Paper 134-2013

# Tips for Generating Percentages Using the SAS® TABULATE Procedure

Kathryn McLawhorn, SAS Institute Inc.

## ABSTRACT

PROC TABULATE is one of the few Base SAS® procedures that calculate percentages. The procedure is unique in that it has many default statistics for generating percentages and also provides the ability to customize denominator definitions. Determining the right denominator definition is an important but often challenging aspect of calculating percentages.

Written for intermediate users, this paper discusses techniques for enhancing PROC TABULATE output with percentage statistics. Using examples, the paper illustrates how to add standard percentages to the PROC TABULATE output, and it shares tips for calculating percentages that you might have thought not possible. The paper also illustrates how to avoid common pitfalls that are related to structuring denominator definitions and how to format table output.

## INTRODUCTION

In the earliest SAS release, PROC TABULATE included only the PCTN and PCTSUM statistics for calculating percentages. However, determining the correct denominator definition to get the desired percentage was tricky. The statistics introduced in a subsequent release (SAS® 7) simplified some common percentage requests. These statistics include COLPCTN, COLPCTSUM, PAGEPCTN, PAGEPCTSUM, REPPCTN, REPPCTSUM, ROWPCTN, and ROWPCTSUM.

This paper provides examples of standard percentage requests, including some challenging requests in which denominator definitions are required. Examples also show special percentage requests that require manipulating the data in advance. Included are tips for avoiding common mistakes when you construct denominator definitions. Finally, you learn techniques for applying formatting to enhance percentage statistics in PROC TABULATE output.

Note that most of the examples in this paper use the CARS data set that is found in the SASHELP library. For all examples, the following statements are submitted at the beginning of the code:

```
options nocenter nodate nonumber linesize=100;
title;
```

## STANDARD PERCENTAGE REQUESTS

Before you can learn how to enhance your PROC TABLUATE output, you first need to understand how to use the default statistics to calculate percentages. You can then build on that knowledge by learning how to construct custom denominator definitions. It is also helpful to know how percentage statistics are named in an output data set that is created by PROC TABULATE.

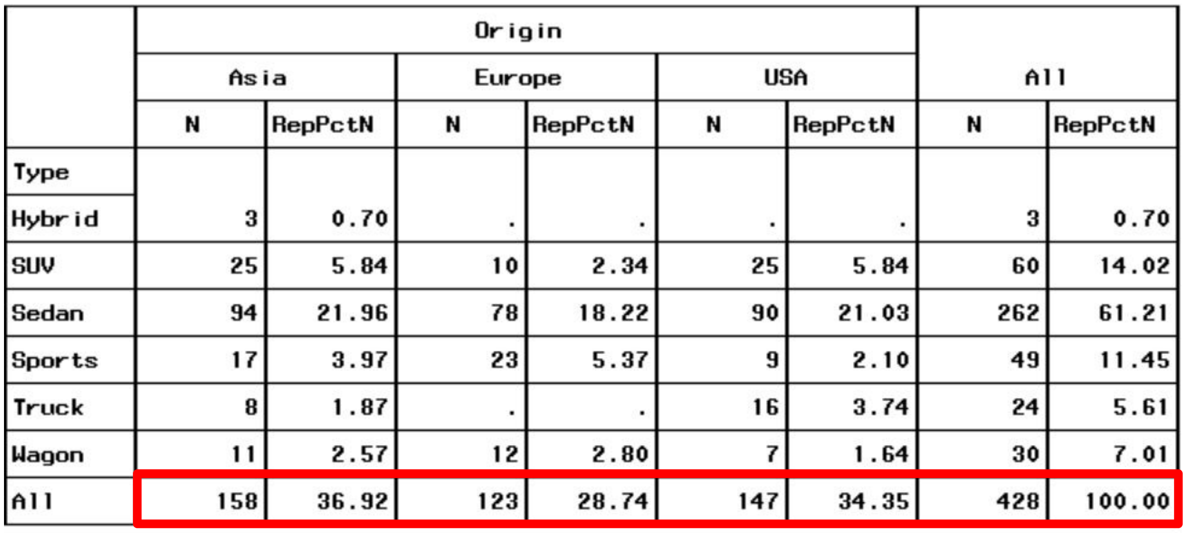
### UNDERSTANDING AND USING DEFAULT PERCENTAGE STATISTICS

The PCTN family of statistics includes COLPCTN, PAGEPCTN, REPPCTN, and ROWPCTN. The percentages that are generated from these statistics are based on the frequency count (*N*) of the whole report or on the specific page, column, or row. The PCTSUM family of statistics includes PCTSUM, PAGEPCTSUM, COLPCTSUM, REPPCTSUM, and ROWPCTSUM. The percentages that are generated from these statistics are based on summarized values (*SUM*) within the whole report or on the specific page, column, or row. The PCTSUM family of statistics requires nesting with a numeric analysis variable. The examples in the following sections show you how to calculate percentages with some of these statistics.

#### The REPPCTN and REPPCTSUM Statistics

The REPPCTN and REPPCTSUM statistics generate the same results as the PCTN and PCTSUM statistics without a denominator definition. They print the percentage of the value in a single table cell in relation to the total of the values in the report. The REPPCTN statistic is used in the following example. The resulting table is shown in Output 1.

```
proc tabulate data=sashelp.cars format=8.2;
   class origin type;
   table type all, (origin all)*(n*f=8. reppctn) / rts=10;
run;
```



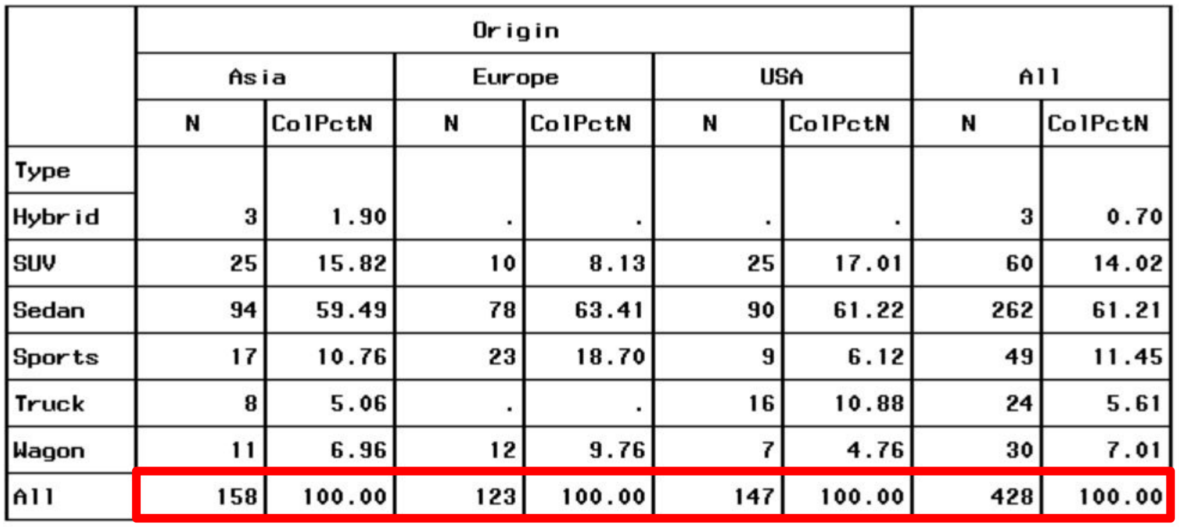
| | Origin | | | | | | All | |
|---|---|---|---|---|---|---|---|---|
| | Asia | | Europe | | USA | | | |
| | N | RepPctN | N | RepPctN | N | RepPctN | N | RepPctN |
| Type | | | | | | | | |
| Hybrid | 3 | 0.70 | . | . | . | . | 3 | 0.70 |
| SUV | 25 | 5.84 | 10 | 2.34 | 25 | 5.84 | 60 | 14.02 |
| Sedan | 94 | 21.96 | 78 | 18.22 | 90 | 21.03 | 262 | 61.21 |
| Sports | 17 | 3.97 | 23 | 5.37 | 9 | 2.10 | 49 | 11.45 |
| Truck | 8 | 1.87 | . | . | 16 | 3.74 | 24 | 5.61 |
| Wagon | 11 | 2.57 | 12 | 2.80 | 7 | 1.64 | 30 | 7.01 |
| All | 158 | 36.92 | 123 | 28.74 | 147 | 34.35 | 428 | 100.00 |

**Output 1. Output Showing the Results Generated by the REPPCTN Statistic**

## The COLPCTN and COLPCTSUM Statistics

The COLPCTN and COLPCTSUM statistics print the percentage of the value in a single table cell in relation to the total of the values in the column. The COLPCTN statistic is used in the following example. The resulting table is shown in Output 2.

```
proc tabulate data=sashelp.cars format=8.2;
   class origin type;
   table type all, (origin all)*(n*f=8. colpctn) / rts=10;
run;
```



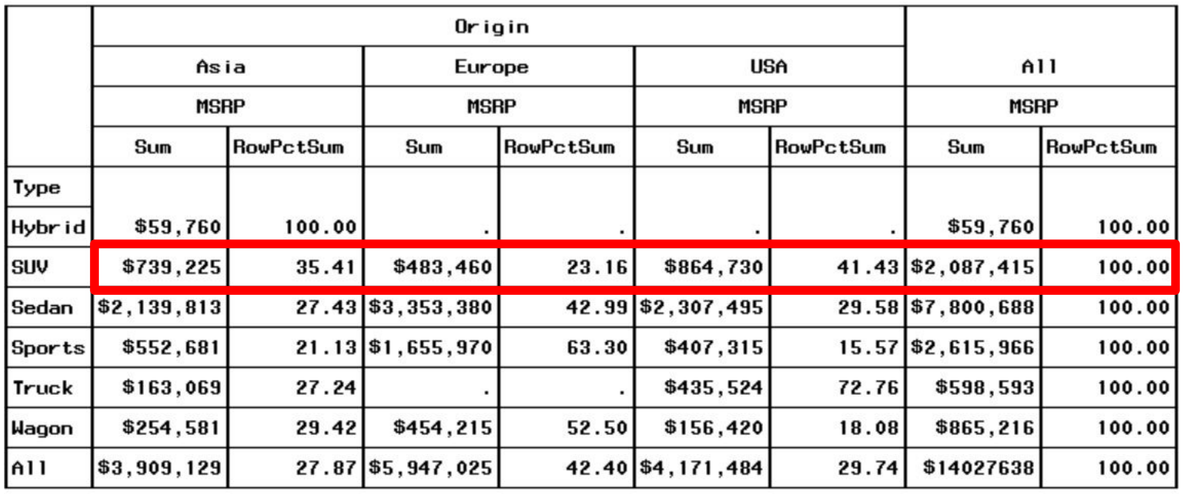
| | Origin | | | | | | All | |
|---|---|---|---|---|---|---|---|---|
| | Asia | | Europe | | USA | | | |
| | N | ColPctN | N | ColPctN | N | ColPctN | N | ColPctN |
| Type | | | | | | | | |
| Hybrid | 3 | 1.90 | . | . | . | . | 3 | 0.70 |
| SUV | 25 | 15.82 | 10 | 8.13 | 25 | 17.01 | 60 | 14.02 |
| Sedan | 94 | 59.49 | 78 | 63.41 | 90 | 61.22 | 262 | 61.21 |
| Sports | 17 | 10.76 | 23 | 18.70 | 9 | 6.12 | 49 | 11.45 |
| Truck | 8 | 5.06 | . | . | 16 | 10.88 | 24 | 5.61 |
| Wagon | 11 | 6.96 | 12 | 9.76 | 7 | 4.76 | 30 | 7.01 |
| All | 158 | 100.00 | 123 | 100.00 | 147 | 100.00 | 428 | 100.00 |

**Output 2. Output Showing the Results Generated by the COLPCTN Statistic**

## The ROWPCTN and ROWPCTSUM Statistics

The ROWPCTN and ROWPCTSUM statistics print the percentage of the value in a single table cell in relation to the total of the values in the row. The ROWPCTSUM statistic is used in the following example. The resulting table is shown in Output 3.

```
proc tabulate data=sashelp.cars format=10.2;
   class origin type;
   var msrp;
   table type all, (origin all)*msrp*(sum*f=dollar10. rowpctsum) / rts=8;
run;
```



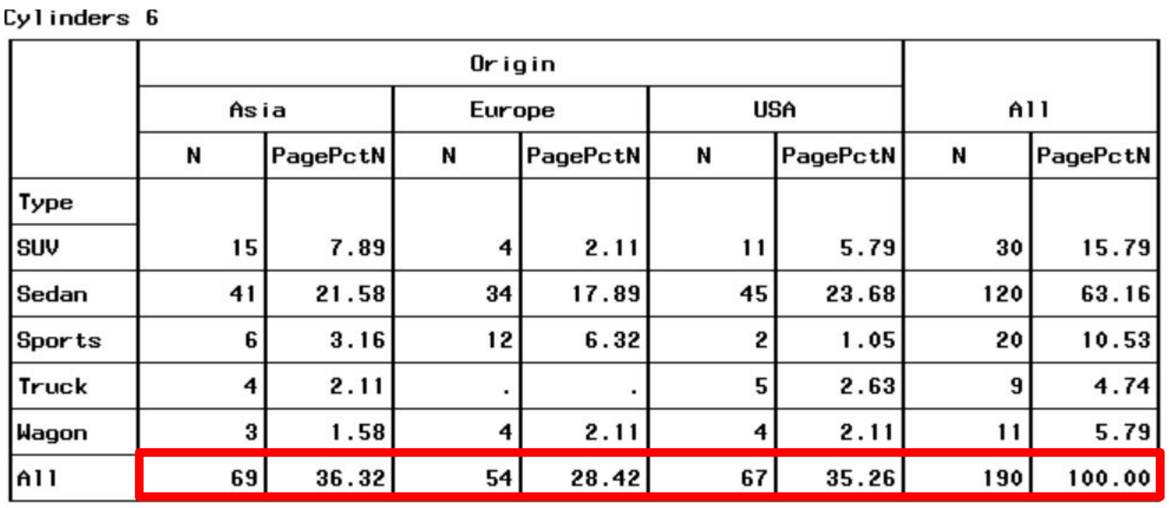
| | Origin | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Asia | | Europe | | USA | | All | |
| | MSRP | | MSRP | | MSRP | | MSRP | |
| | Sum | RowPctSum | Sum | RowPctSum | Sum | RowPctSum | Sum | RowPctSum |
| Type | | | | | | | | |
| Hybrid | $59,760 | 100.00 | . | . | . | . | $59,760 | 100.00 |
| SUV | $739,225 | 35.41 | $483,460 | 23.16 | $864,730 | 41.43 | $2,087,415 | 100.00 |
| Sedan | $2,139,813 | 27.43 | $3,353,380 | 42.99 | $2,307,495 | 29.58 | $7,800,688 | 100.00 |
| Sports | $552,681 | 21.13 | $1,655,970 | 63.30 | $407,315 | 15.57 | $2,615,966 | 100.00 |
| Truck | $163,069 | 27.24 | . | . | $435,524 | 72.76 | $598,593 | 100.00 |
| Wagon | $254,581 | 29.42 | $454,215 | 52.50 | $156,420 | 18.08 | $865,216 | 100.00 |
| All | $3,909,129 | 27.87 | $5,947,025 | 42.40 | $4,171,484 | 29.74 | $14027638 | 100.00 |

**Output 3. Output Showing the Results Generated by the ROWPCTSUM Statistic**

### The PAGEPCTN and PAGEPCTSUM Statistics

The PAGEPCTN and PAGEPCTSUM statistics print the percentage of the value in a single table cell in relation to the total of the values in the page. The PAGEPCTN statistic is used in the following example. The resulting table is shown in Output 4.

```
proc tabulate data=sashelp.cars format=8.2;
   class cylinders origin type;
   table cylinders, type all, (origin all)*(n*f=8. pagepctn) / rts=10;
run;
```



Cylinders 6

| | Origin | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Asia | | Europe | | USA | | All | |
| | N | PagePctN | N | PagePctN | N | PagePctN | N | PagePctN |
| Type | | | | | | | | |
| SUV | 15 | 7.89 | 4 | 2.11 | 11 | 5.79 | 30 | 15.79 |
| Sedan | 41 | 21.58 | 34 | 17.89 | 45 | 23.68 | 120 | 63.16 |
| Sports | 6 | 3.16 | 12 | 6.32 | 2 | 1.05 | 20 | 10.53 |
| Truck | 4 | 2.11 | . | . | 5 | 2.63 | 9 | 4.74 |
| Wagon | 3 | 1.58 | 4 | 2.11 | 4 | 2.11 | 11 | 5.79 |
| All | 69 | 36.32 | 54 | 28.42 | 67 | 35.26 | 190 | 100.00 |

**Output 4. Output Showing the Results Generated by the PAGEPCTN Statistic**

## CONSTRUCTING A CUSTOM DENOMINATOR DEFINITION

The default percentage statistics enable you to obtain basic calculations, but they might not give you the percentage that you actually need. For example, you want a percentage that reflects a subtotal in the table. In this case, PROC TABULATE gives you the ability to build your own denominator definition.

The denominator definition uses PCTN or PCTSUM as a starting point. Then, in angle brackets (<>) after the statistic name, you specify an expression that tells PROC TABULATE which values should be used to calculate the denominator for the percentage that you request. These expressions can include the following:

- a single variable (a class variable or an analysis variable)
- a class variable and the ALL universal class variable
- crossings of class variables
- crossings of class variables and the ALL variable
- crossings that contain class variables and one analysis variable
- concatenations of any of the items above

Note that all class variables used in a denominator definition must appear in a dimension expression in the TABLE statement.

Determining which denominator definition is correct to use can be challenging. It is not always intuitive which expression you should use when you are constructing the denominator definition. In addition, if there are concatenated elements in an expression, the order of the elements matters. Whereas several combinations of expressions might produce output, not all combinations produce the desired percentage. Consider what is required when you create a subtotal percentage. The table consists of multiple subtables, so you need to figure out which combinations of the class variables and, if included, the ALL variable, contribute to each subtable.

A strategy for creating more complex denominator definitions starts with simplifying the TABLE statement: Remove all formats, statistics, and labels. Eliminate any parentheses by distributing (multiplying out) the variables. The resulting TABLE statement should include simply the variable names, asterisks, and commas. Next, determine the list of possible denominators. Two denominators that always work are the entire column dimension and the entire row dimension. Some other possible denominator definitions include one item from each of the parts that make up the row or column dimension.

With the following sample TABLE statement, where A, B, C, and D are defined in a CLASS statement, you can use the process described above to discover potential denominator definitions:

```
table A*B, C D;
```

Expanding the crossings in the table results in `A*B*C A*B*D`.

Here are possible denominator definitions for this table:

```
A*B

C D

A

B

A*C A*D

B*C B*D
```

As shown above, a good way to obtain a better understanding of this concept is to experiment with different combinations of valid denominators for the PCTN statistic.

### Calculating a Subtotal Percentage

In this example, the PROC TABULATE output includes multiple subtables. The percentage statistics in the output represent the subtotal percentage of each subtable. In order to define the subtotal for the denominator, you need to determine which class variables' values to sum in order to get the desired subtotal.

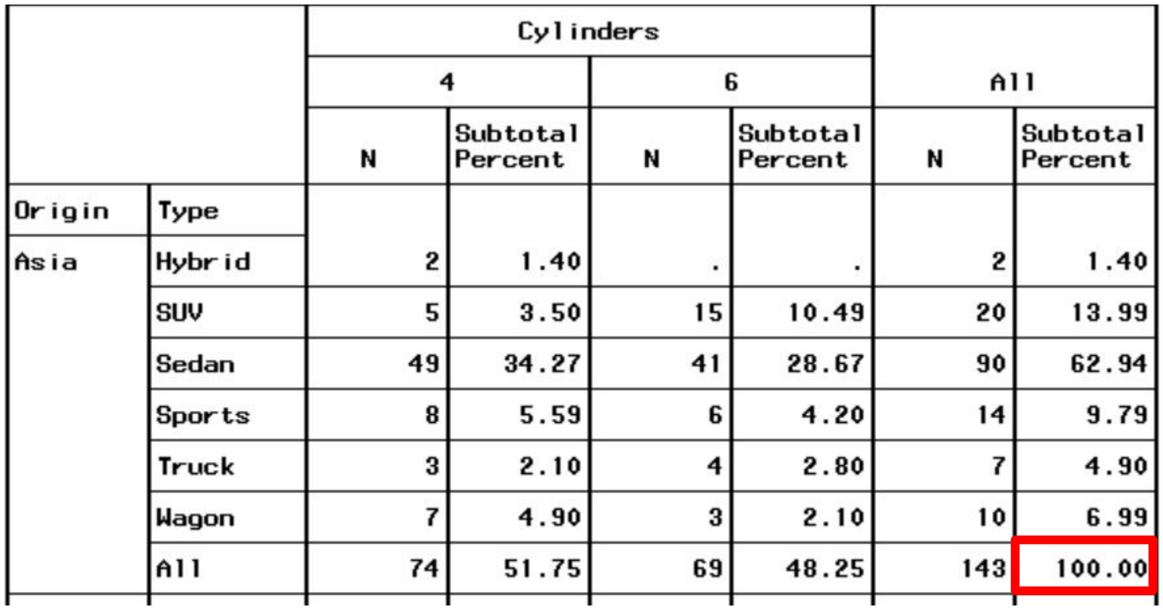
For example, you want a percentage that represents the proportion of each cylinder within type for a specific origin (as shown in Output 5). This percentage is the number of cylinders in each cell of the table divided by the total number of cylinders for each origin. In Output 5, the denominator for the Asia subtable is 143. This subtotal results from summing frequency counts for two class variables, TYPE and CYLINDERS. This is done for each ORIGIN. The values are not summed across all values of ORIGIN. Therefore, only class variables TYPE and CYLINDERS are included in the denominator definition.

The sample output also includes concatenated tables, (TYPE ALL) and (CYLINDERS ALL). You include these concatenations in the denominator definition. The resulting output is a concatenation of four subtables:

- TYPE and CYLINDERS—the number of 4-cylinder engines in each type or the number of 6-cylinder engines in each type
- TYPE and ALL—the total number of cylinders in each type
- ALL and CYLINDERS—the total number of 4-cylinder engines or the total number of 6-cylinder engines
- ALL and ALL—the total number of cylinders in all types

The implied crossing ALL*ALL can be simplified to ALL. Combine the elements above to create the denominator definition as shown in the following code example and the table in Output 5:

```
proc tabulate data=sashelp.cars format=8.2;
   class origin type cylinders;
   table origin*(type all), (cylinders all)*(n*f=8.
      pctn<type*cylinders type*all all*cylinders all>
         ='Subtotal Percent') / rts=20;
   where cylinders in (4,6);
run;
```



| | | Cylinders | | | | | |
|---|---|---|---|---|---|---|---|
| | | 4 | | 6 | | All | |
| | | N | Subtotal Percent | N | Subtotal Percent | N | Subtotal Percent |
| Origin | Type | | | | | | |
| Asia | Hybrid | 2 | 1.40 | . | . | 2 | 1.40 |
| | SUV | 5 | 3.50 | 15 | 10.49 | 20 | 13.99 |
| | Sedan | 49 | 34.27 | 41 | 28.67 | 90 | 62.94 |
| | Sports | 8 | 5.59 | 6 | 4.20 | 14 | 9.79 |
| | Truck | 3 | 2.10 | 4 | 2.80 | 7 | 4.90 |
| | Wagon | 7 | 4.90 | 3 | 2.10 | 10 | 6.99 |
| | All | 74 | 51.75 | 69 | 48.25 | 143 | 100.00 |

**Output 5. Output Showing the Subtotal Percentage**

### Calculating a Percentage from a Concatenated Table Request

In the following example of a concatenated table request, multiple class variables separated by spaces are included in the row dimension expression. Because an implied crossing exists between dimension expressions, the following crossings affect results of the PCTN statistic, shown in the table in Output 6:

```
origin*cylinders

type*cylinders
```

The denominator definition must include an expression for each of these crossings. In this case, you want the denominator to be the total within each cylinder, not across both cylinders, resulting in a percentage based on column totals. Therefore, you do not include CYLINDERS in the denominator definition. The values for a class variable that is not included in the denominator definition remain distinct, and totals are calculated for each of the distinct values. Here is the correct denominator definition for the PCTN statistic:

```
proc tabulate data=sashelp.cars format=8.2;
   class origin type cylinders;
   table origin type, cylinders*(n*f=8. pctn<origin type>) / rts=10;
   where cylinders in (4,6);
run;
```

| | Cylinders | | | |
|---|---|---|---|---|
| | 4 | | 6 | |
| | N | PctN | N | PctN |
| Origin | | | | |
| Asia | 74 | 54.41 | 69 | 36.32 |
| Europe | 25 | 18.38 | 54 | 28.42 |
| USA | 37 | 27.21 | 67 | 35.26 |
| Type | | | | |
| Hybrid | 2 | 1.47 | . | . |
| SUV | 7 | 5.15 | 30 | 15.79 |
| Sedan | 96 | 70.59 | 120 | 63.16 |
| Sports | 11 | 8.09 | 20 | 10.53 |
| Truck | 6 | 4.41 | 9 | 4.74 |
| Wagon | 14 | 10.29 | 11 | 5.79 |

**Output 6. Output Showing a Percentage from a Concatenated Table Request**

If you choose to include ALL in the TABLE request to show totals, you must include ALL in the denominator definition, as shown below:

```
proc tabulate data=sashelp.cars format=8.2;
   class origin type cylinders;
   table origin all type all, cylinders*(n*f=8. Pctn
      <origin all type all>) / rts=10;
   where cylinders in (4,6);
run;
```

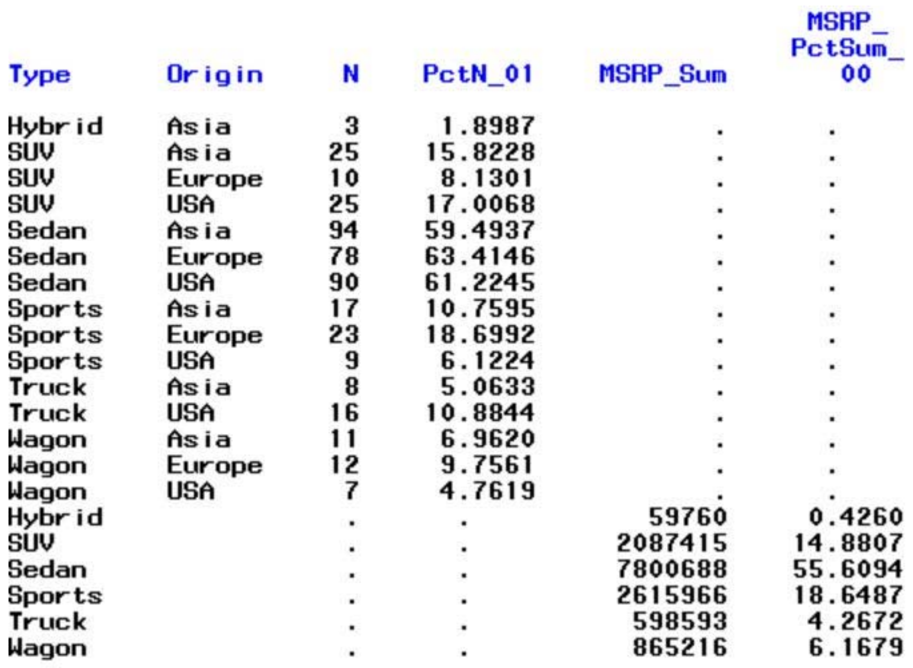
## UNDERSTANDING THE NAMING CONVENTIONS FOR PERCENTAGE STATISTICS IN AN OUTPUT DATA SET

PROC TABULATE gives you the ability to create an output data set. If you want to create a data set from your initial table results, it is helpful to know how the statistics, particularly the percentage statistics, are named in this data set. The naming conventions for the percentage statistics in a PROC TABULATE output data set are related to the interaction of the class variables. The naming convention is PCTN_*n*... or varname_PCTSUM_*n*..., where *n* is a binary value of `0` or `1`. A `0` in the *i*th position means that the *i*th class variable does not participate in the denominator crossing; a `1` means that it does.

In the following sample code and output (Output 7), the PCTN_01 variable in the output data set represents ORIGIN*COLPCTN in the TABLE statement. The `0` in the first position means that the class variable TYPE did not participate in the denominator crossing; the `1` in the second position means that the class variable ORIGIN did participate in the denominator crossing. In addition, the MSRP_PCTSUM_00 variable represents MSRP*COLPCTSUM in the TABLE statement. Neither of the class variables participate in this denominator crossing.

```
proc tabulate data=sashelp.cars out=out_pct(drop=_type_ _table_ _page_);
   class type origin;
   var msrp;
   table type, origin*(n colpctn) msrp*(sum colpctsum);
run;

proc print data=out_pct noobs;
run;
```



```
                                                          MSRP_
                                                         PctSum_
Type       Origin       N     PctN_01      MSRP_Sum        00

Hybrid     Asia         3      1.8987             .          .
SUV        Asia        25     15.8228             .          .
SUV        Europe      10      8.1301             .          .
SUV        USA         25     17.0068             .          .
Sedan      Asia        94     59.4937             .          .
Sedan      Europe      78     63.4146             .          .
Sedan      USA         90     61.2245             .          .
Sports     Asia        17     10.7595             .          .
Sports     Europe      23     18.6992             .          .
Sports     USA          9      6.1224             .          .
Truck      Asia         8      5.0633             .          .
Truck      USA         16     10.8844             .          .
Wagon      Asia        11      6.9620             .          .
Wagon      Europe      12      9.7561             .          .
Wagon      USA          7      4.7619             .          .
Hybrid                  .       .             59760     0.4260
SUV                     .       .           2087415    14.8807
Sedan                   .       .           7800688    55.6094
Sports                  .       .           2615966    18.6487
Truck                   .       .            598593     4.2672
Wagon                   .       .            865216     6.1679
```

**Output 7. An Output Data Set Created by PROC TABULATE**

## SPECIAL PERCENTAGE REQUESTS

You might want to control how missing values are handled in the PROC TABULATE output, or you might want to control which class variable values appear in the output. However, you want all class variable values to be included in the percentage calculations. These percentage requests require manipulating the data in advance by adding variables in a DATA step.

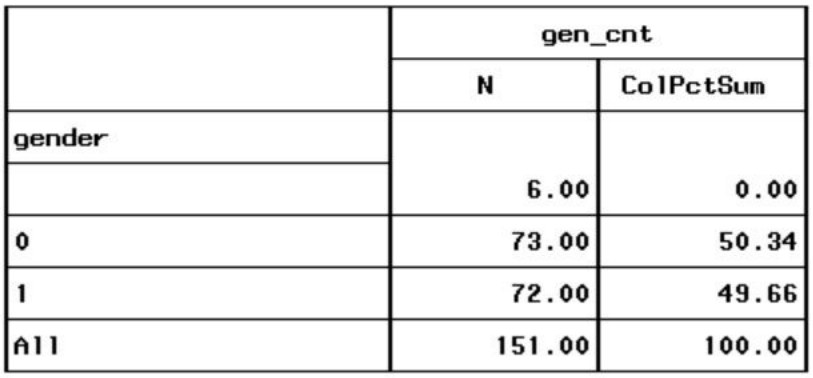
### INCLUDING MISSING CLASS VARIABLE VALUES IN THE COUNT BUT NOT IN THE PERCENTAGE CALCULATION

In your table output, you would like to show the frequency count of missing values for a particular class variable, but you do not want that total reflected in the overall percentage. To include missing class variable values in the output, use the MISSING option in the PROC TABULATE statement. In a prior DATA step, a new variable with the value of `0` is created when the class variable value is missing. Otherwise, the value is `1`. Then you use this new variable in the VAR statement in PROC TABULATE. The new variable is nested with the appropriate PCTSUM statistic in the TABLE statement in order to obtain the desired percentage. By basing the denominator on the summed value, the observations with missing values do not contribute toward the calculation of the percentage because they are always assigned a value of `0`. This is shown through the following sample code and table in Output 8:

```
data one;
   input gender $ freq;
   cards;
. 6
0 73
1 72
;
run;

data one;
   set one;
   if gender=. then gen_cnt=0;
   else gen_cnt=1;
run;

proc tabulate data=one missing;
   class gender;
   var gen_cnt;
   table gender all, gen_cnt*(n colpctsum);
   freq freq;
run;
```



| | gen_cnt | |
|---|---|---|
| | N | ColPctSum |
| gender | | |
| | 6.00 | 0.00 |
| 0 | 73.00 | 50.34 |
| 1 | 72.00 | 49.66 |
| All | 151.00 | 100.00 |

**Output 8. Output Showing Missing Values Excluded from a Percentage Calculation**

## EXCLUDING A CLASS VARIABLE VALUE FROM THE OUTPUT BUT INCLUDING IT IN THE PERCENTAGE CALCULATION

You have data that includes YES and NO responses. In your table output, you want to show only the YES responses, but you want the percentage to reflect all responses. You can do this by manipulating the data in advance. In a prior DATA step, a new variable with the value `0` is created for the class variable values that you want to exclude. Otherwise, the value is `1`. You also create another new variable that has a value of `1` for every observation. This value represents the denominator. You include both of these variables in the VAR statement. The new variable is nested with the PCTSUM statistic, and the new denominator variable becomes the denominator definition, as shown in this example:

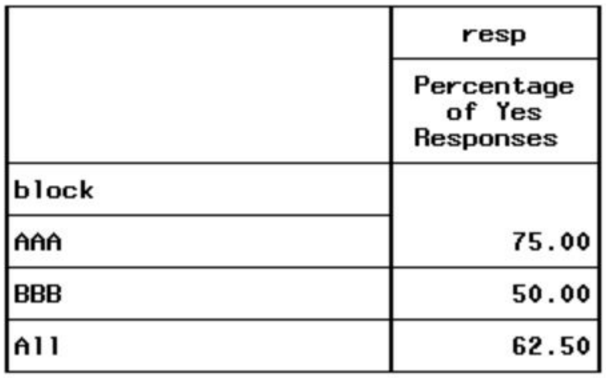
```
counter*pctsum<denom>
```

By making the denominator variable equal to `1` for every observation, you are including all possible values in the total. However, the values that you want to exclude do not contribute toward the calculation of the percentage because they are assigned a value of `0`. This is shown through the following sample code and the table in Output 9:

```
data test;
   input block $ response $;
   cards;
AAA Yes
AAA No
AAA Yes
AAA Yes
BBB No
BBB Yes
BBB No
BBB Yes
;
run;

data test;
   set test;
   denom=1;
   if response='Yes' then resp=1; else resp=0;
run;

proc tabulate data=test;
   class block;
   var denom resp;
   table block all, resp*pctsum<denom>='Percentage of Yes Responses';
run;
```



| | resp |
|---|---|
| | Percentage of Yes Responses |
| block | |
| AAA | 75.00 |
| BBB | 50.00 |
| All | 62.50 |

**Output 9. Output showing a Class Variable Value Excluded from Percentage Calculation**

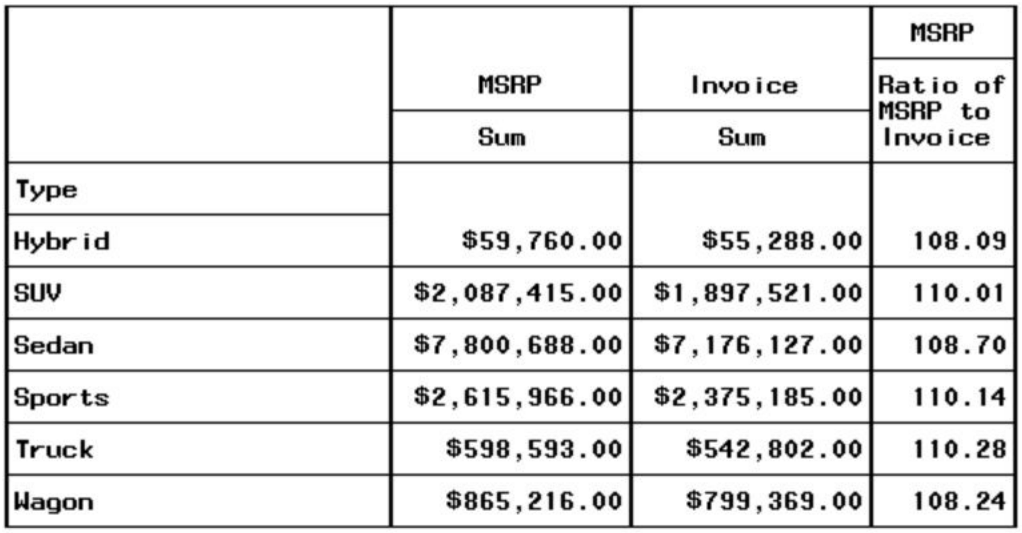
## CALCULATING A RATIO

A ratio is another special percentage that you can calculate. A *ratio* is a relationship between two numbers. Because a ratio can be expressed as a percentage, the PCTSUM statistic is used. Using two variables that are defined in a VAR statement, the ratio is calculated as follows:

```
var1*pctsum<var2>
```

The result represents VAR1 as a percentage of VAR2, as shown in the following code and table in Output 10:

```
proc tabulate data=sashelp.cars format=dollar14.2;
class type;
var invoice msrp;
table type, msrp invoice msrp*pctsum<invoice>=
                         'Ratio of MSRP to Invoice'*f=8.2;
run;
```



|  | MSRP | Invoice | MSRP |
|---|---|---|---|
|  | Sum | Sum | Ratio of MSRP to Invoice |
| Type |  |  |  |
| Hybrid | $59,760.00 | $55,288.00 | 108.09 |
| SUV | $2,087,415.00 | $1,897,521.00 | 110.01 |
| Sedan | $7,800,688.00 | $7,176,127.00 | 108.70 |
| Sports | $2,615,966.00 | $2,375,185.00 | 110.14 |
| Truck | $598,593.00 | $542,802.00 | 110.28 |
| Wagon | $865,216.00 | $799,369.00 | 108.24 |

**Output 10. Output Showing a Ratio**

## COMMON MISTAKES AND ERRORS

There are some common mistakes and errors that can occur when you work with percentages. This section discusses issues related to the order of elements in a denominator definition and messages that you might encounter in your SAS log.

### ORDER MATTERS WHEN YOU BUILD DENOMINATOR DEFINITIONS

PROC TABULATE uses the first valid denominator definition that it encounters for each expression in the TABLE statement to create the denominator for calculating percentages. The denominator definition must list elements in order from the most detailed to the least detailed. The first element of a denominator definition should never be ALL. Put ALL or elements crossed with ALL at or near the end of the denominator definition.

In the two TABLE statements that are shown in the code below, the only difference between them is the order of the expression in the denominator definition. In the resulting tables (Output 11 and Output 12), the difference is the PCTN statistic across the first row—in the first table, the values are all 100%. *ALL* is referred to as the universal class variable. It represents a class variable with one level because it summarizes all of the levels for class variables in a parenthetical group or dimension. Remembering that there is an implied crossing between the dimensions of a table, the class variable crossings for the first row of the table below are as follows:

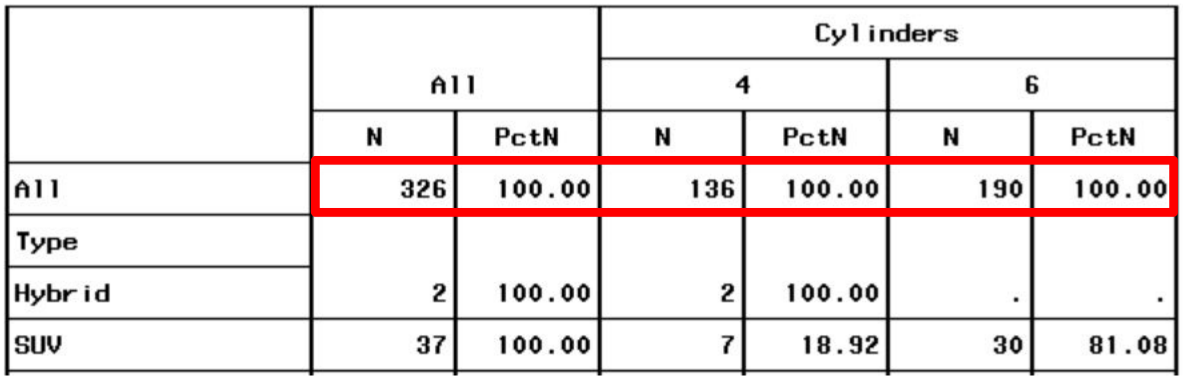
```
all*all

all*cylinders
```

The frequency count for each value of cylinder is the numerator for the PCTN statistic in the first row. In the first table, PROC TABULATE applies the denominator expression ALL to the crossing ALL*CYLINDERS and sums the classification levels represented by ALL to create the denominator. Because ALL represents only one classification level, the denominator for each cylinder is the same as the numerator, resulting in a PCTN statistic that is equal to 100%.

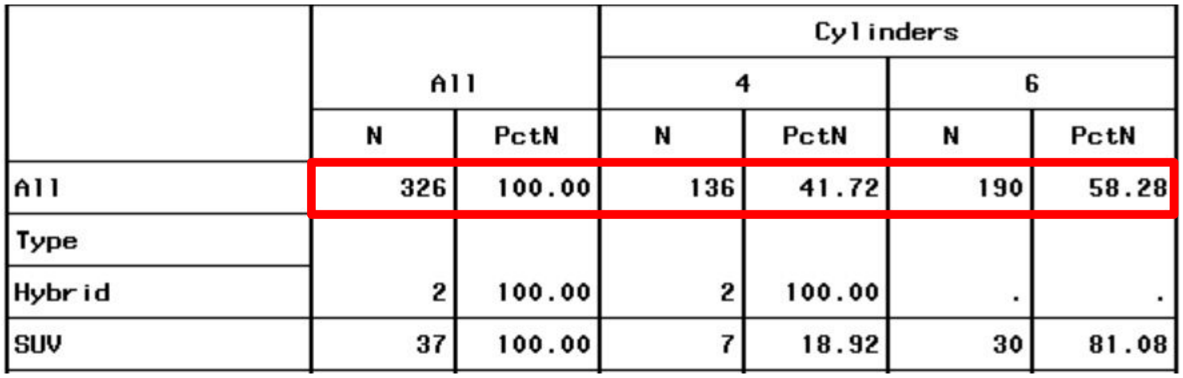
The denominator expression CYLINDERS indicates that the two levels of CYLINDERS should both contribute to the denominator count. Because CYLINDERS and ALL both appear in the crossing ALL*CYLINDERS, you avoid problems with concatenated tables by ensuring that PROC TABULATE uses the CYLINDERS (most detailed) denominator expression first by placing it before ALL (least detailed) in the denominator definition.

```
proc tabulate data=sashelp.cars format=8.2;
   class type cylinders;
   table all type, (all cylinders)*(n*f=8. pctn<all cylinders>) / rts=20;
   table all type, (all cylinders)*(n*f=8. pctn<cylinders all>) / rts=20;
   where cylinders in (4,6);
run;
```



| | | | Cylinders | | | |
|---|---|---|---|---|---|---|
| | All | | 4 | | 6 | |
| | N | PctN | N | PctN | N | PctN |
| All | 326 | 100.00 | 136 | 100.00 | 190 | 100.00 |
| Type | | | | | | |
| Hybrid | 2 | 100.00 | 2 | 100.00 | . | . |
| SUV | 37 | 100.00 | 7 | 18.92 | 30 | 81.08 |

**Output 11. Output Showing the Results When ALL Is First in the Denominator Definition**



| | | | Cylinders | | | |
|---|---|---|---|---|---|---|
| | All | | 4 | | 6 | |
| | N | PctN | N | PctN | N | PctN |
| All | 326 | 100.00 | 136 | 41.72 | 190 | 58.28 |
| Type | | | | | | |
| Hybrid | 2 | 100.00 | 2 | 100.00 | . | . |
| SUV | 37 | 100.00 | 7 | 18.92 | 30 | 81.08 |

**Output 12. Output Showing the Results When ALL Is Positioned Correctly in the Denominator Definition**

## ERROR AND WARNING MESSAGES EXPLAINED

You might encounter several different errors or warnings that are related to your denominator definition. Some common messages are explained below.

### ERROR: PCTN base is not in table. A PCTN crossing has no denominator.

This message means that the denominator definition does not match the table structure. For example, you have the following TABLE statement:

```
table x all, y*pctn<x>;
```

Because ALL is part of the row dimension, the statement should be written as follows:

```
table x all, y*pctn<x all>;
```

The solution to this problem is to use the appropriate denominator definition for the table structure.

### ERROR: Expecting a name.

Parentheses, such as those shown in the following code, are not allowed in a denominator definition:

```
pctn<(x y)*z>
```

The solution to this problem is to expand all the crossings, as shown below:

```
pctn<x*z y*z>
```

**WARNING: Invalid denominator nesting element: varname**

Using a denominator definition with percentage statistics other than PCTN and PCTSUM is not necessary and will generate the preceding warning. In SAS 9.3 and earlier releases, using a denominator definition with percentage statistics other than PCTN or PCTSUM also generates a Read Access Violation.

The solution to this problem is to remove any denominator definition for percentage statistics other than PCTN or PCTSUM.

## ENHANCING YOUR PERCENTAGES WITH FORMATTING TECHNIQUES

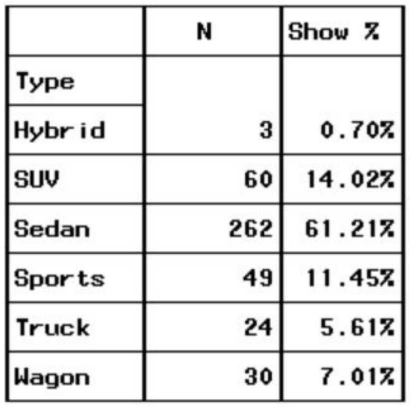
You can enhance the way in which percentages are displayed in your PROC TABULATE output in all destinations by applying a format or by adding colors if you are routing the output to any of the non-listing Output Delivery System (ODS) destinations. These methods are shown below.

### ADDING A PERCENT SIGN (%) TO DISPLAYED OUTPUT

By default, percentages are displayed in PROC TABULATE in 12.2 format. You might want your percentage to be displayed with a percent sign. The PCTN and PCTSUM families of statistics already multiply by 100. Therefore, using the PERCENT*w.d* format, which also multiplies by 100, gives overinflated results. To add a percent sign, you must use a picture format that is created with the PICTURE statement in PROC FORMAT. This is shown through the following sample code and the table in Output 13:

```
proc format;
   picture pctfmt (round) other='009.99%';
run;

proc tabulate data=sashelp.cars format=8.2;
   class type;
   table type, n*f=8. pctn='Show %'*f=pctfmt. / rts=10;
run;
```



|  | N | Show % |
|---|---|---|
| Type |  |  |
| Hybrid | 3 | 0.70% |
| SUV | 60 | 14.02% |
| Sedan | 262 | 61.21% |
| Sports | 49 | 11.45% |
| Truck | 24 | 5.61% |
| Wagon | 30 | 7.01% |

**Output 13. Output Showing a Percent Sign**

### HIGHLIGHTING PERCENTAGES IN SAS® OUTPUT DELIVERY SYSTEM (ODS) OUTPUT

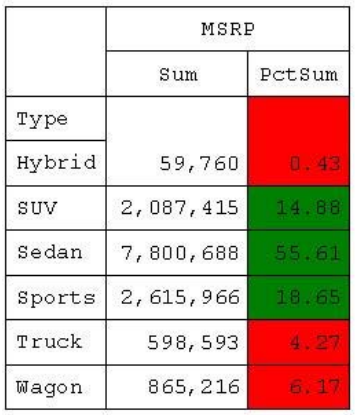
You might want to highlight percentages in your output to draw attention to them. You can do this with the SAS® Output Delivery System if you are routing your output to any of the non-listing destinations. To highlight a percentage, create a format that assigns colors based on the value. Then use the format by applying it to a style attribute, such as FOREGROUND= or BACKGROUND=, in a nesting with the percentage statistic. This is shown through the following sample code and the table in Output 14:

```
proc format;
   value colfmt low-<10='red'
                10-high='green';
run;

ods listing close;
ods rtf file='test.rtf' style=minimal;

proc tabulate data=sashelp.cars format=8.2;
   class type;
   var msrp;
   table type, msrp*(sum*f=comma12. pctsum*[style=[background=colfmt.]]);
run;

ods rtf close;
ods listing;
```



| | MSRP | |
|---|---|---|
| | Sum | PctSum |
| Type | | |
| Hybrid | 59,760 | 0.43 |
| SUV | 2,087,415 | 14.88 |
| Sedan | 7,800,688 | 55.61 |
| Sports | 2,615,966 | 18.65 |
| Truck | 598,593 | 4.27 |
| Wagon | 865,216 | 6.17 |

**Output 14.  RTF Output Showing Percentages Highlighted**

## CONCLUSION

This paper provides examples of standard percentage requests, including some more challenging requests in which denominator definitions are required.  Additional examples show how to manipulate the data to get the desired percentage.  This paper also instructs you in ways to avoid common mistakes and in how to apply formatting to the output.  The methods discussed in this paper help you gain a better understanding of how to use PROC TABULATE to calculate percentages and provide you with the ability to enhance your PROC TABULATE output in new ways.

## ACKNOWLEDGMENTS

The author acknowledges Bari Lawhorn, Russ Tyndall, Jerry Leonard, and David Kelley from SAS for their time and efforts to make this paper better for our customers.

## RECOMMENDED READING

Haworth, Lauren. 1999.  *PROC TABULATE by Example*. p. 65-93.  Cary, NC:  SAS Institute, Inc.

SAS Institute Inc. 2012. "TABULATE Procedure." *Base SAS® 9.3 Procedures Guide: Second Edition*. Cary, NC: SAS Institute Inc. Available at support.sas.com/documentation/cdl/en/proc/65145/PDF/default/proc.pdf.

## CONTACT INFORMATION

Your comments and questions are valued and encouraged.  Contact the author at:

Kathryn McLawhorn
SAS Institute, Inc.
SAS Campus Drive
Cary, NC 27513
E-mail: support@sas.com
Web: support.sas.com