



# The Missing Semicolon™

TECHNICAL ASSISTANCE FOR THE SAS® SOFTWARE PROFESSIONAL

Volume 2, Number 3

October, 1999

## ARE YOU CONNECTING WITH SAS/CONNECT®?

Most sites today have multiple platforms. If your company has SAS on two or more platforms, the SAS/CONNECT product can allow users to run code and access data from those different computers almost as easily as if the programs and data resided on a single machine.

In other words, SAS/CONNECT is a cooperative processing product that provides client/server services between a local SAS session and one or more remote SAS sessions. While this is most often thought of as an interactive process, it can also give excellent connections in certain batch environments.

Earlier documentation often referred to a client and a server, or a PC and a mainframe, with the server (mainframe) usually being more powerful and in control.

Lately, as connected machines and the connecting protocols are more equal, the terms *local* and *remote* are being used to describe the machines. The *local/remote* terminology is more appropriate as either machine may be in control and the data may flow in lots of directions.

### Using SAS/CONNECT

Requirements for using SAS/CONNECT include the following:

- SAS must reside on each platform.
- SAS/CONNECT software must be licensed on each machine.
- There must be a telecommunications connection between the two systems.
- A "script" file that describes the type of connection needs to be used.

To use the software, the user:

1. Signs onto one of the systems.
2. Starts a SAS session. (This is usually an interactive SAS session, but it could also be a batch job in some environments.)
3. Optionally assigns the filename RLINK to the script file describing the connection desired.
4. Signs on to the remote SAS system

5. providing required user ids and passwords.
6. Uses SAS/CONNECT services to submit code remotely and to transfer data.
7. Signs off the remote system when finished there.
8. Eventually signs off the local system.

### Script Files

A script file has some very exact instructions to get SAS/CONNECT functioning. A script has to do at least three things:

1. Invoke SAS on the remote machine.
2. Set up the appropriate communications options for the remote.
3. Determine when the remote SAS session is ready to communicate with the local session.

Scripts can also automate the sign on/sign off process, handle error conditions, and do much more.

The bad news about scripts is that they are very tedious to write and must be coded exactly for each connection. The good news is that there is ample documentation and help available for users to write scripts correctly. Also, once a script is correct, the user only needs to remember the name.

Using SAS/CONNECT with a manual connection might go like this:

1. Log on to the local system.
2. Log on to the remote.
3. Invoke SAS on the local system.
4. Assign RLINK to our script file. An example of linking to an MVS system is:

```
FILENAME RLINK 'PROD.MVS.SCR';
```

5. Specify the communications access method and remote name with an OPTIONS statement. An example is:

```
OPTIONS COMAMID=TCP REMOTE=REMOTE1;
```

6. Sign on to the remote system.

Continued on Page 3.....

## IN THIS ISSUE

**Are You Connecting with SAS/CONNECT®?:** Steve First explains the benefits of this powerful SAS product .....Page 1

**Multi-Line Per Observation Files:** Kim Kolbe-Ritzow demonstrates how to read these special files.....Page 2

**And ARRAY We Go!:** Andrea Decker demonstrates how to use ARRAYS effectively.....Page 4

**SAS® Help Desk I/O:** Robert Purvis and Chandy Karnum answer a question regarding how to round properly.....Page 5

**SAS® Talents Unveiled:** Russ Lutz explains Systems Seminar Consultants' specialty areas.....Page 7

**Technical Credit:**.....Page 7

**Public Class Schedule:**.....Page 8



SYSTEMS SEMINAR CONSULTANTS, INC.

2997 Yarmouth Greenway Drive  
Madison, WI 53711

Website: www.sys-seminar.com  
E-mail: train@sys-seminar.com  
Phone: (608) 278-9964  
Fax: (608) 278-0065



## NEW SAS® PRODUCTS ARE CHANGING THE WAY WE DO BUSINESS



The SAS world is changing. SAS Version 7 has been released and Version 8 is right on its heels. New products are constantly being offered. SAS Institute, SAS products, and companies using SAS have recently won numerous awards for excellence. SAS users everywhere are excited to get their hands on new products and produce the results they have never been able to produce before.

At Systems Seminar Consultants, Inc., we are committed to keeping pace with the evolving SAS software products. We have trained our staff on the newest technologies and are updating our training materials. New course material includes information on the newest versions of SAS and uses a PowerPoint® format, allowing us to illustrate complex concepts even better.

Our consulting projects are also affected by new SAS products:

- Clients who have recently purchased a new SAS product often ask us to help them take full advantage of their investment.
- Clients who already license SAS products may ask us to enhance their current system to maximize return.

Besides Base SAS®, many of our new projects utilize SAS/ACCESS®, SAS/AF®, SAS/CONNECT®, SAS/EIS®, SAS/FSP®, SAS/GRAPH®, SAS/IntrNet™, SAS/MDDDB®, SAS/SHARE®, SAS/STAT®, SAS/Warehouse Administrator™, and other web-based tools.

We are excited about the new results we have been able to achieve with SAS and look forward to future software offerings from the SAS Institute.

Sincerely,

Steve First  
President



## MULTI-LINE PER OBSERVATION FILES HOW TO READ THEM

A multi-line per observation file has a single observation's values spanning more than one line on the input file. The key to identifying these types of files is that each observation has the SAME NUMBER of lines and the lines come in the same order. These types of files usually have some kind of identifier to help determine which type of record is being read. An example of a multi-line per observation file:

```
0103181028211234567891212223
020318122322245522343234
03031821211112232112200001
0106711341121455636388902021
020671288387378791005807
03067138387877177277474773
```

### Steps to Read This File

One of the safest ways to read this type of file is to:

- On the first INPUT statement code the variables that are common for each input line. In our example, the record indicator (01,02,03) and the survey id number (0318, 0671, etc.) appear on every line of the flat file.
- After checking the indicator variable, which specifies the current record being read, begin conditionally coding INPUT statements based on the type of record read in.
- Be sure to RETAIN variables (hold the value from one record to the next).
- OUTPUT the record (write it out) on the last line within the observation.

This method requires more coding, but it assures the data will be read correctly if a line is missing in the data.

### Programming Example

This example assumes the '03' record exists:

```
DATA READIT;
  INFILE 'where-the-data-are-located';
  INPUT @1 REC_TYPE $CHAR2.
        @3 SURVEYNO $CHAR4. @;
  IF REC_TYPE='01' THEN
    DO;
      RETAIN CITY AGE SEX Q1A1-Q1A17;
      INPUT @7 CITY $2.
            @9 AGE 2.
            @11 SEX $1.
            @12 (Q1A1-Q1A17) (+0 1.) ;
    END;
  ELSE IF REC_TYPE='02' THEN
    DO;
      RETAIN Q2A1-Q2A18;
      INPUT@7 (Q2A1-Q2A18) (+0 1.);
    END;
  ELSE IF REC_TYPE='03' THEN
    DO;
      RETAIN Q3 Q4 Q5A1-Q5A18;
      DROP REC_TYPE;
      INPUT @7 Q3 $1.
            @8 Q4 $1.
            @9 (Q5A1-Q5A18) (+0 1.);
      OUTPUT READIT;
    END;
RUN;
```



**SYSTEMS SEMINAR CONSULTANTS, INC.**

Copyright © 1999 Systems Seminar Consultants, Inc. Madison, WI

All rights reserved. Printed in USA. The Missing Semicolon is a trademark of

Systems Seminar Consultants, Inc. SAS, Base SAS, SAS/ACCESS, SAS/AF, SAS/CONNECT,

SAS/EIS, SAS/FSP, SAS/GRAPH, SAS/IntrNet, SAS/MDDDB, SAS/SHARE,

SAS/STAT, and SAS/Warehouse Administrator are registered trademarks

or trademarks of SAS Institute Inc. in the USA and other countries.

PowerPoint is a registered trademark of Microsoft in the USA and other countries.

There are at least three other ways to read data in multi-line per observation files.

### Alternative One

The first alternative involves coding separate INPUT statements for each line of data. An example of this is:

```

DATA READIT;
  INFILE 'where-the-data-are-located';
  INPUT @1 REC_TYPE $CHAR2.
        @3 SURVEYNO $CHAR4.
        @7 CITY $2.
        @9 AGE 2.
        @11 SEX $1.
        @12 (Q1A1-Q1A17) (+0 1.);
  INPUT @7 (Q2A1-Q2A18) (+0 1.);
  INPUT @7 Q3 $1.
        @8 Q4 $1.
        @9 (Q5A1-Q5A18) (+0 1.);
RUN;

```

### Alternative Two

A second alternative involves coding one INPUT statement, a forward slash (/) to read the next lines, and one semicolon at the end:

```

DATA READIT;
  INFILE 'where-the-data-are-located';
  INPUT @1 REC_TYPE $CHAR2.
        @3 SURVEYNO $CHAR4.
        @7 CITY $2.
        @9 AGE 2.
        @11 SEX $1.
        @12 (Q1A1-Q1A17) (+0 1.)
        /
        @7 (Q2A1-Q2A18) (+0 1.)
        /
        @7 Q3 $1.
        @8 Q4 $1.
        @9 (Q5A1-Q5A18) (+0 1.);
RUN;

```

### Alternative Three

The third alternative involves coding one INPUT statement, one semicolon, and line pointers (#). An example of this is:

```

DATA READIT;
  INFILE 'where-the-data-are-located';
  INPUT #1 @1 REC_TYPE $CHAR2.
        @3 SURVEYNO $CHAR4.
        @7 CITY $2.
        @9 AGE 2.
        @11 SEX $1.
        @12 (Q1A1-Q1A17) (+0 1.);
  INPUT #2 @7 (Q2A1-Q2A18) (+0 1.);
  INPUT #3 @7 Q3 $1.
        @8 Q4 $1.
        @9 (Q5A1-Q5A18) (+0 1.);
RUN;

```

Each of these methods assumes that every line will be represented in the data. If a type 2 or 3 record was missing for a given observation, the three methods outlined above would incorrectly read the data. The previous method of using conditional INPUT statements, RETAINS, and an OUTPUT statement could be modified to allow for the possibility of missing rows of incoming data. For this reason, using conditional INPUT statements, RETAINS, and an OUTPUT statement is the preferred method for reading multi-line per observation files.



## QUICK TIP

When reading raw files use the FIRSTOBS=record-number to begin reading the input data at the record number specified. This is also helpful when you don't want to read a header record, usually stored in the first record of the input file.

Example: INFILE RAWFILE FIRSTOBS=2;



## ARE YOU CONNECTING...?

CONTINUED FROM PAGE 1

At this point, the two SAS sessions start communicating with each other and a message is written to the log indicating that the connection is complete (unless some error occurs). The user may have to provide userid and password information if prompted.

One last thing to note is that there are many shortcuts to do the above. Some options are:

- use SAS menus
- embed commands as statements in a program
- use the SAS autoexec facilities

Again, extensive documentation is available for all of the above and once a procedure is correct, it doesn't change much. Once a connection is made between two systems, the user can either make another connection to another remote or start moving programs and data around.

### SAS/CONNECT Services

There are three major services available:

**Compute Services (remote submit):** allows users to submit code on a local machine, run some or all of that code on a remote machine, and then transfer the results back to the local machine.

**Data Transfer Services:** allows users to upload and download SAS data files, text files, and binary files.

**Remote Library Services:** allows data that resides on a remote machine appear as if it resides on the local machine.

This article will discuss Compute Services. The next two issues of this newsletter will contain information on the other two services.

Continued on Page 6.....

# AND ARRAY WE GO!

## AN INTRODUCTION TO ARRAYS

When someone says “arrays” to you, do you hear “a raise” and immediately think about that new car and long-postponed trip to Tahiti? Hold onto those thoughts. Arrays aren’t quite as much fun, but they can save you some bulky coding.

SAS arrays are used to group and refer to variables in an easier way. Arrays can be used for many purposes: reading data, performing repetitive calculations, performing table lookups, creating several related variables, and rotating datasets.

Arrays do not take up any extra space. Essentially, they are just another name for existing variables.

### Defining Arrays

Arrays are defined in the DATA step and must be defined in each DATA step, as they disappear once the step ends. The ARRAY statement is a compiler statement, which means the variable references are defined at compile time.

Defining an array is very simple. The basic syntax for the statement is:

```
ARRAY name {n} $ length elements
      (initial values);
```

*Name* gives a name to the array. It can be any valid SAS name. The dimension, *n*, which is the number of elements of the array, is stated within brackets. The dollar sign is necessary only if the elements in the array are character variables. The *length* specifies the maximum length for each of the elements in the array. This parameter is not required for numerics, as the default value is 8 bytes. The *initial values* for the elements can also be set if desired.

### Using Arrays

The following DATA step illustrates the potential need for array processing. The data set reads in four employees for each division. It then calculates the sales profit amount by subtracting the expenses from the sales for each employee.

```
DATA BADEX;
  INPUT DEPT $ EMP1 $ SALE1 EXP1
        EMP2 $ SALE2 EXP2
```

```
        EMP3 $ SALE3 EXP3
        EMP4 $ SALE4 EXP4;
  PROFIT1=SALE1-EXP1;
  PROFIT2=SALE2-EXP2;
  PROFIT3=SALE3-EXP3;
  PROFIT4=SALE4-EXP4;
  DATALINES;
H AD 20 5 RP 30 8 BF 25 15 CK 10 3
S JS 45 8 RT 50 25 EG 15 10 KM 35 15
;
RUN;
```

Without an array defined, the input statement must be defined to read each variable and the calculations must be done on an individual basis. If there were more than four employees in each division, the DATA step could conceivably be very long, and would contain very similar calculations.

If three arrays are defined, the calculations can be done very easily within a DO loop. The previous data set is read in and new variables are created using arrays.

```
DATA GOODEX;
  SET BADEX;
  ARRAY PROFIT{4} PROFIT1-PROFIT4;
  ARRAY SALES{4} SALE1-SALE4;
  ARRAY EXPENSES{4} EXP1-EXP4;
  DO J=1 TO 4;
    PROFIT{J}=SALES{J}-EXPENSES{J};
  END;
  DROP J;
RUN;
```

### Program Data Vector

The program data vector is included to illustrate the three arrays created in the data step.

	EMP1	EMP2	EMP3	EMP4
Sales	{1}	{2}	{3}	{4}
	SALE1	SALE2	SALE3	SALE4
Expenses	{1}	{2}	{3}	{4}
	EXP1	EXP2	EXP3	EXP4
Profit	{1}	{2}	{3}	{4}
	PROFIT1	PROFIT2	PROFIT3	PROFIT4

### Advantages of Arrays

As you can see, the code is much shorter when arrays are used. This also prevents typing errors that may occur when code is repeated over several lines.

Now that you have added SAS array skills to your little black bag, talk to the boss. Maybe you can get array to Tahiti this winter...

## QUICK TIP

To read a range of records from a raw file you can use the FIRSTOBS option combined with OBS=record-number, where record-number specifies the last record that you want to read from an input file.

Example: INFILE RAWFILE FIRSTOBS=10 OBS=30, results in 21 records being read, 10 through 30.



# SAS® HELP DESK I/O

SOLUTIONS FROM OUR HELP DESK SERVICE

**Q:** "I am building a custom format for displaying numeric values using a PICTURE statement. My results are not rounding properly. What is wrong?"

**A:** The PICTURE statement allows you to build custom formats. Rounding is implicit in the formats that SAS provides but must be specified when using the PICTURE statement. You must also make sure that the template is large enough to accommodate all the values displayed for that variable. If the specification is too short, values will truncate at the higher end because numbers get right justified when displayed.

## ROUND Option

There is a ROUND option that can be used on a PICTURE statement that works just like the ROUND function. The ROUND option rounds to the nearest digit and then formats the value according to the picture template. Note that the SAS system rounds the values before formatting. The ROUND option affects the entire statement, not just ranges included in the definition.

## Example

To see the results of a PICTURE statement with and without the ROUND option, let's format some account balance data. For our example, we want the data to be displayed with a '+' or '-' as a suffix, leading zeroes filled with '\*', and '\$' as a prefix. First we will create a PICTURE format with no rounding specified. A DATA step proceeds to create a table with three rows and columns for the example.

```
DATA TEST1;
  INFILE CARDS;
  INPUT PURCHASE PAYMENT;
  BALANCE1 = PURCHASE - PAYMENT;
  BALANCE2 = PURCHASE - PAYMENT;
DATALINES;
9563.98      3773.61
8563.67      13120.93
11254.55     1886.67
;
```

```
PROC FORMAT;
  PICTURE NOROUND LOW-<0 = '0,000,000-'
                  (FILL = '*' PREFIX = '$')
                  0-HIGH = '0,000,000+'
                  (FILL = '*' PREFIX = '$') ;
```

```
PROC PRINT;
TITLE '*** Output without Rounding ***';
FORMAT PURCHASE 9.2 PAYMENT DOLLAR9.2 BALANCE1
NOROUND. BALANCE2 DOLLAR9.2;
SUM PURCHASE PAYMENT BALANCE1 BALANCE2;
RUN;
```

## The Resulting Output - No Rounding

```
*** Output without Rounding ***
```

OBS	PURCHASE	PAYMENT	BALANCE1	BALANCE2
1	9563.98	\$3,773.61	***\$5,790+	\$5,790.37
2	8563.67	\$13120.93	***\$4,557-	\$-4557.26
3	11254.55	\$1,886.67	***\$9,367+	\$9,367.88
=====				
	29382.20	\$18781.21	**\$10,600+	\$10600.99

## QUICK TIP

— If you are running out of space (disk space) when creating new data sets try the data set OPTION COMPRESS=YES; This reduces the data file storage size by using a compression technique.

run;

The decimal portion of the BALANCE1 values was truncated, not rounded. One result is the total value is \$1 less than it should be.

Now we will create a PICTURE format with the ROUND option.

```
PROC FORMAT;
  PICTURE YESROUND (ROUND)
  LOW-<0 = '0,000,000-'
  (FILL = '*' PREFIX = '$')
  0-HIGH = '0,000,000+'
  (FILL = '*' PREFIX = '$') ;

PROC PRINT;
TITLE '*** Output with Rounding ***';
FORMAT PURCHASE 9.2 PAYMENT DOLLAR9.2 BALANCE1
YESROUND. BALANCE2 DOLLAR9.2;
SUM PURCHASE PAYMENT BALANCE1 BALANCE2;
RUN;
```

We run a PROC PRINT as before but format BALANCE1 with the YESROUND format.

## The Resulting Output - Rounding

```
*** Output with Rounding ***
```

OBS	PURCHASE	PAYMENT	BALANCE1	BALANCE2
1	9563.98	\$3,773.61	***\$5,790+	\$5,790.37
2	8563.67	\$13120.93	***\$4,557-	\$-4557.26
3	11254.55	\$1,886.67	***\$9,368+	\$9,367.88
=====				
	29382.20	\$18781.21	**\$10,601+	\$10600.99

The values for BALANCE1 are now properly rounded before display, and the sum is now correctly rounded to \$10,601.

PICTURE formats have many powerful uses but with power should come caution. Test your formats carefully to make sure that the results are correct.

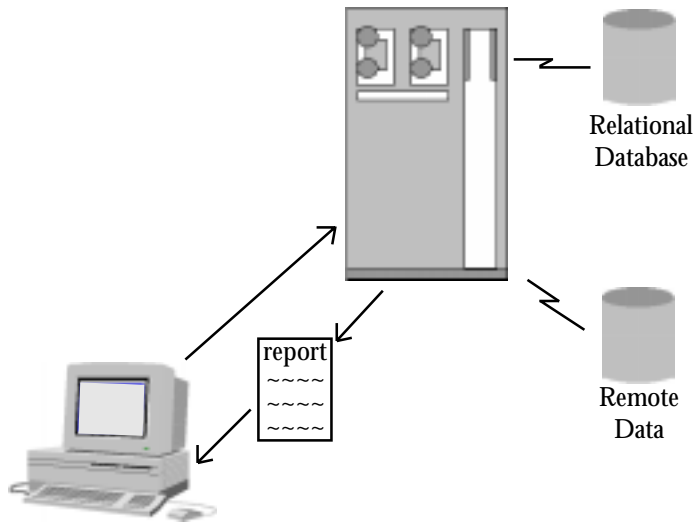
run;

# ARE YOU CONNECTING...?

CONTINUED FROM PAGE 3

## A Diagram of Compute Services

Compute Services allows the user to use a local SAS session to send a program off to a remote. The remote processes the code and returns the results to the local machine.



When a user types SUBMIT, the job stays on the local machine and does all of its processing there. When the command RSUBMIT is used instead, the job is sent to the remote. The job is executed on the remote, and the results are returned to the local machine. This of course can be done many ways: SAS menus may be used to SUBMIT/RSUBMIT, a command may be typed, or the RSUBMIT statement may be imbedded in a program to remotely submit part of a program.

However it is done, the Compute Services technique provides the tremendous resources of remotes to the local machine, while still allowing the user a familiar local interface and access to local printers and other devices.

Here is an example of a program submitted from a Windows local machine to a remote MVS system, using RSUBMIT. Note that the

naming of libraries and files needs to reflect the remote (MVS) type names.

```
libname mvslib 'prod.mysas.data';
proc contents dta=mvslib._all_;
run;
```

Another option is to use the RSUBMIT and ENDRSUBMIT *statements* intermixed with the desired code. The entire program is submitted locally, using SUBMIT. SAS/CONNECT will RSUBMIT all lines between RSUBMIT and ENDRSUBMIT. This allows the user to always submit the same way (locally) but gives the flexibility to run some code locally and some remotely, within one submission.

### Example

```
rsubmit;                               /*run on remote */
  libname mvslib 'prod.mysas.data';
  proc contents dta=mvslib._all_;
  run;
endrsubmit;
proc print data=work.data2;run; /*run on local */
```

Even though part of the example code runs locally and the rest runs remotely, it is submitted once and all output comes back to the local.

One disadvantage of Version 6 SAS/CONNECT is that while the remote code is running, the local session is tied up. Asynchronous processing is supported in Version 7; control is returned immediately to the local while the remote code is still running.

### SAS/GRAPH Support

Remote graphics are also supported by SAS/CONNECT and SAS/GRAPH. Many mainframes are difficult to use for displaying and printing graphics. With SAS/CONNECT, graphs can be built on the remote but displayed on the local machine via the special GRLINK device.

### Example

```
GOPTIONS DEVICE=WIN; /*local graphics driver */
RSUBMIT;
  GOPTIONS DEVICE=GRLINK; /*remote driver */
  PROC GSLIDE;TITLE 'X';
RUN;
ENDRSUBMIT;
```

### Access to Remote Relational Databases

Often the remote contains data in a relational database which can be accessed from a remote session as well. This requires a combination of SAS/CONNECT and the appropriate SAS/ACCESS software for your database.

SAS/ACCESS is another tool that can be used in many ways. Below is an example of a job that uses the PROC SQL Pass-Through facility to select certain rows and columns from a SYBASE table located on a remote UNIX system. Again, the program is submitted

Continued on Page 8.....

## QUICK TIP

When using PROC SQL, if you need to know the number of the observation that you are reading in, use the option NUMBER. It will print a column on the report labeled 'ROW'.

```
Example: PROC SQL NUMBER;
         SELECT * FROM INDATA;
         QUIT;
```



# SAS® TALENTS UNVEILED

## MULTIPLE WAYS SSC CAN HELP YOU

Some of you who know us by our quality consulting may be surprised to learn that we train over 1,000 students a year to use SAS products or that we answer questions for beginners and experts alike through our help desk services.

One of our fundamental principles of doing business is that we must know what we teach, teach what we know, and always strive to learn more. As consultants, we are expert communicators, and our teaching experience assists in the knowledge transfer to our customers. As instructors, we teach others how to use the SAS tools we use everyday in our consulting practice. In fact, many customers will request a particular consultant who has done project work for them, to also provide training, thereby gaining the extra benefit of having an instructor who knows their company's systems and data.

Our services are organized into the following areas:

### SAS Training Services

SSC is dedicated to making SAS software easier for you to understand and use. We train over 1,000 students each year on a variety of courses from our basic Introduction to SAS® course to more sophisticated courses like SAS/AF® and SAS® Macros. Our instructors are highly rated and are the same experts who provide consulting and help desk services to you.

We provide both public and private on-site training options depending upon your needs. Our rates are very competitive and can save you money. We can even customize our classes to meet your individual business or data requirements.

### SAS Consulting Services

We are well-versed in a variety of business areas. We provide consulting support in data warehousing, production reporting systems, application development, and market research and analysis. Our recent customers include well-known companies from the financial services, insurance, manufacturing, marketing, and retail sectors, as well as state and federal agencies.

We guarantee your satisfaction and will not close the book on a project until you are happy.

### SAS Help Desk Services

We are available to help solve your everyday SAS problems. Users of this service simply call us during business hours. One of our SAS experts will always be available to answer your questions or help you with a program. We will own the problem until you get the help you need.

You can learn more about these services by calling (608) 278-9964 or by visiting our website [www.sys-seminar.com](http://www.sys-seminar.com).

## QUICK TIP

To include text on a report and avoid modifying each PROC, create a macro variable at the top of the program and refer to it in the PROC. Simply modify the macro the next time you run the same job.

Example: %LET MONTH=August 1999;  
TITLE1 "Sales Summary as of &MONTH";



## TECHNICAL CREDIT AND RECOGNITION



**Steve First**  
President  
Author of this issue's:  
Are You Connecting with SAS/CONNECT?, Page 1



**Kim Kolbe-Ritzow**  
Trainer/Consultant  
Author of this issue's:  
Multi-Line Per Observation Files, Page 2



**Andrea Decker**  
Trainer/Consultant  
Author of this issue's:  
And ARRAY We Go!, Page 4



**Robert Purvis**  
Trainer/Consultant  
Co-author of this issue's:  
SAS® Help Desk I/O, Page 5



**Chandy Karnum**  
Trainer/Consultant  
Co-author of this issue's:  
SAS® Help Desk I/O, Page 5



**Russ Lutz**  
Vice President, Operations  
Author of this issue's:  
SAS® Talents Unveiled, Page 7



**Antonio Correa**  
Trainer/Consultant  
Author of this issue's:  
Quick Tips, Pages 3-7

Publisher .....Jodie Schmidt  
Editors .....David Beam, Russ Lutz, & Cindy Kersten



# ARE YOU CONNECTING...?

CONTINUED FROM PAGE 6

locally, it executes remotely, and the results are printed locally.

```
DM 'signon';
rsubmit;
PROC SQL OUTOBS=10;
CONNECT TO SYBASE(USER=XXXXXX          /*TO SYBASE      */
                  PASSWORD=yyyyy
                  DATABASE=MAST_CLAIMS
                  SERVER=UNIX01_);
CREATE TABLE WORK.BIO AS /*SAS TABLE OUT */
SELECT MEMBER,          /*OUTPUT COLUMNS */
       GROUP,
       MEMSEQ,
       DATEPART(BIRTH) AS BIRTH
       FORMAT=MMDDYY10. ,
       SEX
FROM CONNECTION TO SYBASE /*FROM SYBASE IN  */
(SELECT *                /*ALL COLUMNS    */
 FROM MAST_BIO           /*INPUT SYBASE TABLE */
 WHERE BIRTH BETWEEN
 '1996/01/01' AND '1996/01/31') A ;
QUIT;
PROC PRINT DATA=WORK.BIO; /*PRINT THEM      */
TITLE 'WORK.BIO';RUN;
endrsubmit;
```

The data remains on the remote but the report is sent to the local device.

For more information about SAS/CONNECT, feel free to call us or consult the SAS/CONNECT documentation which is excellent. In the next issue, we will address the data transfer (upload/download) capabilities of SAS/CONNECT.



## PUBLIC CLASS SCHEDULE

### Introduction to SAS®

October 25-27	\$675	Madison, WI
November 9-11	\$675	St. Paul, MN
December 6-8	\$675	Madison, WI
December 6-8	\$675	St. Paul, MN

### Advanced SAS®

October 18-20	\$675	St. Paul, MN
December 13-15	\$675	St. Paul, MN
December 9-10	\$500	Madison, WI

### Introduction to Proc Report

October 29	\$350	Madison, WI
------------	-------	-------------

### SAS® Macros

October 21-22	\$500	St. Paul, MN
---------------	-------	--------------

### The SAS® SQL Procedure

October 28	\$350	Madison, WI
November 12	\$350	St. Paul, MN

We also provide a variety of private on-site SAS classes. Call (608) 278-9964, ext. 311 for details. Course outlines for the SAS classes we offer are available on our website.

To register call (608) 278-9964 or visit [www.sys-seminar.com](http://www.sys-seminar.com).



**SYSTEMS SEMINAR CONSULTANTS, INC.**

2997 Yarmouth Greenway Drive  
Madison, WI 53711

BULK RATE  
U.S. POSTAGE  
**PAID**  
MADISON, WI  
PERMIT #2783

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