

USING PROC REPORT TO GENERATE 'IMPOSSIBLE' TOTALS

PROC REPORT is a powerful and easy procedure used to generate attractive reports. The programmer controls the appearance and content of every column, thus making it more flexible than PROC PRINT. The DATA step can also create the same reports but usually requires more coding and time spent counting columns for data placement. PROC REPORT also has the ability to create summary lines wherever they are needed. Problems arise, however, when calculating summary lines containing percentages.

Let's Look at an Example

The data used to generate the reports will be sample credit information for 16 people who have purchased a certain number of items at a particular department in one of two stores. Variables shown include the card holder's name (NAME) and credit balance (BALANCE), the annual percentage rate (APR), the number of items purchased within the last year (NUMPURCH), a store code (STORE) of 10 or 20, and a department code (DEPT) within that store. PROC REPORT will translate DEPT into a character string with a format.

```
PROC FORMAT;
  VALUE $DEPTFMT
    '201' = 'Clothing'
    '202' = 'Hardware'
    '203' = 'Shoes'
    '204' = 'Sports'
  ;
RUN;
```

Calculating Simple Percentages

Percentage calculations, such as APR, must be handled in a special way with PROC REPORT. The bulk of the problem lies in the summary lines the PROC produces. By default, the summary lines produce a SUM statistic by simply adding up the column. This is not the value desired, as you will see below.

```
PROC REPORT DATA=ANDREA.RPTTEST
  NOWINDOWS /*send to output */
  SPLIT='*' /*splits labels */
  HEADLINE /*underline header */
  HEADSKIP; /*skip line after */
  COLUMNS STORE DEPT NAME
            /*report col order */
  NUMPURCH BALANCE APR;
```

```
DEFINE STORE / ORDER
  WIDTH=5 /*order values */
  'STORE*NAME'; /*column width */
  /*label */
DEFINE DEPT / ORDER
  ORDER=FORMATTED
  WIDTH=15 /*by formatted val*/
  FORMAT=$DEPTFMT. /*values format */
  'DEPARTMENT';
DEFINE NAME / ORDER
  ORDER=INTERNAL
  WIDTH=10
  'CARDHOLDER*NAME';
DEFINE NUMPURCH / ANALYSIS
  /*default sum stat */
  WIDTH=10
  'NUMBER OF PURCHASES';
DEFINE BALANCE / ANALYSIS
  WIDTH=15
  FORMAT=DOLLAR15.2
  'CURRENT*BALANCE';
DEFINE APR / ANALYSIS
  WIDTH=10
  FORMAT=PERCENT10.2
  'APR';
```

To create summary lines, a BREAK statement must be coded controlling summary line placement. The statements below will create a summary line once the DEPT and STORE values change, resulting in DEPT and STORE total lines. The SUMMARIZE option on the BREAK statement will produce the statistics for all ANALYSIS variables. By default, the SUM statistic is used. As the columns have been defined above, the NUMPURCH, BALANCE, and APR will be added to achieve a summary.

```
BREAK AFTER DEPT / SUMMARIZE
  /*analysis vars stats */
  SUPPRESS /*don't print dept val */
  SKIP /*skip line after sum */
  OL; /*overline */
BREAK AFTER STORE / SUMMARIZE
  SUPPRESS SKIP DOL;
RUN;
```

(Figure 1 at www.sys-seminar.com/prtotals.doc)

All analysis variables have the SUM statistic associated with them. The break statements do all the summarization of the analysis variables. An average of the APR column would make more sense here, since we do not want the addition of the percentages. By adding the MEAN statistic to the DEFINE statement of the

Continued on page 5.....

IN THIS ISSUE

Using PROC REPORT to Generate 'Impossible' Totals: Andrea Decker explains how to sum percentage calculations with PROC REPORT.....Page 1

Puzzler #3 Solution: Steve First shares the solution and winners from the puzzler in our January issue.....Page 2


Are You Sufficiently Efficient?: Roz Gusinow explains how you should be sorting your data setsPage 3

SAS® Help Desk I/O: Robert Tyllo answers a question on how to confirm if a user is browsing or viewing an output file before trying to append specific data.....Page 5

Technical Credit:.....Page 7

Upgrade Your Skills: Jodie Schmidt explains which SAS, SyncSort, and JCL/ISPF classes are availablePage 8

Public Class Schedule:.....Page 8



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

IT STAFF SUPPLEMENTATION

A UNIQUE OPTION



As we mentioned in our last issue, our company now offers several new training courses and services to our customers. I want to focus on one of the new service areas that really has us excited: IT staff supplementation.

Our staff supplementation service is designed to provide you with high quality, cost effective IT support in a number of technology disciplines.

This temporary support can be invaluable for major project work with tight deadlines or when specific technology expertise is needed for 'one time only' projects. We can also get a consultant to you quickly to fill in should you experience unanticipated or untimely separation of key personnel.

Staff supplementation can be a very cost effective alternative as you only pay for the technology *resources that you need* in the *time period that you need them*. We have a wide range of skilled developers available who are knowledgeable in tools ranging from COBOL and Oracle, to Cold Fusion, MS Access, Visual Basic, and C++.

If you want to learn more about this service, please call Russ Lutz at (608) 278-9964 (ext 305). He'll be happy to tell you more about this new area. Discuss your requirements with Russ and he can forward resumes of qualified consultants to you. We carefully screen our candidates and only present you consultants who have the proper technology expertise and experience. If you find someone whose background interests you, Russ will arrange an interview.

As always, we are committed to providing the highest quality SAS training, consulting, and help desk support in the business. That will never change. We bring the same quality commitment to this new service area.

Sincerely,

Steve First
President



PUZZLER #3 SOLUTION

FROM JANUARY'S PUZZLER

Our instructors and consultants use SAS software to solve business problems every day and they run across a variety of challenging issues. We really enjoy solving the most difficult problems and hope you enjoy the challenge too.

Puzzler #3

MVS JCL's SET statement assigns symbolic names to values that can be used in the JCL stream. The SYSPARM option passes parameters from JCL to SAS.

In the example that follows, a non-SAS program, PGM1, needs to be given a parameter and it also needs to be passed to SAS. How can you alter the EXEC SAS statement to pass the SASPARM value to SAS and set the SYSPARM macro variable to 01JAN1999 from the SET statement?

```
//XXXX JOB (XXXX)
//SETUP SET SASPARM='01JAN1999'
//S01 EXEC PGM=PGM1,PARM=&SASPARM
//S02 EXEC SAS <--line to alter
PUT ***** SYSPARM=&SYSPARM;
```

Desired Log

NOTE: SAS SYSTEM OPTIONS SPECIFIED ARE:
SYSPARM="01JAN1999"

```
%PUT ***** SYSPARM=&SYSPARM;
***** SYSPARM=01JAN1999
```

Contestants

We received several solutions to the problem. Many solutions focused on quoting. Some also avoided the OPTIONS symbolic and coded the underlying PARM which was sent to the SAS supervisor.

Sample Solution

```
//XXXX JOB (XXXX)
//SETUP SET SASPARM='01JAN1999'
//S01 EXEC PGM=PGM1,PARM=&SASPARM
//S02 EXEC SAS,PARM='SYSPARM="&SASPARM"'
PUT ***** SYSPARM=&SYSPARM;
```

Desired Log

NOTE: SAS SYSTEM OPTIONS SPECIFIED ARE:
SYSPARM="01JAN1999"

```
%PUT ***** SYSPARM=&SYSPARM;
***** SYSPARM=01JAN1999
```

The Winners

The first three people to code a solution correctly were:

- Kevin Faes of John Deere Parts Distribution
- Robert Mengis of Bear Creek Corporation
- Mark MacMullen of Hallmark Information Services

Congratulations to our winners. Each of you will receive a free coffee mug and a \$100 credit towards our training service.

Thanks to all the respondents for some very clever solutions.



SYSTEMS SEMINAR CONSULTANTS, INC.

Copyright © 2000 Systems Seminar Consultants, Inc. Madison, WI
All rights reserved. Printed in USA. The Missing Semicolon is a trademark of
Systems Seminar Consultants, Inc., SAS is a registered trademark of SAS Institute Inc., and
SyncSort is a registered trademark of SyncSort, Inc.
in the USA and other countries.

ARE YOU SUFFICIENTLY EFFICIENT

WITH PROC SORT?

Sorting data sets is one of the most commonly used tasks for processing information. Sequenced data sets are needed when the data set is passed into a PROC step that has a BY statement. They are also needed when the data set is passed to a DATA step that uses a BY statement, as in a match-merge or activity update. While SORT is one of the simplest operations, it is one of the most resource intensive procedures.

Using PROC SORT to reorder a data set and create a separate output data set requires three or more passes through the file. It requires enough memory and/or sort work areas to store two copies of your data set during sequencing. It also requires an intense amount of time in terms of I/O time and CPU time, as well as clock time.

Example

```
// 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

When using large data sets, it may not be possible to acquire this much space or be practical to use the required amount of time. Are there ways of handling data sets so that you can have the data sets in the order you need them and either avoid sorting, make sorting less space intensive, or make sorting less time intensive?

Avoiding Sorting - Case 1

If your procedure allows the use of the CLASS statement, you will not need to presort the data. Instead of sorting your data set and then using PROC MEANS with a BY statement, simply use PROC MEANS with the CLASS statement. Note that this requires more memory.

Example

```
/* Adequate Program */
proc sort data=midwest;
class state;
run;

proc means data=midwest;
by state;
run;

/* Better Program */
proc means data=midwest nway;
class state;
run;
```

Avoiding Sorting - Case 2

If the data in a raw (non-SAS) file is already in the correct sequence, you do not need to sort the data. However, if the data set is passed to PROC SORT, PROC SORT will not recognize that the data set is already sorted. PROC SORT will resort the data set.

QUICK TIP

Use the minimum (><) or maximum (<>) operators in dataset statements.

Example:

statements	result
Y=9; X=3	
MAX=Y<>X;	MAX=9
MIN=Y><X;	MIN=3



Example

Raw Data CENSUS

AZ	MOHAVE	KINGMAN	132,303
IN	DUBOIS	JASPER	38,303
WI	DANE	MADISON	660,393
WY	NATRONA	CASPER	73,202

```
data findcity;
infile census;
input @1 state $2.
@4 county $8.
@13 city $10.
@24 populate comma7.;
run;
```

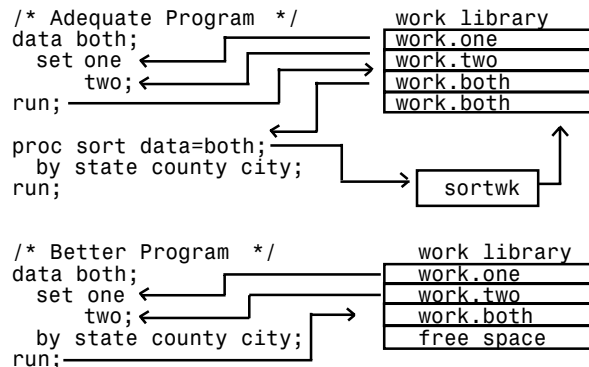
```
proc sort data=findcity; /* a sort is not needed */
by state county;
run;
```

If your data is already in the sequence you need, do not sort the data. Later steps, that process the data with BY logic, will be able to properly handle the information.

Avoid Sorting - Case 3

If two data sets are already in the proper sequence, use the SET statement and a BY statement to interleave the observations rather than concatenating the data sets and then sorting them.

Example



Sorting Efficiently - Tip 1

By reducing the size of the input file, less space and time will be used during processing. When creating the original input data set remove any unnecessary observations by using WHEREs, IFs, or DELETES.

Continued on page 4.....

ARE YOU SUFFICIENTLY EFFICIENT?

CONTINUED FROM PAGE 3

Sorting Efficiently - Tip 2

When creating the original input data set remove any unnecessary variables by using KEEP or DROP. Use the KEEP or DROP data set option on the input data set to eliminate unneeded variables. Group the procedures that need to use the data in this order together.

Example

```
/* Adequate Program */      /* Better Program */
proc sort data=midwest;      proc sort data=midwest;
  by state;                  by state;
run;                          run;

proc print data=midwest;     proc print data=midwest;
  by state;                  by state;
run;                          run;

proc sort data=midwest;     proc chart data=midwest;
  by county;                by state;
run;                          vbar x y;
run;                          run;

proc print data=midwest;    proc sort data=midwest;
  by county;                by county;
run;                          run;

proc sort data=midwest;     proc print data=midwest;
  by state;                  by county;
run;                          run;

proc chart data=midwest;    proc print data=midwest;
  by state; vbar x y;run;    by county;
run;                          run;
```

Sorting Efficiently - Tip 3

When sorting, use SAS sort options to save time and or space.

Sort Option 1 - NODUP

Use the NODUP option to eliminate identical observations from the input data set.

Example

```
data cityds;
  infile census;
  input @1 state $2.
        . . . rest input
run;

proc sort data=cityds out=scities nodup;
  by state county city;
run;
```

Sort Option 2 - NODUPKEY

To delete observations with identical values of the BY variable, use the NODUPKEY option.

Example

```
data findcity;
  infile census;
  input @1 state $2.
        . . . rest of input
run;

proc sort data=cityds out=scities nodupkey;
  by state county city;
run;
```

Sort Option 3 - NOEQUALS

Consider the use of the NOEQUALS option on the PROC SORT statement when CPU time is a particular problem. In the normal sort procedure, observations with identical BY variable values maintain their order from the input data set to the output data set. Specifying the NOEQUALS option on the PROC SORT statement tells SAS that the order of the observations, when the key values are the same, need not be maintained. This can save both CPU time and memory.

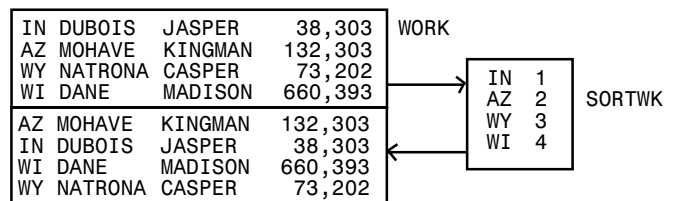
Example

```
PROC SORT DATA=CITYDS NOEQUALS;
  BY STATE COUNTY;
RUN;
```

Sort Option 4 - TAGSORT

When memory is of more concern than processing (CPU) time, you may wish to consider using the TAGSORT option. The TAGSORT option on the PROC SORT statement tells SAS to use 'tags' for sorting the data set instead of using the entire observation. The tag is composed of the values of the key variables specified on the By statement and the observation number of the row. The tags are then sorted, and when the sorting is finished, the tags are used to access the observation on the original data set in sorted order. You can see that if the number of bytes in the tag is much less than the size of the entire observation, this could substantially reduce the amount of temporary storage required. This may, however, substantially increase the amount of time it takes to process the observation.

Example



```
PROC SORT DATA=CITYDS TAGSORT;
  BY STATE COUNTY;
RUN;
```

While PROC SORT is a valuable procedure for processing your data sets, it must be used wisely in order to avoid unnecessary resource consumption. We hope this article provided you with a few options to consider the next time you read or combine information with PROC SORT. Future articles will address how to create indexes for your data set when it is created.

QUICK TIP

Use the GETOPTION function to create variables to hold SAS option values.

Example:

```
YEARCUT=GETOPTION('YEARCUTOFF');
```

run;

run;

SAS® HELP DESK I/O

SOLUTIONS FROM OUR HELP DESK SERVICE

Q: "I have a weekly production job that appends results to an existing output file. If someone else is viewing this output file as I'm trying to append to it, my program abends. How can I check to see if someone is browsing or viewing the output file before I try to append data?"

A: The following code, inserted right before the file is appended, will check to see if someone else is browsing the file. If there is someone in the file, the loop waits 15 seconds before checking again. The loop repeats this cycle at least 5 times before proceeding. One could adjust the counter and delay time accordingly.

Code:

```
/* ***** */
/* THIS DATA STEP TRIES TO OPEN AN EXTERNAL          */
/* FILE WITH DISP=MOD TO ALLOW THE PROGRAM TO         */
/* LATER APPEND TO THE FLAT FILE. IF ANY USER        */
/* IS READING OR USING THAT FILE, WE WANT TO         */
/* TRY SEVERAL TIMES TO OPEN THE FILE. IF AFTER      */
/* SEVERAL TRIES THE FILE CANNOT BE OPENED,         */
/* CANCEL THIS PROGRAM WITH 'ABORT' AND DISPLAY      */
/* A MESSAGE IN THE LOG.                             */
/* ***** */
```

```
DATA _NULL_;
    * MAIN LOOP WILL PROCESS TIL FILE
    CAN BE OPENED OR 5 TRIES;

    MAXTRIES=5;
    DO UNTIL(STOP);
        COUNT+1;
        * IF FUNCTION RETURNS ZERO, FILE IS
        AVAILABLE WITH DISP=MOD;
        * DDNAME FILENAME DISP;
        RC=FILENAME('OUTPUT1','BEN222.SAS.BEAMT1',, 'MOD');
        PUT 'TRYING TO OPEN FILE - RETURN CODE =' RC;
        * SEE IF FILENAME FUNCTION WORKED;
        IF RC = 0 THEN STOP=1;
        ELSE
            DO;
                * IF FOLLOWING LOOP TRIED TO
                MAXTRIES, ABORT JOB;
                IF COUNT GT MAXTRIES THEN
                    DO;
                        PUT 'JOB WILL NOT RUN - TRIED TO OPEN FILE
                        SEVERAL TIMES';
                        ABORT;
                    END;
                * FOLLOWING DOES A 'TIMER' LOOP FOR
                15 SECONDS;
                STOPTIME=TIME()+15;
                DO UNTIL(TIME() GT STOPTIME);
                    END;
                * GO BACK AND TRY TO OPEN THE FILE
                AGAIN;
            END;
        * LEAVE THIS NEXT 'STOP' HERE FOR
        SAFE PROGRAMMING;
    END;
END;
STOP;
RUN;
```



QUICK TIP

———— If you have a long WHERE clause which contains an AND, you can break it into two clauses. The SAME-AND operator adds requirements to your first WHERE clause.

Example:

```
WHERE SALES>=500;
WHERE SAME AND EXPENSE<=100;
```



GENERATING 'IMPOSSIBLE' TOTALS

CONTINUED FROM PAGE 1

APR column, the summarized lines will show the average of the column, not the sum.

```
DEFINE APR / ANALYSIS
    MEAN
    WIDTH=10
    FORMAT=PERCENT10.2
    'APR';
```

(See Figure 2 at www.sys-seminar.com/prtotals.doc)

The BREAK statements can remain the same. The SUMMARIZE option will now use the average of the APR.

Reporting at a higher level (summarizing detail lines into departments and showing only store totals) requires a little more coding. To summarize the detail lines down to just DEPT and STORE totals, we need to include a PROC SUMMARY beforehand. The NWAY option is used to get the highest level of interaction between the STORE and DEPT. The SAS data set created from the OUTPUT statement contains the detail lines summarized into departments within the two stores.

```
PROC SUMMARY DATA=ANDREA.RPTTEST NWAY;
    CLASS STORE DEPT;
    VAR NUMPURCH BALANCE APR;
    OUTPUT OUT=SUMM1OUT
        SUM(NUMPURCH)=PURCHTOT
        SUM(BALANCE)=BALTOT
        MEAN(APR)=MEANAPR;
RUN;
```

The PROC REPORT has not changed. SUMM1OUT will be fed into the procedure to create the report. This new dataset has different variable names, but the new define statements look identical to their unsummarized counterparts. The MEAN statistic is coded into the MEANAPR column to provide us with the average of the APR, not the sum, in the summary lines.

```
PROC REPORTDATA=SUMM1OUT NOWINDOWS
    SPLIT='*' HEADLINE HEADSKIP;
    COLUMNS STORE DEPT PURCHTOT BALTOT MEANAPR;
    DEFINE STORE / (same as previous PROC REPORT);
    DEFINE DEPT / (same as previous PROC REPORT);
    DEFINE PURCHTOT / ANALYSIS WIDTH=10
        'NUMBER OF*PURCHASES';
    DEFINE BALTOT / ANALYSIS WIDTH=15
        FORMAT=DOLLAR15.2
        'CURRENT*BALANCE';
```

Continued on page 6.....

GENERATING 'IMPOSSIBLE' TOTALS

CONTINUED FROM PAGE 5

```
DEFINE MEANAPR / ANALYSIS MEAN
  WIDTH=10 'APR'
  FORMAT=PERCENT10.2;
BREAK AFTER STORE / SUMMARIZE SUPPRESS SKIP OL;
RUN;
```

(See Figure 3 at www.sys-seminar.com/prtotals.doc)

Since the detail lines are already summarized by DEPT, only one BREAK statement was coded. We only need to compute the STORE totals.

Computing Weighted Percentages

Getting simple averages of percentages is not complicated. It becomes more difficult when the columns represent a weighted percentage. PROC REPORT does support a WEIGHT statement; this statement will weight all analysis variables. This is not desired as only one variable needs to be weighted. Another way to get a weighted variable is to use the PROC SUMMARY, but again all variables will be weighted. Therefore, two PROC SUMMARY steps are needed.

The first step looks identical to the previous example, but the average of the APR variable is removed. The sum of the number of purchases and balances is calculated at store-department level. NWAY statements are used in both steps, as we are only interested in the highest level of interaction between STORE and DEPT.

```
PROC SUMMARY DATA=ANDREA.RPTTEST NWAY;
CLASS STORE DEPT;
VAR NUMPURCH BALANCE;
OUTPUT OUT=SUMM2A
  SUM(NUMPURCH)=PURCHTOT
  SUM(BALANCE)=BALTOT;
RUN;
```

In order to get the APR to be weighted by the balance, we must use a second step to compute the weighted variable and roll up the data to store-department level.

```
PROC SUMMARY DATA=ANDREA.RPTTEST NWAY;
CLASS STORE DEPT;
WEIGHT BALANCE;
VAR APR;
OUTPUT OUT=SUMM2B MEAN(APR)=WTDAPR;
RUN;
```

The resulting SAS data sets must be merged together by STORE and DEPT to give one SAS data set we can plug into the PROC REPORT.

```
DATA SUMM2;
MERGE SUMM2A SUMM2B;
BY STORE DEPT;
RUN;
```

The resulting SAS data set contains the store and departments along

with the total purchases, total balances, and an annual percentage rate weighted by the balance.

```
PROC REPORT DATA=SUMM2 NOWINDOWS
  SPLIT='*' HEADLINE HEADSKIP;
COLUMNS STORE DEPT PURCHTOT BALTOT WTDAPR;
DEFINE STORE / (same as previous PROC REPORT);
DEFINE DEPT / (same as previous PROC REPORT);
DEFINE PURCHTOT / (same as previous PROC REPORT);
DEFINE BALTOT / (same as previous PROC REPORT);
DEFINE WTDAPR / ANALYSIS MEAN
  WIDTH=10
  FORMAT=PERCENT10.2
  'WEIGHTED*APR';
BREAK AFTER STORE / SUMMARIZE SUPPRESS SKIP OL;
RUN;
```

(See Figure 4 at www.sys-seminar.com/prtotals.doc)

The MEAN statistic was included with the WTDAPR definition to avoid adding the percentages. This is not the desired result, since the summary lines are the average of the weighted percentages, not the weighted percentage at the store level. Getting the weighted percentage at store level requires quite a bit more coding and becomes quite complex. We eventually want to let PROC REPORT calculate the summary line.

Store Weighted APR=

Sum of $\left(\frac{\text{WTDAPR for store-dept} * \text{store-dept BAL}}{\text{store BAL}} \right)$ for each dept

The information must be summarized using techniques similar to those used in the previous example. The store balance must be attached to each observation because it is the denominator of the Store Weighted APR formula.

PROC SUMMARY is used to generate two SAS data sets in one step. NWAY was used in the previous examples to keep only the highest level of interaction. If we take this option off, we get all levels. Observations where _TYPE_ =2 are the individual store totals. This SAS data set will contain the sum of the balances for each store, which is the denominator of the Store Weighted APR formula. The second SAS data set contains the _TYPE_=3, or the highest level of interaction. This SAS data set contains the details for the report and the store-department balance for the numerator of the Store Weighted APR formula.

```
PROC SUMMARY DATA=ANDREA.RPTTEST;
CLASS STORE DEPT;
VAR NUMPURCH BALANCE;
OUTPUT OUT=SUMM2A(WHERE=( _TYPE_=3))
  SUM(NUMPURCH)=PURCHTOT
  SUM(BALANCE)=BALTOT;
OUTPUT OUT=SUMM2A2(WHERE=( _TYPE_=2))
  SUM(BALANCE)=STBALTOT;
RUN;
```

Again PROC SUMMARY is used to create a SAS data set which has the APR weighted by balance. NWAY is used because only the highest level of interaction between STORE and DEPT are needed.

```
PROC SUMMARY DATA=ANDREA.RPTTEST NWAY;
CLASS STORE DEPT;
WEIGHT BALANCE;
VAR APR;
OUTPUT OUT=SUMM2B MEAN(APR)=WTDAPR;
```

QUICK TIP

To format a SAS date with a four digit year, two digit month, and two digit day without slashes (e.g., 20000401), use the YYMMDDN8. format.

run;

RUN;

The SAS data sets, containing the highest level of interaction, are merged. This dataset will look identical to the previous report's created SAS dataset.

```
DATA SUMM2;
MERGE SUMM2A SUMM2B;
BY STORE DEPT;
RUN;
```

The balance for the store totals needs to be attached to the SAS data set just created in order for the weighted APR column to be computed correctly. The SAS data set SUMM2A2, which contains the store balance totals, is now merged with the previously created data set. The only variables being added to the merge from SUMM2A2 are the balance and the store number. This will prevent any necessary variables from being overridden. One iteration of the Store Weighted APR is also calculated in this data step.

```
DATA SUMM2;
MERGE SUMM2 SUMM2A2(KEEP=STBALTOT STORE);
BY STORE;
TOTAPR=(WTDAPR*BALTOT)/STBALTOT;
RUN;
```

The SAS data set is now ready for the PROC REPORT. The columns are the same as previous reports, but with one addition, TOTAPR. This is the value created from the previous data step. In the DEFINE statement for this variable, the NOPRINT option is specified. This will prevent this column from showing up on the final report. PROC REPORT will add the TOTAPR column to create the weighted APR for the store summary lines.

```
PROC REPORTDATA=SUMM2 NOWINDOWS
SPLIT='*' HEADLINE HEADSKIP;
COLUMNS STORE DEPT PURCHTOT BALTOT WTDAPR TOTAPR;
DEFINE STORE / (same as previous PROC REPORT);
DEFINE DEPT / (same as previous PROC REPORT);
DEFINE PURCHTOT / ANALYSIS
WIDTH=10
FORMAT=COMMA10.0
'NUMBER OF*PURCHASES';
DEFINE BALTOT / ANALYSIS
WIDTH=15
FORMAT=DOLLAR15.2
'CURRENT*BALANCE';
DEFINE WTDAPR / ANALYSIS
WIDTH=10
FORMAT=PERCENT10.2
'WEIGHTED*APR';
DEFINE TOTAPR / ANALYSIS
NOPRINT;
BREAK AFTER STORE / OL SUPPRESS SKIP;
COMPUTE AFTER STORE ;
LINE @33 PURCHTOT.SUM COMMA10.0
+2 BALTOT.SUM DOLLAR15.2
+2 TOTAPR.SUM PERCENT10.2;
ENDCOMP;
RUN;
```

(See Figure 5 at www.sys-seminar.com/prtotals.doc)

The BREAK statement lacks the SUMMARIZE option. To compute the summary lines, use a COMPUTE block with a LINE statement; this is similar to a PUT statement in the DATA step, but it does not support all options. Absolute and relative pointers are used to point to where the information will be placed on the report. Formats are used to insert commas, dollar signs, and percent signs. The variables are referred to by two-level names: variable name and summarization

QUICK TIP

Use the TIMEAMPW. format to display times with AM or PM.

Example:

Value	Format	Result
'18:15'T	TIME5.	18:15
	TIMEAMPW8.	6:15 PM



statistic, respectively. The PURCHTOT and BALTOT totals will be the same as when a SUMMARIZE option was used on the BREAK statement. The WTDAPR column is not used; the TOTAPR column sum, which is the correct weighted APR at the store level, is shown instead.

PROC REPORT is an easy and powerful reporting procedure capable of producing a variety of reports, from simple reports with simple summary lines to complex reports with complex summary lines. When used with PROC SUMMARY and some additional computing statements, PROC REPORT can be manipulated to produce calculations often thought "impossible."



TECHNICAL CREDIT AND RECOGNITION



Andrea Decker
Trainer/Consultant
Author of this issue's:
Using PROC REPORT to Generate
'Impossible' Totals, Page 1



Steve First
President
Author of this issue's:
Puzzler #3 Solution, Page 2



Roz Gusinow
Trainer/Consultant
Author of this issue's:
Are you Sufficiently Efficient with PROC
REPORT?, Page 3



Robert Tyllo
Trainer/Consultant
Author of this issue's:
SAS Help Desk I/O, Page 5

Upgrade Your Skills.....Jodie Schmidt
Quick TipsAndrea Decker & David Beam
PublisherJodie Schmidt
EditorsDavid Beam, Russ Lutz, & Cindy Kersten



UPGRADE YOUR SKILLS

WE PROVIDE QUALITY ON-SITE TRAINING

Systems Seminar Consultants, Inc. offers highly-rated technical training. Could the employees at your company benefit from one or more of the following classes?

Introduction to SAS®	Acquaint yourself with SAS language to analyze data and write reports.
SAS® in the Windows Environment	Translate your mainframe skills to PC SAS.
The SAS® SQL Procedure	Master this data access, report writing, and joining tool.
Introduction to PROC Report	Enhance your report writing skills with this powerful procedure.
SAS® Report Writing	Generate a wide variety of reports using advanced techniques.
SAS/GRAPH®	Produce high quality, color graphics and charts.
SAS® Efficiencies < NEW!	Learn efficiency techniques and save system resources.
Advanced SAS®	Read complex data files, use SAS functions, process arrays, learn advanced joining methods, and master efficiency techniques.
SAS/ACCESS® with Relational Databases	Learn to interface with relational database products.
SAS® Macros	Learn to automate and eliminate repetitive code.
SAS/AF®	Develop interactive screens for your SAS systems.
SAS® Client Server Computing	Design client/server solutions to perform

PUBLIC CLASS SCHEDULE

Introduction to SAS®

April 3-5	\$725	Madison, WI
April 10-12	\$725	St. Paul, MN
May 15-17	\$725	St. Paul, MN
June 12-14	\$725	St. Paul, MN
June 19-21	\$725	Madison, WI

The SAS® SQL Procedure

May 18	\$350	St. Paul, MN
--------	-------	--------------

SAS® Report Writing

April 6-7	\$525	Madison, WI
April 13-14	\$525	St. Paul, MN

Advanced SAS®

June 7-9	\$725	St. Paul, MN
June 22-23	\$525	Madison, WI

To register call (608) 278-9964 or visit www.sys-seminar.com.

run;

data transfers, submit code remotely, access remote data, and allow multi-user access.

SAS/FSP®

Learn to develop data entry systems.

Upgrading to SAS®

Version 7 & 8 < NEW! Learn features of the newest versions of SAS.

JCL/ISPF® < NEW!

Learn basic mainframe editing and JCL syntax.

SyncSort® < NEW!

Learn to use the SyncSort tool.

Our prices are competitive, and we welcome the opportunity to provide your next SAS, SyncSort, or JCL/ISPF training class.

Call us at (608) 278-9964 ext. 311 for a copy of our IT Training Course Catalog to learn more about our classes or call (608) 278-9964 ext. 306 to obtain a quote on a specific training course, location, and date.

run;



SYSTEMS SEMINAR CONSULTANTS, INC.

2997 Yarmouth Greenway Drive
Madison, WI 53711

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

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