Paper 153

Advanced Features of PROC TABULATE

Thomas J. Winn Jr., Texas State Comptroller's Office, Austin, Texas

ABSTRACT

The hands-on workshop associated with this paper will provide participants with practical information regarding some advanced features of PROC TABULATE. It will include a review of the basic syntax for PROC TABULATE. It also will cover methods for customizing the appearance of tables generated by PROC TABULATE (formatting, modifying table outlining characters, removing separator lines, changing headings, and changing the width of cells), for using PROC TABULATE to compute and to display percentages and other quotients, and for using PROC TABULATE for displaying statistical results.

DATA USED FOR EXAMPLES

This paper will utilize a certain SAS data file, PRDSALE, from the Release 7.00 SASHELP data library, as well as a SAS data file created by the following SAS DATA step (representing a hypothetical used car dealer's current inventory of Saturn automobiles):

```
DATA SATURN;
    LENGTH  COLOR $10  TRANSM $9;
    INPUT YEAR MODEL $  PRICE MILEAGE
            COLOR $  TRANSM $  AC;
 DATALINES;
96  SC2  12475  23705  white  automatic  1
96  SL2   8488  55460  black   manual    1
99  SL1  14007   3015  dark_green  automatic  0
96  SL2  11048  44480  purple  automatic  0
96  SC2  11224  14433  red  manual  0
94  SC2   7884  85201  med-blue   automatic  0
97  SW2  12668  31912  white  automatic  1
95  SL1   9464  29734  blue_green automatic  1
96  SL2  11264  24424  copper  automatic  1
93  SL2  8442  49770  blue_green  automatic  0
97  SL2  12996  19830  white  automatic  1
98  SC2  13848  20004  white  automatic  0
98  SL1  12998  13454  silver  automatic  1
96  SL2  10994  67663  light_plum  manual  1
97  SW2  13628  20789  gold  automatic  1
96  SL1  9996  44084  dark_green  automatic  1
95  SL1  8588  67716  med_blue  manual  1
95  SC2  8968  80931  red  manual  0
93  SL2  7965  77004  med_blue  automatic  1
97  SC2  13998  29778  silver  automatic  1
96  SL2  12248  14543  gold  automatic  1
94  SC1  8996  57688  plum  manual  1
95  SC1  9428  49862  red  manual  1
96  SC2  11468  64401  purple  automatic  1
97  SW2  12994  17422  white  manual  0
;;;;
```

REVIEW OF ELEMENTARY CONCEPTS REGARDING PROC TABULATE

Introduction to PROC TABULATE

PROC TABULATE is used to build tabular summary reports containing descriptive statistical information, including hierarchical relationships among variables. PROC TABULATE is the SAS® System's implementation of TPL (Table Producing Language), which was developed at the U.S. Bureau of Labor Statistics during the 1970's, for generating tabular reports of descriptive statistics involving employment data. PROC TABULATE is more powerful for producing tabulations than PROC FREQ, and it is a more flexible statistical report writer than PROC MEANS.

Basic Syntax for PROC TABULATE

A general form for a PROC TABULATE step is:

```
PROC TABULATE <option-list>;
  CLASS class-variable-list;
  VAR analysis-variable-list;
  TABLE <<page-expression,>  row-expression,>
          column-expression </ table-option-list>;
  BY <NOTSORTED> <DESCENDING> variable-1
      < … <DESCENDING> variable-n>;
  FORMAT variable-list-1 format-1
           < … variable-list-n format-n>;
  FREQ variable;
  KEYLABEL keyword-1='description-1'
              < … keyword-n="description-n'>;
  LABEL variable-1='label-1' <…variable-n='label-n'>;
  WEIGHT variable;
  TITLE 'text';
```

The PROC TABULATE statement invokes the procedure and specifies certain options. The most commonly-used options are: DATA=*SAS-data-file*, which specifies the data set to be used as input by the procedure, and FORMAT=*numeric-format*, which determines the width of each cell (the intersection of a row and a column of a PROC TABULATE summary table) in the table. The MISSING option requests that missing values be regarded as valid levels for classification variables. Unless the MISSING option is specified, observations with missing values for class variables will not be included in the analysis. The NOSEPS option removes the interior horizontal lines from the printed report.

It is a requirement that every PROC TABULATE step must include a PROC TABULATE statement, either a CLASS statement or a VAR statement, or both, and a TABLE statement. Each variable appearing in a TABLE statement must be mentioned in either the CLASS statement or the VAR statement. Variables may not be mentioned in both the CLASS and the VAR statements.

Coding PROC TABULATE

Before writing any SAS code for PROC TABULATE, the programmer needs to decide what the final report should look like. It is essential to have the fundamental design of the pages, row, and columns of the table clearly in mind. This is the vision which will determine the particular code to be used.

The following is an example of a table generated by PROC TABULATE from the SATURN data set:

		TRANSM		
		automatic	manual	All
		N	N	N
YEAR	MODEL			
93	SL2	2	0	2
94	SC1	0	1	1
	SC2	1	0	1
95	SC1	0	1	1
	SC2	0	1	1
	SL1	1	1	2
96	SC2	2	1	3
	SL1	1	0	1
	SL2	3	2	5
97	SC2	1	0	1
	SL2	1	0	1
	SW2	2	1	3
98	SC2	1	0	1
	SL1	1	0	1
99	SL1	1	0	1
All		17	8	25

The simplest way to approach coding for PROC TABULATE is in terms of the following five steps:

- Specification of classification variables,
- Specification of analysis variables,
- Definition of dimensions of table,
- Identification of desired statistics, and
- Labeling and formatting the table.

For a complete explanation of this approach, see the papers by Virgile (1997) or Winn (1998), noted in the **References** at the end of this paper.

Step 1 -- Specification of Classification Variables

The CLASS statement is used to specify any categorical variables which will be used for grouping purposes in the analysis. These variable-names will be the page, row, and column headers of the table. In most cases, the CLASS statement is required.

```
PROC TABULATE DATA=SATURN;
    CLASS YEAR MODEL TRANSM; . . .
```

Step 2 -- Specification of Analysis Variables

The VAR statement is used to list any variables which will be used for computing the statistics which are intended to appear in the body of the table. Frequency counts can be computed without an analysis variable. However, under most conditions, the VAR statement is required.

```
PROC TABULATE DATA=SATURN;
    CLASS YEAR MODEL TRANSM;
     VAR PRICE MILEAGE; . . .
```

Step 3 -- Definition of the Table's Dimensions

The TABLE statement is used to define both the arrangement of the rows and columns of the table, as well as the requests for any summary statistics. It is the tricky aspect of using PROC TABULATE. So let's consider this complex statement one piece at a time.

In a TABLE statement, the comma is a very important symbol, because it separates the dimensions of the table. If two commas were specified, then the table would have three dimensions, and the order would be pages, rows, and columns. If only one comma was specified, then the table would have two dimensions, and the order would be rows, columns. No comma would be interpreted to mean that the table's only dimension would be the column dimension. The table would only have one row.

So, TABLE statements look like the following:

```
TABLE page-expression, row-expression, column-
expression . . .;
TABLE row-expression, column-expression . . .;     or
TABLE column-expression . . .;
```

In this context, an expression can consist of variables, statistics, operators, format specifications, and label assignments. Only three operators are needed to specify the page, row and column headings which identify the structure of a table.

An asterisk (*) can be used to cross the classification variables; that is, to arrange them in a *nested* manner, according to the order listed (top, middle, and lower). A blank space is used to *concatenate* two classification variables (which will appear in the table: top-to-bottom for row headings, left-to-right for column headings). Parentheses ( ) are used to group the elements of an expression, and to associate an adjacent operator with each concatenated element inside the parentheses.

Step 4 -- Identification of Desired Statistics

Up to this point, we have only focused on the dimensions of the tables, and our TABLE statements have been incomplete. The TABLE statement also specifies which summary statistics should be produced, and pertaining to which analysis variables. Each statistic is identified by a keyword.

```
N = the number of observations, the frequency count,
MIN = the smallest value,
MAX = the largest value,
MEAN = the arithmetic mean, or the average,
STD = the standard deviation,
VAR = the variance,
SUM = the sum of the values,
PCTN = the percentage that one frequency is of another
frequency,
```

PCTSUM = the percentage that one sum is of another sum,
. . . and other descriptive statistics.
(Note – PROC TABULATE in Version 7 of the SAS System has numerous statistics not available in previous releases.)

Whenever you cross a variable with a keyword for a statistic, you are identifying the statistic to be applied to that variable (which tells PROC TABULATE what type of calculation to perform). You can cross class variables only with the N or PCTN statistics. By default, if the TABLE statement does not include an analysis variable or a statistic, then PROC TABULATE automatically crosses the N statistic with the indicated class variables. Analysis variables can be crossed with any statistic. By default, if the TABLE statement includes an analysis variable but without crossing it with any statistic, PROC TABULATE automatically crosses it with SUM.

Here are some examples:

```
PROC TABULATE  DATA=SATURN;
   CLASS YEAR MODEL TRANSM;
   VAR PRICE;
   TABLE YEAR*MODEL, TRANSM;
   TABLE YEAR*MODEL, TRANSM*PCTN;
   TABLE YEAR*MODEL, TRANSM*PRICE*MEAN;
   TABLE YEAR*MODEL, TRANSM*PRICE*PCTSUM;
```

PROC TABULATE has a universal class variable, ALL, which can be used to generate totals for any specified class variable. Just concatenate the keyword ALL into the row or column expression of a TABLE statement. Whenever the ALL is concatenated with other elements in a dimension (page, row, or column), PROC TABULATE summarizes all of the categories within that dimension. Moreover, whenever the ALL is used within a grouping defined by parentheses, PROC TABULATE will summarize only the categories within the grouping.

```
PROC TABULATE  DATA=SATURN;
     CLASS YEAR MODEL TRANSM;
     VAR PRICE;
     TABLE YEAR*MODEL  ALL,  N  PRICE*MEAN;
     TABLE YEAR*MODEL ALL,  PRICE*PCTSUM;
     TABLE (YEAR ALL)*MODEL,
                (TRANSM ALL)*(N  PRICE*MEAN);
```

Step 5 -- Labeling and Formatting the Table

Now that we know how to define the basic structure of the tables we will generate, we would like to be able to make the tables more self-explanatory; that is, easier to read and interpret.

As in many other SAS procedures, you can use a LABEL statement to replace variable names with more descriptive headings for your class variables.

```
PROC TABULATE  DATA=SATURN;
     CLASS YEAR MODEL TRANSM;
     VAR MILEAGE;
     LABEL MILEAGE='Odometer Reading';
     TABLE YEAR*MODEL  ALL,  N  MILEAGE*MEAN;
```

There also is a way to specify temporary labels in a TABLE statement.

```
PROC TABULATE  DATA=SATURN;
     CLASS YEAR MODEL TRANSM;
          VAR MILEAGE;
     TABLE YEAR*MODEL  ALL='All Autos Combined',
               N  MILEAGE='Odometer Reading'*MEAN;
```

To assign labels to procedure-generated statistics and the universal class variable, we use the KEYLABEL statement.

```
KEYLABEL  N = 'Count'
          ALL = 'Total'
          PCTN = 'Percent';
```

To suppress the label for a procedure-generated statistic, use a statement like the following:

```
KEYLABEL N = '   ';
```

You can use the FORMAT= option of the PROC TABULATE statement and the F= format modifiers in the TABLE statement to format the contents of table cells, and to set the width of a column.

TITLE and FOOTNOTE statements also can be used to enhance the tabular reports generated by PROC TABULATE, in the same manner as these statements are used in other SAS report-writing procedures.

COMPUTING AND DISPLAYING PERCENTAGES

Using PROC TABULATE to compute percentages is a bit tricky. PCTN and PCTSUM are the keywords which specify, respectively, the percentage of the value in a particular cell to the value in another cell, or to the total of a group of cells. Whenever someone performs percentage calculations, it is essential to be very clear concerning which number is to be used as the *denominator* in the computation. PROC TABULATE allows a programmer to identify the specific denominator to be used in computing percentages.

By default, if a denominator is not explicitly defined in a TABLE statement which includes percentages, then PROC TABULATE uses the total of the N-frequency-counts for the entire data set as the denominator in a PCTN calculation, and the total of all of the SUM cells for the denominator in a PCTSUM calculation – in each case, the percentage is computed based upon *all* of the observations in the data set. If that is what a programmer intends, then specifying PCTN or PCTSUM in a TABLE statement isn't much different than requesting any other summary statistic. In that case, you wouldn't need to define the denominator, because the default denominator is implicitly assumed (see the examples, above, or the first two of the examples which follow).

However, whenever it is necessary to use a particular total as the denominator in a PROC TABULATE percentage calculation, then a *denominator definition* must be specified in the TABLE statement. To specify how the percentage is to be calculated, the "less-than" (<) and "greater-than" (>) inequality signs are used as brackets to enclose an expression which tells PROC TABULATE how to identify the appropriate denominator. Now, this probably doesn't work the way you might think that it should, so pay close attention to the examples, below.

Here are some examples involving percentage calculations:

```
PROC TABULATE  DATA=SATURN;
     CLASS  YEAR MODEL TRANSM;
     VAR  PRICE;
     LABEL  TRANSM = 'Transmission';
     KEYLABEL  N='Count'
               ALL='Total'
               SUM='$'
               PCTN='% of Autos'
               PCTSUM='% of $';
     TABLE  YEAR*MODEL ALL,
            (TRANSM ALL)*(N*F=5.0  PCTN*F=6.2);
     TABLE  MODEL  ALL,
            (TRANSM ALL)*PRICE*(SUM PCTSUM);
     TABLE  YEAR*MODEL, TRANSM
             *(N*F=5.0 PCTN<YEAR*MODEL>*F=6.2);
     TABLE  YEAR*MODEL, TRANSM
             *(N*F=5.0 PCTN<TRANSM>*F=6.2);
     TABLE  YEAR*MODEL,
            TRANSM*PRICE
             *(SUM PCTSUM<YEAR*MODEL>*F=6.2);
     TABLE  YEAR*MODEL,
            TRANSM*PRICE
             *(SUM PCTSUM<TRANSM>*F=6.2);
```

The first two TABLE statements do not contain an explicit *denominator definition*, therefore the percentage calculations are carried out with respect to the entire data set. The first TABLE statement would generate a breakdown of the counts and frequency percentages for each combination of values of YEAR, MODEL, and TRANSM, where the percentages are calculated with respect to the entire data set. The second TABLE statement produces a breakdown of the sum of the values of PRICE and percentage of the sum of PRICE, for each combination of values of YEAR, MODEL, and TRANSM, and where the percentages are calculated with respect to the entire data set.

The third TABLE statement would generate breakdowns of the counts and the associated frequency percentages, for each combination of values of YEAR, MODEL, and TRANSM, where the percentages are calculated in a column-wise manner (using the combined counts for each value of TRANS, across all values of combinations of YEAR and MODEL, as the denominator for all cells in that column). The fourth TABLE statement would generate breakdowns of the counts and the associated frequency percentages, for each combination of values of YEAR, MODEL, and TRANSM, where the percentages are calculated in a row-wise manner (using the total counts for each particular combination of YEAR and MODEL, across all values of TRANSM, as the denominator for all cells in that row).

The fifth TABLE statement would generate breakdowns of the sum of the values of PRICE and percentage of the sum of PRICE, for each combination of values of YEAR, MODEL, and TRANSM, where the percentages are calculated in a column-wise manner (using the total of PRICE for each value of TRANS, across all values of combinations of YEAR and MODEL, as the denominator for all cells in that column). Finally, the sixth TABLE statement would generate breakdowns of the sum of the values of PRICE and percentage of the sum of PRICE, for each combination of values of YEAR, MODEL, and TRANSM, where the percentages are calculated in a row-wise manner (using the total of PRICE for each particular combination of YEAR and MODEL, across all values of TRANSM, as the denominator for all cells in that row).

Observe that, to obtain percentages by row, we use the column-expression in the "denominator definition"; and to obtain percentages by column, we use the row-expression in the "denominator definition".

Notice that PROC TABULATE allows the use of denominator definitions which contain crossings. However, PROC TABULATE does not permit denominator definitions which contain groupings which are defined by the use of parentheses.

Following are a couple of examples which illustrate the use of the universal class variable, ALL, in a denominator definition for percentage calculations. Notice that whenever row- or column-percentages are to be produced for a column- or row-expression which include ALL, then ALL also must be included in the denominator definition.

```
PROC TABULATE  DATA=SATURN;
     CLASS  YEAR MODEL TRANSM;
     VAR  PRICE;
     LABEL  TRANSM = 'Transmission';
     KEYLABEL  N='Count'
               ALL='Total'
               SUM='$'
               PCTN='% of Autos'
               PCTSUM='% of $';
     TABLE  YEAR*MODEL  ALL,  (TRANSM  ALL)
          *(N*F=5.0 PCTN<YEAR*MODEL ALL>*F=6.2);
     TABLE  YEAR*MODEL  ALL,
            (TRANSM ALL)*PRICE
              *(SUM PCTSUM<TRANSM ALL>*F=6.2);
```

CUSTOMIZING THE APPEARANCE OF TABLES

PROC TABULATE provides several ways for customizing the appearance of tables. We have already discussed some of the ways of changing the default headings which are used in the tables. You also may want to use formats to change the headings which correspond to values of a class variable.

The default for displaying cells with missing numeric values is a period. You can change the way missing values are displayed by using the MISSTEXT= option to define up to twenty characters of text which will print in the table cells whenever a particular combination of class variable values is not found in the input data set.

```
PROC TABULATE  DATA=SATURN;
     CLASS YEAR MODEL TRANSM;
     TABLE YEAR*MODEL ALL,
           TRANSM='Transmission'  ALL
            / MISSTEXT='0';
     TITLE 'Saturn Automobiles in Used Car Inventory';
```

Formats can be used to substitute labels for values of the classification variables. Formats also can be used to combine many values of the classification variables into a much smaller number of values to be printed in the report. We create custom formats by using PROC FORMAT, and we invoke those formats in PROC TABULATE either through a FORMAT statement, or by crossing F=*format-name.* in the TABLE statement with the particular variable.

```
PROC FORMAT;
   VALUE  ODOMFMT   low -9999 = 'less than 10,000'
                    10000-34999 = '10,000 to 34,999'
                    35000-59999 = '35,000 to 59,999'
                      60000-high = 'at least 60,000';

PROC TABULATE  DATA=SATURN  FORMAT=8.0;
     CLASS YEAR MODEL MILEAGE;
     KEYLABEL N='Number of Autos'
              ALL='All Autos Combined';
     FORMAT MILEAGE ODOMFMT.;
     TABLE YEAR*MODEL  ALL,
           MILEAGE='Odometer Reading'  ALL;
     TITLE 'Saturn Automobiles in Used Car Inventory';
```

The RTSPACE= (or RTS=) option defines the total amount of space for the row headings. If there are several levels of headings for row, then the space is divided equally among the levels, after subtracting the spaces which are needed for the vertical lines.

Whenever a table produced by PROC TABULATE is too wide to fit on a single page, the procedure automatically splits the table, to span as many separate pages as are necessary for printing. For short, wide tables, the CONDENSE option could be specified on the TABLE statement, in order to print as many *logical pages* as possible on a single page, one below the other.

The FORMCHAR= option in the PROC TABULATE statement may be used to change the characters which are used for outlining the table. To replace all of the outlining characters with blanks, use the option

FORMCHAR(1,2,3,4,5,6,7,8,9,10,11)=' '

(that is, 11 blanks, enclosed in quotes).

USING PROC TABULATE FOR DISPLAYING STATISTICAL RESULTS

Sometimes we may desire to produce tables which include statistics which are not calculated by PROC TABULATE, or we may want to combine several statistics in a single table. In general, either of these situations will require the use of a statistical PROC which is capable of writing its results to an output DATA set (such as PROC MEANS, PROC SUMMARY, PROC UNIVARIATE, or PROC CORR), and/or an intermediate DATA step which restructures the data into a form which PROC TABULATE is able to utilize.

Consider the SAS data file, PRDSALE, from the SASHELP data library. PRDSALE contains 1,440 observations, including predicted and actual sales data for various products, divisions, and locations, covering the period of January 1993 through December 1994. Here is some information from the PROC CONTENTS output for this data set:

#	Variable	Type	Len	Pos	Format	Label
1	ACTUAL	Num	8	0	DOLLAR12.2	Actual Sales
3	COUNTRY	Char	10	40	$CHAR10.	Country
5	DIVISION	Char	10	60	$CHAR10.	Division
10	MONTH	Num	8	32	MONNAME3.	Month
2	PREDICT	Num	8	8	DOLLAR12.2	Predicted Sales
6	PRODTYPE	Char	10	70	$CHAR10.	Product type
7	PRODUCT	Char	10	80	$CHAR10.	Product
8	QUARTER	Num	8	16	8.	Quarter
4	REGION	Char	10	50	$CHAR10.	Region
9	YEAR	Num	8	24	4.	Year

Let us suppose that we are interested in examining the accuracy of the predictions, by computing the percentage error [PCTERR = 100*(PREDICT-ACTUAL)/ACTUAL] for each value of PRODUCT, and in each month. This statistic is not calculated by PROC TABULATE. We could calculate it in a DATA step, or with some combination of a PROC step and a DATA step, and then use TABULATE to write the report; or we could "trick" PROC TABULATE into doing most of the work anyway, by using an intermediate DATA step just before a PROC TABULATE step with a carefully coded TABLE statement. Let us explore both of these approaches.

```
PROC SUMMARY DATA=SASHELP.PRDSALE
          NOPRINT NWAY;
  CLASS  PRODUCT YEAR MONTH;
  VAR ACTUAL PREDICT;
  OUTPUT OUT=SUMRY SUM=;

DATA SUMRY;
     SET SUMRY;
     PCTERR = 100*(PREDICT-ACTUAL)/ACTUAL;

PROC TABULATE DATA=SUMRY;
     CLASS PRODUCT YEAR MONTH;
     VAR PREDICT ACTUAL PCTERR;
     KEYLABEL SUM=' ';
     TABLE  PRODUCT, YEAR*MONTH,
              PREDICT  ACTUAL  PCTERR;
     TITLE 'Accuracy of Sales Predictions in'
           ' SASHELP.PRDSALE Data Set';
```

Here is another approach to the same problem:

```
DATA SALES;
     SET SASHELP.PRDSALE;
          ERR100 = (PREDICT-ACTUAL)/100;

PROC TABULATE DATA=SALES  FORMAT=9.0;
     CLASS PRODUCT YEAR MONTH;
     VAR PREDICT ACTUAL ERR100;
     KEYLABEL SUM='  '
                PCTSUM='  ';
     LABEL ERR100 = 'Percent Error';
     TABLE PRODUCT, YEAR*MONTH,
          PREDICT  ACTUAL
          ERR100*(PCTSUM<ACTUAL>)
              *F=PERCENT9.3;
     TITLE 'Accuracy of Sales Predictions in'
           ' SASHELP.PRDSALE Data Set';
```

The preceding example also illustrates something else about PROC TABULATE. Strictly speaking, TABULATE does not compute "pure" quotients of sums; however, it does calculate the percentage that one sum represents of another sum (PCTSUM). Therefore, if we want to use PROC TABULATE to compute quotients of sums, then pre-divide the summands of the numerator sums by 100, and then use PROC TABULATE to cross the numerator sums with PCTSUM, where the *denominator definition* is the analysis variable for the intended denominator sums. The reason for pre-dividing by 100 is to ensure that the decimal point is correctly positioned in the quotient.

We have already noted that, in a TABLE statement, the dimensions of the table are separated by commas. If all of the variables to be used in the table occur in the same dimension, and if there is no comma, then PROC TABULATE interprets the TABLE statement to mean that the table's only dimension

would be the column dimension. So, the table would only have one row, and the table would be a *horizontal* display, with each variable occupying its own column. Well, suppose that a *vertical* display is desired. Maybe you will be directed to prepare a tabular report in which various statistics are to be arranged in a single column. What then? A simple illustrative example follows.

Let us aggregate the PRDSALE data according to product, using a separate variable for the sales corresponding to each value of PRODUCT.

```
DATA PRODS;
     SET SASHELP.PRDSALE;
     KEEP YEAR MONTH BEDS CHAIRS
          DESKS SOFAS TABLES;
     IF PRODUCT = 'BED' THEN BEDS = ACTUAL;
        ELSE IF PRODUCT = 'CHAIR'
             THEN CHAIRS = ACTUAL;
        ELSE IF PRODUCT = 'DESK'
             THEN DESKS = ACTUAL;
        ELSE IF PRODUCT = 'SOFA'
             THEN SOFAS = ACTUAL;
        ELSE IF PRODUCT = 'TABLE'
             THEN TABLES = ACTUAL;

PROC TABULATE DATA=PRODS;
     VAR BEDS CHAIRS DESKS SOFAS TABLES;
     KEYLABEL SUM='$';
     TABLE  (BEDS CHAIRS DESKS
              SOFAS TABLES)*SUM, ALL = ' ';
      TITLE 'Total Sales for Each Type of Product'
            ' in SASHELP.PRDSALE Data Set';
```

In the example, the combined (over all time periods in the data set) sales for each of the product types are crossed with ALL= ' '. This produces the desired vertical display, instead of the horizontal arrangement which one might expect.

VERSION 7 ENHANCEMENTS

Release 7.00 of the SAS System includes several enhancements concerning PROC TABULATE. Using the CLASSDATA= option, you can specify which combinations of class variables are to be included in the analysis. Now, you also can name an output data set from PROC TABULATE. PROC TABULATE has an expanded set of available statistics, including quartiles, and the most commonly-requested percentiles. Version 7 includes the new Output Delivery System, which allows you to create customized HTML files from PROC TABULATE.

CONCLUSION

PROC TABULATE is a very useful and very powerful procedure for constructing tabular reports containing descriptive statistical information, including hierarchical relationships among variables. It is well-worth the necessary investment of time and effort for learning the intricacies and subtleties of its syntax.

REFERENCES:

- Jeffery M. Abolafia & Stephen M. Noga, "The TABULATE Procedure: One Step Beyond the Final Chapter",
  - SESUG '97 Proceedings (1997), pp. 293-300; and
  - Proceedings of the Twenty-Third Annual SAS Users Group International Conference (1998), pp. 839-844.
- Dan Bruns, "The Utter Simplicity of the TABULATE Procedure",
  - Proceedings of the Sixteenth Annual SAS Users Group International Conference (1991), pp. 365-371; and
  - Proceedings of the Twentieth Annual SAS Users Group International Conference (1995), pp. 363-368; and
  - Proceedings of the Seventh Annual South-Central SAS Users' Conference (1997), pp. 54-60.
- Dan Bruns, "The Utter 'Simplicity?' of the TABULATE Procedure", Proceedings of the Seventeenth Annual SAS Users Group International Conference (1992), pp. 216-220.
- Dan Bruns, "Advanced Features of PROC TABULATE -- or -- The Utter Simplicity of the TABULATE Procedure - The Sequel", Proceedings of the Twenty-First Annual SAS Users Group International Conference (1996), pp. 242-247.
- Dan Bruns, "The Utter 'Simplicity?' of the TABULATE Procedure -- The Final Chapter",
  - Proceedings of the Twenty-Second Annual SAS Users Group International Conference (1997), pp. 251-256 ; and
  - Proceedings of the Seventh Annual South-Central SAS Users' Conference (1997), pp. 61-66.
- Michele M. Burlew, "A Stepwise Approach to Using PROC TABULATE", Chapter 5 of Reporting From the Field (SAS Institute Inc., 1994), pp. 67-88.
- SAS Institute Inc., "The TABULATE Procedure", Chapter 37 of SAS Procedures Guide, Version 6, Third Edition (1990).
- SAS Institute Inc., SAS Guide to TABULATE Processing, Second Edition (1990).
- SAS Institute Inc., "Summary Reports -- PROC TABULATE", Chapter 5 of SAS Views: SAS Report Writing (1987).
- Bob Virgile, "The Right Approach to Learning PROC TABULATE", SESUG '97 Proceedings (1997), pp. 189-195.
- Tom Winn, "Introduction to Using PROC TABULATE", Proceedings of the Eighth Annual South-Central SAS Users' Conference (1998), pp.316-326.

SAS is a registered trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

AUTHOR INFORMATION:

Thomas J. Winn Jr.
Fiscal Management Support,
Comptroller of Public Accounts
L.B.J. State Office Building
111 E 17th Street
Austin, TX 78774

Telephone: (512) 463-4907
E-Mail: tom.winn@cpa.state.tx.us