

SAS[®] Excels!!



Microsoft Excel window showing a report titled "Softsale Sales by State and Division". The report is displayed in a table format with columns for State, Div., Average Years, Sales Amt., and Expenses. The data is as follows:

State	Sales Div.	Average Years	Sales Amt.	Expenses
IL	H	4.0	\$4,009.21	322
	S	2.0	\$743.22	159
MN	H	11.0	\$3,231.75	645
	S	11.7	\$7,732.44	1,339
WI	H	9.4	\$12,185.10	2,786
	S	9.0	\$8,232.11	3,339
Totals:			\$36,133.83	\$8,590.00

And now you can add your own formulas in Excel - to a report created in SAS!!



SYSTEMS SEMINAR CONSULTANTS, INC.

Gerald Frey

2997 Yarmouth Greenway Drive Madison, WI 53711

(608) 278-9964

www.sys-seminar.com

SAS[®] Excels!! Exporting HTML to Other Programs



It is very common to send data and report output to Excel or other programs.

Techniques:

- Use File Export Data menu choice from SAS to create a XLS file
- Write a data step to create a CSV file that can be imported
- Use a macro that does the above
- Use ODBC to create a connection to a data source
- Use the EXCEL libname engine (SAS 9) to access Excel Data
- Use ODS to create a report file ready for import
 - as HTML
 - as CSV
 - as XML

File Export Facility in Interactive SAS®



Advantages:

- Easy
- Almost automatic
- Best format is chosen for variables

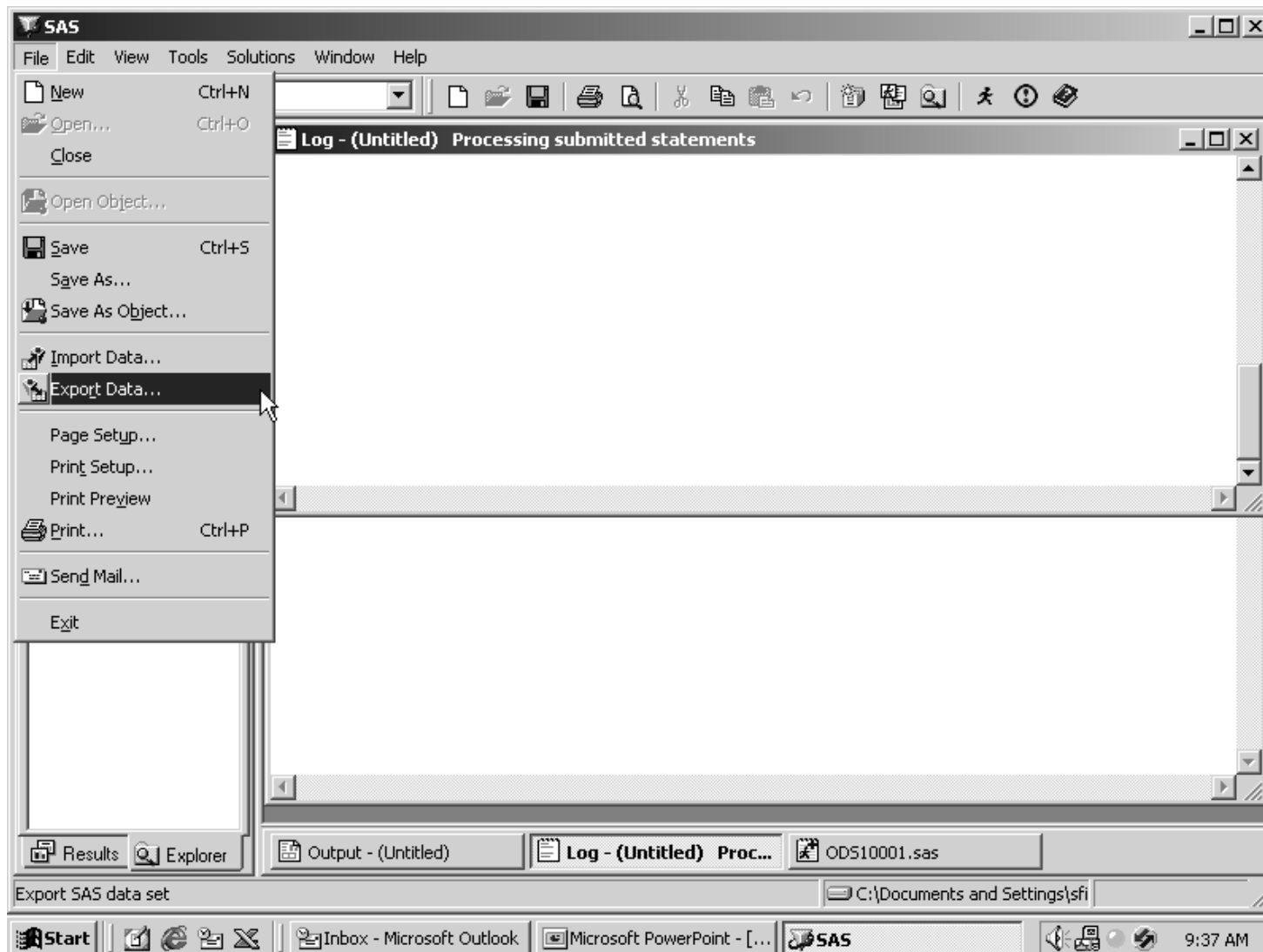
Disadvantages:

- Not available on all systems
- Not available for batch jobs
- SAS/Access necessary for file types other than .CSV and .TXT files



File Export Facility in Interactive SAS®

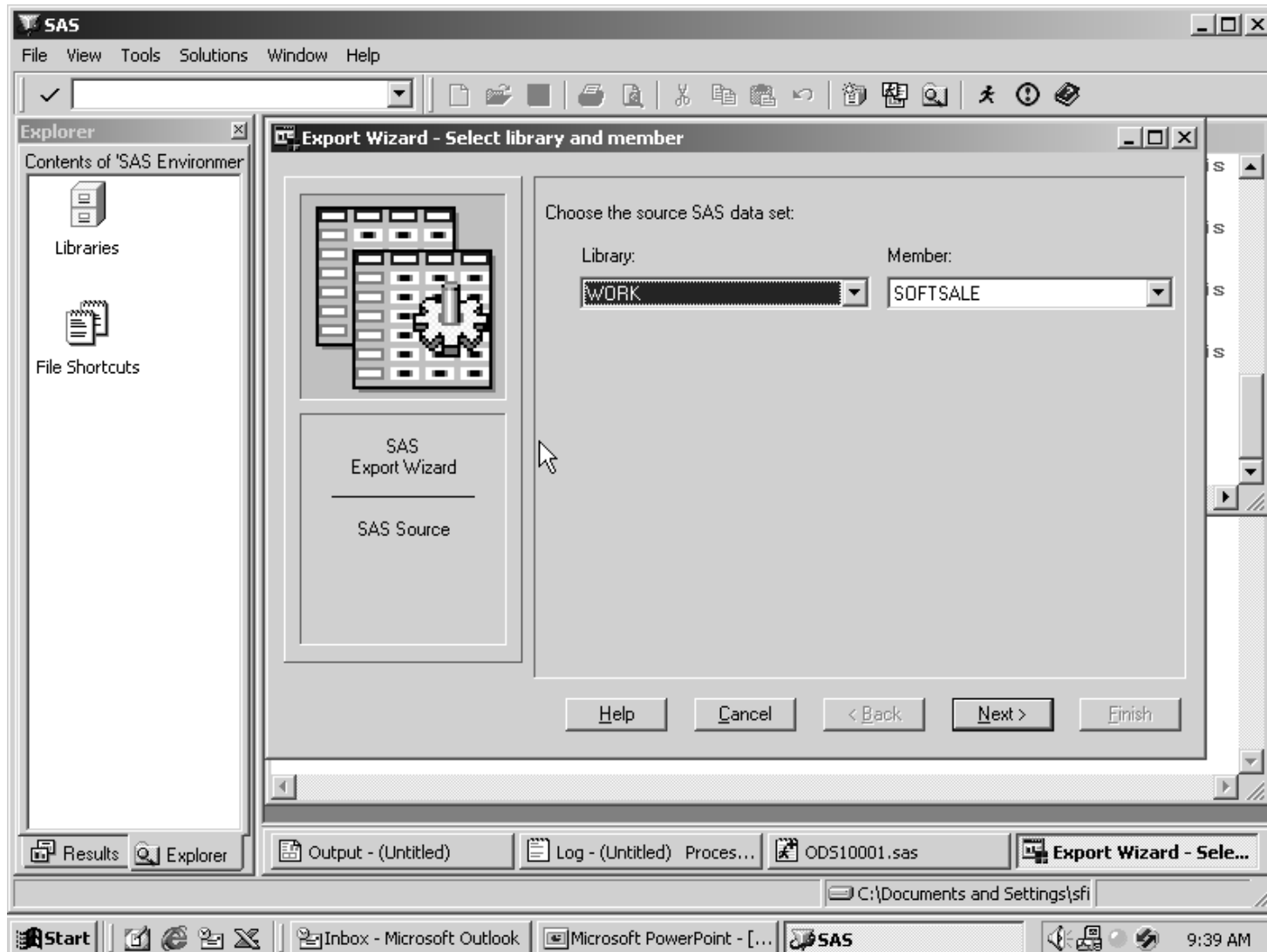
File Extract can convert SAS datasets to various formats.





File Export Facility (continued)

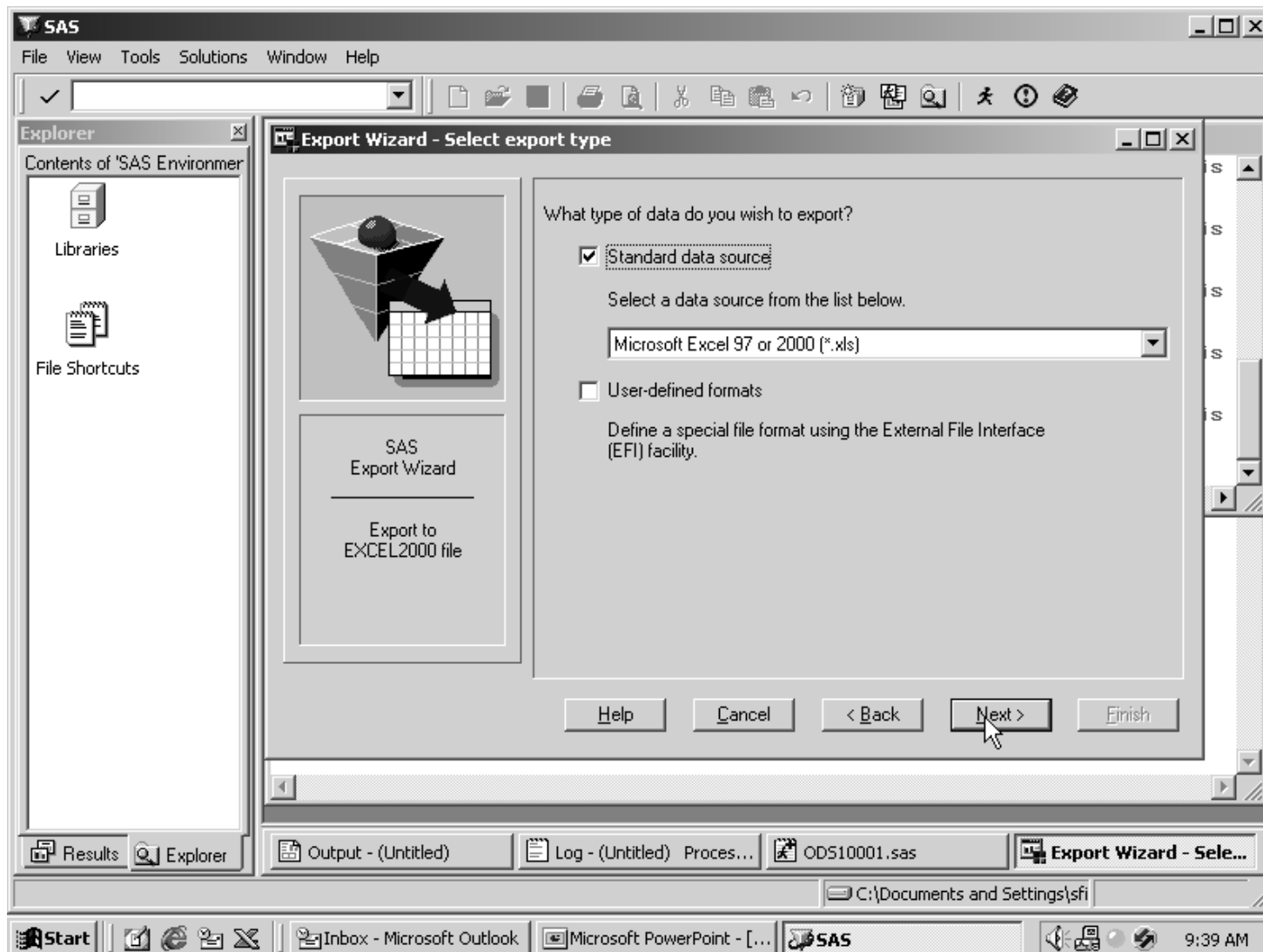
You can choose your input dataset.





File Export Facility (continued)

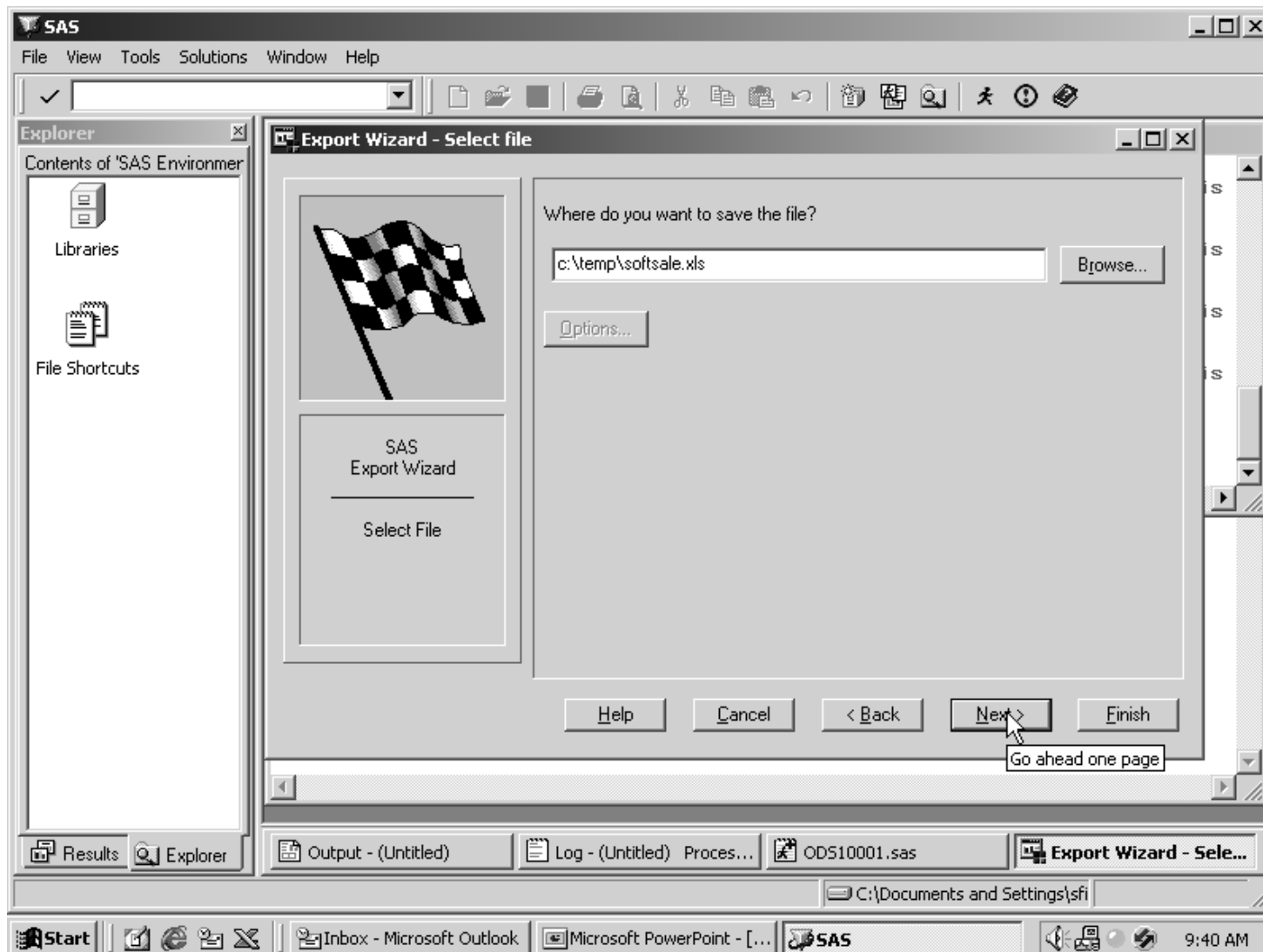
You can then choose output file type.





File Export Facility (continued)

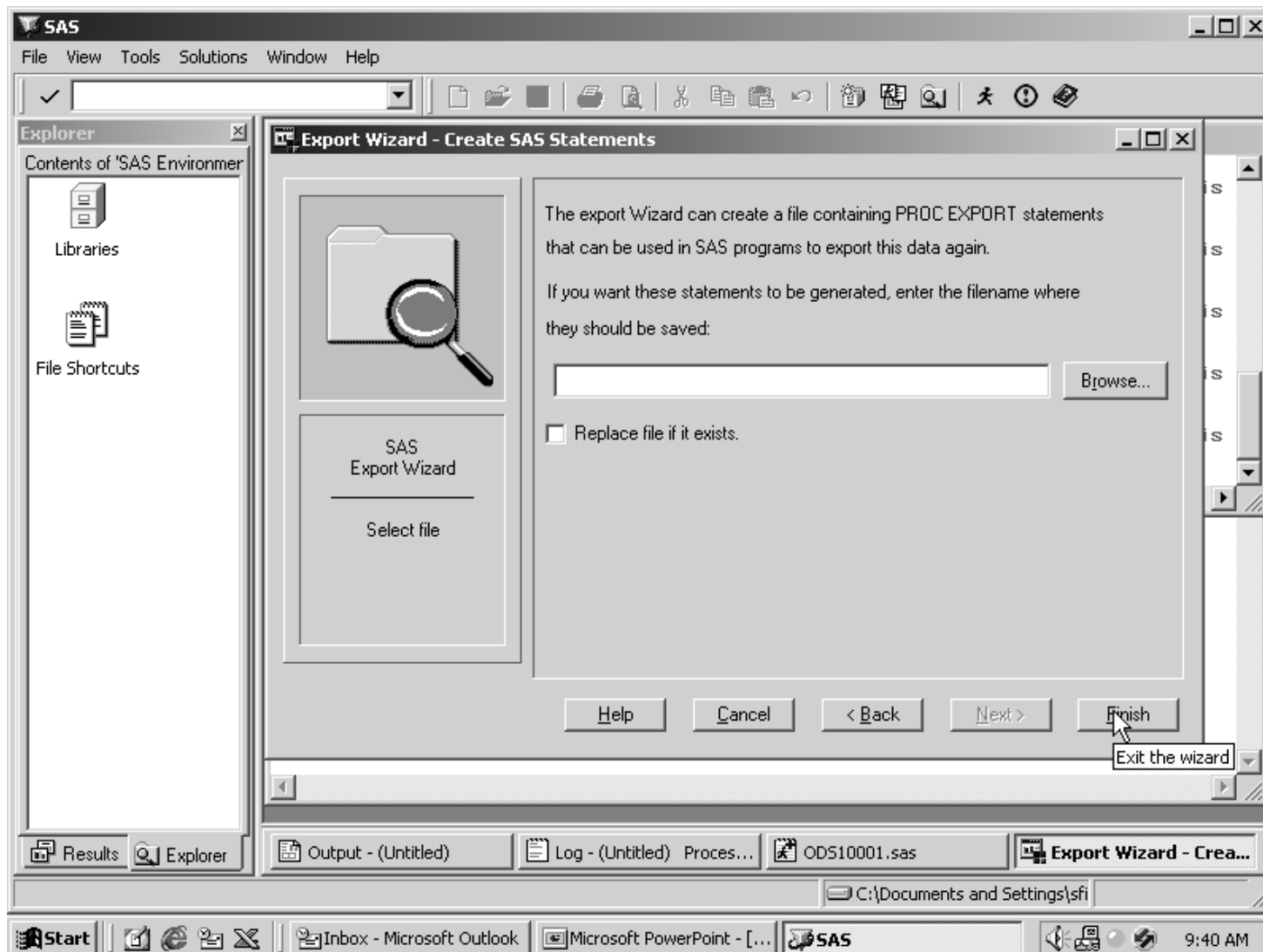
You can then name the output file.





File Export Facility (continued)

You can save the export statements for later reruns.





File Export Facility (continued)

The resulting worksheet in Excel (Full screen view).

	A	B	C	D	E	F
1	Name	Division	Years	Sales	Expense	State
2	CHRIS	H	2	233.11	94.12	WI
3	MARK	H	5	298.12	52.65	WI
4	SARAH	S	6	301.21	65.17	MN
5	PAT	H	4	4009.21	322.12	IL
6	JOHN	H	7	678.43	150.11	WI
7	WILLIAM	H	11	3231.75	644.55	MN
8	ANDREW	S	24	1762.11	476.13	MN
9	BENJAMIN	S	3	201.11	25.21	IL
10	JANET	S	1	98.11	125.32	WI
11	STEVE	H	21	6153.32	1507.12	WI
12	JENNIFER	S	1	542.11	134.24	IL
13	JOY	S	12	2442.22	761.98	WI
14	MARY	S	14	5691.78	2452.11	WI
15	TOM	S	5	5669.12	798.15	MN
16	BETH	H	12	4822.12	982.1	WI
17						
18						
19						
20						
21						
22						

Creating a Custom CSV File in a Data Step



Advantages:

- Very flexible
- Runs in all environments
- Can be tailored for any dataset
- Individual formats can be used

Disadvantages:

- User must know variable names and formats
- A different and difficult program for each dataset

Creating a Custom CSV File in a Data Step



A data step with lots of quotes and variables.

```
data _null_;          /* no sas ds needed */
  set softsale;      /* read input ds      */
  file 'c:\temp\softsale.csv'; /* output flat file */
  if _n_ =1 then     /* before first rec */
    put
      ' "Name", "Division", "Years", "Sales", "Expense", "State" ';
  put ' "' name      +(-1) '" , ' /* put out values */
      ' "' division +(-1) '" , ' /* quoted and commas*/
      ' "' years    +(-1) '" , ' /* between          */
      ' "' sales    +(-1) '" , '
      ' "' expense  +(-1) '" , '
      ' "' state    +(-1) '" ' ;
run;                  /* end of step      */
```



The Resulting Flat File

Quotes and commas are around field names and values.

```
"Name", "Division", "Years", "Sales", "Expense", "State"  
"CHRIS", "H", "2", "233.11", "94.12", "WI"  
"MARK", "H", "5", "298.12", "52.65", "WI"  
"SARAH", "S", "6", "301.21", "65.17", "MN"  
"PAT", "H", "4", "4009.21", "322.12", "IL"  
"JOHN", "H", "7", "678.43", "150.11", "WI"  
"WILLIAM", "H", "11", "3231.75", "644.55", "MN"  
"ANDREW", "S", "24", "1762.11", "476.13", "MN"  
"BENJAMIN", "S", "3", "201.11", "25.21", "IL"  
"JANET", "S", "1", "98.11", "125.32", "WI"  
"STEVE", "H", "21", "6153.32", "1507.12", "WI"  
"JENNIFER", "S", "1", "542.11", "134.24", "IL"  
"JOY", "S", "12", "2442.22", "761.98", "WI"  
"MARY", "S", "14", "5691.78", "2452.11", "WI"  
"TOM", "S", "5", "5669.12", "798.15", "MN"  
"BETH", "H", "12", "4822.12", "982.1", "WI"
```



Opening With Excel

File Open will convert the CSV file to XLS format while reading.

The screenshot shows a Microsoft Excel spreadsheet with the following data:

	A	B	C	D	E	F
1	Name	Division	Years	Sales	Expense	State
2	CHRIS	H	2	233.11	94.12	WI
3	MARK	H	5	298.12	52.65	WI
4	SARAH	S	6	301.21	65.17	MN
5	PAT	H	4	4009.21	322.12	IL
6	JOHN	H	7	678.43	150.11	WI
7	WILLIAM	H	11	3231.75	644.55	MN
8	ANDREW	S	24	1762.11	476.13	MN
9	BENJAMIN	S	3	201.11	25.21	IL
10	JANET	S	1	98.11	125.32	WI
11	STEVE	H	21	6153.32	1507.12	WI
12	JENNIFER	S	1	542.11	134.24	IL
13	JOY	S	12	2442.22	761.98	WI
14	MARY	S	14	5691.78	2452.11	WI
15	TOM	S	5	5669.12	798.15	MN
16	BETH	H	12	4822.12	982.1	WI
17						
18						
19						
20						
21						
22						

A Different Approach



The DSD option inserts commas automatically.

```
data _null_;                                /* no sas ds needed */
  set softsale;                              /* read input ds    */
  file 'c:\temp\softsale.csv' dsd;
  put name
      division
      years
      sales
      expense
      state ;
run;                                          /* end of step     */
```

Notes:

- You still need to know the field names.
- You don't get the header record from the previous program.

The Resulting Flat File



The file is similar to before and imports similarly.

```
CHRIS,H,2,233.11,94.12,WI
MARK,H,5,298.12,52.65,WI
SARAH,S,6,301.21,65.17,MN
PAT,H,4,4009.21,322.12,IL
JOHN,H,7,678.43,150.11,WI
WILLIAM,H,11,3231.75,644.55,MN
ANDREW,S,24,1762.11,476.13,MN
BENJAMIN,S,3,201.11,25.21,IL
JANET,S,1,98.11,125.32,WI
STEVE,H,21,6153.32,1507.12,WI
JENNIFER,S,1,542.11,134.24,IL
JOY,S,12,2442.22,761.98,WI
MARY,S,14,5691.78,2452.11,WI
TOM,S,5,5669.12,798.15,MN
BETH,H,12,4822.12,982.1,WI
```



Creating A CSV File With SSCFLAT Macro

A macro to convert any SAS[®] dataset to a flat file.

Features:

- will convert any SAS dataset
- runs under windows, OS/390, UNIX
- user does not need to know the contents
- can provide a header row with field names
- will honor most SAS formats assigned to variables
- displays record and byte counts
- written and supported by SSC



A SSCFLAT Example

Just name the input and output datasets.

```
data softsale;  
  etc.  
Run;
```

```
%include 'a:\sscflat.sas';
```

```
%sscflat(msasds=softsale,mflatout=c:\temp\softsale.csv,  
         mlabel=YES)
```

Notes:

- SAS 9 has a new macro called %DS2CSV(data=dsn, csvfile=file)

The Resulting Flat File



The file is similar to before and imports similarly.

```
"Employee Name", "Division", "Years", "Sales", "Expense", "State"  
"CHRIS", "H", 2, 233.11, 94.12, "WI"  
"MARK", "H", 5, 298.12, 52.65, "WI"  
"SARAH", "S", 6, 301.21, 65.17, "MN"  
"PAT", "H", 4, 4009.21, 322.12, "IL"  
"JOHN", "H", 7, 678.43, 150.11, "WI"  
"WILLIAM", "H", 11, 3231.75, 644.55, "MN"  
"ANDREW", "S", 24, 1762.11, 476.13, "MN"  
"BENJAMIN", "S", 3, 201.11, 25.21, "IL"  
"STEVE", "H", 21, 6153.32, 1507.12, "WI"  
"JOY", "S", 12, 2442.22, 761.98, "WI"  
"MARY", "S", 14, 5691.78, 2452.11, "WI"  
"TOM", "S", 5, 5669.12, 798.15, "MN"
```



SSCFLAT Example under OS/390

```
//JOB1 JOB (XXXX, 'SSC TEST',MSGCLASS=A,MSGLEVEL=(2,1)
//SAS EXEC SAS
data softsale;
  infile, input, etc.
run;
%inc 'my.progs(sscflat)';
%sscflat(msasds=softsale,mprefix=my.data.);
```

Creates MY.DATA.SOFTSALE.DAT

Accessing Excel using ODBC connection(V8)



Advantages:

- File reference in Windows, can be global.
- Can read or reference MANY types of files, not just EXCEL

Disadvantages:

- Extra layer to run thru
- ODBC may not be efficient

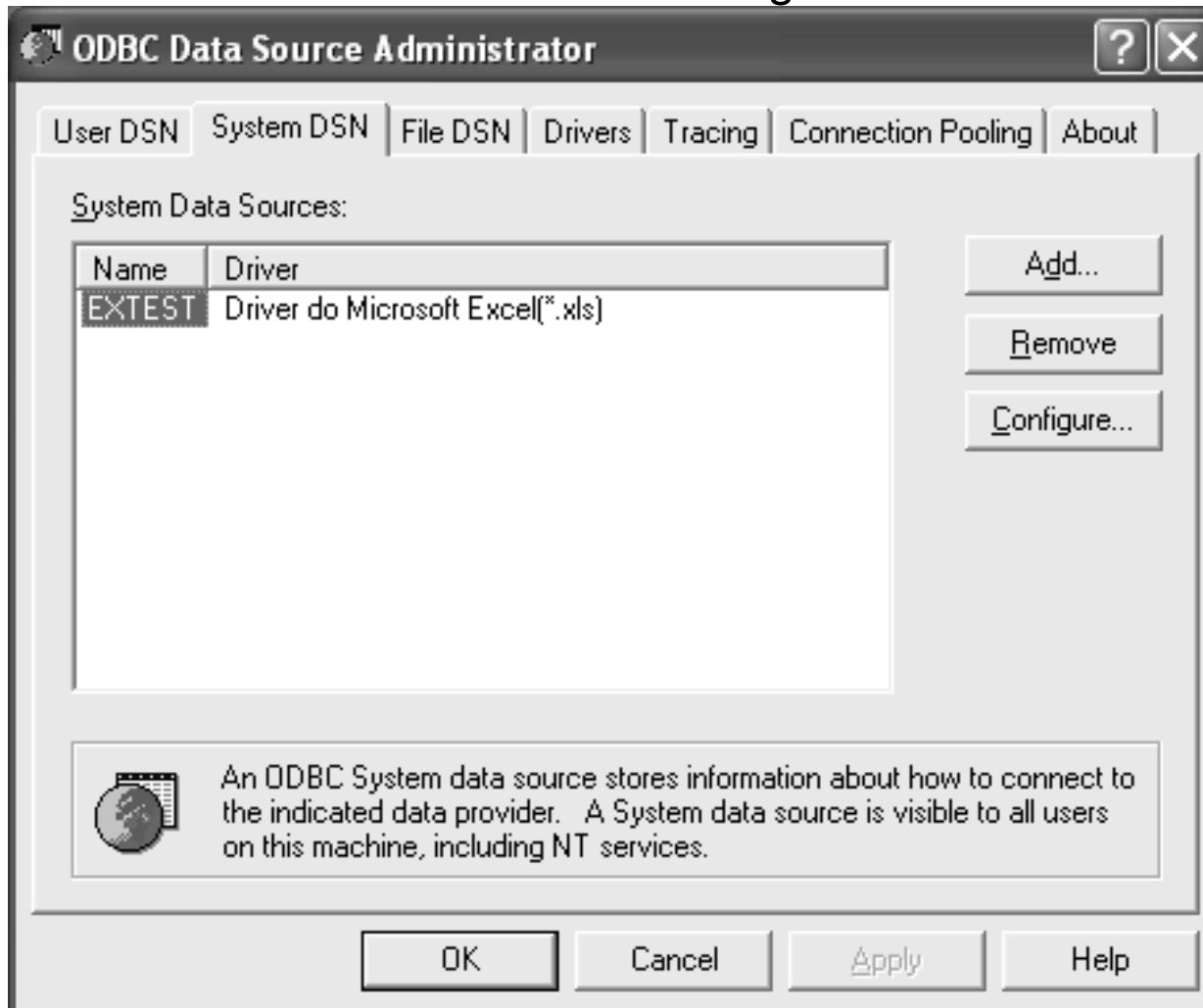
Notes:

- ODBC connections are maintained in the Windows environment.



Using ODBC connections(V8)

- Open the ODBC administrator to manage connections.



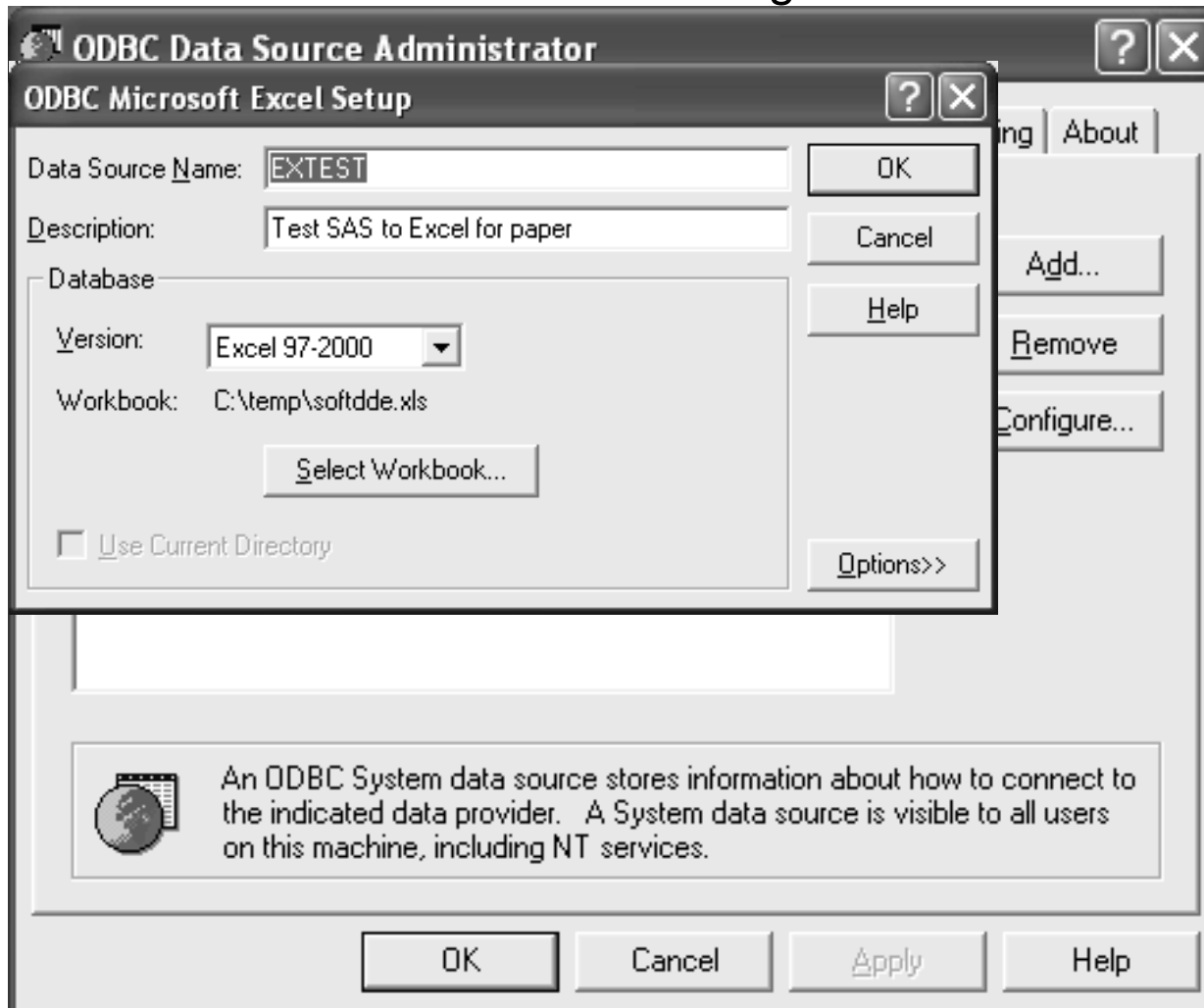
Notes:

- Select an existing connection or click ADD to create a new connection.



Setup ODBC connections(V8)

- Open the ODBC administrator to manage connections.



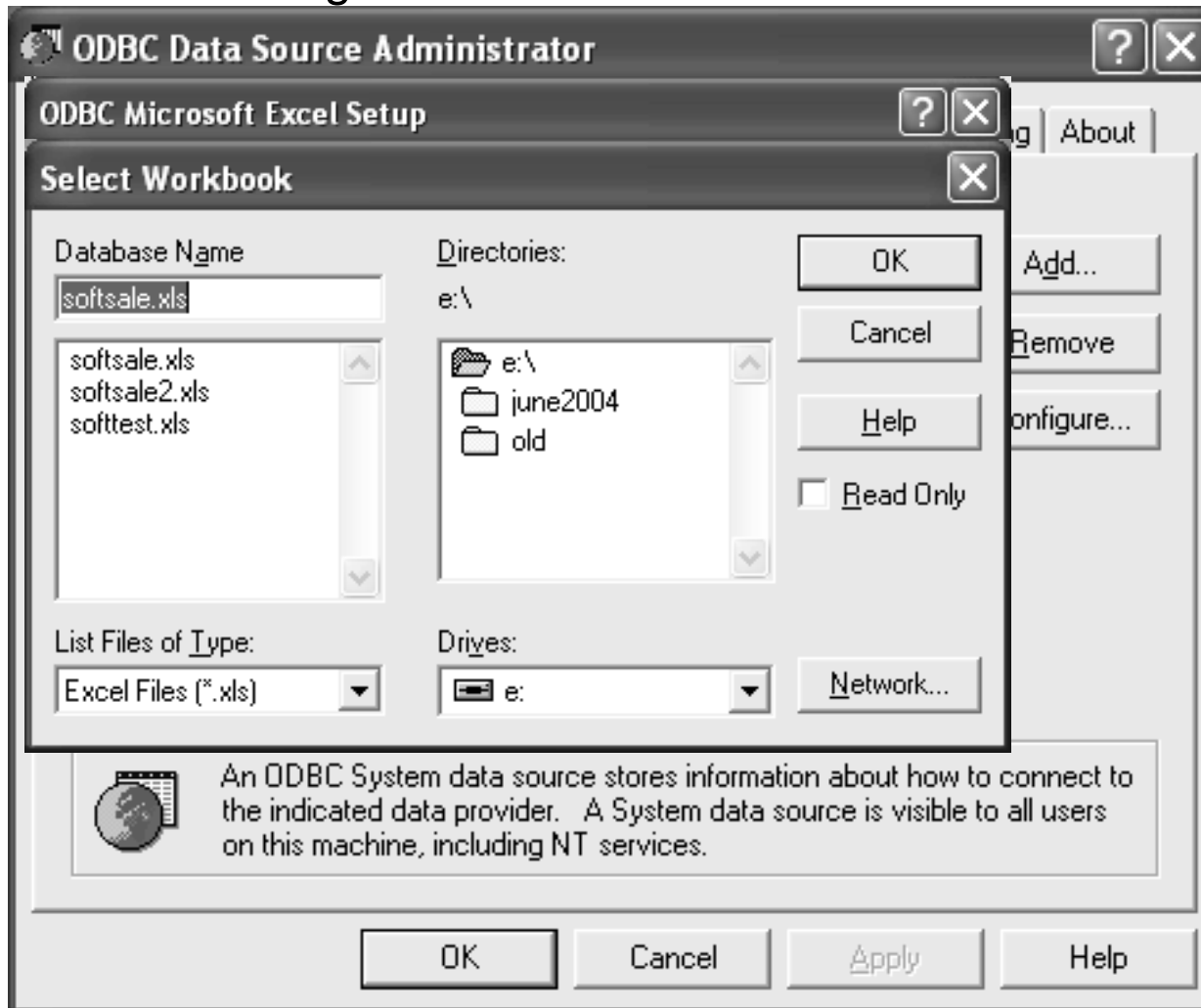
Notes:

- Select an existing connection or click ADD to create a new connection.
- Click on Select workbook to manage data source.



Select Data Source

- Click on the existing workbook and OK to use.



Notes:

- To create a new spreadsheet give it a name.



Using the ODBC connection

- Use the ODBC connections in Windows as a File reference in SAS.

```
libname myodbc odbc dsn=extest;  
Proc contents data=myodbc.softsale;  
run;
```

```
proc print data=myodbc.softsale;  
run;
```



The Resulting Contents Report in SAS

Note Lengths are longer than we expect!

The SAS System		1				
The CONTENTS Procedure						
Data Set Name	MYODBC.softsale	Observations .				
Member Type	DATA	Variables 6				
Engine	ODBC	Indexes 0				
Created	.	Observation Length 0				
Last Modified	.	Deleted Observations 0				
.						
Alphabetic List of Variables and Attributes						
#	Variable	Type	Len	Format	Informat	Label
2	Division	Char	255	\$255.	\$255.	Division
5	Expense	Num	8			Expense
1	Name	Char	255	\$255.	\$255.	Name
4	Sales	Num	8			Sales
6	State	Char	255	\$255.	\$255.	State
3	Years	Num	8			Years



The Starting Excel File

Note Softsale and Softdiv are the worksheet names.

	A	B	C	D	E	F	G	H	I
1	Name	Division	Years	Sales	Expense	State			
2	CHRIS	H	2	233.11	94.12	WI			
3	MARK	H	5	298.12	52.65	WI			
4	SARAH	S	6	301.21	65.17	MN			
5	PAT	H	4	4009.21	322.12	IL			
6	JOHN	H	7	678.43	150.11	WI			
7	WILLIAM	H	11	3231.75	644.55	MN			
8	ANDREW	S	24	1762.11	476.13	MN			
9	BENJAMIN	S	3	201.11	25.21	IL			
10	JANET	S	1	98.11	125.32	WI			
11	STEVE	H	21	6153.32	1507.12	WI			
12	JENNIFER	S	1	542.11	134.24	IL			
13	JOY	S	12	2442.22	761.98	WI			
14	MARY	S	14	5691.78	2452.11	WI			
15	TOM	S	5	5669.12	798.15	MN			
16	BETH	H	12	4822.12	982.1	WI			
17									
18									
19									
20									
21									
22									
23									
24									
25									



Using the EXCEL Libname engine

Advantages:

- Direct access to EXCEL workbooks
- Can create multiple worksheets
- Can stored formatted data values

Disadvantages:

- Runs in Windows/UNIX environments
- Does not support REPLACE of existing table. (Can create New or Read existing.)
- SAS labels, formats, and lengths are not stored

Notes:

- EXCEL libname engine is new in SAS 9.



Libname Options using the EXCEL engine

Read using the EXCEL engine.

```
LIBNAME libref <engine-name> <physical-file-name>  
      <SAS/ACCESS-engine-connection-options>  
      <SAS/ACCESS-libname-options>;
```

Assign a Libref to an existing EXCEL workbook in 'e:\softsale.xls'.

```
libname myexcel excel 'e:\softsale.xls';
```



List existing Table worksheet names

Read an existing worksheet and create a new worksheet.

```
libname myexcel excel 'e:\softsale.xls';
```

```
proc contents data=myexcel._all_ nods;  
run;
```



The Resulting Report in SAS

Note Worksheets are listed.

```
The SAS System 1
The CONTENTS Procedure

      Directory
Libref          MYEXCEL
Engine          EXCEL
Physical Name   e:\softsale.xls
Schema/Owner   Admin

      DBMS
#  Name          Member  Member
#  Name          Type    Type
1  SOFTSALE      DATA  TABLE
2  SOFTSALE$    DATA  TABLE
3  Softdiv$     DATA  TABLE
```

Creating a worksheet using EXCEL libname engine



Read an existing worksheet and create a new worksheet.

```
libname myexcel excel 'e:\softsale.xls';

data myexcel.harddiv;          /* new excel sheet name */
  set myexcel.softsale;      /* existing worksheet   */
  where upcase(division)='H'; /* filter input        */
run;                          /* end of step         */
```



The Resulting Excel File

Note Harddiv is the name of the new worksheet.

The screenshot shows a Microsoft Excel window titled 'Microsoft Excel - softsale.xls'. The worksheet 'harddiv' contains a table with the following data:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Name	Division	Years	Sales	Expense	State									
2	CHRIS	H	2	233.11	94.12	WI									
3	MARK	H	5	298.12	52.65	WI									
4	PAT	H	4	4009.21	322.12	IL									
5	JOHN	H	7	678.43	150.11	WI									
6	WILLIAM	H	11	3231.75	644.55	MN									
7	STEVE	H	21	6153.32	1507.12	WI									
8	BETH	H	12	4822.12	982.1	WI									
9															
10															
11															
12															
13															
14															
15															
16															
17															
18															
19															
20															
21															
22															
23															
24															
25															
26															
27															
28															
29															
30															
31															
32															
33															
34															

Creating a worksheet using EXCEL libname engine



Read an existing worksheet and create a new worksheet.

```
libname myexcel excel 'e:\softsale.xls';

data myexcel.harddiv2;          /* new excel sheet name */
  set myexcel.softsale;       /* existing worksheet   */
  where upcase(division)='H';  /* filter input         */
  format sales dollar12.2;     /* set format           */
run;                           /* end of step          */
```



The Resulting Excel File

Note Sales values are stored as Formatted values.

The screenshot shows a Microsoft Excel window titled "Microsoft Excel - softsale.xls". The spreadsheet contains the following data:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Name	Division	Years	Sales	Expense	State									
2	CHRIS	H	2	\$233.11	94.12	WI									
3	MARK	H	5	\$298.12	52.65	WI									
4	PAT	H	4	\$4,009.21	322.12	IL									
5	JOHN	H	7	\$678.43	150.11	WI									
6	WILLIAM	H	11	\$3,231.75	644.55	MN									
7	STEVE	H	21	\$6,153.32	1507.12	WI									
8	BETH	H	12	\$4,822.12	982.1	WI									
9															
10															
11															
12															
13															
14															
15															
16															
17															
18															
19															
20															
21															
22															
23															
24															
25															
26															
27															
28															
29															
30															
31															
32															
33															
34															



Sending Reports to Excel With ODS

Advantages:

- Probably the easiest way to get reports to Excel (and other programs)
- Available on all platforms both in batch or interactively
- ODS captures **reports**
- Excel can capture the **data** shown on the ODS report
- Output will look nice in Excel and others

Disadvantages:

- File sizes can get large if reports are large.



Sending Reports to Excel With ODS

Since Excel automatically converts HTML to XLS format during File Open, to send any SAS[®] report to Excel:

1. Creating a CSV file with ODS or
2. Create a HTML file with ODS or
3. Create a XML file with ODS

Then:

4. Use Excel File Open to read the CSV, HTML or XML file
5. Format as required in Excel
6. Use Excel File Save As to specify XLS format.

A Typical Report



PROC TABULATE produces a table.

```
proc tabulate data=softsale;
  title 'Softsale Sales and Expenses by Division';
  class state division ;
  var sales expense;
  table state,division*sales*sum
          division*expense*sum;
run;
```

The Current Text Report



How can we get this report into a worksheet?

Softsale Sales and Expenses by Division

	Division		Division	
	H	S	H	S
	Sales	Sales	Expense	Expense
	Sum	Sum	Sum	Sum
State				
IL	4009.21	743.22	322.12	159.45
MN	3231.75	7732.44	644.55	1339.45
WI	12185.10	8232.11	2786.10	3339.41



Creating A CSV file using ODS

ODS CSVALL destination produces a file.

```
ods listing close;
ods csvall body='c:\temp\tabulate.csv';
proc tabulate data=softsale;
  title 'Softsale Sales and Expenses by Division';
  class state division ;
  var sales expense;
  table state,division*sales*sum
         division*expense*sum;
run;
ods csv close;
ods listing;
```

Notes: The CSVALL destination is new in SAS 9.



The Resulting Spreadsheet

Opening the file(1kb) shows:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1															
2															
3		Division		Division											
4				H	S	H	S								
5				Sales	Sales	Expense	Expense								
6				Sum	Sum	Sum	Sum								
7	State	4009	743.2	322.1	159.5										
8	IL														
9	MN	3232	7732	644.6	1339										
10	WI	12185	8232	1786	1339										
11															
12															
13															
14															
15															
16															
17															
18															
19															
20															
21															
22															
23															
24															
25															
26															
27															
28															
29															

Notes: There may be some formatting issues, for example: the title is dropped and some column headings are shifted.



Example of Exporting HTML to Excel

Route the report to an HTML file.

```
ods listing close;
ods html body='c:\temp\tabulate.html';
proc tabulate data=softsale;
  title 'Softsale Sales and Expenses by Division';
  class state division ;
  var sales expense;
  table state,division*sales*sum
         division*expense*sum;
run;
ods html close;
ods listing;
```



The Resulting Worksheet

The HTML file(6kb) is automatically converted to XLS format.

The screenshot shows an Excel spreadsheet with the following data:

<i>Softsale Sales and Expenses by Division</i>				
	Division		Division	
	H	S	H	S
	Sales	Sales	Expense	Expense
	Sum	Sum	Sum	Sum
State				
IL	4009.2	743.22	322.12	159.45
MN	3231.8	7732.4	644.55	1339.45
WI	12185	8232.1	1786.1	1339.41

Notes:

Upon using “File Save As”, the worksheet should be saved in XLS format.



Giving the file a XLS extension.

When the file is given XLS as an extension

- The file still contains HTML even though suffix is XLS
- Users and Explorer etc. view it as XLS
- Excel File Open will automatically opens in XLS format
- Excel File Save will automatically save as XLS format



Reducing File Size.

There are several ways to reduce the ODS file size.

- Use the ODS HTML option **STYLE=MINIMAL**
- Using a custom **STYLESHEET**(Not covered in this presentation)
- ODS PHTML to produce basic HTML
- ODS CHTML for compact HTML



STYLE=MINIMAL and a XLS extension.

Reproduce the PROC TABULATE Output.

```
options nocenter;
ods html body='c:\temp\tabulatemin.xls' style=minimal;
proc tabulate data=softsale;
  title 'Softsale Sales and Expenses by Division';
  class state division ;
  var sales expense;
  table state,division*sales*sum
         division*expense*sum;
run;
ods html close;
```



The Resulting Worksheet

Excel has a normal worksheet(3kb).

The screenshot shows an Excel spreadsheet with a table titled "Softsale Sales and Expenses by Division". The table is structured as follows:

State	Division		Division	
	H	S	H	S
	Sales	Sales	Expense	Expense
	Sum	Sum	Sum	Sum
IL	4009.21	743.22	322.12	159.45
MN	3231.75	7732.44	644.55	1339.45
WI	12185.1	8232.11	1786.1	1339.41

Notes:

- This is probably the easiest way to get any report to Excel
- You may have to do some slight formatting



Using ODS PHTML.

Change to use the ODS PHTML destination.

```
options nocenter;
ods phtml body='c:\temp\tabulatep.xls';
proc tabulate data=softsale;
  title 'Softsale Sales and Expenses by Division';
  class state division ;
  var sales expense;
  table state,division*sales*sum
         division*expense*sum;
run;
ods phtml close;
```

Notes: PHTML reduces formatting in HTML.

- The PHTML destination is experimental in SAS version 8.2.



The Resulting Worksheet

Excel has a normal worksheet(3kb).

The screenshot shows an Excel spreadsheet with the following data:

Softsale Sales and Expenses by Division					
State	Division		Division		
	H	S	H	S	
	Sales	Sales	Expense	Expense	
	Sum	Sum	Sum	Sum	
IL	4009.21	743.22	322.12	159.45	
MN	3231.75	7732.44	644.55	1339.45	
WI	12185.1	8232.11	1786.1	1339.41	

(Continued)

Notes:

- This is very similar to ODS HTML with Style= minimal



Using ODS CHTML.

If formatting is not required, but saving space is, use the ODS CHTML destination.

```
options nocenter;
ods chtml body='c:\temp\tabulatec.xls';
proc tabulate data=softsale;
  title 'Softsale Sales and Expenses by Division';
  class state division ;
  var sales expense;
  table state,division*sales*sum
         division*expense*sum;
run;
ods chtml close;
```

Notes: CHTML produces compact HTML.

- The CHTML destination is experimental in SAS version 8.2.
- SAS Excels! - Systems Seminar Consultants, Inc.



The Resulting Worksheet

Excel has a normal worksheet(2kb).

The screenshot shows an Excel spreadsheet window with the following data:

Softsale Sales and Expenses by Division				
State	Division		Division	
	H	S	H	S
	Sales	Sales	Expense	Expense
	Sum	Sum	Sum	Sum
IL	4009.21	743.22	322.12	159.45
MN	3231.75	7732.44	644.55	1339.45
WI	12185.1	8232.11	1786.1	1339.41

Notes:

- This is very similar to ODS HTML with Style= minimal.



Miscellaneous Notes

- Excel may format things differently
Example: Leading zeros don't show.
- Titles may not span columns in the worksheet the way you would like it to
 - We can Span Columns
 - Use ODS HTMLCSS
- Use RS=NONE on the HTML statement under OS/390 for a text file:
ODS HTML BODY=HTMLOUT STYLE=MINIMAL **RS=NONE**;

Tricks:

- Use the colspan HTML tag in titles that you want to span
- Use style sheets as required for Excel to preserve leading zeros

Title Spanning example



- Titles can be forced to span columns.
- In the quoted Title string add HTML tags including `colspan=n` where *n*= number of columns to span.

Title Example:

```
ODS HTML body='c:\temp\title.xls';  
title '<td colspan=4>Softsale Sales by State and  
    Division</td>';  
proc print data=softsale;  
  var state division sales expense years;  
run;  
ODS HTML close;
```

The Resulting Report In Excel(13kb)



The screenshot shows an Excel spreadsheet with the following data:

Obs	State	Division	Sales	Expense	Years
1	WI	H	233.11	94.12	2
2	WI	H	298.12	52.65	5
3	MN	S	301.21	65.17	6
4	IL	H	4009.21	322.12	4
5	WI	H	678.43	150.11	7
6	MN	H	3231.75	644.55	11
7	MN	S	1762.11	476.13	24
8	IL	S	201.11	25.21	3
9	WI	S	98.11	125.32	1
10	WI	H	6153.32	507.12	21
11	IL	S	542.11	134.24	1
12	WI	S	2442.22	761.98	12
13	WI	S	5691.78	452.11	14
14	MN	S	5669.12	798.15	5
15	WI	H	4822.12	982.1	12

Notes: Title starts in cell 2.



HTMLCSS example

- Titles can be forced to span columns.
- In the quoted Title string add HTML tags including `colspan= n` where n = number of columns to span.

HTMLCSS Example:

```
ODS HTMLCSS body='c:\temp\title2.xls';  
title 'Softsale Sales by State and Division';  
proc print data=softsale;  
  var state division sales expense years;  
run;  
ODS HTMLCSS close;
```

Notes: The ODS HTMLCSS destination produces HTML that refers to Cascading Style Sheets.

- The HTMLCSS destination is experimental in SAS version 8.2.
- SAS Excels! - Systems Seminar Consultants, Inc.



The Resulting Report In Excel(6kb)

The screenshot shows an Excel spreadsheet window with the following data:

Obs	State	Division	Sales	Expense	Years
1	WI	H	233.11	94.12	2
2	WI	H	298.12	52.65	5
3	MN	S	301.21	65.17	6
4	IL	H	4009.21	322.12	4
5	WI	H	678.43	150.11	7
6	MN	H	3231.75	644.55	11
7	MN	S	1762.11	476.13	24
8	IL	S	201.11	25.21	3
9	WI	S	98.11	125.32	1
10	WI	H	6153.32	507.12	21
11	IL	S	542.11	134.24	1
12	WI	S	2442.22	761.98	12
13	WI	S	5691.78	452.11	14
14	MN	S	5669.12	798.15	5
15	WI	H	4822.12	982.1	12

Notes: Title starts in cell 1.



Preserving Leading Zeros into Excel

For Office 97:

```
ODS HTML body='c:\temp\ProcPrint.xls' style=minimal;
title '<td colspan=4>Softsale Sales by State and
      Division</td>';
proc print data=softsale;
  var state division sales expense;
  var years / style={htmlstyle="vnd.ms-
    excel.numberformat:@"};
  format years z4.;
run;
ODS HTML close;
```



Preserving Leading Zeros into Excel

For Excel newer than 97:

```
ODS HTML file='c:\temp\ProcPrint.xls'  
  Headtext='<style>td{mso-number-format:\@;}</style>';  
  
title '<td colspan=4>Softsale Sales by State and  
  Division</td>';  
proc print data=softsale;  
  var state division sales expense years;  
  format years z4.;  
run;  
ODS HTML close;
```



The Resulting PROC PRINT in Excel 97

Obs	State	Division	Sales	Expense	Years
1	WI	H	233.11	94.12	0002
2	WI	H	298.12	52.65	0005
3	MN	S	301.21	65.17	0006
4	IL	H	4009.2	322.12	0004
5	WI	H	678.43	150.11	0007
6	MN	H	3231.8	644.55	0011
7	MN	S	1762.1	476.13	0024
8	IL	S	201.11	25.21	0003
9	WI	S	98.11	125.32	0001
10	WI	H	6153.3	507.12	0021
11	IL	S	542.11	134.24	0001
12	WI	S	2442.2	761.98	0012
13	WI	S	5691.8	452.11	0014
14	MN	S	5669.1	798.15	0005
15	WI	H	4822.1	982.1	0012

HTML from other SAS[®] Procedures



Try a PROC REPORT to HTML:

```
ods html body='c:\temp\ProcReport.xls' style=minimal;
title '<td colspan=4>Softsale Sales by State and
  Division</td>';
proc report data=softsale nowd;
  columns state division sales expense;
  define state / group width=6 'State';
  define division / group width=5 'Sales/Div.';
  define sales / analysis width=12
    format=dollar12.2 'Sales Amt.';
  define expense / analysis width=8
    format=comma8. 'Expenses';
run;
ods html close;
```



The Resulting PROC REPORT in Excel

State	Sales Div.	Sales Amt.	Expenses
IL	H	\$4,009.21	322
	S	\$743.22	159
MN	H	\$3,231.75	645
	S	\$7,732.44	1,339
WI	H	\$12,185.10	1,786
	S	\$8,232.11	1,339

Notes:

- The Excel worksheet can be modified as you see fit.



Best of both worlds

Create a HTML file and allow the client to view it in a browser, AND still have the option to save it as an Excel file!

We need to create a style using PROC TEMPLATE

We need to create the HTML Report using the style we just created.

Then we can display the report in a web browser (HTML) with a button labeled 'Download to Excel'.

The Template



Copy an existing style and add the click to save button.

```
proc template;
  define style styles.test;
    parent=styles.default;
    style body from body /
      prehtml='<input
onclick="document.execCommand(' 'SAVEAS' ',true,
  ' 'c:\\temp\\test.xls' ')" value= "Download To Excel"
type="button">';
  end;
run;
```

Notes: This opens the SAVE AS dialog box. Because we are entering Javascript commands, some quotes and slashes are doubled.

The Template



Now use the changed style to create a HTML report for viewing in a Browser:

```
ods html file='c:\temp\click.html' style=styles.test;
proc tabulate data=softsale;
  title 'Softsale Sales and Expenses by Division';
  class state division ;
  var sales expense;
  table state,division*sales*sum
         division*expense*sum;
run;
ods html close;
```



The Resulting HTML REPORT in Web Browser

Opening the HTML file in a Web Browser:

The screenshot shows a Microsoft Internet Explorer browser window displaying an HTML report. The address bar shows the file path C:\temp\click.html. The report title is "Softsale Sales and Expenses by Division". A "Download To Excel" button is visible at the top left of the report content. The report contains a table with the following data:

	Division		Division	
	H	S	H	S
	Sales	Sales	Expense	Expense
	Sum	Sum	Sum	Sum
State				
IL	4009.21	743.22	322.12	159.45
MN	3231.75	7732.44	644.55	1339.45
WI	12185.10	8232.11	1786.10	1339.41

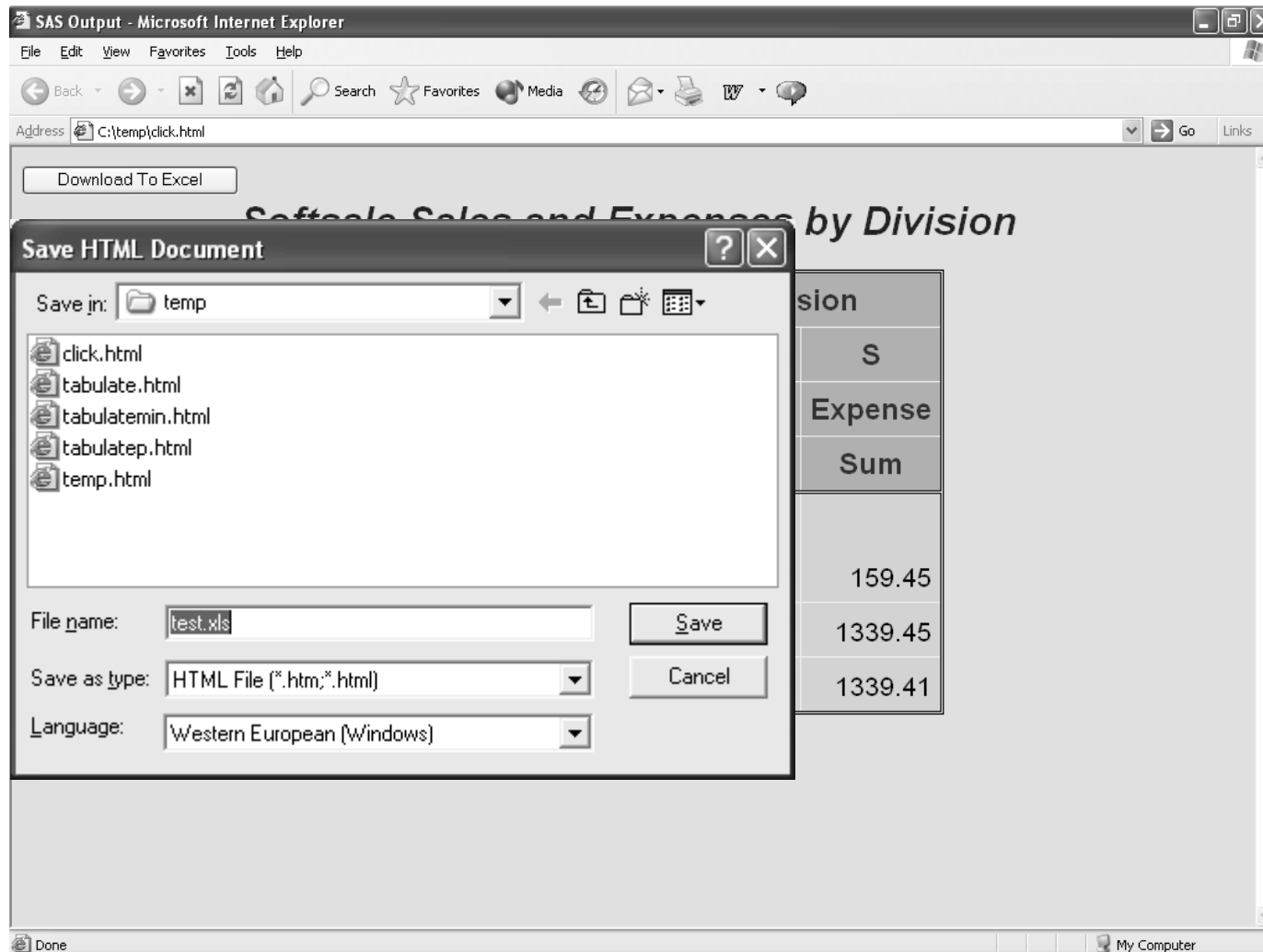
Notes:

- The HTML report displays nicely.
- SAS Excels! - Systems Seminar Consultants, Inc.



The SAVE AS Dialog Box

Clicking on DOWNLOAD TO EXCEL button results:



Notes:

- The default location/name was specified in the TEMPLATE.
- SAS Excels! - Systems Seminar Consultants, Inc.

Questions and Comments?



Gerald Frey

608 278 964 x321

gfrey@sys seminar.com

References:

- 1) The Creative Consultants at Systems Seminar Consultants, Inc.
- 2) “Using ODS to Generate Excel Files”, Chevell Parker,
SAS Institute