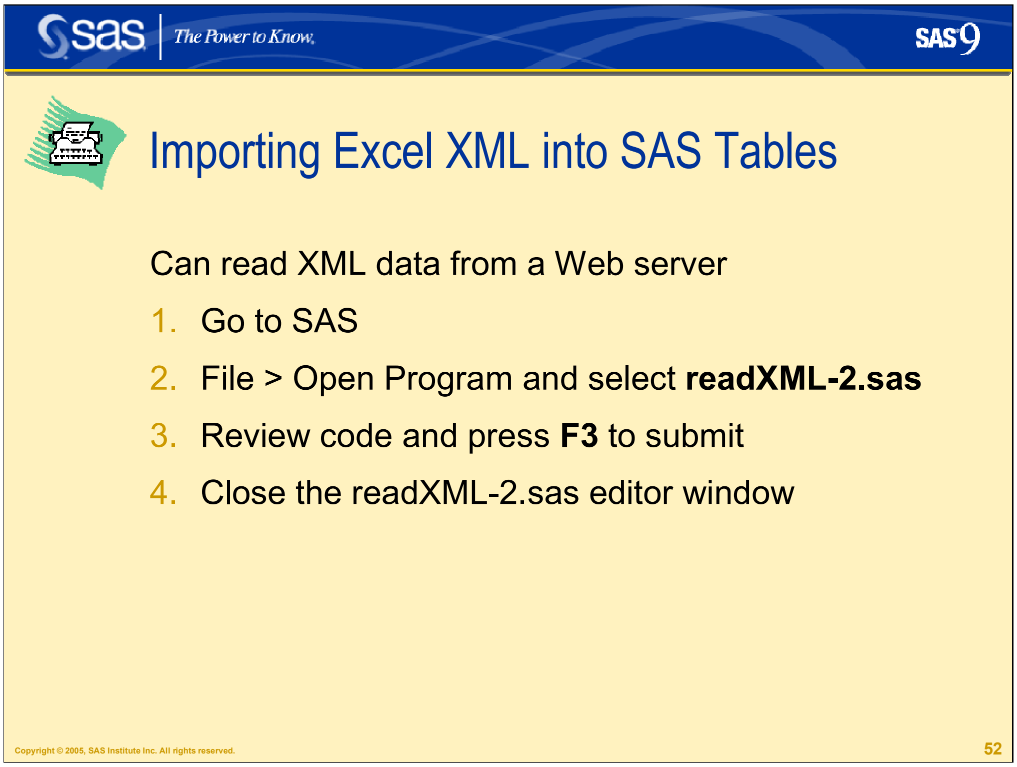
1. View > Explorer to open the SAS Explorer
2. Click on the **Work** library on the left side
3. Double-click the **Chemicals** table and compare to Excel worksheet. Close the Viewtable window.
4. Double-click the **Graphics_cards** table and compare to Excel worksheet. Close the Viewtable window.
5. Close the SAS Explorer window

Chemicals table

- SAS column labels match the Excel worksheet column labels
- The column "CAA 112(r) TQ" is correctly typed as numeric
- All other columns correctly typed as character

Graphics_cards table

- SAS column labels match the Excel worksheet column labels
- The SAS columns "Pixel Pipelines" and "Memory bus width (bits)" are correctly typed as character.
- All other columns correctly typed as numeric



Importing Excel XML into SAS Tables

Can read XML data from a Web server

1. Go to SAS
2. File > Open Program and select **readXML-2.sas**
3. Review code and press **F3** to submit
4. Close the readXML-2.sas editor window

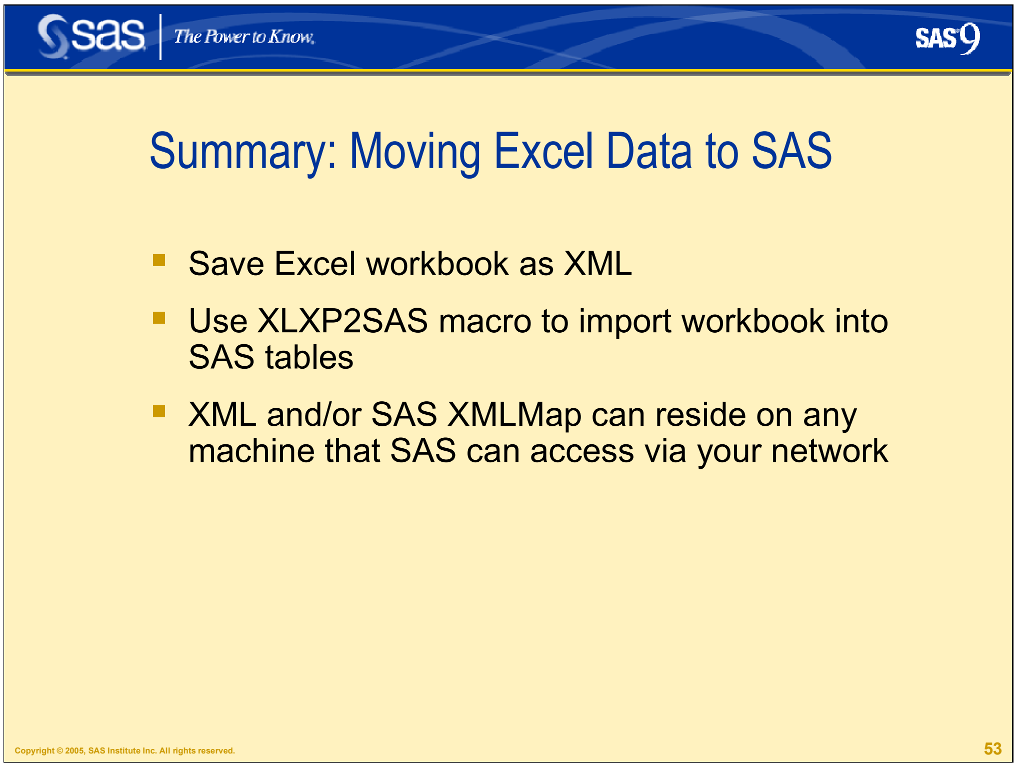
This code is the same as the last program we ran with one exception: SAS will read the input XML file from a Web server instead of the local file system. This is accomplished via the URL access method that was used in conjunction with the FILENAME statement.

This technique is useful when the XML file is stored on a machine other than the one where SAS is installed. For example, you may have the XML stored on a Windows machine and SAS is installed on a mainframe. Your SAS program running on the mainframe would read the file from the Web server running on the Windows machine, and convert the XML to one or more SAS tables.

The same holds true for the SAS XMLMap – it, too, could be retrieved from any Web server, providing SAS has network access to that machine. You specify the location of the XMLMap using the MAPFILE argument of XLSP2SAS.

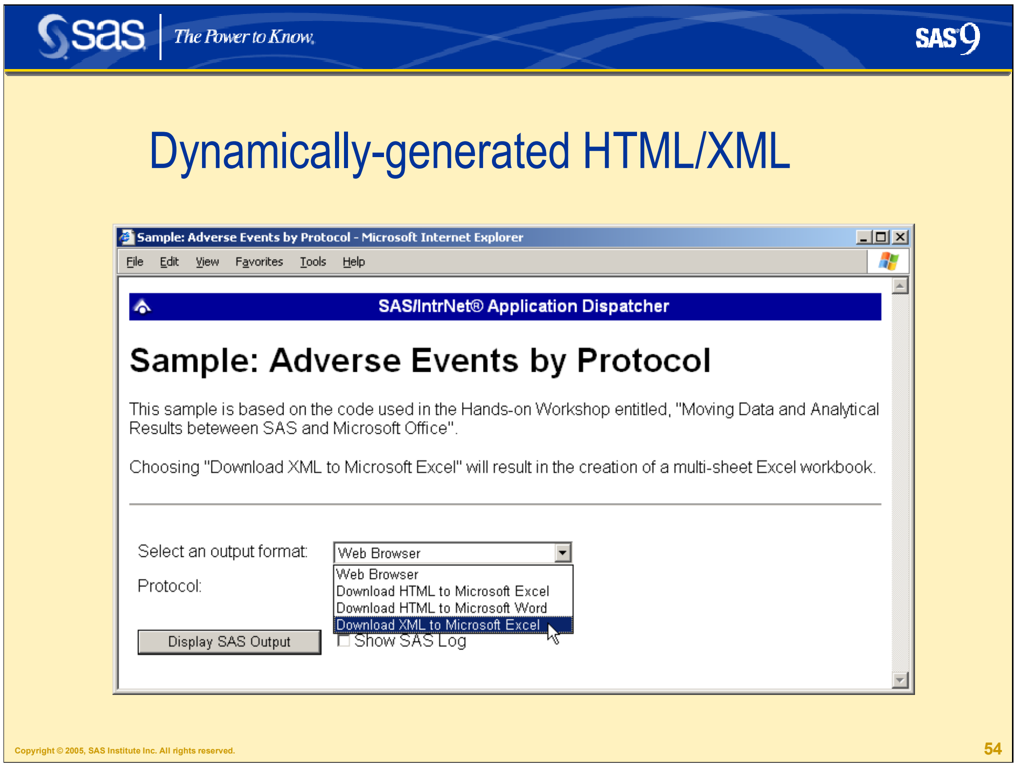
Note that *any* FILEREF can be used with the EXCELFILE and MAPFILE arguments, not just FILEREFs that utilize the URL access method.

Refer to the accompanying paper for documentation on the XLXP2SAS macro.



Summary: Moving Excel Data to SAS

- Save Excel workbook as XML
- Use XLXP2SAS macro to import workbook into SAS tables
- XML and/or SAS XMLMap can reside on any machine that SAS can access via your network



Dynamically-generated HTML/XML

Sample: Adverse Events by Protocol - Microsoft Internet Explorer

SAS/IntrNet® Application Dispatcher

Sample: Adverse Events by Protocol

This sample is based on the code used in the Hands-on Workshop entitled, "Moving Data and Analytical Results beteween SAS and Microsoft Office".

Choosing "Download XML to Microsoft Excel" will result in the creation of a multi-sheet Excel workbook.

Select an output format: Web Browser
- Web Browser
- Download HTML to Microsoft Excel
- Download HTML to Microsoft Word
- Download XML to Microsoft Excel

Protocol:

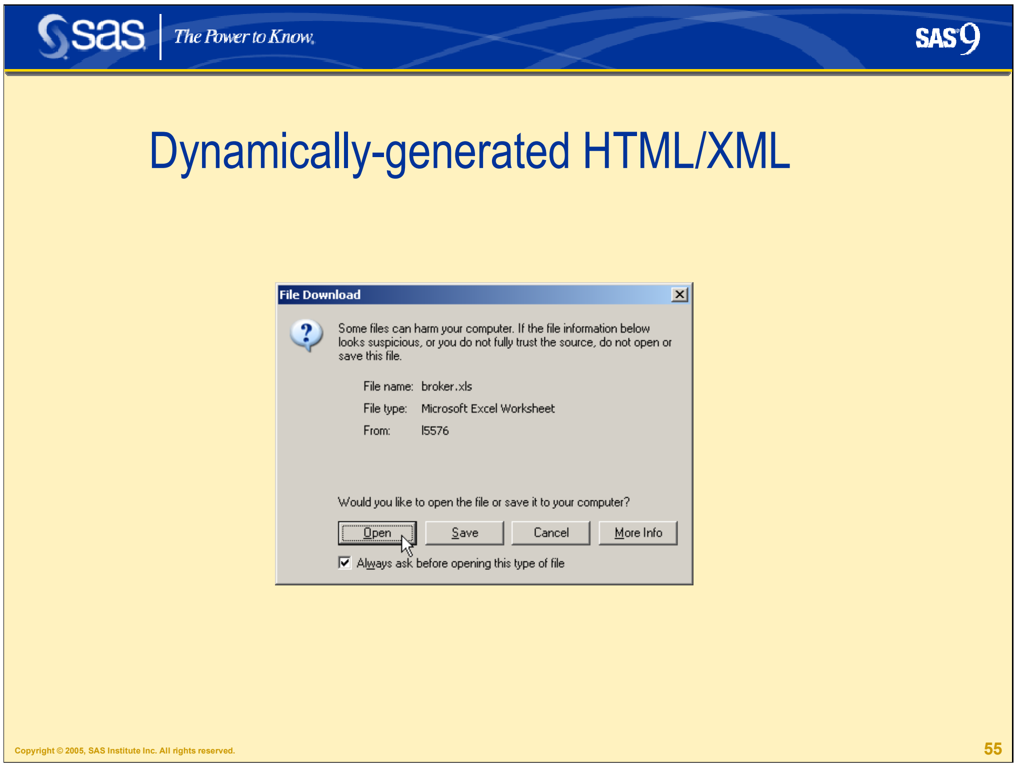
☐ Show SAS Log

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 have been using to generate HTML and XML files, with a few changes to "Web-enable" it.

The image above indicates that we will be generating XML for use with Microsoft Excel.

Clicking "Display SAS Output" sends the choices made in the Web page to the SAS server as global macro variables. Those macro variables are used by the SAS program to control the type of output generated.



Dynamically-generated HTML/XML

File Download

Some files can harm your computer. If the file information below looks suspicious, or you do not fully trust the source, do not open or save this file.

File name: broker.xls

File type: Microsoft Excel Worksheet

From: l5576

Would you like to open the file or save it to your computer?

☑ Always ask before opening this type of file

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

Note that you are presented with an Open/Save dialog, instead of the results being displayed in the browser. Clicking Open will cause the SAS output to be displayed in Excel, provided that Excel is installed on the machine.

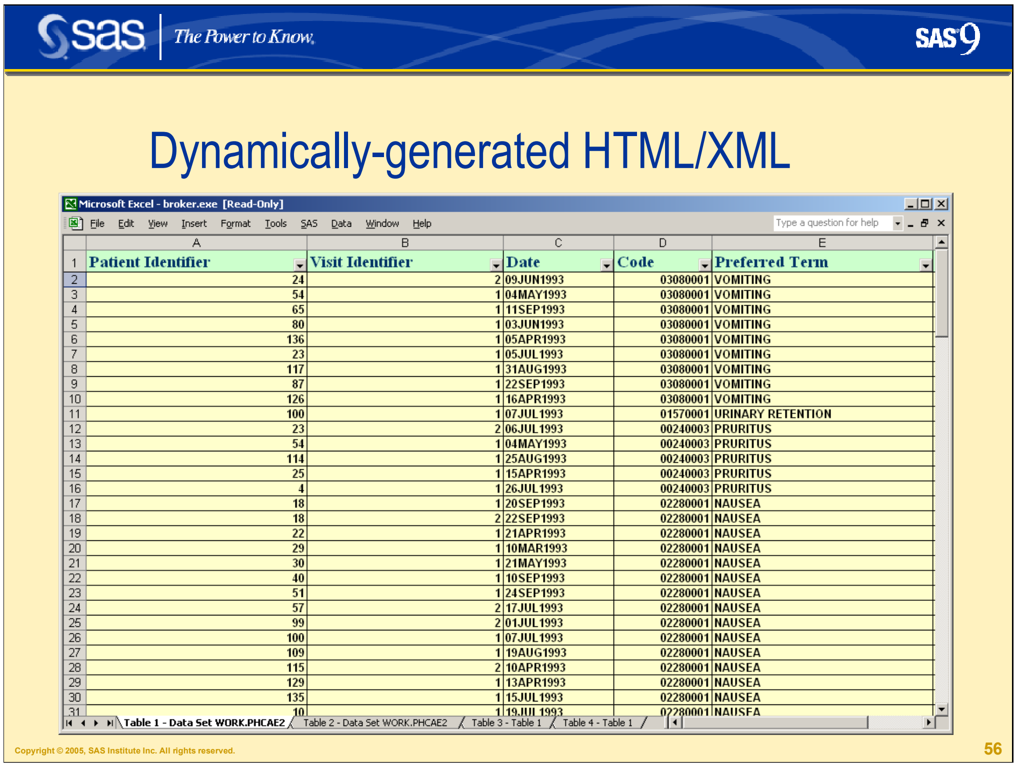
This SAS/IntrNet-specific code, which must be specified before any ODS statements, was used to cause the SAS output to be displayed in Excel:

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

Similar code can be used to cause the SAS output to be displayed in Word:

```
%let RV =%sysfunc(appsrv_header(Content-type,application/msword));
```

Note: Neither of the above statements should be used if you wish to display the SAS output in a Web browser.



The Power to Know.

Dynamically-generated HTML/XML

Microsoft Excel - broker.exe [Read-Only]

	A	B	C	D	E
1	Patient Identifier	Visit Identifier	Date	Code	Preferred Term
2	24	2	09JUN1993	03080001	VOMITING
3	54	1	04MAY1993	03080001	VOMITING
4	65	1	11SEP1993	03080001	VOMITING
5	80	1	03JUN1993	03080001	VOMITING
6	136	1	05APR1993	03080001	VOMITING
7	23	1	05JUL1993	03080001	VOMITING
8	117	1	31AUG1993	03080001	VOMITING
9	87	1	22SEP1993	03080001	VOMITING
10	126	1	16APR1993	03080001	VOMITING
11	100	1	07JUL1993	01570001	URINARY RETENTION
12	23	2	06JUL1993	00240003	PRURITUS
13	54	1	04MAY1993	00240003	PRURITUS
14	114	1	25AUG1993	00240003	PRURITUS
15	25	1	15APR1993	00240003	PRURITUS
16	4	1	26JUL1993	00240003	PRURITUS
17	18	1	20SEP1993	02280001	NAUSEA
18	18	2	22SEP1993	02280001	NAUSEA
19	22	1	21APR1993	02280001	NAUSEA
20	29	1	10MAR1993	02280001	NAUSEA
21	30	1	21MAY1993	02280001	NAUSEA
22	40	1	10SEP1993	02280001	NAUSEA
23	51	1	24SEP1993	02280001	NAUSEA
24	57	2	17JUL1993	02280001	NAUSEA
25	99	2	01JUL1993	02280001	NAUSEA
26	100	1	07JUL1993	02280001	NAUSEA
27	109	1	19AUG1993	02280001	NAUSEA
28	115	2	10APR1993	02280001	NAUSEA
29	129	1	13APR1993	02280001	NAUSEA
30	135	1	15JUL1993	02280001	NAUSEA
31	10	1	19JUL1993	02280001	NAUSEA

Table 1 - Data Set WORK.PHCAE2
Table 2 - Data Set WORK.PHCAE2
Table 3 - Table 1
Table 4 - Table 1

The image above shows the ODS-generated XML output viewed using Excel.

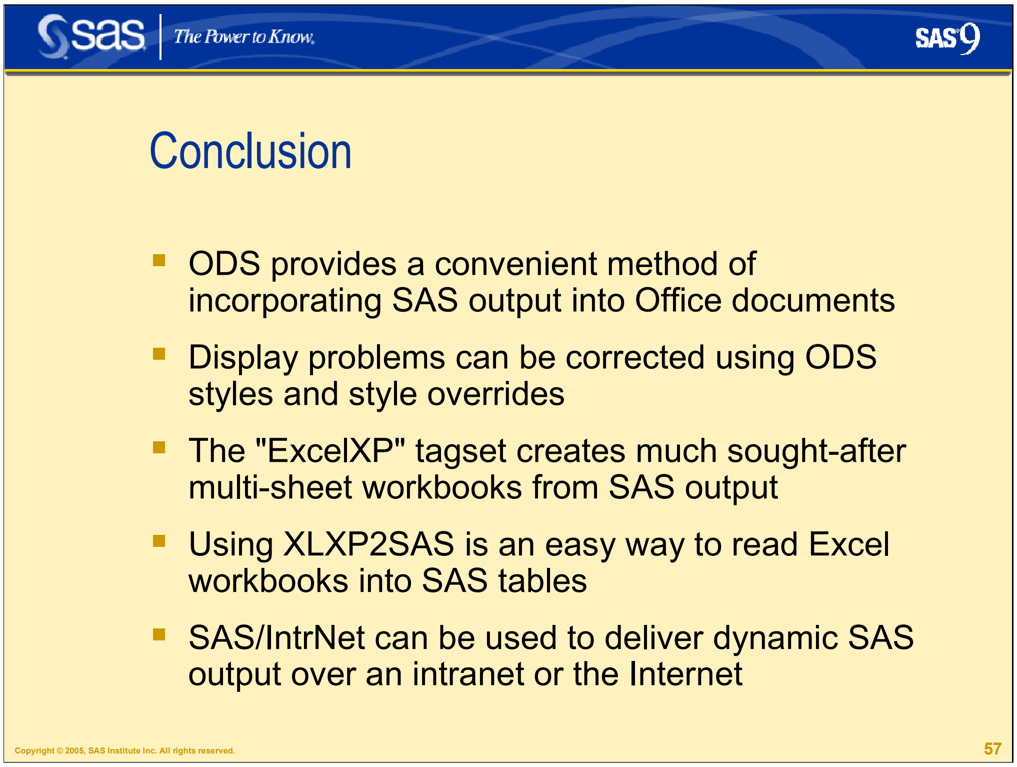
Note that this file contains autofilters, and column and row headers are "frozen". These two features were implemented using this code:

```
ods tagsets.ExcelXP options(autofilter='all'
                            frozen_headers='yes'
                            frozen_rowheaders='yes') ...
```

To see all the features supported by the ExcelXP tagset, submit this code:

```
ods tagsets.ExcelXP file='c:\temp\temp.xml'
  options(doc='help');
ods tagsets.ExcelXP close;
```

Information will be printed to the SAS Log.



Conclusion

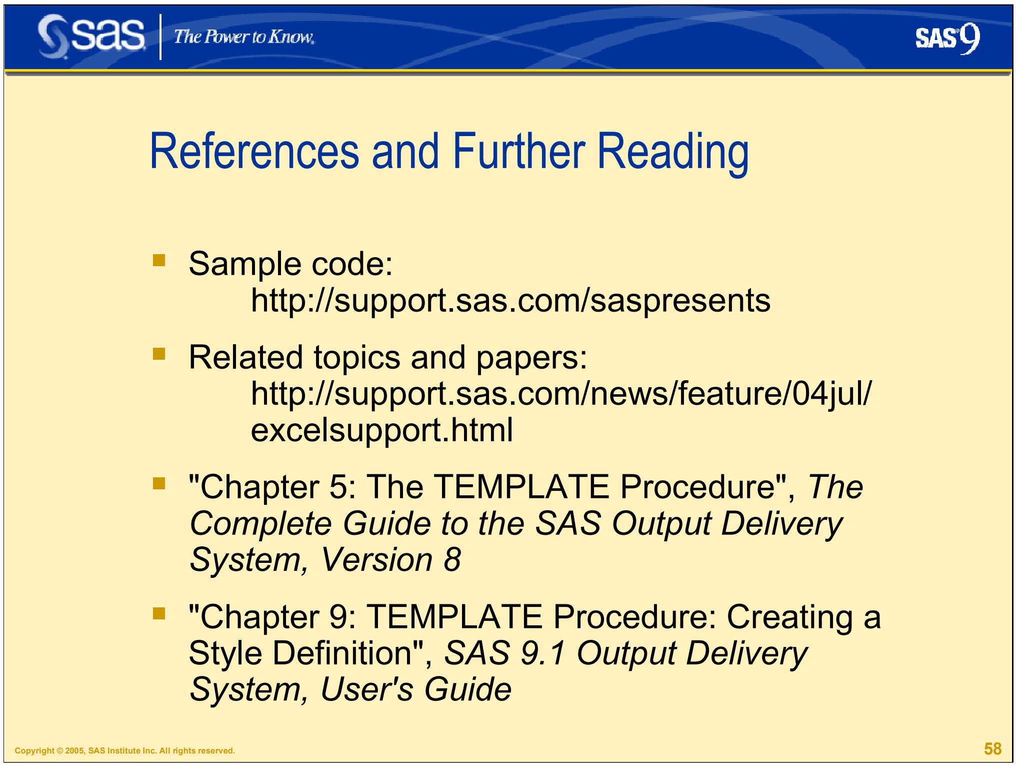
- ODS provides a convenient method of incorporating SAS output into Office documents
- Display problems can be corrected using ODS styles and style overrides
- The "ExcelXP" tagset creates much sought-after multi-sheet workbooks from SAS output
- Using XLXP2SAS is an easy way to read Excel workbooks into SAS tables
- SAS/IntrNet can be used to deliver dynamic SAS output over an intranet or the Internet

Using ODS to generate HTML and XML output is an effective method of incorporating SAS output in Excel and Word documents. Although you may initially encounter formatting problems when using this technique, by using ODS style overrides and the TEMPLATE procedure you can correct these problems.

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.

While it may be a bit cumbersome to use the SXLE and XMLMaps to import Excel data to SAS, the use of the XLXP2SAS macro greatly simplifies the task.

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 Office applications.



References and Further Reading

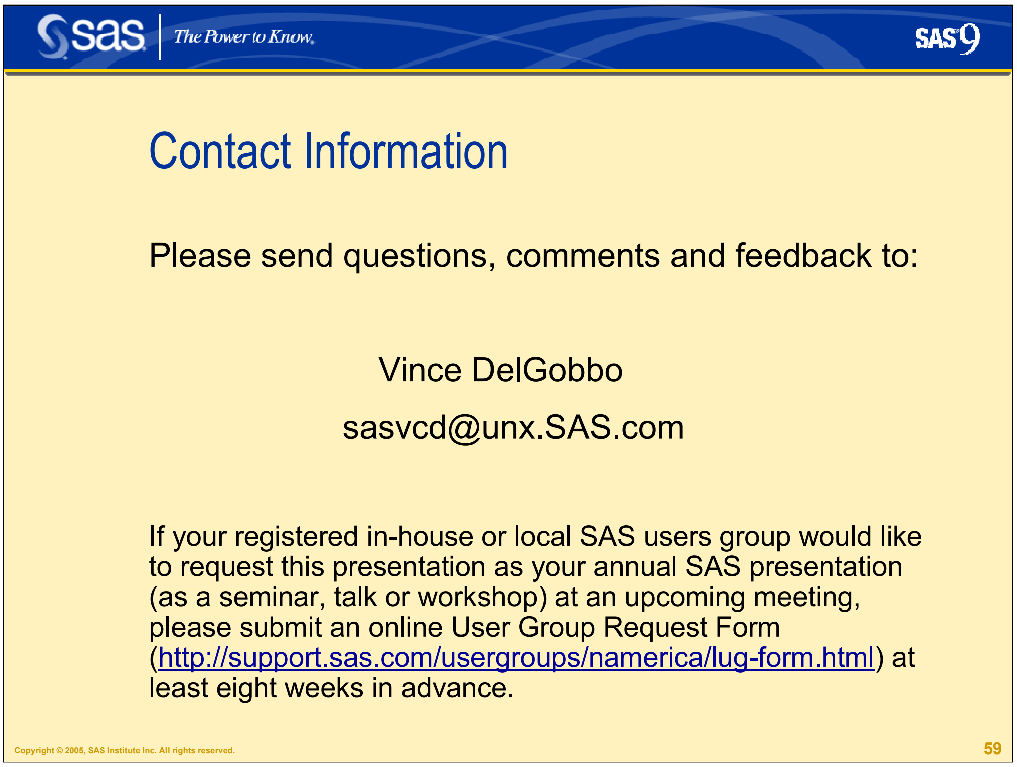
- Sample code:
  http://support.sas.com/saspresents
- Related topics and papers:
  http://support.sas.com/news/feature/04jul/excelsupport.html
- "Chapter 5: The TEMPLATE Procedure", *The Complete Guide to the SAS Output Delivery System, Version 8*
- "Chapter 9: TEMPLATE Procedure: Creating a Style Definition", *SAS 9.1 Output Delivery System, User's Guide*

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/saspresents and find the entry "Moving Data and Analytical Results between SAS and Microsoft Office".

IMPORTANT NOTE: Review the "Installation" and "Usage" sections of the file named "readme.txt" for important information on using the sample files.

A number of related papers can be found at:
http://support.sas.com/news/feature/04jul/excelsupport.html

Refer to the ODS documentation for your release of SAS to find out more about styles, style attributes and the TEMPLATE procedure.



Contact Information

Please send questions, comments and feedback to:

Vince DelGobbo
sasvcd@unx.SAS.com

If your registered in-house or local SAS users group would like to request this presentation as your annual SAS presentation (as a seminar, talk or workshop) at an upcoming meeting, please submit an online User Group Request Form (http://support.sas.com/usergroups/namerica/lug-form.html) at least eight weeks in advance.

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. Vince has been a SAS Software user since 1982, and joined SAS in 1992.