



The Missing Semicolon™

TECHNICAL ASSISTANCE FOR THE SAS® SOFTWARE PROFESSIONAL

Volume 2, Number 2

June, 1999

CREATING HTML-READY DOCUMENTS WITH SAS®

The promise of computers to eliminate paperwork has eluded computer users for years. Printed reports may contain useful summaries and appropriate detail, but burrowing through a "doorstop" report can be time-consuming. A report that can be opened with a common software tool may be just what the user needs. HTML formatted reports viewed with a browser, such as Internet Explorer or Netscape, can improve functionality while saving paper.

SAS HTML Formatting Tools

The SAS Institute has been active in helping SAS users generate HTML output. There are several macros (%OUT2HTML, %TAB2HTML, and %DS2HTML) available on the SAS Institute website for download, along with documentation on their use. These macros enable SAS to generate HTML formatted reports from procedures and tables, with considerable capability to customize appearance. Version 7 contains the SAS Output Delivery System (ODS), a feature that includes the ability to automatically create output from any SAS

procedure in HTML, RTF, PS, or PCL formats, and the ability to edit the output. Note: Version 8 will not require SAS ODS and will simplify the steps described in this article.

Dynamic Content - SAS/IntrNet

It is also possible to write SAS programs that can be run through a web browser to create dynamic web content. SAS/IntrNet tools allow the user to enter information into the browser. This information is then passed to a SAS program running on a different platform. The output is returned to the user's browser for viewing, filing, or printing.

Back to the Grindstone

There is often a need for less sophisticated approaches. We do not all have the time and expertise to use the SAS/IntrNet tools to create dynamic web content. The HTML macros are great for generating quick results but at a price – size. Each cell in a table may contain lots of tags for format control. File size is very important if users are downloading pages via phone lines. If size is a concern, we can rely on one of the most basic tools in the SAS system for lean, mean HTML output – the DATA step.

Big Secret Revealed

The web browser interprets tag pairs in a document for display purposes. A few basic examples:

```
<HTML> Interpret until the end tag pair </HTML>
<TITLE> Title bar of browser </TITLE>
<HEAD> Report heading </HEAD>
<BODY> Body of the report</BODY>
<TABLE> Table some data </TABLE>
<B> Bold text </B>
<PRE> Preformatted text </PRE>
```

As you might guess from the artful appearance of many web pages, there are many more tags available to enhance and control the appearance of your product. Images can also be included. While surfing the internet, keep an eye out for HTML formats you like. Use your browser's 'view source' or 'view document source' option to expose this code.

Continued on page 6.....

IN THIS ISSUE

Creating HTML-Ready Documents with SAS®: Robert Purvis shares a technique to create value-added reports from SAS.....Page 1

The SQL Pass-Through Facility: Steve First discusses differences between the Pass-Through and view descriptor.....Page 2

Puzzler #2 Solution: David Beam solves last issue's puzzler and announces the winners of \$100 training certificates.....Page 4

SAS® Help Desk I/O: David Beam shares additional uses of INDEX keys.....Page 5

Decoding the _TYPE_ Variable: Katie Minten explains a useful tip for PROC SUMMARY.....Page 6

SAS® Support Services: get the most out of the SAS software product you are licensing.....Page 8

PRIORITY SAS-Y2K SUPPORT

We have a solution to your SAS-related Year 2000 problems.

Systems Seminar Consultants, Inc. is launching a new SAS-related service: *On-demand, Priority SAS-Y2K Support.*

This service provides immediate support for production and user SAS programs and allows you to avoid business complications associated with the new millennium.

Call Russ Lutz at (608) 278-9964, ext. 305 for details.



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



EXCHANGING SAS® TIPS

SUGI 24



Our annual trip to SUGI proved interesting and informative. It was also great to be on the beach in April! Presenters of papers, hands-on workshops, and posters provided attendees with numerous ways to learn more about the SAS system.

We enjoyed presenting our workshops and papers: SAS Macro Variables and Simple Macro Programs and An Introduction to PROC SQL. We

received a lot of positive feedback and appreciated your attendance at our sessions. If you were unable to attend but would like a copy of the presented material please let us know.

One thing we notice at conferences, whether they are SUGI or local meetings, is the excitement over sharing and learning those little tips that can make a big difference in your efficiency. We highly recommend attending conferences to interact with other SAS users and learn about these tips.


One of the goals of this newsletter is to provide you with some of those same "little tips" that can make your job easier. We are always looking for new and more efficient ways to do things. If you have a SAS tip you would like to share, please e-mail or fax it to us under the subject "New Tip". If we print it, we'll send you a SSC coffee mug.

We always invite your comments and suggestions. One suggestion we recently received was for more in-depth articles. In this issue we tried to follow your suggestion.

I hope you enjoy this issue of *The Missing Semicolon*™. If there is anything we can do for you or your company, please contact us.

Sincerely,

Steven First,
President

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, SAS/IntrNet, and SAS/ACCESS are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries.

THE SQL PASS-THROUGH FACILITY

VERSUS VIEW DESCRIPTORS

When SAS Version 6 was introduced, a new method of accessing relational databases was included in the SAS/ACCESS product.

Even though relational tables are similar in shape to SAS datasets, accessing them in SAS was complicated by two things:

1. SAS names were limited to 8 characters (many databases allow up to 40 characters in their table and column names).
2. SAS names did not allow special characters such as dashes and spaces (special characters are often used in database names).

The basic design was to come up with a logical view of the data that would make the database table appear to be a SAS dataset and, at the same time, map database names to legal SAS names. This structure is called a *view descriptor*.

Once set up, a view is very simple to use and transparent to the SAS user. Views appear like most other SAS datasets and in most cases can be used like a SAS dataset.

Below is a batch job that builds and uses a SAS view to access data from a DB2 table.

```
proc access dbms=sdb2;ssid=abcd;
  create sasuser.sales.access;
  table= user42.sales;
  assign=yes;
  format date mmdyy8. amount 8.2;
  create sasuser.sales.view;
  select all;
run;

proc print data=sasuser.sales;
  title 'Sales dbms table';
run;
```

Sales dbms table				
OBS	CUSTID	SALESID	DATE	AMOUNT
1	10003	900386	01/03/89	2579.00
2	10003	900386	04/03/89	8337.00
.

This illustration shows how simple view descriptors can be to use.

The performance of the structures, though, was unacceptable for some very large tables. As a result, SAS Institute developed a way for SAS jobs to "pass through" SQL code to the database system without altering it.

The 'Pass-Through' facility was implemented as part of PROC SQL in release 6.07. Although the performance problem was solved, making Pass-Through part of PROC SQL caused some minor confusion:

1. PROC SQL has numerous capabilities and many users are not aware of the Pass-Through facility.

2. The documentation for Pass-Through is not included in the SAS/ACCESS manuals. Instead, it is found technical reports starting with *SAS Technical Report P-222 Changes to and Enhancements to Base SAS Software. Release 6.07.*

In any case, the most common syntax is:

```
proc sql;
connect to dbms-name;
create table table-name as
select-statement from connection to dbms-name
(dbms-query);
disconnect from dbms-name;
quit;
```

This facility allows users to write their own SQL code. Everything between the connect statement and the disconnect is “passed through” to the DBMS without SAS altering it. A useful exception to this is that SAS does honor the SAS macro language here, so macro code will be expanded. Several examples of PROC SQL programs follow.

Example 1

Create and print a SAS work file from a DB2 table:

```
proc sql;
connect to db2(ssid=abcd);
create table members as /*memb is out sas ds */
select memgrp, /*vars from in table */
L_name,
F_name,
M_init,
Dob_ymd format=yymmdd10.,
ssn
from connection to db2
(select /*vars from in table */
mem_memgrp as memgrp, /*rename long names */
mem_l_name as l_name, /*to short sas names */
mem_f_name as f_name,
mem_m_initial as m_init,
mem_dob_ymd as dob_ymd,
mem_ssn as ssn
from qdspr.ppcmemmstr /*input table */
where mem_dob_ymd between '1970-01-01' and
'1996-12-31'
);
%put &sqlxrc &sqlxmsg; /*error codes & msgs */
disconnect from db2;
quit;
proc print data=members;
title 'print of members'; run;
```

Because there is a mixture of SAS names and DB2 names, it can be confusing to determine which system you are dealing with. This tip might help: generally items inside the parentheses refer to the database system and items outside the parentheses refer to SAS.

Example 2

Create and print a SAS work view from a DB2 table:

```
proc sql;
connect to db2(ssid=abcd);
create view members as
/*memb is out sasview*/
select memgrp, /*vars from in table */
L_name,
F_name,
M_init,
Dob_ymd format=yymmdd10.,
```

QUICK TIP

Use the PROC PRINTTO procedure to output a proc print to a file. This is a great way to store a report to a flat file for input to a later step.

Example: FILENAME DUMP 'C:\FILE.TXT';
PROC PRINTTO PRINT=DUMP NEW;
PROC PRINT;
FORMAT VAR1 MMDYY8.8;
RUN;
PROC PRINTTO; /*TURN OFF */



```
ssn
.
.
quit;
proc print data=members;
title 'print of members';
run;
```

The only difference between Example 1 and Example 2 is the use of the "create TABLE members..." statement versus the "create VIEW members..." statement.

A VIEW does not create a SAS data file until it is used (i.e., in a PROC PRINT), and, as such, can save intermediate work space.

Example 3

Join two DB2 tables:

```
proc sql;
connect to db2(ssid=abcd);
create table members as select *
/*select all vars */
From connection to db2
(select /*vars in */
a.mem_memgrp as memgrp,
a.mem_l_name as l_name,
a.mem_f_name as f_name,
a.mem_dob_ymd as dob_ymd,
a.mem_ssn as ssn,
b.sub_addr1 as addr1,
b.sub_hm_phone_nbr as hphone
from qdspr.ppcmemmstr a, qdspr.ppcsubscriber b
where a.mem_memgrp=b.sub_memgrp and mem_dob_ymd
between '1970-01-01' and '1996-12-31'
);
%put &sqlxrc &sqlxmsg; /*codes and msgs */
disconnect from db2;
quit;
proc print data=members;run;
```

Example 4

Join a DB2 Table with matching obs in a SAS data file:

```
data subset;
infile ddin;
input acct_num $1-19;
run;
```

Continued on page 7.....

PUZZLER #2 SOLUTION

MERGE OPERATIONS

We received an overwhelming response to February's puzzler and would like to thank everyone who participated. We received our first solution February 3rd and continued to receive faxes and e-mails about the puzzler until the end of April.

To refresh your memory, here is the problem. A recent SAS student stated that she never performed calculations directly after merging SAS files. She instead would merge the data in one step, SET the dataset in a second step, and then do any necessary calculations. The student's datasets, program, and output are below.

Datasets Involved and Student's Program and Output

ONE			TWO		
OBS	ACCT	RATE	OBS	ACCT	BALANCE
1	A	1	1	A	10
			2	A	20
			3	A	30

```
DATA BOTH1;
MERGE ONE(IN=ONONE) TWO (IN=ONTWO);
BY ACCT;
IF ONONE AND ONTWO;
```

```
DATA BOTH1;
SET BOTH1;
IF BALANCE=20 THEN RATE=2;
RUN;
```

```
PROC PRINT DATA=BOTH1;
TITLE 'BOTH1 (CORRECT RATES)';
RUN;
```

BOTH1 (CORRECT RATES)			
OBS	ACCT	RATE	BALANCE
1	A	1	10
2	A	2	20
3	A	1	30

The Instructor's Program and Output

```
DATA BOTH2;
MERGE ONE(IN=ONONE) TWO(IN=ONTWO);
BY ACCT;
IF ONONE AND ONTWO;
IF BALANCE=20 THEN RATE=2;
RUN;
```

```
PROC PRINT DATA=BOTH2;
TITLE 'BOTH2 (INCORRECT RATES)';
RUN;
```

BOTH2 (INCORRECT RATES)			
OBS	ACCT	RATE	BALANCE
1	A	1	10
2	A	2	20
3	A	2	30

Explanation

The problem is how SAS retains values coming from SAS data files during a MERGE operation. In our example, the values are not re-initialized until the value of the BY variable changes. This results in the value of RATE coming in as 1 for the first row, being changed to 2 for the second row, and this value being retained for the rest of the rows with the same ACCT value. The solution requires using a new variable to hold RATE in the output data set. Note the use of the RENAME clause here to eliminate the extra column:

Solution

```
DATA BOTH1 (KEEP= ACCT BALANCE NEWRATE
RENAME=(NEWRATE=RATE));
MERGE ONE(IN=ONONE)
TWO(IN=ONTWO);
BY ACCT;
IF ONONE AND ONTWO;
IF BALANCE=20 THEN NEWRATE=2;
ELSE NEWRATE=RATE;
RUN;

PROC PRINT DATA=BOTH3;
TITLE 'BOTH3 (SOLUTION)';
RUN;
```

BOTH3 (SOLUTION)			
OBS	ACCT	RATE	BALANCE
1	A	1	10
2	A	2	20
3	A	1	30

Winners

The first two readers to send in the correct solution were:

- Mark Holmberg of BlueCross BlueShield of Minnesota
- Lucy Mackey Bilaver of The University of Chicago

These two winners have each received a \$100 training certificate, good towards any one of SSC's SAS training programs.

Secondary prizes have been sent to 10 readers who submitted solutions that also worked. These note-worthy runner-ups include:

- Patricia Quinnett of The Associates
- Deborah Gleason of The Musicland Group
- Russell Burns of Fleet Technology Solutions
- David Walker of Centers for Disease Control and Prevention
- Brad Barnhill of Lexis Law Publishing
- Shuxuan Zhao of 3M Pharmaceuticals
- Dave Hapeman of IBM
- Paul Baard of CUNA Mutual Insurance Group
- Robert Mengis of Bear Creek Corporation
- Snow Fu of Rhone-Poulenc Rorer.

Congratulations to all our winners. We hope everyone enjoyed this puzzler. Look for our next puzzler in the *The Missing Semicolon's* October issue.

QUICK TIP

Use the descending option within the VBAR or HBAR statement to produce bar charts in descending order of magnitude. Example: the following code results in a bar graph with descending values for the variable GRPPCT by grouping:

```
...HBAR GROUPING/DESCENDING SUMVAR=GRPPCT...
```



SAS® HELP DESK I/O

SOLUTIONS FROM OUR HELP DESK SERVICE

Q: "I would like to do two things. I first want to read a large flat (non-SAS) file for customer numbers and compare it with a second, smaller list of customers. I then want to produce a report showing which customers from the second, smaller list were NOT found in the larger file. If possible, I would like to avoid having to create a SAS file from the first, very large set of customer numbers. How can I do this?"

A: In the last issue, we discussed using INDEX keys with the SET statement to merge files, by directly retrieving observations from an indexed SAS data file. This help desk question can be solved by another creative use of INDEX keys.

For our example, the following represents a large flat file of customer information, which contains a customer identification number:

Large RAWDATA Flat File

CUSTID	CUSTID
11111	77777
22222	88888
33333	99999
44444	AAAAA
55555	BBBBB
66666	CCCCC

There is also a second, smaller list of customers, and you need to know which of these 'look-up' customer identification numbers are NOT found in the large flat file:

Smaller NUMBERS List of Customers

CUSTID
22222
12345
98765

Step 1

Create a SAS data set of the customer numbers you want to report if NOT found in the first file. This data set needs to be created with an INDEX on the CUSTID variable, which can be done with PROC SORT:

```
PROC SORT DATA=NUMBERS
          OUT=NUMBERS ( INDEX= ( CUSTID= ( CUSTID ) ) );
BY CUSTID;
RUN;
```

Step 2

Now use the indexed 'look-up' data set created in Step 1 while reading the large flat file. If the flat file customer identification number is NOT in the 'look-up' data set, do not output the customer identification number to the SAS data set:

QUICK TIP

Utilize PROC GREPLAY to overlay plots with different symbols or shading to visually highlight differences.. The following is an example where OLAY is a full screen template with one panel:

```
PROC GREPLAY NOFS IGOUT=MYGRAPHS
TC=MYTEMPL TEMPLATE=OLAY;
TREPLAY 1: GPLOT 1: GPLOT 1;
RUN;
```

```
OPTIONS ERRORS=0;
DATA MYDATA;
INFILE RAWDATA;
INPUT @1 CUSTID $CHAR5.;

/* SEE IF CUSTOMER NUMBER IS IN THE DESIRED LIST
   SET NUMBERS KEY=CUSTID / UNIQUE;

/* IF NOT FOUND, SAS RETURNS NON-ZERO IN _IORC_ AND THEN YOU CAN READ THE NEXT RECORD.
   IF _IORC_ NE 0 THEN
   DO;
   DELETE; /* GET NEXT CUSTOMER RECORD
   END;

* ONLY OUTPUT IF FOUND;
OUTPUT MYDATA;
RUN;
```

Step 3

Sort the 'found' customers, removing duplicates.

```
PROC SORT DATA=MYDATA OUT=INCOMING NODUPKEY;
BY CUSTID;
RUN;
```

Step 4

Merge 'found' customers with the smaller selection list. Only keep customer identification numbers if NOT on the first 'found' file.

```
DATA NOTFOUND (KEEP=CUSTID);
MERGE INCOMING (IN=ONA)
      NUMBERS;
BY CUSTID;
* OUTPUT IF NOT ON THE "FOUND" DATA SET;
IF NOT ONA;
RUN;

* NOW PRINT LIST OF DESIRED CUSTID NUMBERS THAT WERE NOT FOUND;
TITLE 'SELECTED CUSTID NUMBERS NOT FOUND ON RAWDATA FILE';
PROC PRINT DATA=NOTFOUND;
RUN;
```

Results

OBS	CUSTID
1	12345
2	98765

DECODING THE _TYPE_ VARIABLE IN PROC SUMMARY AND PROC MEANS

The summary procedure is a powerful tool which produces summary statistics for all possible combinations of class variables. Each unique combination is denoted with a unique _type_ value. It is often confusing to figure out which _type_ value goes along with a particular combination of class variables without looking at the data. The following steps outline how to decode the _type_ variable, so you can select the statistics you need.

Example SAS Program

```
PROC SUMMARY DATA=TEST;
  VAR SALES;
  CLASS DEPT STORE CITY REGION;
  OUTPUT OUT=SALESDAT
         SUM(SALES)=TOTSALES;
RUN;
```

Step 1

Write your class variables on a piece of paper.

```
DEPT  STORE  CITY  REGION
```

Step 2

Start with the right-most variable, and number that variable 1.

```
DEPT  STORE  CITY  REGION
                        1
```

Step 3

Work from right to left, numbering each variable as the previous number times two. This is a base 10 representation of a binary number.

```
DEPT  STORE  CITY  REGION
  8      4      2      1
```

Step 4

To find what _type_ value represents a combination of variables, add the numbers of the variables together. Note that _type_ 0 is always the grand total.

```
DEPT (8) + STORE (4) = 12
DEPT (8) + STORE (4) + CITY (2) = 14
DEPT (8) + STORE (4) + CITY (2) + REGION (1) = 15
STORE (4) + REGION (1) = 5
```

Step 5

Once you know the type, you can select the statistics you need. For example, to print the Dept and Store totals, use:

```
PROC PRINT DATA=SALESDAT;
  WHERE _TYPE_=12;
RUN;
```

A Final Example

Your client comes to you and says, "Can you roll up the numbers for departments within city and also at the region level?"

```
DEPT  STORE  CITY  REGION
  8      4      2      1
```

To get statistics for DEPT within CITY we need _type_ = 8 + 2 = 10.
To get statistics for REGION, _type_ = 1.

The Final Report

```
PROC PRINT DATA=SALESDAT;
  WHERE _TYPE_ IN (1,10);
RUN;
```



CREATING HTML DOCUMENTS CONTINUED FROM PAGE 1

DATA Step Example

A SAS table named YTDSALES has columns for product name, total sales, and commission. We need a report on the performance of the five leading products. Sort the table rows by descending total sales, then present the top five names.

```
PROC SORT DATA=YTDSALES;
  BY DESCENDING TOTSALES;
RUN;

DATA NULL ;
  SET YTDSALES (OBS=5) END=EOF;
  FILE 'C:\TEMP\YTDSALES.HTM' NOTITLE;

  IF _N_=1 THEN PUT '<HTML>' /*header block */
                  /*browser title */
                  '<TITLE>Acme Sales Leaders</TITLE>'
                  '<h2>Top 5 Sales Leaders in the Acme Widget
                   Company</h2>' /*size: 1 largest */
                  '<h4>Fiscal Year 1999</h4>'
                  '<BODY>' /
                  '<PRE>' / /*preformatted block */
                  @1 'Product name' /*column header */
                  @25 'Total Sales'
                  @45 'Commission';

  PUT @1 NAME /*data block */
      @25 TOTSALES comma11.
      @45 COMMISSN comma10.2;

  IF EOF THEN PUT /*footer block */
                '</PRE>' / /*end preform block */
                'No distribution rights allowed.'
                'Acme Widget Company' /
                '</BODY>' /
                '</HTML>' /;
RUN;
```

QUICK TIP

Utilize the following 3 lines of code within an annotate dataset to custom initialize all your graphs:

```
FUNCTION='LABEL'; TEXT='YOUR COMP/DEPT'; SIZE=2;
X=91; Y=5; OUTPUT;
FUNCTION='LABEL'; TEXT='MM/YY'; SIZE=2; X=95;
Y=5; OUTPUT;
FUNCTION='LABEL'; TEXT='YOUR INITIALS'; SIZE=2;
X=93; Y=3; OUTPUT;
```



Output Viewed with Browser

Product name	Total Sales	Commission
Topgama.jsp	8,000	2,280.00
Widgetwizwit	4,870	1,342.50
Widgetwizwan	4,460	1,235.00
Widgetwizbit	4,000	1,090.00
Widgetwizbitbit	1,478	419.50

No distribution rights allowed. Acme Widget Company

Simple Power

This simple example will enable you to pack lots of information into a relatively small file. Files can contain imbedded links, enabling you to construct indexes that allow users to navigate to reports with a click of a mouse button. Design strategy can enable users to search quickly through your site to glean the information that they need and print only what they need.

Examples

There are excellent examples of programs on SAS Institute's website, www.sas.com, as well as information on more advanced tools such as style sheets. There are also HTML and web page design resources on the internet. A tutorial by Jeff Barta at <http://junior.apk.net/~jbarta> will get you started. Follow links from this site to other sources of reference materials.



THE SQL PASS-THROUGH FACILITY

CONTINUED FROM PAGE 3

```
proc sql;
  connect to db2(ssid=abcd);
  create table work.joins as
  select acct_num,
         eom_dt,
         logo_cde,
         ins_pre2,
         cycle_da
  from connection to db2
  (select *
   from abcdefg.acct_ins_pr_eom) a,
  (select acct_num as acct_nu2
   From work.subset) b
  where a.acct_num = b.acct_nu2;
  %put &sqlxrc &sqlxmsg; /* errors and msgs */
  disconnect from db2;
quit;
proc print data=joins;run;
```

Even though the above does join the two files, it has to, in effect, download the entire DB2 table to SAS before it can do the matching. The above program will be much more efficient if you can use a WHERE clause as part of the DB2 query.

QUICK TIP

When utilizing by-group processing for graphs you can assign a null label to suppress the variable name and equal sign in the subtitle. For example, if you were graphing by location, the line "LABEL LOCATION='00'X;" would suppress the "LOCATION=" portion of the subtitle.



Advantages of View Descriptors

1. Once created, views are transparent to the user.
2. Views can be used with SAS/FSP to update DBMS data.
3. Views can be stored temporarily or permanently, as needed.
4. WHERE, ORDER BY, and other statements can be included transparently with the view.

Advantages of Pass-Through

1. DBMS can optimize queries especially when using large tables, summary functions, group-by functions, and computed functions.
2. The user can have more complete control over the query without SAS getting in the way.

Version 7 and Beyond

SAS Version 7, by adding long name and special character support, greatly simplifies this process. The whole concept of a view will be unnecessary because there is no mapping required. Instead, a simple option on the libname statement will let SAS/ACCESS know which type of table is being accessed transparently. Performance will also improve when doing joins, as SAS will perform joins in the most efficient location. SAS Institute has indicated that PROC SQL queries will have continued supported in Version 7 and later versions.

In summary, using the SQL Pass-Through Facility can make it easier to access and effectively use relational databases. For more complete information please refer to the following:

- *SAS Technical Report P-222 Changes and Enhancements to Base SAS Software. Release 6.07*
- *SAS Technical Report P-242 SAS Software: Changes and Enhancements Release 6.08.*



TECHNICAL CREDIT

AND RECOGNITION

Creating HTML-Ready Documents with SAS®.....Robert Purvis
The SQL Pass-Through Facility.....Steve First
Quick Tips.....Robert Tylo
Puzzler #2 Solution.....David Beam
SAS® Help Desk I/O.....David Beam
Decoding the _TYPE_ Variable.....Katie Minten
Publisher.....Jodie Schmidt
Editors.....David Beam, Russ Lutz, & Cindy Kersten

SYSTEMS SEMINAR CONSULTANTS

SAS® SUPPORT SERVICES

At Systems Seminar Consultants, Inc. we focus on providing practical solutions to your information requirements and strive to make SAS software easier to understand, use, and support.

Dynamic Partnerships

By providing quality SAS training, consulting, and help desk services, we foster dynamic partnerships with you, our client, to help achieve your highest goals.

SAS Training Services

We train over 1,000 students each year from private firms, corporations, and government agencies. We offer public classes (see schedule to right) and private on-site courses (see website for details). Our courses are designed to give students more knowledge per training dollar spent. Our prices are quite competitive. Call Cindy Kersten at (608) 278-9964, ext. 306 for a quote on your next SAS training project.

SAS Consulting Services

Our staff of SAS consultants is well-versed in a variety of business areas. Our specialty areas include: Data Systems Development, Decision Support and Business Consultation, and Market Research and Analysis. Need help on an upcoming project? Call Russ Lutz at (608) 278-9964, ext. 305 to discuss project details and learn how we can help.

SAS Help Desk Services

Our team of experts is available to solve your company's daily SAS problems. You'll gain timely assistance with SAS-related problems and save on internal support, time, and money. Call David Beam at (608) 278-9964 ext. 304 to develop a customized support plan.

Guaranteed Satisfaction

Our client loyalty and our reputation for excellence are a result of our commitment to you. We will not close the book on a project until you are completely satisfied with our results.



PUBLIC CLASS SCHEDULE

Introduction to SAS®

June 21-23	\$675	Madison, WI
June 28-30	\$675	St. Paul, MN
August 16-18	\$675	Madison, WI
August 23-25	\$675	St. Paul, MN
September 27-29	\$675	St. Paul, MN
October 11-13	\$675	St. Paul, MN
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

SAS® Report Writing

Sept 30 - Oct 1	\$500	St. Paul, MN
-----------------	-------	--------------

Advanced SAS®

June 16-18	\$675	St. Paul, MN
June 24-25	\$500	Madison, WI
October 18-20	\$675	St. Paul, MN
December 13-15	\$675	St. Paul, MN
December 9-10	\$675	Madison, WI

Introduction to Proc Report

August 26	\$350	St. Paul, MN
October 29	\$350	Madison, WI

SAS® Macros

August 19-20	\$500	Madison, WI
October 21-22	\$500	St. Paul, MN

The SAS® SQL Procedure

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

To register call (608) 278-9964 or visit 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™ .