



The Missing Semicolon™

TECHNICAL ASSISTANCE FOR THE SAS® PROFESSIONAL

Volume 1, Number 1

June, 1998

WHAT'S NEW IN SAS® VERSION 7.0

Attending the 23rd annual SUGI Conference in Nashville, Tennessee this year provided us with the opportunity to hear about the enhancements that will be part of the forthcoming Version 7.0 of the SAS System. As with any major SAS release, Version 7.0 will be packed with hundreds of enhancements, which we wouldn't be able to cover here. We did try to highlight a few of the major features in the following overview.

Naming Conventions

To be more consistent with database management terminology, the SAS System will now use the terms *table*, *row*, and *column* in place of *data set*, *observation*, and *variable*. The old names, of course, may still appear in some SAS documentation for a while.

Long Names

Among the long-awaited programming enhancements is the ability to have longer variable names, catalog names, and file names - all are increased from 8 to 32 character maximums. This enhancement, along with the ability to use special characters such as spaces, dashes, and others in naming, will make SAS more descriptive as well as provide almost complete transparency to database access, without the field mapping and translation that is currently required.

Longer Character Variables

Character variables have been increased from a maximum of 200 bytes to 32K bytes.

Versioning of SAS Data Sets

This allows you to have multiple data sets with the same name but with different version/generation numbers, very much like generation data groups in a MVS environment.

User-Specific Integrity Constraints

This enables you to have SAS automatically verify your data before an observation is added, updated, or deleted from your data set. There are five types of checking that can be done in this manner.

- **Not Null** - this indicates that missing values are not permitted for this variable.
- **Check** - this allows you to use any valid *Where* expression to determine the variable value.
- **Unique** - this indicates that the values in this column must be unique.
- **Primary Key** - this indicates that all data values must be unique and not null.
- **Foreign Key** - this links the record in one data set to a specific record (primary key) in another data set.

Concatenating SAS Libraries & Catalogs

You can now reference two or more SAS libraries via a single libref in all environments. Entries from two or more SAS catalogs can also be concatenated.

.....continued on page 7

TABLE OF CONTENTS

MVS Space Issues: SSC President Steve First explains a valuable mainframe concept.....Page 2

Reading Packed Dates: a technical tip by SAS consultant David Beam.....Page 4

Puzzler Dataset: refer to this dataset to solve SAS consultant Iain Robertson's puzzler.....Page 4

Questions & Answers: SAS instructor Kim Kolbe-Ritzow shares great SAS questions and answersPage 5

File Import/Export Wizard: SAS consultant David Beam explains a useful SAS technique...Page 7

PUZZLER

Data very often contain "gaps" that interfere with the layout of a report. For example, you may want all accounts in a report to have the same number of years, even when there were no data for a particular year. PROC Report/Tabulate cannot do this.

Write a program to fill in the missing years of the "Puzzler Dataset" on page 4 so that each account has the same number of years.

The final report should show each account's sales summarized at the year level for all possible years (94-97).

The code should be as maintenance-free as possible - don't just hardcode the missing years with zeros!

SSC coffee mugs will be sent to the 3 most imaginative solutions e-mailed to train@sys-seminar.com by June 31, 1998.



SYSTEMS SEMINAR CONSULTANTS

2997 Yarmouth Greenway Drive
Madison, WI 53711

(608) 278-9964 Fax (608) 278-0065
www.sys-seminar.com

THE MISSING SEMICOLON™

A NEW KIND OF NEWSLETTER

Welcome to a dynamic new partnership between Systems Seminar Consultants and you, our customer. This is the first edition of our newsletter series designed to share solutions to common SAS problems among our customers, and to publicize available SAS resources. We hope that you will find it useful.



SSC President Steve First

SSC has always been dedicated to making SAS easy to understand, easy to use, and easy to support. This newsletter is another way to help us do just that.

As you read this issue and ones to follow, we invite your input. We think that the best ideas and solutions occur whenever SAS users come together and share information at SAS users groups, SAS classes, and in print. Hopefully the Missing Semicolon™ will be another way to disseminate SAS knowledge.

If you have ideas, comments, or experiences that you would be willing to share with our audience, please fax, e-mail, or mail them to us and we will address them in future issues.

Sincerely,

Steve First,
President



SYSTEMS SEMINAR CONSULTANTS

2997 Yarmouth Greenway Drive, Madison, WI 53711
(608) 278-9964 1 Fax (608) 278-0065 1 www.sys-seminar.com

Copyright © 1998. Systems Seminar Consultants, Madison, WI.

All rights reserved. SAS® is a trademark of SAS Institute, Inc.

Printed in U.S.A.

MVS SPACE ISSUES

HEADACHES YOU CAN AVOID

The most common SAS help desk question that we receive at our site involves a user running out of disk space in the SAS WORK library. Usually some rather large files are being processed and the SAS job can only partially finish.

Some possible causes for this condition are:

1. The data is just too big for the WORK library.
2. Temporary SAS files are being kept needlessly.
3. There are more records and variables than necessary.
4. Multiple copies of a SAS file are kept while recreating or sorting.
5. Variable widths are longer than necessary.

Possible solutions are:

1. Make the WORK library bigger.
2. Use an alternate temporary or permanent disk library.
3. Use tape datasets instead of disk.
4. Clean up the program to avoid carrying any unneeded data.
5. Understand and minimize sorting space demands.
6. Use logical *views* of the data instead of physical data files.

The SAS WORK library is a required MVS sequential dataset that is normally allocated at the beginning of each SAS session. It has a finite amount of space that is set by invoking a JCL batch proc or interactive clist. Space can be allocated in cylinders, tracks, or blocks, and usually is specified as a primary space and a secondary space that the systems will repeat up to 15 times. The actual amount of space varies at each site, but the total available to your SAS session can be checked by viewing the appropriate proc or clist.

A partial batch SAS proc:

```
//SAS PROC ENTRY=SASXAL ,  
// WORK=' 20, 10'  
//WORK DD UNIT=SYSDA, SPACE=(CYL, (&WORK) ) ,
```

A partial SAS clist:

```
PROC 0  
WORK(' 20, 10')  
ALLOC F(&DDWORK) SP(&WORK) CYL UNIT(SYSDA)
```

In both cases above, the WORK library will be allocated as 20 cylinders plus up to 15 times 10 cylinders for a total of 170 cylinders. A cylinder on a 3390 device contains approximately 850,000 bytes.

Note: Space can also be allocated in tracks (TRK) which is approximately 56,664 bytes on a 3390 device, or it can be an actual byte count such as 6160, which is the actual number of bytes in each block.

Knowing the above sizes, we can multiply the allocation unit (CYL, TRK, or block) by the number required (20,10 in this example) allocated to come up with a total WORK library size in bytes if

MVS Space Issues Continued.....

necessary. In the previous example, the maximum size would be 170 * 850,000 or 144.5 million bytes.

To estimate how much work space we *need*, we can use a very simple calculation which allows 20% for overhead: $\text{space-in-bytes} = \text{obs-length} * \text{no-of-obs} * 1.2$.

Obs-length and no-of-obs can be estimated, or PROC CONTENTS can be used to display some of the dataset's information. The space required for all datasets needed must be added up at one time; this sum must fit in the WORK library.

One more problem has to do with replacing datasets in the WORK library. If an existing dataset is being rewritten, the *old* copy is kept until the *new* dataset is successfully written. The most common example of this is PROC SORT sorting a dataset in place.

Below is a typical job that creates a few datasets with the following characteristics:

A: contains 100,000 obs, 290 obs length, or 34.8 million bytes
 B: contains 200,000 obs, 187 obs length, or 44.8 million bytes
 C: contains 100,000 obs, 333 obs length, or 40.0 million bytes
 WORK space is (CYL,(20,10)) or 144.5 million bytes.

CODE	WORKSPACE
// EXEC SAS DATA A; RUN;	A 34.8 mil bytes Free 109.7 mil bytes
PROC SORT DATA=A; BY ID; RUN;	A 34.8 mil bytes A 34.8 mil bytes Free 74.9 mil bytes
DATA B; RUN;	A 34.8 mil bytes B 44.8 mil bytes Free 64.9 mil bytes
PROC SORT DATA=B; BY ID; RUN;	A 34.8 mil bytes B 44.8 mil bytes B 44.8 mil bytes Free 20.1 mil bytes
DATA C; MERGE A B; BY ID; RUN;	A 34.8 mil bytes B 44.8 mil bytes C 40.0 mil bytes Free 24.9 mil bytes
PROC SORT DATA=C; BY NAME; RUN;	A 34.8 mil bytes B 44.8 mil bytes C 40.0 mil bytes C Not Enough Space!

QUICK TIP

— To stop SAS from issuing page breaks between pages of a reports, set the FORMDLIM option equal to a quoted blank, i.e. `OPTIONS FORMDLIM=' '`;



In the preceding figure, the left side is the SAS code, while the right side represents the WORK library at various points in the job.

Note that the last step fails because there is no longer enough room to hold dataset A, dataset B, and two copies of dataset C while the sort program runs. To run this job without altering the SAS code, the WORK space would have to be at least 159.6 million bytes. Working our math backwards (since our proc is allocated in cylinders), requires us to divide 159.6 million by 850,000, giving a minimum WORK library of 188 cylinders. Rather than specifying the minimum, we may want to add a little extra for growth (without asking for too much) such as:

```
// EXEC SAS.WORK='200,10' (Batch)
or SAS WORK('200,10') (Interactive)
```

A better approach for this program is to do some housekeeping. After dataset C is built, there is no longer a need to keep datasets A and B. The following PROC DATASETS step can be inserted after C is created:

```
PROC DATASETS LIBRARY=WORK;
DELETE A B;
RUN;
```

After the above step runs, the PROC SORT step will run as long as there is room for two copies of C. Here C occupies 40 million bytes, so two copies fit easily in our WORK space of 144.5 million bytes.

Don't forget that if datasets are large we should always try to decrease dataset size by:

1. Using KEEP or DROP to discard unneeded variables.
2. Using LENGTH to specify the minimum variable widths.
3. Using DELETE, IF, or WHERE to include only necessary rows.

In future issues I plan to address permanent SAS dataset usage as well as logical data step views. There are some interesting things that can be done to minimize data passes as well as eliminating WORK space.

I hope that this helps explain some of the WORK space issues. Please feel free to contact us with any questions or comments.



TECHNICAL ADVICE

SAS u TIPS u SAS u TIPS

Reading packed dates from external files often requires a three-step process of first, unpacking the field, second, converting it to a character value with leading zeros, and third, converting it to a SAS date value. Because there is no packed date SAS format, there is no simple and direct way of reading these types of fields short of using the steps as outlined above.

An example of a file containing a packed date looks like:

```
RULE:  ----+----1----+  
  
1 CHAR  ...@  
  ZONE  0107  
  NUMR  019C  
2 CHAR  ..r@  
  ZONE  0197  
  NUMR  109C
```

A packed field is read like an accordion (up down, up down). The last bit within the byte (in our example, the 'C') stores the sign of the value (whether it is positive or negative). The advantage of packed fields is that you can store more information in less space; hence, the name *packed* fields. In this example, a user can store in 4 bytes a value up to 7 digits in length.

A packed date field using SAS is read like:

```
DATA DATES;  
  INFILE 'path-name';  
  INPUT DATE1 PD4.;          /*Note 1 */  
  DATE2=PUT (DATE1,Z6.);    /*Note 2 */  
  DATE3=INPUT (DATE2,DDMMYY6.); /*Note 3 */  
  DROP DATE1 DATE2;       /*Note 4 */  
  FORMAT DATE3 DATE9.;    /*Note 5 */  
RUN;  
  
PROC PRINT DATA=DATES NOOBS;  
RUN;
```

Note 1: This line of code unpacks the field and stores it as a numeric variable in the SAS data set. It now looks more recognizable (e.g.: 11097 and 110997). Since DATE1 is now a numeric variable in SAS, leading zeros are dropped, which will cause a problem with the first observation value because the leading zero is significant in reading its value correctly as a SAS date value later on.

Note 2: In this line the PUT function converts a numeric to character, which is important, because the format Zn., where n stands for the length of the newly created variable, will place leading zeros back into the value. The values now look something like '011097' and '110997'.

Note 3: This line of code uses the INPUT function to convert a character to a numeric. Use the appropriate SAS date informat which matches the style of the incoming value (e.g., if the value had been in month-day-year order then the MMDDYYn. informat would be used). DATE3 will now have the value stored as a SAS date value (representing the values as *the number of days that the date is from January 1, 1960*).

Note 4: This line of code drops these variables, since they are only needed for the converting process.

Note 5: This line of code formats the variable so you can recognize its value. Since DATE3 is a SAS date variable, formatting any SAS date format can be used to change its *appearance* on the report. Formatting does not physically change the way that the value is actually stored on the SAS data set.

Additional Note: The creation of DATE3 could have been done in one step versus using two nesting functions. They were broken into separate steps simply for illustration.

An example of resulting output looks like:

```
DATE3  
  
01OCT1997  
11SEP1997
```



PUZZLER DATASET

USE TO SOLVE PUZZLER ON PAGE ONE

Account	Year	Sales	Account	Year	Sales
0001	95	7890	0002	96	3210
0001	96	1234	0002	97	1298
0001	96	3567	0002	97	1234
0001	97	5678	0003	96	1234
0001	97	2345	0003	96	2345
0001	97	1234	0003	96	3456
0002	94	7652	0003	97	9876
0002	95	4321	0003	97	8765

QUICK TIP

— The colon can be used as a wildcard in a variable list. For example, BAL: refers to all variables beginning with BAL.



QUESTIONS & ANSWERS

BENEFIT FROM OTHERS

Q: "I have several variables that are related which sit contiguously next to one another in a flat file (see example of data below). Short of coding a very long and repetitive input statement, is there some kind of short cut that I can take?"

```
AK065 AZ101 AL092 AR078 ....WY054
```

A: Formatted lists are useful when reading data that have an identifiable and repetitive pattern associated. Coding the input statement this way can help reduce the chance of potentially miscoding a column of pointer reference and can reduce the amount of redundant typing.

```
DATA TEMPS;  
  INFILE 'C:\FLATFILE\TEMPDATA.DAT';  
  INPUT @1 (STATE1-STATE50) ($2.+4)  
        @3 (TEMP1-TEMP50) (3.+3)  
RUN;
```

The names of the variables to be created are coded in the first set of (required) parenthesis after the pointer above. Creating several related variables in SAS is done by providing a constant prefix followed by a numerical suffix. The informat to be used in reading the data and the number of increments their values are apart from one another are specified in the second set of the above (required) parenthesis.



Q: "I am having trouble reading a variable length file (see example of data below) using SAS, is there any "trick" to doing it?"

```
13456 29 Y N 134 Y CHANGED MEDS  
39892 63 Y Y 112 N FOLLOW UP IN 2 MOS  
38382 43 N N 221 N CHECK HDLS
```

A: A variable length file contains data that differs in length for some or all of the records. If the column or formatted input were used to read in the last variable on the file, we would have problems because the records physically end in different positions on each line. Without some special tricks and techniques, these files can be difficult to read.

There are several ways that a variable length file can be read. One of the easiest ways is to use the LRECL= option in addition to the PAD option on the INFILE statement. The purpose of the LRECL= option is to specify the length of the longest record being read in (this

QUICK TIP

— Data sets have labels too! The data set label is a good place to store critical information about the dataset. As long as the data exists, the label exists, e.g., DATA NEW(LABEL="NEW STUFF").

information can usually be found with the documentation on the file or through the use of some system utility like FILEAID). The PAD option will then pad out the remaining bytes on the file to the length specified on the LRECL= with blanks. These two options essentially turn an otherwise variable length file into a fixed length file which can be read like any other fixed length file.

```
DATA CONSUMER;  
  INFILE 'C:\FLATFILE\EAT7.DAT'  
  LRECL=45 PAD;  
  INPUT  
  @1      ID          $5.  
  @7      AGE         2.  
  @9      MED_DIET    $1.  
  @10     HEL_DIET    $1.  
  @12     WEIGHT      3.  
  @18     DIABETIC    $1.  
  @20     DESCRIB1    $24.'  
RUN;
```

Do you have a technical question that you would like us to answer? You can e-mail us, fax us, or send us your technical question by mail and we will try to respond to it in an upcoming Questions & Answers section of our newsletter.



SYSTEMS SEMINAR CONSULTANTS

Attn: Technical Questions & Answers
2997 Yarmouth Greenway Drive
Madison, WI 53711

Fax: (608) 278-0065

E-mail: train@sys-seminar.com

IMPORT/EXPORT WIZARD

LET YOUR SAS DATA TRAVEL

We frequently need to move data from SAS into popular PC-based software packages, such as Excel or dBASE.

If you have PC/SAS for Windows available, this movement of data is now easy to perform with the "File Import/Export Wizard". This feature includes several common output formats that are recognized by most PC software packages. With the addition of SAS/ACCESS for PC File Formats, additional interfaces are available.

This article shows a very simple SAS data set, and the resulting outputs created by using the Export Wizard. Learning to use the Wizard is a snap! The sample data set and export files took our tester, David Beam, less than 10 minutes to create and test.

"To test the Export feature, I created a SAS data set with five variables. Notice I used a 'FORMAT' statement for my date variable, this way the resulting output is created with the date values appropriate to the software selected."

Data Set: WEATHER					
#	Variable	Type	Len	Pos	Format
1	CITY	Char	20	0	
3	DATE	Num	8	22	DATE9.
5	HIGH	Num	8	38	
4	LOW	Num	8	30	
2	STATE	Char	2	20	

CITY	STATE	DATE	LOW	HIGH
Madison	WI	04AUG1997	54	101
Portland	OR	04AUG1997	62	83
Palm Spring	CA	27JUL1997	86	112
New York	NY	31JUL1997	73	86

"I then used the Export Wizard to create five different output files. Keep in mind that the dBASE and Excel formats require the additional SAS/ACCESS software for PC File Formats."

File Format	File Created
dBASE	weather.dbf
Excel Spreadsheet	weather.xls
Delimited File	weather.dat
Comma Separated	weather.csv
Tab Delimited	weather.txt

"The following shows the resulting output I got from the 'delimited' and the 'comma separated' files. The dBASE and Excel files can be opened directly in the appropriate software packages. A delimited or comma separated file however, requires some manipulation to import into another software package, but the file can easily be read into almost any PC software."

```
File Weather.dat
CITY STATE DATE LOW HIGH
Madison WI 04AUG1997 54 101
Portland OR 04AUG1997 62 83
"Palm Springs" CA 27JUL1997 86 112
"New York" NY 31JUL1997 73 86
```

"Notice that the 'delimiter' above is a space between fields. The 'comma separated values' (CSV), example below, is often a more flexible interface between SAS and other software packages."

```
File Weather.csv
CITY,STATE,DATE,LOW,HIGH
Madison,WI,04AUG1997,54,101
Portland,OR,04AUG1997,62,83
Palm Springs,CA,27JUL1997,86,112
New York,NY,31JUL1997,73,86
```

"Unfortunately the Export Wizard requires you to be running PC SAS under Windows. For those of you running SAS on the mainframe, Systems Seminar Consultants has developed a macro, *SSCFlat*, that enables users to move SAS mainframe data into a spreadsheet. Call (608) 278-9964 for a copy of this macro."

"What are my conclusions? The Export/Import Wizard offers a very easy interface between SAS and other software packages. Other interfaces that are available, such as the ODBC Drivers and Dynamic Data Exchange (DDE) may require more technical expertise to setup. However, if simple movement of data is required, the use of the new Import/Export Wizard is worth looking at."



QUICK TIP

— To save a backup copy of your log to a file, use the ALTLOG option.





What's New in SAS Version 7.0 Continued.....

Direct Access to Compressed Observations

Features have been added to allow direct access to compressed data set observations using the observation number in the Point and Firstobs options. Performance enhancements have been made to take greater advantage of single and composite indices.

The SAS Output Delivery System (ODS)

This feature includes the ability to automatically create output from any SAS procedure in HTML, rich-text, Postscript, or PCL formats as well as the ability to edit the output.

SAS/GRAPH Enhancements

New three dimensional pie and bar charts are available. Annotation text can "clip" the background around it.

SAS/AF Enhancements

SCL has been renamed as the SAS Component Language. The language itself is much more object oriented, and should be much easier to use for development.

Windows Enhancements

The SAS workspace has been redesigned to resemble the Windows95 desktop. The SAS Explorer, which is modeled on the Windows Explorer, is a central point for managing basic SAS software tasks. Also included are enhancements such as GUI features, a print preview utility, additional print options, command bar changes, Program Editor changes, and support for Microsoft's IntelliMouse™. System enhancements that have been made to take greater advantage of NT features include I/O optimization, monitoring tools, and more support for e-mail, Lotus notes, ODBC, and SAS System Viewer.

There is a new External File Interface (EFI) that is a point and click interface for reading and writing external data, as well as an Import/Export Wizard for transfer to PC File formats.

PROC SQL Enhancements

Even though long names will require less mapping of database table and column names, all former PROC SQL pass through facilities are still in place. In addition, PROC SQL will, whenever possible, pass joins to the DBMS instead of doing the joins itself. This was a major performance problem in the past when joining a small SAS dataset with a large DBMS table because the large table would be completely downloaded before joining could take place. PROC SQL can now query up to 32 views or tables and perform inner joins on up to 32 tables.

Miscellaneous PROC Enhancements

PROC FREQ, MEANS, TABULATE, and UNIVARIATE all are supporting more features and statistics.

Client Server and Web Enhancements

Version 7.0 allows for new client-server capabilities that allow for distributed processing. This includes the ability to remotely submit jobs

QUICK TIP

— Have you ever needed to input data following a specific word? Using *INPUT @ 'text' positions the input pointer directly after the word 'text'.*



from a local SAS session and return to the local session. Cross environment data access (CEDA) allows Version 7.0 SAS datasets that are created on any directory-based platform to be read by the SAS System running on any other directory-based platform.

Like every other software system, SAS is changing to allow access to the world wide web. Not only can reports be distributed through the web, but interactive browsers can interface with SAS applications to build queries with SAS in the background. This area, of course, is huge and will be covered in more detail in future issues.

Summary

In the seventh major release of the SAS System we continue to see some major shifts in emphasis along with hundreds of minor enhancements. We will do our best to keep you informed of changes as they occur. Other sources of information are SAS training classes, local, regional, and national SAS Users Group meetings, and the SAS website.



**COMING UP
OVERLOOKED SAS PRODUCTS**

Over the course of the next few issues, SAS Consultant Iain Robertson will discuss existing SAS products that can be used for reporting and also review new products that SAS is releasing. Specifically he will look at: SAS/EIS, Dynamic Data Exchange, SAS/IntrNet, and Enterprise Reporter.

**TECHNICAL CREDIT
AND RECOGNITION**

Version 7.0.....	Steve First & Rosalind Gusinow
A New Kind of Newsletter.....	Steve First
MVS Space Issues.....	Steve First
Quick Tips.....	Justin Zinter
Technical Tips.....	David Beam
Question & Answers.....	Kim Kolbe-Ritzow
Import/Export.....	David Beam
Beyond the Basics.....	Jodie Schmidt
Puzzler.....	Iain Robertson
Publisher.....	Jodie Schmidt
Editors.....	David Beam, Laurie Holman, & Cindy Bergman



SYSTEMS SEMINAR CONSULTANTS
2997 Yarmouth Greenway Drive
Madison, WI 53711

CLASS SCHEDULE

INTRODUCTION TO SAS®

June 2-4	\$625	Columbus, OH
June 3-5	\$625	St. Paul, MN
June 15-17	\$625	Madison, WI
Aug 12-14	\$625	St. Paul, MN
Aug 17-19	\$625	Madison, WI
Sept 21-23	\$625	St. Paul, MN
Oct 21-23	\$625	St. Paul, MN
Nov 18-20	\$625	St. Paul, MN

ADVANCED SAS®

June 18-19	\$425	Madison, WI
June 24-26	\$625	St. Paul, MN
Oct 28-30	\$625	St. Paul, MN
Dec 9-11	\$625	St. Paul, MN

SAS® REPORT WRITING

Aug 24-26	\$625	Madison, WI
Sept 24-25	\$475	St. Paul, MN

SAS® MACROS

June 24-25	\$475	Madison, WI
Oct 26-27	\$475	St. Paul, MN

THE SAS® SQL PROCEDURE

June 26	\$325	Madison, WI
---------	-------	-------------

To register call Cindy at (608) 278-9964, ext. 301
or visit our website at
www.sys-seminar.com.

BEYOND THE BASICS RECOMMENDED COURSES

Each year hundreds of students from a variety of backgrounds complete our *Introduction to SAS®* class. Although our *Advanced SAS®* is often thought of as the next step, recent course feedback shows students benefit most when one of the following classes is used as a follow-up to *Introduction to SAS®*:

- **SAS® Report Writing:** Make writing reports and turning your data into information easier. This 2 or 3 day class will provide you with an in-depth understanding of the reporting features of SAS.
- **PROC Report:** Take full advantage of the powerful report writing capabilities of PROC Report by attending this 1 day course. PROC Report is also offered on the third day of a 3-day "SAS Report Writing" class.
- **The SAS® SQL Procedure:** Enhance your current SAS applications by mastering PROC SQL syntax in just 1 day. Take full advantage of PROC SQL's joining and report features.

Private On-site Courses: Course feedback also proves that companies gain more knowledge per training dollar spent by holding a private on-site class. During on-site classes our instructors are able to address specific issues which directly affect your company and are able to teach supplemental materials that provide solutions to these problems.

Systems Seminar Consultants is your one source provider of SAS based support. Please keep us in mind for all your SAS training, consulting, and help desk needs. For more information call (608) 278-9964 or visit www.sys-seminar.com.

