

READING COMMA-DELIMITED DATA INTO SAS

In the October 2000 issue we showed how our SAS macro called "SSCFLAT" can be used to export data easily from SAS to a comma-delimited flat file. This flat file can be imported into a variety of software packages, such as Excel or Word.

I received a question shortly after the issue was published asking how that type of flat file can be read back into a SAS program. It seems the user often receives data files from a vendor with the data in comma-delimited format.

Let's look at the issues involved in reading the following text file into a SAS data step. Suppose the raw data file looks like this:

```
"NAME","CITY","STATE","SALES DOLLARS","SELDATE"
"Jones, Tom","Phoenix","AZ",3456.85,02/18/2000
"Lake, Amy","New York","NY",,02/13/1999
"Cook, Bob","Chicago","IL",245.1,02/06/2000
```

Puzzler #4

A recent request required several balances on an individual account to be summed together and the account average balance to be determined.

Once all of the individual account balances and account averages were calculated, the grand total balance and the total average balance needed to be found and output to a report. The dataset, program, and output are shown on page 3.

The actual results in the report were significantly higher than the expected results. Why was this so? What is the most efficient way to get the correct results?

The first three readers to fax correct answers to us at (608) 278-0065 will each win a \$100 training certificate and a coffee mug. See our next issue, April 2001, for the results.

Problems

The variable names or descriptions are on the first line, which we have to skip over while reading the file. The data contains a mixture of character and numeric data, and we need to be concerned about the length of the character fields. Notice the last field is a date, which we would like to read into SAS as a numeric date value.

SAS cannot read the above file and figure out the length of the character fields without help from the programmer. So we will build the data step using several SAS statements to describe the data being read in. We will also tell SAS the raw data is comma-delimited.

INFILE Statement

The INFILE statement has several options available, two of which we will use here:

FIRSTOBS=2 tells SAS to start reading the file from the second record, bypassing the row with the field names.

DSD signifies comma-delimited and tells SAS to treat two consecutive delimiters as a missing value. Quotation marks will be removed from character values.

LENGTH Statement

We can tell SAS the size of new variables being created with the following code. Notice the use of the '\$' symbol signifying character variables. We also are setting the date variable to 4 bytes numeric, as this is large enough for date values:

```
LENGTH name $20 city $15 state $2
selldate 4;
```

INPUT Statement

We will use the "list" style of input statement, which can read this data by naming the variables in the order they appear on the raw data line. The date field can be read in correctly by using an "informat" of MMDDYY10., which

Continued on page.....7

IN THIS ISSUE

Reading Comma-Delimited Data Into SAS: David Beam explains how to read flat files back into your SAS programs.....Page 1

Puzzler #4: Your chance to win a \$100 Training Certificate.....Page 1


Creating SAS/GRAPH® Output for the World Wide Web: Steve First gives an overview and shows examples of how to create graphics for the web using SAS/GRAPH drivers....Page 2

Using Macros to Generate Titles - Two Approaches: Steve First shows two ways you can use macros to generate titles that place certain text in specific columnsPage 4

SAS® Help Desk I/O: Katie Minten Ronk answers help desk questions on merging using PROC SQL and on permanent formats stored in a catalog.....Page 5

Technical Credit:.....Page 7

Public Class Schedule:.....Page 8



SYSTEMS SEMINAR CONSULTANTS, INC.
 2997 Yarmouth Greenway Drive
 Madison, WI 53711
 Web site: www.sys-seminar.com
 Email: train@sys-seminar.com
 Phone: (608) 278-9964
 Fax: (608) 278-0065

A NEW YEAR AND A NEW CLASS

INTRODUCTION TO SAS CLASS REVISED



I'm writing this issue's greeting while Steve First is on a well-deserved vacation, in a bit warmer climate! Here in Wisconsin the snow is really piling up; I've already shoveled more snow this winter than the past several winters combined!

As we head into the new millennium (this time the real one), we are proud to announce our revised *Introduction to SAS* training course. This major update to our most popular training class has been in the works for some time. New features of Version 8 have been included. We have also incorporated many suggestions received from students over the years.

In our humble opinion, the revised materials are well-paced, with more frequent hands-on exercises than in the past. As with all of our training classes, the introductory course can be held on-site at your company or you can attend one of our public training sessions in Minneapolis, MN or Madison, WI.

As the SAS software continues to evolve, so do we at SSC. We always strive to keep our skills current. Several of our consulting projects in recent months have incorporated some of the internet capabilities of SAS software, and we look forward to continuing this evolution. One thing about this industry is certain - changes will continue, and we hope to stay at the forefront of those changes.

Here's wishing you and your organizations a safe, exciting and enjoyable New Year!

Sincerely,

David Beam
Vice President



CREATING SAS/GRAPH® OUTPUT

FOR THE WORLD WIDE WEB

For many years, SAS/GRAPH has been able to produce high resolution color graphics for a variety of devices. Graphics can be captured to a file. The file can then be referenced through a web page, allowing users to view the page through a web browser from literally anywhere in the world.

Graphs can be routed to the web either with or without help from the SAS Output Delivery System (ODS). The emphasis of this article is to produce the web output without ODS. ODS and SAS/GRAPH will be revisited in a future article.

The general design without ODS is to use a compatible SAS/GRAPH driver to produce a file that can be referenced in an HTML page. When the web browser displays the page, it links to the graphics file, reads it, and displays the corresponding image.

There are at least two kinds of image files that are widely used for web graphics:

- GIF (Graphics Interchange Format) files are used for reduced color images, icons, and other drawings. SAS/GRAPH usually produces graphics of this type. There are several drivers with different resolutions available.
- JPEG (Joint Photographic Experts Group) files are typically used for photorealistic images.

Using SAS/GRAPH Drivers

This partial list of SAS/GRAPH drivers gives us several choices for our application. The driver names and approximate sizes of images are:

GIF	792 x 598 pixels	GIF160	160 x 120 pixels
GIF260	260 x 195 pixels	GIF373	373 x 280 pixels
GIF570	570 x 480 pixels	GIF733	733 x 550 pixels
JPEG	600 x 300 pixels		

Image sizes are approximately 100 pixels per inch at 800 x 600 pixel resolution. Other drivers are available and are documented in the SAS documentation. The GOPTIONS DEVICE= option can be used to choose one of the above drivers.

Example

The job below produces a bar chart of interest rates and creates a GIF file called `examp1.gif`.

```
filename gsasfile 'c:\temp\examp1.gif';
goptions gaccess=gsasfile device=gif;
proc gchart data=mortdata;
  vbar year/sumvar=rate discrete
        sum space=2 width=5 ;
  pattern1 c=red v=solid;
  title c=black 'Home Mortgage Rates';
  title2 c=black '(Annual Percentage Rates)';
  where region='East';
run;
```



SYSTEMS SEMINAR CONSULTANTS, INC.

Copyright © 2001 Systems Seminar Consultants, Inc. Madison, WI

All rights reserved. Printed in USA. The Missing Semicolon is a trademark of Systems Seminar Consultants, Inc., SAS is a registered trademark of SAS Institute Inc., SyncSort is a registered trademark of SyncSort, Inc. in the USA and other countries.

In order to see the output via a web browser, an HTML file can be edited and an image tag inserted that points to the GIF file. A typical HTML file might look like the following:

```
<HTML>
<HEAD>
<TITLE>SAS/GRAPH</TITLE>
</HEAD>
<P>
<IMG SRC="examp1.gif">
</HTML>
```

Viewing the output with a browser displays the image. Note that the graph doesn't completely display, but scroll bars can be used to view the remainder of the graph. An HTML file can contain more than one image tag to display two graphics at one time. Scroll bars can be used to view both graphics.



Using Pixel Options to Define Size

The XPIXELS= and YPIXELS= options can be specified to give exact horizontal and vertical measurement in pixels, so that the entire image shows at one time.

```
filename gsasfile 'c:\temp\examp2.gif';
goptions gaccess=gsasfile device=gif
        border xpixels=500 ypixels=400;
proc gchart data=mortdata;
vbar year/sumvar=rate discrete
    sum space=2 width=5;
pattern1 c=red v=solid;
title c=black 'Home Mortgage Rates';
title2 c=black
    '(Annual Percentage Rates)';
where region='East';
run;
```



Continued on page6

QUICK TIP

The "FILEEXIST()" function can tell you if a file exists on the system. It returns 0 if the file does NOT exist and 1 if it does.

Examples:

SAS Code	file exist?	X value
X=FILEEXIST('ABC.XYZ.TEST');	Yes	1
X=FILEEXIST('C:\MYDATA\XYZ.DAT');	No	0
X=FILEEXIST('ABC.XYZ.PDS(GOOD)');	Yes	1



PUZZLER DATASET, PROGRAM, & OUTPUT

PUZZLER APPEARS ON PAGE 1

Dataset involved

The DSONE dataset used in the puzzler is shown below:

Obs	ACCT	BAL1	BAL2	BAL3
1	A	15000	20000	18500
2	B	17500	21000	22750
3	C	13000	19500	16500

Initial Program and Output (Unexpected Results)

```
DATA CALCS;
SET DSONE;
ARRAY BAL (3) BAL1-BAL3;
DO I = 1 TO 3;
    BALSUM + BAL(I);
END;
AVGBAL = BALSUM / 3;
RUN;

PROC SUMMARY DATA = CALCS;
VAR BALSUM
    AVGBAL;
OUTPUT OUT = TOTALS
    SUM(BALSUM) = TOTALBAL
    SUM(AVGBAL) = TOTALAVG;
RUN;

PROC PRINT DATA = TOTALS;
TITLE 'UNEXPECTED AMOUNTS';
VAR TOTALBAL
    TOTALAVG;
FORMAT TOTALBAL
    TOTALAVG
    DOLLAR9.;
RUN;
```

UNEXPECTED AMOUNTS		
Obs	TOTALBAL	TOTALAVG
1	\$332,000	\$110,667

Expected Results

EXPECTED AMOUNTS		
Obs	TOTALBAL	TOTALAVG
1	\$163,750	\$54,583



USING MACROS TO GENERATE TITLES

TWO APPROACHES

A recent project required titles that put certain text in specific columns. For example, the company name had to be in column 3, the date in column 72, and so on. It wasn't an option to use DATA step reporting as the system used many reporting procs.

We decided to write a macro that would use the call below specifying the respective title number, and the starting column and text for up to three blocks of text.

```
%ssctitlm(1,
           3,'left edge',
           20,'sample center',
           40,"date run:&sysdate")
```

The job of the macro would be to insert the correct number of spaces in the title's text string, so that the text ended up in the correct columns. At first it seemed to be a simple problem that should be solved easily. The macro below, after several iterations of coding and testing, worked.

First Approach

```
%macro ssctitlm(mtitlno,mcol1, mtext1,
                mcol2, mtext2,
                mcol3, mtext3);
/*****
/* macro ssctitlm
/* purpose: create sas title variables for sas procs
/* input: title number, starting col, text for up
/* to 3 segments of title.
/* output: title vars that can be used by sas
/*****
/* need data step logic to build data step variables,
/* put in macro variables at end.
/*****
%let noblank=%eval(&mcol1-1); /* # blanks on left */
%let mtext=%str(); /* set to null */
%do i=1 %to &noblank; /* do this # blanks */
%let mtext=%str(&mtext)%str( ); /* insert blank */
%end; /* end of blank loop*/

%let tlen=%eval(%length(&mtext1)-2); /* len(mtext1),no 's'*/
/* strip out 's */
%let mtext1=%substr(%str(&mtext1),2,&tlen);
%let mtext=%str(&mtext)%str(&mtext1); /* append to mtext */
/* # blanks to mtext2*/
%let noblank=%eval(&mcol2-1-%length(&mtext));
%do i=1 %to &noblank; /* do this # blanks */
%let mtext=%str(&mtext)%str( ); /* insert blank */
%end; /* end of blank loop*/

%let tlen=%eval(%length(&mtext2)-2); /* len(mtext2),no 's'*/
/* strip out 's */
%let mtext2=%substr(%str(&mtext2),2,&tlen);
%let mtext=%str(&mtext)%str(&mtext2); /* append to mtext */
/* # blanks to mtext3*/
%let noblank=%eval(&mcol3-1-%length(&mtext));
%do i=1 %to &noblank; /* do this # blanks */
%let mtext=%str(&mtext)%str( ); /* insert blank */
%end; /* end of blank loop*/

%let tlen=%eval(%length(&mtext3)-2); /* len(mtext3),no 's'*/
/* strip out 's */
%let mtext3=%substr(%str(&mtext3),2,&tlen);
%let mtext=%str(&mtext)%str(&mtext3); /* append to mtext */

title&mtitlno "&mtext"; /* make title stmt */
/***** end of macro ssctitlm *****/
%mend ssctitlm;
```

```
%ssctitlm(1,
           3,'left edge',
           20,'sample center',
           40,"date run:&sysdate")
```

The above macro call generated the following title statement:

```
title1 " left edge sample center date run:13DEC00";
```

As you can see it was not a trivial effort, because the macro language is somewhat limited in text substitution, calculation, and in looping capabilities, as well as being a fairly difficult language to code, test, and debug. In any case it worked just fine.

Second Approach

Another approach might be to use a more advanced language to do most of the work, and use macro features only when required. The two obvious choices that make sense would be to use the DATA step, or to use the SAS SCL language.

SCL can generate SAS code similar to what the macro facility produces, and many programmers like the power and function library available.

The DATA step can't really generate a lot of SAS code directly, but it can create macro variables that can be passed to the macro facility. This coupled with a very rich language to manipulate text, access to over 300 functions, as well as being a familiar and easy to debug language, makes the DATA step a good possible alternative.

The following macro generated essentially the same title statement, but with a lot less coding effort.

```
%macro ssctitle(mtitlno,mcol1,mtext1,
                mcol2,mtext2,mcol3,mtext3);
/*****
/* macro ssctitle
/* purpose: create sas title variables for sas procs
/* input: title number, starting col, text for up
/* to 3 segments of title.
/* output: title vars that can be used by sas
/*****
/* need data step logic to build data step variables,
/* put in macro variables at end.
/*****
data _null; /* use a data step */
length text $ 160; /* ds var to max len*/
text=' '; /* blank it out */
substr(text,&mcol1)=&mtext1; /* first text */
substr(text,&mcol2)=&mtext2; /* second text */
substr(text,&mcol3)=&mtext3; /* third text */
call symput("mtitle&mtitlno",text); /* ds vars->mac vars*/
run; /* end of data step */
title&mtitlno "&mtitle&mtitlno"; /* move to title */
/***** end of macro ssctitle *****/
%mend ssctitle;

%ssctitle(1,
          3,'left edge',
          20,'sample center',
          40,'date run:!!put("&sysdate"d,mmys5.) )
```

The above macro call generated the following title statement:

```
title1 " left edge sample center date run:12/00";
```

Continued on page.....5

SAS® HELP DESK I/O

SOLUTIONS FROM OUR HELP DESK SERVICE



"I am trying to join two datasets using PROC SQL. I am joining by two concatenated variables (the combination of first name and last name). Here is my program:

```
proc sql;
  create table mergfile as
  select a.fname, a.lname, a.age,
         b.adrln1, b.citycd, b.state,
         b.zipcod
  from   demos a left join
         address b
  on a.fname||a.lname=b.fname||b.lname;
quit;
```

I am not getting any results in my address fields even though I know I should have matches. Why isn't this working?"



When concatenating variables, SAS does not trim leading or trailing blanks; it keeps the length of each variable in creating the new variable. For example, if the value of fname only has 3 characters, yet the length of the variable is 8, there will be 5 blank spaces between the fname (first name) and the lname (last name) variable.

The reason the files are not joining properly is the first variable (first name) probably has a different length in each of the datasets being joined. If the lengths are not the same, there will be a different number of spaces between the two variables (first and last name), and thus, not an exact match. For example, if the length of first name in the first and second dataset are 8 and 10, respectively, here is what each of the concatenated values would look like:

```
FIRST DATASET : 12345678901234567890
                AMY      JONES
SECOND DATASET: AMY      JONES
```

Because these values would not have an exact match, the variables from the second file would be missing. To verify that this is the problem, run a PROC CONTENTS on each dataset to see what the lengths are for the variable FNAME. If the lengths are different, the problem can be corrected by using the trim or compress function to get rid of any trailing blanks around the first variable. The trim function will only remove trailing blanks, while the compress function will remove all blanks in the value. Here is code that will solve the problem:

```
proc sql;
  create table mergfile as
  select a.fname, a.lname, a.age,
         b.adrln1, b.citycd, b.state,
         b.zipcod
  from   demos a left join
         address b
  on trim(a.fname)||a.lname=trim(b.fname)||b.lname;
quit;
```

QUICK TIP

New in Version 8... To rename the label on the OBS column in PROC PRINT code the OBS= option.

Example:

```
PROC PRINT DATA=TEST OBS='Survey Number';
  Title 'Results of December Survey';
RUN;
```

Results of December Survey		
Survey Number	Question	Result
1	1	Yes
2	2	No
3	3	Unsure



"Someone in my company has permanent formats stored in a catalog. How do I get at these formats so I can use them in my own program?"



To access these permanent formats, first, set up a library reference pointing to the library where the catalog is located. If the desired formats are located in a directory 'J:\SAS\SHARED' your libref would look something like this:

```
libname fmtlib 'J:\SAS\SHARED';
```

Next, you need to set the FMTSEARCH option to let SAS know where to search for formats. The syntax for fmtsearch is:

```
FMTSEARCH=(libref1 libref2 ... );
```

SAS will always search your work library first for formats, unless otherwise specified in the FMTSEARCH option. To get SAS to first search the 'J:\SAS\SHARED' and then the WORK directory, the syntax is:

```
OPTIONS FMTSEARCH=(FMTLIB WORK);
```



USING MACROS TO GENERATE TITLES

CONTINUED FROM PAGE 4

An advantage of using the DATA step, in addition to being much simpler, is that the date can be reformatted with the PUT function to use a different style of date. Disadvantages might be that now the macro cannot be used within a DATA step, since the macro itself runs a DATA step. A second disadvantage is that since symput is used to generate macro variables this technique would probably make sense only when generating a small amount of code. Using the macro language would be superior when generating many lines of SAS code.

In summary, I think that there are many ways to solve this problem, and if at all possible, it may be advantageous to minimize the macro coding and instead use a more powerful language such as the DATA step.



CREATING SAS/GRAPH® OUTPUT

CONTINUED FROM PAGE 3

A Line Plot

This job produces a line plot with PROC GPLOT.

```
filename gsasfile 'c:\temp\examp3.gif';
goptions gaccess=gsasfile device=gif
        noborder xpixels=500 ypixels=400;
proc gplot data=mortdata;
  plot rate * year /vaxis=5 to 10 by 1
        haxis=1988 to 2000 by 2;
  symbol1 i=spline color=red v=star;
title c=black 'Home Mortgage Rates';
title2 c=black '(Annual Percentage Rates)';
  where region='East';
run;
```



Special SAS/GRAPH Drivers

Another special driver that is available is the HTML SAS/GRAPH driver. The HTML SAS/GRAPH driver will automatically produce a single file called INDEX.HTML that will contain multiple tags, each pointing to a separate GIF file. This works only for a single PROC invocation that produces several graphs. Examples of PROCs that could be used are any PROC with multiple BY groups or run groups, or PROC GREPLAY, which can display several previously built graphs.

In the example below, the BY statement produces a plot for each of the four regions with the INDEX.HTML file pointing to the four graphs. Note that in this program, a slightly different GOPTIONS statement is used to point to a directory instead of a single file along with the driver name of HTML.

```
filename grafout "c:\temp";
goptions reset=all device=html
        gsfname=grafout
        xpixels=500 ypixels=400;
proc gplot data=mortdata;
  plot rate * year /vaxis=5 to 10 by 1
        haxis=1988 to 2000 by 2;
  symbol1 i=spline color=red v=star;
title c=black 'Home Mortgage Rates';
```

QUICK TIP

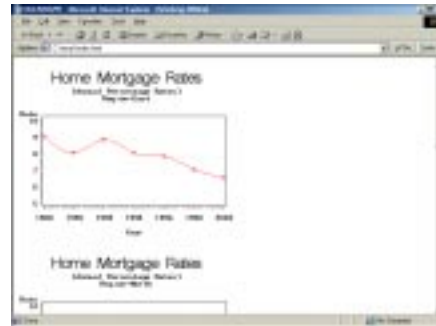
By default, all printed reports do not skip any lines at the top of the page. Change this with the SKIP=nn global option.

OPTIONS SKIP=5;
TITLE "This Title will start on line 5 of the page, not line 1";
PROC PRINT DATA=TEST; etc...



```
title2 c=black
        '(Annual Percentage Rates)';
  by region;
run;
```

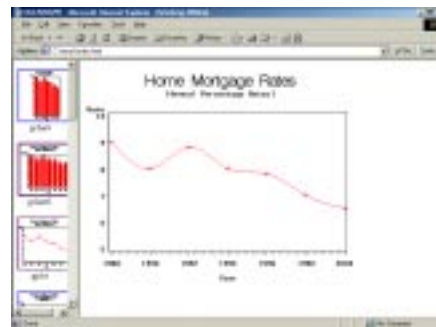
The multiple graphs can again be viewed via the scroll bars.



The WEBFRAME driver not only produces an index file along with multiple graphs, but it also creates a series of thumbnail graphs that display on the left of the display. Any of the thumbnails can be clicked to display the corresponding full size graphs. This is a very easy way to catalog a large number of graphics without using custom programming or menus.

The following job displays all of the previously built graphics and builds the INDEX.HTML file.

```
goptions reset=all device=webframe
        xpixels=500 ypixels=400
        display gsfname=grafout;
proc greplay igout=work.gseg nofs;
  replay _all_;
run;
quit;
```



Continued on page7

READING COMMA-DELIMITED DATA . . .

CONTINUED FROM PAGE 1

converts the character string to a numeric date equivalent.

The informat is named after a colon on the input statement, as in:

```
INPUT selldate : MMDDYY10.;
```

The DATA Step

The following DATA step will read the raw data:

```
FILENAME RAWIN 'c:\temp\a.txt';
DATA newfile;
  INFILE rawin FIRSTOBS=2 DSD;
  LENGTH name $20 city $15 State $2 selldate 4;
  INPUT name $ city $ state $ Sales
        selldate : MMDDYY10.;
  FORMAT selldate YMMDD8.;
RUN;
PROC CONTENTS DATA=newfile;
RUN;
PROC PRINT DATA=newfile;
RUN;
```

PROC CONTENTS Results

#	Variable	Type	Len	Pos	Format
5	Sales	Num	8	0	
3	State	Char	2	47	
2	city	Char	15	32	
1	name	Char	20	12	
4	selldate	Num	4	8	YMMDD8.

PROC PRINT Results

name	city	State	selldate	Sales
Jones, Tom	Phoenix	AZ	00-02-18	3456.85
Lake, Amy	New York	NY	99-02-13	.
Cook, Bob	Chicago	IL	00-02-06	245.10

The options and statements used above allow great flexibility in reading a variety of data formats. The power of SAS never ceases to amaze me.



CREATING SAS/GRAPH® OUTPUT

CONTINUED FROM PAGE 6

There are obviously many more SAS/GRAPH capabilities and HTML features that can be used to produce web displayed graphics. In a future article, I will discuss how SAS ODS can be used along with SAS graphics and printed output to display even more involved output.



QUICK TIP

Suppose you have a SAS dataset where some records will be chosen, in multiple programs, or changing the selection is the only change made to the program on a regular basis. For example, a customer number or part number from a list may change.

Try using a macro variable for the WHERE statement.

Example:
%LET WHERE=XVAR='AAA';

```
DATA PARTS;
  INPUT XVAR $;
CARDS;
AAA
BBB
CCC
DDD
;
RUN;
```

```
PROC PRINT DATA=PARTS;
  WHERE &WHERE;
  /* SAS substitutes WHERE XVAR='AAA' */
RUN;
```



TECHNICAL CREDIT AND RECOGNITION



Steve First
President

Author of this issue's:
Creating SAS/GRAPH®, Page 2 and Using Macros to Generate Titles, Page 4



David Beam
Vice President

Author of this issue's:
Reading Comma-Delimited Data Into SAS®, Page 1



Katie Minten Ronk
Trainer/Consultant

Author of this issue's:
SAS® Help Desk I/O, Page 5



Gerry Frey
Trainer/Consultant

Author of this issue's:
Quick Tips, Pages 3,5,6,7

Puzzler #4Teresa Schudrowitz
 PublisherCindy Kersten
 EditorsDavid Beam & Cindy Kersten



PUBLIC CLASS SCHEDULE

	St. Paul, MN		Madison, WI	
Introduction to SAS® Acquaint yourself with SAS language to analyze data and write reports.	January 23-25	\$750	February 6-8	\$750
	March 12-14	\$750	April 2-4	\$750
	April 9-11	\$750	June 18-20	\$750
	May 21-23	\$750		
	June 26-28	\$750		
SAS /FSP® Learn to develop data entry systems with this 'user friendly' SAS product.	January 26	\$350		
SAS® Report Writing Generate a wide variety of reports using advanced techniques.	February 5-6	\$550	April 5-6	\$550
Introduction to PROC Report Enhance your report writing skills with this powerful procedure.	May 24	\$350		
The SAS® SQL Procedure Master this data access, report writing and joining tool.	April 12	\$350		
Advanced SAS® Read complex data files, use SAS functions, process process arrays, and learn advanced joining methods.	February 7-9	\$750	February 20-22	\$750
	June 12-14	\$750		
SAS® Efficiencies Learn advanced methods and save system resources.	April 13	\$350	February 23	\$350
SAS® Macros Learn to automate and eliminate repetitive code.	March 15-16	\$550	June 21-22	\$550
SyncSort®	June 15	\$350		

For more information or to register for a class call **(608) 278-9964** or visit **www.sys-seminar.com**.



SYSTEMS SEMINAR CONSULTANTS, INC.

2997 Yarmouth Greenway Drive
 Madison, WI 53711

PRSRST STD
 U.S. POSTAGE
PAID
 MADISON, WI
 PERMIT #2783

Call or visit our web site for your
 free subscription to
 The Missing Semicolon™ .