



The Missing Semicolon™

TECHNICAL ASSISTANCE FOR THE SAS® SOFTWARE PROFESSIONAL

Volume 8, Number 1

Winter 2006

Customizing ODS Excel Output

ODS makes it much easier to take the reports we create in SAS and move them to Excel. With just a few lines of code, the SAS report is ready to open in Excel.

```
ODS HTML FILE =
'C:\TEMP\SOFTSALE.XLS';
```

```
PROC PRINT DATA = SOFTSALE;
  TITLE 'SOFTSALE DATA';
RUN;
TITLE;
```

```
ODS HTML CLOSE;
```

While this code is relatively straightforward, there are some limitations on the Excel output. For many reports, landscape orientation would be preferable, but by default, Excel opens the report in portrait orientation regardless of the layout of the report.

PROC TEMPLATE step before we output the report to the ODS destination. This article will focus on building options into the MSOffice2K tagset, in particular, sending output to Excel or Adobe Acrobat in landscape page orientation. In addition to setting the page orientation, other useful options are: adding a sheet name, setting the zoom, freezing the header, setting the paper size for the printer, and adding titles to each printed page.

A tagset is part of the ODS markup language that must be formatted by a third-party. The SAS Institute describes tagsets as a way to “enable the user to apply styles to output objects that are used by applications other than SAS”. SAS has created a variety of tagsets, but you can also create your own if the existing tagsets don't contain all the options you desire, as is the case with the example in this article.

Now let's take a look at creating this tagset. The first step is to code PROC TEMPLATE and give your tagset a name. In this case, the tagset is called 'test'. We want to build onto an existing tagset, 'MSOffice2K', so we code parent = tagsets.MSOffice2K. Now we have inherited all the options already built into the MSOffice2K tagset as our base and we can build other options into the 'test' tagset.

```
PROC TEMPLATE;
  DEFINE TAGSET TAGSETS.TEST;
  PARENT=TAGSETS.MSOFFICE2K;
  MVAR SHEET_NAME;
```

For each part of the report, we define the event. An event defines what is written to the output file and can be called throughout the program. An event has a unique name and can include start and/or finish sections. Start and finish define portions of output to be effected by the event. For example: DEFINE EVENT DOC; directs the output to Excel, and DEFINE EVENT DOC_HEAD; defines options for the page orientation.

Continued on page 2.....

The dotted line down the middle of the report represents the print layout in portrait orientation. The report will come out on two pages and probably be difficult to understand. In order to get this report to print out on one page, the user will have to go into the print settings in Excel and manually change the page orientation to landscape.

Getting a report to print in landscape page orientation without having to set up the print settings in Excel or Adobe Acrobat is fairly simple. We can define some of our own settings within a



SYSTEMS SEMINAR CONSULTANTS, INC.

2997 Yarmouth Greenway Drive
Madison, WI 53711

www.sys-seminar.com
train@sys-seminar.com
1-800-997-7081

IN THIS ISSUE

Customizing ODS Excel Output.....

Sarah Chronquist.....Page 1

President's Letter.....

Steve First.....Page 2

Report Prep 101.....

Timothy Broeckert....Page 4

Help, My Format's Not Working.....

Rosalind Gusinow.....Page 7

Quick Tip 1.....

Katie Ronk.....Page 7

Quick Tip 2.....

Katie Ronk.....Page 8

Technical Credit and Recognition.....Page 8

SAS Training Solutions.....

.....Page 8



Letter From the President



Dear SAS User:

As a company of successful SAS consultants for over 20 years, we have come up with many secret tips, tricks, and techniques that we use to wow our clients. We have our own unique options and programming techniques to improve our code. We have discovered our own cool uses of

common SAS procedures, functions, and statements.

But they're not secrets anymore! Now programmers of all levels can pick up our tips in our new class *Tips, Tricks, and SAS Techniques*. This one day class is an eclectic mix of SAS tips. It has short examples from a wide variety of topics presented in lecture format, along with short exercises. So, you'll get to practice these time-saving techniques.

Topics include:

- ◆ Naming conventions for files, programs, and variables
- ◆ Techniques to validate input data
- ◆ Cleaning data values and finding unique keys
- ◆ Techniques to work with periodic report processing
- ◆ Percentage problems
- ◆ Methods to compare and audit data sets
- ◆ Useful functions
- ◆ Macro and format tips
- ◆ Verify the existence of files and SAS data sets
- ◆ Data step tips
- ◆ Useful tricks to debug code
- ◆ IBM Mainframe utilities
- ◆ SAS as a programmer's utility

Every student will leave with some clever tips. We hope you'll save CPU time, programming time, and lots of headaches!

Sincerely,

Steven First,
President



Customizing ODS Excel Output

(CONTINUED FROM PAGE 1)

```
DEFINE EVENT DOC;  
START:  
  PUT '<HTML XMLNS:X="URN:SCHEMAS-MICROSOFT-  
COM:OFFICE:EXCEL">' NL;  
FINISH:  
  PUT "</HTML>" NL;  
END;
```

We begin to set up the orientation options and we identify the option as landscape. The options coded throughout this program will be called later in the ODS step.

```
DEFINE EVENT DOC_HEAD;  
START:  
  PUT "<HEAD>" NL;  
  DO /IF  
  CMP($OPTIONS["ORIENTATION"], "LANDSCAPE");  
  PUT "<STYLE> @PAGE {MSO-PAGE-  
ORIENTATION:LANDSCAPE} </STYLE>" NL;  
  DONE;  
  PUT VALUE NL;  
FINISH:
```

```
PUT "<!-- [IF GTE MSO 9]><XML>" NL;  
PUT "<X:EXCELWORKBOOK>" NL;  
PUT "<X:EXCELWORKSHEETS>" NL;  
PUT "<X:EXCELWORKSHEET>" NL;
```

A sheet name option can be added with the following statements:

```
DO / IF $OPTIONS["SHEET_NAME"];  
SET $$SHEET_NAME $OPTIONS["SHEET_NAME"];  
PUT "<X:NAME>$$SHEET_NAME "</X:NAME>" NL;  
ELSE;  
PUT "<X:NAME>SHEET1</X:NAME>" NL;  
DONE;
```

Open Positions

**We are currently filling these positions,
all requiring the use of SAS software:**

Chicago, IL	Financial Analysts (all levels)
Madison, WI	Quality Improvement Analyst
Madison, WI	SAS Consultant / Trainer
Madison, WI	Cognos Developer
Minneapolis, MN	Senior Financial Analyst
Minneapolis, MN	SAS Contractors

Please apply online at: <http://www.sys-seminar.com/onlineapp.php>



SYSTEMS SEMINAR CONSULTANTS, INC.

Copyright © 2006 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 following statements will allow the user to set a zoom option.

```
PUT "<X:WORKSHEETOPTIONS>" NL;
DO / IF $OPTIONS["ZOOM"];
SET $ZOOM $OPTIONS["ZOOM"];
PUT "<X:ZOOM>" $ZOOM "</X:ZOOM>" NL;
```

Add a frozen header as an option:

```
DO / IF
CMP($OPTIONS["FREEZE_HEADERS"], "YES");
PUT " <X:FREEZEPANES/>" NL;
PUT " <X:FROZENOSPLIT/>" NL;
PUT " <X:SPLITHORIZONTAL>1 </
X:SPLITHORIZONTAL>"
NL;
PUT " <X:TOPROWBOTTOMPANE>1 </
X:TOPROWBOTTOMPANE>"
NL;
PUT " <X:SPLITVERTICAL>1 </X:SPLITVERTICAL>"
NL;
PUT " <X:LEFTCOLUMNRIGHTPANE>1
</X:LEFTCOLUMNRIGHTPANE>" NL;
DONE;
```

A scaling option can be coded to print the output to scale and the paper size can be set, as well.

```
PUT "<X:PRINT>" NL;
DO / IF $OPTIONS["SCALE"];
SET $SCALE $OPTIONS["SCALE"];
PUT "<X:SCALE>" $SCALE "</X:SCALE>" NL;
PUT " <X:VALIDPRINTERINFO/>" NL;

DO / IF
CMP($OPTIONS["PAPERSIZE"], "LEGAL");
PUT " <X:PAPERSIZEINDEX>5 </
X:PAPERSIZEINDEX>" NL;
DONE;
PUT "</X:PRINT>" NL;
PUT "</X:WORKSHEETOPTIONS>" NL;
PUT "</X:EXCELWORKSHEET>" NL;
PUT "</X:EXCELWORKSHEETS>" NL;
PUT "</X:EXCELWORKBOOK>" NL;
```

This code sets the option to add print titles to each of the pages of the report:

```
DO / IF CMP(
$OPTIONS["PRINT_TITLE"], "YES");
PUT "<X:EXCELNAME>" NL;
PUT "<X:NAME>PRINT_TITLES</X:NAME>" NL;
PUT "<X:SHEETINDEX>1</X:SHEETINDEX>" NL;
PUT "<X:FORMULA>=SHEET1!$1:$1 </
X:FORMULA>" NL;
PUT "</X:EXCELNAME>" NL;
DONE;

PUT "</XML><![ENDIF]-->" NL;
PUT "</HEAD>" NL;
END;
```

The final define event header; sets up some vanity details for the print titles and executes more events with the 'trigger' statements.

```
DEFINE EVENT HEADER;
START:
PUT "<TD";
PUTQ " TITLE=" FLYOVER;
TRIGGER CLASSALIGN;
TRIGGER STYLE_INLINE;
TRIGGER ROWCOL;
PUT ">";
TRIGGER CELL_VALUE;
FINISH:
TRIGGER CELL_VALUE;
PUT "</TD>" NL;
END;
END;
END;
RUN;
```

Next, end all steps, and finish the PROC TEMPLATE with a RUN; The TEMPLATE options have been specified and can now be used with ODS destinations. Remember to use the tagset name that you have defined, in this example, we called our tagset 'test'. Within the ODS statement, code tagset=tagsets.test, and define all the options in an OPTIONS parameter. Remember, these are the same options that we set up in the program and now we have to set values for each of the options in the ODS code.

Exploiting SAS ODS

Enhance your SAS reports!

Schedule a class at your business location or
Attend a public class at our site in Madison

- ◆ Understand ODS components
- ◆ Send SAS reports to HTML for publishing to the web
- ◆ Import SAS reports into Excel and other programs
- ◆ Include graphics in your reports
- ◆ Automatically email SAS reports
- ◆ Show emphasis on graphs by changing background colors or fonts
- ◆ And more!

Call 1-800-997-7081 for details!

```

ODS LISTING CLOSE;
ODS MARKUP FILE = 'C:\MY DOCUMENTS\TEST.XLS'
TAGSET=TAGSETS.TEST
OPTIONS (ORIENTATION="LANDSCAPE"
        SHEET_NAME="SOFTSALE"
        FREEZE_HEADERS="YES"
        ZOOM="90"
        SCALE="50"
        PRINT_TITLE="YES"
        PAPERSIZE="LETTER" );
ODS MARKUP CLOSE;

```

Report Prep 101

In SAS, certain reports are more difficult to create than others. This is often due to the current format of the data. Data often needs to be manipulated prior to the report step in order to create the exact report you need. Here is an example of one difficult report layout and the steps to get the results we are looking for.

After we have coded the ODS statements and run the code, we should see the report in Excel will all the options we specified.

Here is a sample of some monthly sales data that is summarized by sales type. The data includes volume split between base and promotional sales for the current and previous months.

Obj	Name	Division	Year	Sal	Expense	State	Status	Hire	Level	Reviet
1	CHRIS	HARDWARE	2	\$233.11	94.12	WISCONSIN	PART-TIME	1/1/2004	ENTRY LEVEL	YES
2	MARK	HARDWARE	5	\$298.12	52.65	WISCONSIN	FULL-TIME	1/1/2001	ENTRY LEVEL	NO
3	SARAH	SOFTWARE SUPPORT	6	\$301.21	65.17	MINNESOTA	PART-TIME	2/1/2000	ENTRY LEVEL	YES
4	PAT	HARDWARE	4	\$4,009.21	322.12	ILLINOIS	FULL-TIME	1/1/2002	MANAGEMENT	NO
5	JOHN	HARDWARE	7	\$678.43	150.11	WISCONSIN	PART-TIME	1/1/1999	ENTRY LEVEL	YES
6	WILLIAM	HARDWARE	11	\$3,231.75	644.55	MINNESOTA	FULL-TIME	1/1/1995	MID-MANAGEMENT	YES
7	ANDREW	SOFTWARE SUPPORT	24	\$1,762.11	476.13	MINNESOTA	FULL-TIME	2/1/1982	MANAGEMENT	NO
8	BENJAMIN	SOFTWARE SUPPORT	3	\$201.11	25.21	ILLINOIS	PART-TIME	2/1/2003	ENTRY LEVEL	NO
9	JANET	SOFTWARE SUPPORT	1	\$98.11	125.32	WISCONSIN	PART-TIME	2/1/2005	ENTRY LEVEL	NO
10	STEVE	HARDWARE	21	\$6,153.32	1507.12	WISCONSIN	FULL-TIME	1/1/1995	MANAGEMENT	NO
11	JENNIFER	SOFTWARE SUPPORT	1	\$542.11	134.24	ILLINOIS	PART-TIME	2/1/2005	ENTRY LEVEL	YES
12	JOY	SOFTWARE SUPPORT	12	\$2,442.22	716.99	WISCONSIN	FULL-TIME	2/1/1994	MID-MANAGEMENT	YES
13	MARY	SOFTWARE SUPPORT	14	\$5,691.78	2452.11	WISCONSIN	FULL-TIME	2/1/1992	MANAGEMENT	NO
14	TOM	SOFTWARE SUPPORT	5	\$5,669.12	798.15	MINNESOTA	FULL-TIME	2/1/2001	MID-MANAGEMENT	NO
15	BETH	HARDWARE	12	\$4,322.12	982.1	WISCONSIN	FULL-TIME	1/1/1994	MID-MANAGEMENT	YES
16	DANELLE	HARDWARE	4	\$430.16	210.1	WISCONSIN	FULL-TIME	1/1/2002	ENTRY LEVEL	YES

Period	PrTrn	BsTrn	TlTrn	PrSale	BsSale	TlSale
Current Month	175	326	501	12084.12	28196.28	40280.40
Last Month	181	272	453	11692.83	17539.26	29232.09

The report below has been requested by management to analyze the sales data above.

This report is now formatted by our tagset 'test'. When the report is opened in Excel, there is a sheet name, the zoom has been set, and the header is frozen. When print preview is selected from the File menu, the report shows up in landscape orientation and on letter size paper, just as we specified with the options statement.

Type	Period	Total Sales	Total Transactions
Base Volume	Current Month	\$\$	
	Last Month	\$\$	
	Monthly Growth %	%	%
Promotional Volume	Current Month	\$\$	
	Last Month	\$\$	
	Monthly Growth %	%	%
Total Volume	Current Month	\$\$	
	Last Month	\$\$	
	Monthly Growth %	%	%

Several obstacles are involved in creating this report. The incoming data does not include monthly growth percentages and is not organized by the type of sales. In addition to this, the report itself does not have consistent formats (%'s & '\$'s) for each column.

The first step is to calculate the percentages. To calculate the missing percentages, the current month's observation will need to be compared to the last month's observation. One way to read two observations at the same time is to use PROC SQL to join the SalesData dataset to itself. Once we have data from both observations available, the calculations are straight forward.

With just one PROC TEMPLATE step, we have set many options that would otherwise have to be changed manually within Excel.

```

PROC SQL;
CREATE TABLE SalesCompare AS
SELECT
CASE WHEN A.Period = 'Current Month'
THEN 'Monthly Gross %'
END AS Period,
(A.PrTrn - B.PrTrn) / B.PrTrn AS PrTrn
(A.BsTrn - B.BsTrn) / B.BsTrn AS BsTrn
(A.TlTrn - B.TlTrn) / B.TlTrn AS TlTrn
(A.PrSale - B.PrSale) / B.PrSale AS PrSale
(A.BsSale - B.BsSale) / B.BsSale AS BsSale
(A.TlSale - B.TlSale) / B.TlSale AS TlSale

```



```

FROM SalesData A, SalesData B
WHERE A.Period = 'Current Month'
AND B.Period = 'Last Month';
QUIT;

```

This PROC SQL step will create a Monthly Growth Percent observation that will need to be appended to the original data.

```

PROC APPEND BASE=SalesData DATA=SalesCompare;
RUN;

```

The next step is to organize the dataset by sales type. The various types can be identified by looking at the variable names. The best way to organize the data for the report will be to transpose it so it looks like the table below. Following the table are two methods for accomplishing this transposition.

Period	Type	Total_Sales	Total_Transactions
Current Month	Pr	12084.12	175
Current Month	Bs	28196.28	326
Current Month	Tl	40280.40	501
Last Month	Pr	11692.83	181
Last Month	Bs	17539.26	272
Last Month	Tl	29232.09	453
Monthly Growth %	Pr	0.033464	-0.033149
Monthly Growth %	Bs	0.607609	0.198529
Monthly Growth %	Tl	0.377951	0.105960

Solution #1 – A Single Data Step

We can manually transpose our data in a single data step by writing three output observations for each input observation. Before each output statement we set values to three new variables identified in our desired table above.

```

DATA Reportset;
  SET SalesData;
  KEEP Period Type Total_Sales Total_Transactions;
  Type = 'Pr';
  Total_Sales = PrSale;
  Total_Transactions = PrTrn;
  OUTPUT;
  Type = 'Bs';
  Total_Sales = BsSale;
  Total_Transactions = BsTrn;
  OUTPUT;
  Type = 'Tl';
  Total_Sales = TlSale;
  Total_Transactions = TlTrn;
  OUTPUT;
RUN;

```

Solution #2 – Proc Transpose

The method above works well for the size of the example dataset. If the incoming dataset were wider, including several more types of sales it would quickly become inefficient. The second method will show how we can accomplish the same task using the TRANSPOSE procedure.

The syntax of the transpose procedure is:

```

PROC TRANSPOSE DATA=dataset_in OUT=dataset_out;
  VAR variables;
  BY byvariables;
RUN;

```

SAS Report Writing

Enhance your SAS reports!

Schedule a class at your business location or
Attend a public class at our site in Madison

- ◆ Create reports from many sources of data in a variety of formats.
- ◆ Various report generation methods
- ◆ Report generating SAS PROCs
- ◆ Organize data in preparation for report generation
- ◆ Data step programming to create flexible format reports
- ◆ Macro language concepts related to report generation
- ◆ PROC Report
- ◆ And more!

Call 1-800-997-7081 for details!

The Transpose procedure can be used to efficiently change the shape of the sample sales dataset. The output dataset will automatically create two new columns called `_NAME_` and `COL1`. `COL1` will contain the data and `_NAME_` will contain the former variable name. The resulting dataset will look like this (partial):

Period	_NAME_	COL1
Current Month	BsSale	28196.28
Current Month	BsTrn	326
Current Month	PrSale	12084.12
Current Month	PrTrn	175
Current Month	TlSale	40280.40
Current Month	TlTrn	501

The output of the transpose step has created twice as many records as we need because it is unable to pair the base, promotional, and total variables together. This can be overcome in a data step by writing out 1 observation for every 2 that we read as long as the data is grouped (sorted) by Period and `_NAME_`. Once the records with common sales types are combined the dataset from this solution will match that of solution #1.

The final steps to create our reporting dataset with the Proc Transpose will look like this.

```

/* Transpose SalesData */
PROC TRANSPOSE DATA=SalesData OUT=Transdata;
  BY Period;
RUN;

/* Sort the dataset for following data step */
PROC SORT DATA=Transdata;
  BY Period _NAME_;
RUN;

/* Process 2 incoming obs for every 1 output */
/* Drop unneeded variables */
/* Set Type var by parsing _NAME_ */
/* Populate sales & trans var's with data by */
/* parsing the _NAME_ var again */
DATA Reportset;
  DO I=1 TO 2;
    SET Transdata;
    DROP Len COL1 _NAME_ I;
    Type = SUBSTR(_NAME_,1,2);
    Len = LENGTH(_NAME_);
    IF SUBSTR(_NAME_,Len-3,4) = 'Sale'
      THEN Total_Sales = COL1;
    ELSE Total_Transactions = COL1;
  END;
RUN;

```

The main advantage of using the second method is that it allows the variable names of the original dataset to be parsed. This allows a single looped routine within a data step to set all of the variables.

Creating the Report

Now that a dataset has been created specifically for our report, the only obstacle remaining is correctly formatting the output. The original report request requires formats for the type column as well as both of the data columns. The data in the type column currently contains a two character code which can be translated with the following user defined format.

```

PROC FORMAT;
  VALUE $TypeFmt
    'Bs' = 'Base Volume'
    'Pr' = 'Promotional Volume'
    'Tl' = 'Total Volume';
RUN;

```

Note that the format name begins with a character since it is formatting a string value. Also keep in mind that when this format is used later in the PROC REPORT step, a period will be required (\$TypeFmt.) just like any SAS defined format.

The data in the transactions and sales columns has an additional challenge to it. The format changes by row depending on whether or not the data is a Monthly Growth %. In order to accommodate this, the format will have to be applied as the report is built using a CALL DEFINE statement in a COMPUTE block of the PROC REPORT Step.

The CALL DEFINE statement allows the flexibility to redefine the attributes of our report on the fly. It can be very useful in setting a format to match the data (as in this example) or to modify a style based on a set of conditions (i.e. traffic lighting). The syntax of this statement is as follows:

```
CALL DEFINE (column-id,attribute-name,value);
```

The PROC REPORT step used to generate the final report will look like this:

```

PROC REPORT DATA=Reportset;
  COLUMNS Period Type Total_Sales Total_Transactions;
  DEFINE Period/GROUP;
  DEFINE Type/GROUP FORMAT=$TypeFmt.;
  DEFINE Total_Sales/DISPLAY WIDTH=12 'Total Sales';
  DEFINE Total_Transactions/DISPLAY WIDTH=18
    'Total Transactions';

  COMPUTE Period;
  IF Period = 'Monthly Growth %' THEN DO;
    CALL
  DEFINE('Total_Sales','FORMAT','PERCENT9.2');
    CALL DEFINE('Total_Transactions','FORMAT',
    'PERCENT9.2');
  END;
  ELSE DO;
    CALL DEFINE('Total_Sales','FORMAT','3. ');
    CALL DEFINE('Total_Transactions','FORMAT',
    'DOLLAR10.2');
  END;
  ENDCOMP;
RUN;

```



SAS® Consulting Services

Expert IT Consultants

- ◆ We specialize in reporting and data manipulation.
- ◆ Each member of our team has specialties of their own, which include mainframe applications, SQL and database programming, web development, graphs, and maps.

New Cognos Consulting Partners

Support for Products Includes:

- ◆ **PowerPlay**
- ◆ **ReportNet**

Extensive SAS® Experience

- ◆ We work with a variety of industries, such as healthcare, marketing, manufacturing, finance, insurance, academic, government, and telecommunications.
- ◆ Our specialty areas include Data Systems Development, Decision Support and Business Consultation, Report Design, Development, and Web-enablement.
- ◆ As a SAS Affiliate, we are Consulting Partners of the SAS® Institute.
- ◆ All members of our team are SAS Certified Professionals.

Tips, Tricks, and SAS Techniques

Our newest course!

Schedule a class at your business location or
Attend a public class at our site in Madison

- ◆ Small, clever tips from a wide variety of topics
- ◆ Discover secrets we use for our consulting projects
- ◆ Cool uses of common SAS procedures, functions, & statements
- ◆ Learn unique options and techniques
- ◆ Quick tips to improve code
- ◆ Every student will leave with time-saving tips!

Call 1-800-997-7081 for details!

QUICK TIP

One use of new options in the compress function in SAS 9 is to find a zip code from a third line of an address. In the second parameter in the compress function, specify all characters to be removed from the character string. The 'A' in the third parameter specifies that all alpha characters, both upper and lower case, should also be removed.

```
DATA GETZIP;
  INPUT THIRDLINE $25.;
  LENGTH ZIPCODE $5;
  ZIPCODE=COMPRESS(THIRDLINE, ' , - ', 'A');
  DATALINES;
NEW ORLEANS, LA 70160
MADISON, WI 53713
ST. PAUL, MN 55191
;
```

The compress statement could have also been written:

```
ZIPCODE=COMPRESS(THIRDLINE, ' ', 'ASP');
```

The 'S' in the third parameter removes all spaces and the 'P' removes all punctuation.

Remember to add a LENGTH statement so the new variable being created is as small as possible!



Help, My Format's Not Working

I recently received an email from one of our clients concerning some percentages that were being displayed on a SAS report. Our client was comparing the numbers on the report to summary figures produced from the original non-SAS data set.

It appeared as if some of the percentages that were being created in the SAS program did not round correctly. The figures that were different appeared as if they were being truncated instead of rounded.

In the SAS report, the summaries were off slightly compared to the figures on the report created from the original non-SAS file. We manually calculated the summaries and found the summary percent numbers generated from the original file were correct.

I looked at the program that was sent and looked into two of the calculations, BALANCEPCT that was not being created correctly and CLASSPCT that was being created correctly. BALANCEPCT was a text field created using a user-defined format:

```
Balancepct = put
              (sum(balamt*couponamt)/sum(balamt)
              , pctfmt.)
```

One of the first methods that might come to mind to fix the problem would be to simply add the rounding function to the calculation:

```
Balancepct = put
              (round(sum(balamt*couponamt)/sum(balamt), .001)
              , pctfmt.) /* round */
```

However, why did the percentage calculation for CLASSPCT agree with the non-SAS figure?

Cognos Consulting Partners

Support for Products Includes:

◆ **ReportNet**

◆ **PowerPlay**

We will assist in all levels of the project process:

- ◆ Report Design
- ◆ Project Specifications
- ◆ Project Management
- ◆ Data Modeling
- ◆ Programming
- ◆ Report Creation
- ◆ Final Documentation
- ◆ End-User Training

```
Classpct = put
(sum(Class), comma20.2), put (sum(Class) / sum(curbal)
, percent8.2)
```

Looking at the statement more closely, we see that the BALANCEPCT percentage on the report is being generated by utilizing a user created format called PCTFMT. The CLASSPCT percentage is being generated by utilizing a SAS created format called Percent.

SAS formats round the values they generate; user-defined formats truncate the values they display.

The discrepancy did have to do with the truncation of values vs rounding. PROC FORMAT has an option that allows you to specify that you want your user-defined formats to round rather than truncate the values they generate.

The original PROC FORMAT step was:

```
PROC FORMAT;
    PICTURE pctfmt          low-high=' 009.999 %';
RUN;
```

We created an additional format to test this option called PCTFMTR.

```
PROC FORMAT;
    PICTURE pctfmt          low-high=' 009.999 %';
    PICTURE pctfmtr (round) low-high=' 009.999 %';
QUIT;
```

We then created sample data with three identical variable to test the formats.

```
DATA formatds;
    INPUT amt;
    amt_SAS_percent      = amt;
    amt_trunc_user_pctfmt = amt;
    amt_rnd_user_pctfmtr = amt;
DATALINES;
1.1234
1.5275
1.38923
1.38256
;
RUN;
```

Next we applied the two user-defined formats and displayed one with the SAS format PERCENT so we could compare the results.

```
OPTIONS nocenter nonumber nodate;
TITLE 'User Defined Formats';
/* Use Three Different Formats To Display the
Variables */
PROC PRINT DATA=formatds;
    FORMAT
        /* rounds - SAS-written formats */
        amt_SAS_percent      percent8.2
        /* truncates - user-written format */
        amt_trunc_user_pctfmt pctfmt.
        /* rounds - user-written format */
        amt_rnd_user_pctfmtr pctfmtr.
;
RUN;
```

The format with the round option produces the results we were looking for.

RESULTS:

User Defined Formats				
Obs	amt	amt_SAS_percent	amt_trunc_user_pctfmt	amt_rnd_user_pctfmtr
1	1.12340	112.3%	1.123 %	1.123 %
2	1.52750	152.8%	1.527 %	1.528 %
3	1.38923	138.9%	1.389 %	1.389 %
4	1.38256	138.3%	1.382 %	1.383 %



SAS Training at Your Business Location

- ◆ Introduction to SAS®
- ◆ SAS® Report Writing
- ◆ Introduction to PROC Report
- ◆ The SAS® SQL Procedure
- ◆ Advanced SAS®
- ◆ SAS® Macros
- ◆ Using SAS/ACCESS® with Relational Databases
- ◆ SAS® Efficiencies
- ◆ Mainframes Made Easy
- ◆ SyncSort®
- ◆ What's New in SAS® 9
- ◆ Advanced Macros
- ◆ Tips, Tricks, & SAS Techniques
- ◆ Exploiting the SAS® Output Delivery System

View our course catalog at
www.sys-seminar.com/training.php

Call 1-800-997-7081 to discuss details!

QUICK TIP

PROC SQL can be used to produce a quick report showing all of the observations in a file without a unique key. This code rolls up the data by account number and then prints all observations where the account number was on the file more than once. This can be helpful for data cleaning.

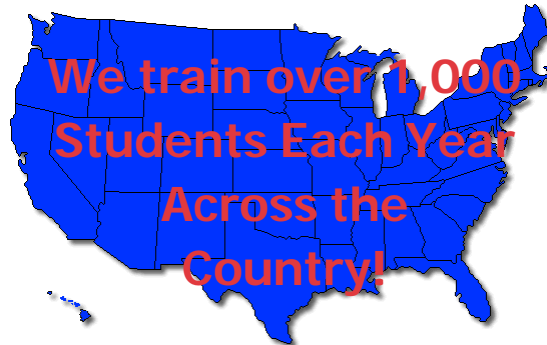
```
PROC SQL;
    SELECT *
    FROM MEMBERS
    GROUP BY ACCT_NUM
    HAVING COUNT(*) > 1;
QUIT;
```



SAS Training Solutions at Your Business Location

Complimentary Follow-up Help Desk
Competitive Prices

Customized Materials & Courses
Quality Training **Nationwide**



We offer more than 16 classes, including our newest...
Tips, Tricks, & SAS Techniques

Interested in training...Unsure of how to coordinate?

- ◆ Have your SAS users take our training survey at www.sys-seminar.com/sscsurvey.php
- ◆ We'll compile reports to **minimize expenses & maximize results**

View our Course Catalog at www.sys-seminar.com

Call 1-800-997-7081 ext. 306 to discover your SAS training solution.

TECHNICAL CREDIT

AND RECOGNITION



Steve First
President

Timothy Broeckert
Trainer/Consultant



Sarah Chronquist
Trainer/Consultant



Katie Ronk
Director of
Operations

Rosalind Gusinow
Trainer/Consultant



Editors

Jennifer First, Katie Ronk