

Advanced Features of PROC REPORT

Kim L. Kolbe Ritzow of Systems Seminar Consultants, Kalamazoo, MI

Due to the length of this paper output to these examples will not appear in the Proceedings. Copies of the entire paper, including examples, can be obtained from the author.

Abstract

PROC REPORT is a powerful new Base SAS[®] PROC which combines the functionality of other BASE SAS PROCs such as PRINT, MEANS, FREQ, and TABULATE into a single PROC step.

This paper is designed for users with existing knowledge of PROC REPORT's more basic report writing features and who are interested in learning additional techniques to generate more sophisticated reports.

This paper will focus on customized breaklines, grouped or stacked column headings, and advanced tricks for improving the appearance of reports.

Skipping More Than One Line Between Totals

The SKIP option will skip one line before the next group begins printing. To skip more than one line, a COMPUTE block can be used:

```
PROC FORMAT;
  VALUE $REGIONF
    'N' = 'NORTH'
    'S' = 'SOUTH'
    'E' = 'EAST'
    'W' = 'WEST'
;
RUN;
```

```
PROC REPORT DATA=SASDATA.STOREDAT NOWINDOWS
  HEADLINE HEADSKIP;
```

```
COLUMNS REGION STORE DEPT SALEAMT TAX NET;
```

```
DEFINE REGION / GROUP ORDER=FORMATTED WIDTH=6
  FORMAT=$REGIONF. 'Region';
DEFINE STORE / GROUP ORDER=FORMATTED 'Store';
DEFINE DEPT / GROUP 'Dept';
DEFINE SALEAMT / ANALYSIS SUM FORMAT=DOLLAR10.2
  WIDTH=10 'Amount/of Sale';
DEFINE TAX / COMPUTED FORMAT=DOLLAR10.2
  WIDTH=10 'Sales/Tax';
```

```
DEFINE NET / COMPUTED FORMAT=DOLLAR10.2
  WIDTH=10 'Net/Sale';
```

```
COMPUTE TAX;
  TAX= SALEAMT.SUM * .05;
```

```
ENDCOMP;
```

```
COMPUTE NET;
  NET= SALEAMT.SUM - TAX;
ENDCOMP;
```

```
BREAK AFTER REGION/OL SUMMARIZE SUPPRESS;
```

```
COMPUTE AFTER REGION;
  LINE '';
  LINE '';
ENDCOMP;
```

```
RBREAK AFTER/DOL DUL SUMMARIZE;
```

```
TITLE1 'AREA SALES FOR SELECT DEPARTMENTS';
TITLE2 'FOR THE WEEK OF 5/02/94 - 5/08/94';
RUN;
```

(see Output #1 for results)

The LINE statement is specified in a COMPUTE block between a COMPUTE and ENDCOMP statement. The LINE statement can also be used to write text in the break lines.

Changing Group Variable's Value in Summary

By default when using the BREAK statement, SAS will print the value of the BREAK variable on the summary line, unless the SUPPRESS option is used, which will prevent the value from printing. A COMPUTE block can be used to print a value other than the BREAK variable's value:

```
DATA SASDATA.STOREDAT;
  LENGTH REGION $14;
  SET SASDATA.STOREDAT;
RUN;
```

```
PROC FORMAT;
  VALUE $REGIONF
    'N' = 'NORTH'
    'S' = 'SOUTH'
    'E' = 'EAST'
    'W' = 'WEST'
;
RUN;
```

```
PROC REPORT DATA=SASDATA.STOREDAT NOWINDOWS
  HEADLINE HEADSKIP;
```

```
COLUMNS REGION STORE DEPT SALEAMT TAX NET;
```

```
DEFINE REGION / GROUP ORDER=FORMATTED
  FORMAT=$REGIONF14. WIDTH=14 'Region';
DEFINE STORE / GROUP ORDER=FORMATTED 'Store';
DEFINE DEPT / GROUP ORDER=FORMATTED 'Dept';
DEFINE SALEAMT / ANALYSIS SUM FORMAT=DOLLAR10.2
  WIDTH=10 'Amount/of Sale';
```

```

DEFINE TAX / COMPUTED FORMAT=DOLLAR10.2
  WIDTH=10 'Sales/Tax';
DEFINE NET / COMPUTED FORMAT=DOLLAR10.2
  WIDTH=10 'Net/Sale';

```

```

COMPUTE TAX;
  TAX= SALEAMT.SUM * .05;
ENDCOMP;

```

```

COMPUTE NET;
  NET=SALEAMT.SUM - TAX;
ENDCOMP;

```

```

BREAK AFTER REGION/OL SKIP SUMMARIZE;

```

```

COMPUTE AFTER REGION;
  REGION='Regional Total';
ENDCOMP;

```

```

RBREAK AFTER/DOL DUL SUMMARIZE;

```

```

TITLE1 'AREA SALES FOR SELECT DEPARTMENTS';
TITLE2 'FOR THE WEEK OF 5/02/94 - 5/08/94';
RUN;

```

(see Output #2 for results)

AS LONG AS THE LENGTH OF THE CUSTOMIZED TEXT DOES NOT EXCEED THE LENGTH OF YOUR BREAK VARIABLE, A VALUE CAN BE ASSIGNED TO YOUR BREAK VARIABLE IN THE COMPUTE BLOCK. In this example, the length of the BREAK variable (REGION) is a 1-byte character. Several things will need to be done before customized text longer than 1-byte can appear on the report. First, the length of REGION must be set to the length of the customized text. This is done in a Data Step prior to PROC REPORT. Secondly, a length value must be specified on the DEFINE statement's FORMAT= option to prevent truncation after the 5th byte of the customized break (longest formatted value is 5; if it were 7, I would get truncation after the 7th byte, and so on). Finally, the DEFINE statement's WIDTH= option must be adjusted as well to accommodate the width of the column. When the customized break text exceeds the length of the BREAK variable, these three things MUST BE DONE in order for the customized break value to print on the report.

When assigning a value to the BREAK variable in the COMPUTE block, as is being done here, the SUPPRESS option on the BREAK statement has no effect upon whether or not the customized text will appear.

Several LINE statements could have also been specified within the COMPUTE block as was done

in an earlier example to print multiple blank lines after or before customized break text.

Another Way of Doing It

Rather than going through the various steps as was done in the previous example to adjust the width so the customized break text would print out, the entire printing of the break line could be controlled through a COMPUTE block:

```

PROC FORMAT;
  VALUE $REGIONF
    'N' = 'NORTH'
    'S' = 'SOUTH'
    'E' = 'EAST'
    'W' = 'WEST'
;

```

```

RUN;

```

```

PROC REPORT DATA=SASDATA.STOREDAT NOWINDOWS
  HEADLINE HEADSKIP;

```

```

COLUMNS REGION STORE DEPT SALEAMT TAX NET;

```

```

DEFINE REGION / GROUP ORDER=FORMATTED WIDTH=6
  FORMAT=$REGIONF. 'Region';
DEFINE STORE / GROUP ORDER=FORMATTED 'Store';
DEFINE DEPT / GROUP 'Dept';
DEFINE SALEAMT / ANALYSIS SUM FORMAT=DOLLAR10.2
  WIDTH=10 'Amount/of Sale';
DEFINE TAX / COMPUTED FORMAT=DOLLAR10.2
  WIDTH=10 'Sales/Tax';
DEFINE NET / COMPUTED FORMAT=DOLLAR10.2
  WIDTH=10 'Net/Sale';

```

```

COMPUTE TAX;
  TAX= SALEAMT.SUM * .05;
ENDCOMP;

```

```

COMPUTE NET;
  NET=SALEAMT.SUM - TAX;
ENDCOMP;

```

```

BREAK AFTER REGION/OL;

```

```

COMPUTE AFTER REGION;
  LINE @1 'Regional Totals '
    @37 SALEAMT.SUM DOLLAR10.2
    @49 TAX DOLLAR10.2
    @61 NET DOLLAR10.2;
  LINE ' ';
ENDCOMP;

```

```

RBREAK AFTER/DOL DUL SUMMARIZE;
TITLE1 'AREA SALES FOR SELECT DEPARTMENTS';
TITLE2 'FOR THE WEEK OF 5/02/94 - 5/08/94';
RUN;

```

(see Output #3 for results)

When writing the code to control the printing of the break line information notice that we have reverted back to the original code without any adjustments to the user-defined format,

FORMAT= and WIDTH= options.

The BREAK statement is controlling the underlining of the values with the OL option and the COMPUTE block is controlling the printing of all the values. Notice on the LINE statement pointers (@) were used to control how and where information will print out. It is important to associate a format with the values on the LINE statement. If formats are not specified, SAS won't display the value and no error or warning message will appear in the SAS Log. When writing out SAS variables notice that the variable's name and the associated statistic (SALEAMT.SUM) were specified. THE STATIC SPECIFIED MUST MATCH THE STATISTIC DEFINED ON THE ANALYSIS STATEMENT FOR THE VARIABLE. ANOTHER STATISTIC CANNOT BE ASSOCIATED WITH THE VARIABLE ON THE COMPUTE BLOCK (such as SALEAMT.MEAN). To associate multiple statistics with the same variable aliases can be used (discussed later in the paper). However, ALIASES CANNOT BE USED IN COMPUTE BLOCKS. The only way to use a variable in a COMPUTE block with a different statistic associated with it other than the one used in the DEFINE statement, is to create a new variable in the Data Step prior to PROC REPORT which contains the same value of the variable you are trying to use.

Notice with COMPUTED variables (such as TAX and NET) they are referred to by their computed name and a statistic is NOT associated with them. If a statistic is associated with them (such as TAX.SUM) missing values for that variable will appear on the report. The usage of a COMPUTED variable CANNOT be changed to something like TAX.MEAN.

If the SUPPRESS option were specified on the BREAK statement, it would have no effect on the printing of the text in the COMPUTE block.

Custom Break Lines

Customized break lines can easily be created by a variation on some of the many things we have just seen. For example:

```
PROC FORMAT;
  VALUE $REGIONF
    'N' = 'NORTH'
    'S' = 'SOUTH'
    'E' = 'EAST'
    'W' = 'WEST'
;RUN;
```

```
PROC REPORT DATA=SASDATA.STOREDAT NOWINDOWS
  HEADLINE HEADSKIP;
```

```
COLUMNS REGION STORE DEPT SALEAMT TAX NET;
```

```
DEFINE REGION / GROUP ORDER=FORMATTED WIDTH=6
  FORMAT=$REGIONF. 'Region';
DEFINE STORE / GROUP ORDER=FORMATTED 'Store';
DEFINE DEPT / GROUP 'Dept';
DEFINE SALEAMT / ANALYSIS SUM FORMAT=DOLLAR10.2
  WIDTH=10 'Amount/of Sale';
DEFINE TAX / COMPUTED FORMAT=DOLLAR10.2
  WIDTH=10 'Sales/Tax';
DEFINE NET / COMPUTED FORMAT=DOLLAR10.2
  WIDTH=10 'Net/Sale';
```

```
COMPUTE TAX;
  TAX= SALEAMT.SUM * .05;
ENDCOMP;
```

```
COMPUTE NET;
  NET= SALEAMT.SUM - TAX;
ENDCOMP;
```

```
COMPUTE AFTER REGION;
  LINE ' ';
  LINE 42* '*';
  LINE ' ';
  LINE '*TOTAL SALES FOR ' REGION $REGIONF. ' WERE:'
    SALEAMT.SUM DOLLAR10.2 ' *';
  LINE ' ';
  LINE 42* '*';
  LINE ' ';
ENDCOMP;
```

```
RBREAK AFTER/DOL DUL SUMMARIZE;
```

```
TITLE1 'AREA SALES FOR SELECT DEPARTMENTS';
TITLE2 'FOR THE WEEK OF 5/02/94 - 5/08/94';
RUN;
```

(see Output #4 for results)

Since we are creating customized break text, the BREAK statement has been removed. Now the COMPUTE block is controlling the entire breaking and text writing process.

Just like when PUTting literal text (anything within quotes) we are in control of the entire process, so if blanks within the literal strings are forgotten, text will run together (notice how blanks were left in the literal strings before the value of variables were written out). Similar to the feature available on PUT statement, multipliers (42* '**') can be specified which will print the character '**' forty-two times.

If a location is not specified with a pointer reference (@), SAS will center the text in the COMPUTE block if the CENTER system option is in effect.

If you happen to have a BREAK statement in effect with a SUPPRESS option specified in addition to the COMPUTE block (you shouldn't really need BREAK statement), the values for REGION will not appear on the report. If for some reason a BREAK statement is needed in addition to a COMPUTE block DO NOT USE THE SUPPRESS OPTION ON THE BREAK STATEMENT.

Notice on the output generated that the format used when writing out values on the LINE statement determines the amount of space used to write its values. Character variables will fill the space with trailing blanks, whereas numeric variables fill the space with leading blanks.

Controlling Space and Grand Totals

The trailing blanks on character variables and leading blanks on numeric variables can be controlled by using the \$VARYING. format on the LINE statement. In addition to using the \$VARYING. format, assignment statements are needed in the COMPUTE blocks to convert values and to check lengths:

```
PROC FORMAT;
  VALUE $REGIONF
    'N' = 'NORTH'
    'S' = 'SOUTH'
    'E' = 'EAST'
    'W' = 'WEST';
;RUN;

PROC REPORT DATA=SASDATA.STOREDAT NOWINDOWS
  HEADLINE HEADSKIP;

  COLUMNS REGION STORE DEPT SALEAMT TAX NET;

  DEFINE REGION / GROUP ORDER=FORMATTED WIDTH=6
    FORMAT=$REGIONF.'Region';
  DEFINE STORE / GROUP ORDER=FORMATTED 'Store';
  DEFINE DEPT / GROUP 'Dept';
  DEFINE SALEAMT / ANALYSIS SUM FORMAT=DOLLAR10.2
    WIDTH=10 'Amount/of Sale';
  DEFINE TAX / COMPUTED FORMAT=DOLLAR10.2
    WIDTH=10 'Sales/Tax';
  DEFINE NET / COMPUTED FORMAT=DOLLAR10.2
    WIDTH=10 'Net/Sale';

  COMPUTE TAX;
    TAX= SALEAMT.SUM * .05;
  ENDCOMP;

  COMPUTE NET;
    NET= SALEAMT.SUM - TAX;
  ENDCOMP;

  COMPUTE AFTER REGION;
    REGIONC=PUT(REGION,$REGIONF.);
    VARLEN1=LENGTH(REGIONC);
```

```
SALESC = LEFT(PUT(SALEAMT.SUM,DOLLAR10.2));
VARLEN2=LENGTH(SALESC);
```

```
LINE ' ';
LINE @17 42* '**';
LINE ' ';
LINE @17 '*' ' TOTAL SALES FOR '
  REGIONC $VARYING. VARLEN1 ' WERE: '
  SALESC $VARYING. VARLEN2 @58 '**';
LINE ' ';
LINE @17 42* '**';
LINE ' ';
ENDCOMP;
```

```
COMPUTE AFTER;
  SALESC = LEFT(PUT(SALEAMT.SUM,DOLLAR10.2));
  VARLEN2=LENGTH(SALESC);
```

```
LINE ' ';
LINE @17 42* '= ' ;
LINE ' ';
LINE @17 '*' ' GRAND TOTAL SALES WERE: '
  SALESC $VARYING. VARLEN2 @58 '**';
LINE ' ';
LINE @17 42* '= ' ;
LINE ' ';
ENDCOMP;
```

```
TITLE1 'AREA SALES FOR SELECT DEPARTMENTS';
TITLE2 'FOR THE WEEK OF 5/02/94 - 5/08/94';
RUN;
```

(see Output #5 for results)

Since the character variable REGION is actually a formatted value, in addition to checking its length with the LENGTH function, the format's value needs to be associated with the variable. This is done by using the PUT function. It could have been done all in one step:

```
VARLEN1=LENGTH(PUT(REGION,$REGIONF.));
```

If the character variable was a normal unformatted character variable, the PUT would not be necessary, only the LENGTH function would be required. Since the \$VARYING. format can only be used on character variables, to control the spacing on a numeric variable it must first be converted to a character using the LEFT and PUT functions and then find its length with the LENGTH function. ALL NUMERIC VARIABLES MUST BE CONVERTED TO CHARACTER, THE LENGTH STATEMENT ALONE IS NOT ENOUGH. Similarly this could have all been done in one step:

```
VARLEN2=LENGTH(LEFT(PUT(SALEAMT.SUM,DOLLAR10.2)));
```

Since the text will now vary in length (it will not be padded out with spaces after characters or before

numerics), the length of the text string will vary. Therefore, the location of the '*' literal with a pointer references will have to be fixed to a location (like @17) when writing out the text line. Once a pointer reference (@) is used anywhere within the COMPUTE block the centering option, if in effect, is turned off for the entire COMPUTE block, that is why the location of the '*' string (@17 42* '*') was also fixed. If left as it was in the last example (42* '*'), the string of asterisks would be left aligned because the centering is automatically turned off.

A grand total has also been added to the report with the COMPUTE AFTER block (note: subtotals are COMPUTE AFTER *variable* blocks). Since the grand totals are customized breaks, the RBREAK statement has been removed.

Group or Stacked Column Headings

A column can be grouped or stacked by defining it as an ACROSS variable:

```
PROC FORMAT;
  VALUE $REGIONF
    'N' = 'NORTH'
    'S' = 'SOUTH'
    'E' = 'EAST'
    'W' = 'WEST'
;
RUN;

PROC REPORT DATA=SASDATA.STOREDAT NOWINDOWS
  HEADLINE HEADSKIP SPLIT='*';

  COLUMNS REGION DEPT SALEDATE;

  DEFINE REGION / GROUP FORMAT=$REGIONF.
    ORDER=FORMATTED WIDTH=6 'Region';
  DEFINE SALEDATE / ACROSS FORMAT=MMDDYY8.
    ORDER=INTERNAL 'Date*of Sale';
  DEFINE DEPT / GROUP ORDER=FORMATTED 'Dept';

  WHERE '07MAY94'D <= SALEDATE <= '08MAY94'D;

  TITLE1 'AREA SALES FOR SELECT DEPARTMENTS';
  TITLE2 'FOR THE WEEKEND OF 5/07/94 - 5/08/94';
RUN;

      (see Output #6 for results)
```

This example subset the dates using a WHERE statement, which is supported on PROC REPORT.

Particularly with date values displayed in one of the three formats that uses slashes to separate values (such as MMDDYY8., DDMMYY8., and YYMMDD8.) it is important to use the SPLIT= option on the PROC statement to change the default split character to something other than a

'/'. If it is not changed, the date value will split to a new line each time it sees the '/' in the date value.

By default ACROSS variables will appear based on their formatted values. Therefore it is useful to specify a different value on the ORDER= option so the values will appear in the desired order.

When a variable is defined as an ACROSS variable and another variable is not specified above or below the ACROSS variable (there are no commas in the COLUMNS statement like in the next example) then the N statistic is displayed by default.

Shared Columns with an Analysis Variable

An ACROSS variable can share a column with an analysis variable by using a comma between the two variables on the COLUMN statement:

```
PROC FORMAT;
  VALUE $REGIONF
    'N' = 'NORTH'
    'S' = 'SOUTH'
    'E' = 'EAST'
    'W' = 'WEST'
;
RUN;

PROC REPORT DATA=SASDATA.STOREDAT NOWINDOWS
  HEADLINE HEADSKIP SPLIT='*';

  COLUMNS REGION DEPT SALEDATE, SALEAMT;

  DEFINE REGION / GROUP FORMAT=$REGIONF.
    ORDER=FORMATTED WIDTH=6 'Region';
  DEFINE SALEDATE / ACROSS FORMAT=MMDDYY8.
    ORDER=INTERNAL 'Date*of Sale';
  DEFINE DEPT / GROUP ORDER=FORMATTED 'Dept';
  DEFINE SALEAMT / ANALYSIS SUM FORMAT=DOLLAR10.2;

  WHERE '07MAY94'D <= SALEDATE <= '08MAY94'D;

  TITLE1 'AREA SALES FOR SELECT DEPARTMENTS';
  TITLE2 'FOR THE WEEKEND OF 5/07/94 - 5/08/94';
RUN;

      (see Output #7 for results)
```

The order in which the variables are defined on the column statement will determine how they appear on the report. The first variable specified will display above the second variable specified. A comma separates the two variables, which will create a stacking effect. In this case, the values of SALEDATE will appear above the word SALEAMT. If they were reversed on the column statement the word SALEAMT would appear above the date values on the report, which is not

as visually pleasing, but provides the same information.

Shared Columns and Multiple Analysis Variables

COMPUTED variables cannot be defined as an ACROSS variable, nor can they share a column with an analysis variable. The variable must be previously defined in a Data Step in order for it to be used as an ACROSS variable.

More than one analysis variable can be specified with an ACROSS variable on the COLUMNS statement. In this case, it produces a side-by-side comparison of SALEAMT and TAX for the specified SALEDATEs.

```
DATA STOREDAT;  
SET SASDATA.STOREDAT;  
TAX=SALEAMT*.05;  
RUN;
```

```
PROC FORMAT;  
VALUE $REGIONF  
'N' = 'NORTH'  
'S' = 'SOUTH'  
'E' = 'EAST'  
'W' = 'WEST'
```

```
;  
RUN;
```

```
PROC REPORT DATA=STOREDAT NOWINDOWS HEADLINE  
HEADSKIP SPLIT='*';
```

```
COLUMNS REGION DEPT (SALEAMT TAX),SALEDATE;
```

```
DEFINE REGION / GROUP FORMAT=$REGIONF.  
ORDER=FORMATTED WIDTH=6 'Region';
```

```
DEFINE SALEDATE / ACROSS FORMAT=MMDDYY8.  
ORDER=INTERNAL 'Date*of Sale' WIDTH=10;  
DEFINE DEPT / GROUP ORDER=FORMATTED 'Dept';  
DEFINE SALEAMT / ANALYSIS SUM FORMAT=DOLLAR10.2;  
DEFINE TAX / ANALYSIS SUM FORMAT=DOLLAR10.2;
```

```
WHERE '07MAY94'D <= SALEDATE <= '08MAY94'D;
```

```
TITLE1 'AREA SALES FOR SELECT DEPARTMENTS';  
TITLE2 'FOR THE WEEKEND OF 5/07/94 - 5/08/94';  
RUN;
```

(see Output #8 for results)

Shared columns that have variables of differing widths will use the width of the variable closest to the first row of the report. Use the WIDTH= option to properly set the width, otherwise the values will appear unformatted and increasing the format width will not help.

Defining a Variable with Multiple Statistics

There are two ways in which a variable can be defined with more than one statistic. One way is to define the associated statistics on the COLUMNS statement:

```
PROC FORMAT;  
VALUE $REGIONF  
'N' = 'NORTH'  
'S' = 'SOUTH'  
'E' = 'EAST'  
'W' = 'WEST'
```

```
;  
RUN;
```

```
PROC REPORT DATA=SASDATA.STOREDAT NOWINDOWS  
HEADLINE HEADSKIP SPLIT='*';
```

```
COLUMNS REGION DEPT (MIN MAX MEAN),SALEAMT;
```

```
DEFINE REGION / GROUP FORMAT=$REGIONF.  
ORDER=FORMATTED WIDTH=6 'Region';  
DEFINE DEPT / GROUP ORDER=FORMATTED 'Dept';  
DEFINE SALEAMT / ANALYSIS SUM FORMAT=DOLLAR10.2;
```

```
TITLE1 'AREA SALES FOR SELECT DEPARTMENTS';  
TITLE2 'FOR THE WEEK OF 5/02/94 - 5/08/94';  
RUN;
```

(see Output #9 for results)

SALEAMT is defined as an ANALYSIS variable and the statistics specified on the COLUMNS statement are telling SAS to display those statistics above the variable SALEAMT (reversing the order of the statements on the COLUMN statement would place the statistics underneath the word SALEAMT on the report).

A statistic should not be specified on the DEFINE statement. If it is, it has no effect; the statistic specified on the COLUMN statement takes precedence. A usage option is not required because the statistic defined on the COLUMN statement determines the usage.

Valid statistics that can be used are the following (these are the same statistics that can be used on the DEFINE statement for ANALYSIS variables):

CSS	NMISS	SUMWGT
CV	PRT	T
MAX	RANGE	USS
MEAN	STD	VAR
MIN	STDERR	SUM

Defining Aliases

Another way of accomplishing the same result as in the previous example is to define aliases:

```

PROC FORMAT;
  VALUE $REGIONF
    'N' = 'NORTH'
    'S' = 'SOUTH'
    'E' = 'EAST'
    'W' = 'WEST'
;
RUN;

PROC REPORT DATA=SASDATA.STOREDAT NOWINDOWS
  HEADLINE HEADSKIP SPLIT='*';

  COLUMNS REGION DEPT SALEAMT=MINSALE
    SALEAMT=MAXSALE SALEAMT=MEANSALE;

  DEFINE REGION / GROUP FORMAT=$REGIONF.
    ORDER=FORMATTED WIDTH=6 'Region';
  DEFINE DEPT / GROUP ORDER=FORMATTED 'Dept';
  DEFINE MINSALE / ANALYSIS MIN FORMAT=DOLLAR7.2
    'Lowest*Sale';
  DEFINE MAXSALE / ANALYSIS MAX FORMAT=DOLLAR7.2
    'Largest*Sale';

  DEFINE MEANSALE / ANALYSIS MEAN
    FORMAT=DOLLAR7.2 'Average*Sale';

  TITLE1 'AREA SALES FOR SELECT DEPARTMENTS';
  TITLE2 'FOR THE WEEK OF 5/02/94 - 5/08/94';
RUN;

```

(see Output #10 for results)

An advantage of defining aliases is it provides you with more control over the appearance of each of the statistics through the use of options on the DEFINE statement.

Aliases cannot be used in COMPUTE blocks, nor defined as an ACROSS variable.

In Summary

PROC REPORT is a powerful new Base SAS Procedure which provides us with more control over our report's appearance than any other existing Base SAS Proc.

We have seen in the course of this paper how PROC REPORT can create customized breaklines, create grouped or stacked column headings, and some advanced tricks were illustrated for improving the appearance of reports.

While not discussed in this paper, PROC REPORT is capable of doing some things that no other Base SAS Procedure is capable of, such as creating multi-column reports and creating percentages.

Trademark Notice

SAS is a registered trademark of the SAS Institute Inc., Cary, NC, USA and other countries.

Useful Publications

SAS Institute Inc. (1990), SAS® Guide to the Report Procedure, Usage and Reference, Version 6, First Edition, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1993), SAS® Technical Report P-258, Using the REPORT Procedure in a Nonwindowing Environment, Release 6.07, Cary, NC.: SAS Institute Inc.

Any questions or comments regarding the paper may be directed to the author:

Kim L. Kolbe Ritzow
 Systems Seminar Consultants
 Kalamazoo Office
 927 Lakeway Drive
 Kalamazoo, MI 49001
 Phone: (616) 345-6636
 Fax: (616) 345-5793