ABSTRACT
Several years ago, one of my clients was in the business of selling reports to hospitals. He used PROC TABULATE to generate part of these reports. He liked the way this procedure 'crunched the numbers', but not the way the final reports looked. He said he would go broke if he had to sell default PROC TABULATE output. So, he wrote his own routine to take TABULATE output and render it through Crystal Reports. This was in the days of the 6.12 release of SAS before there was something called the Output Delivery System (ODS). Once he got his hands on SAS ODS, he kissed his Crystal Reports good-bye. This paper is all about using PROC TABULATE to generate fantastic looking reports. If you want to generate BIG money reports with PROC TABULATE, this presentation is for you.

INTRODUCTION
The TABULATE procedure offers the same statistics as many of the other SAS procedures found in the Base SAS product such as MEANS, SUMMARY and REPORT. In addition to this, the TABULATE procedure provides flexible report writing features such as:

- flexible table construction
- multiple dimensions
- use of labels and formats
- customization with the Output Delivery System (ODS) statements and options.

The general form of the TABULATE step is:

```
PROC TABULATE data = SAS-data-set options ;
   class variables ;
   var variables ;
   table expression;
run;
```

These are the basic statements to get started with this procedure. The CLASS statement allows you to specify the categorical (or class) variables. The VAR statement allows you to specify analysis variables. The TABLE statement is the real work horse of this procedure. You can use special characters to construct a fairly elaborate report. There are many other statements that can be used to make this procedure quite powerful such as:

- BY
- CLASSLEV
- FREQ
- KEYLEVEL
- WEIGHT

Most of these statements will be illustrated in upcoming examples.

The TABLE statement is used to construct the report. The table format along with statistics and variables are specified here. Before variables can be used on the TABLE statement, they must be mentioned on the VAR or CLASS statements first. The 'shape' of the report is controlled by TABLE statement operators.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Symbol</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comma</td>
<td>,</td>
<td>determines the number of dimensions</td>
</tr>
<tr>
<td>Asterisk</td>
<td>*</td>
<td>cross, subgroup or 'within'</td>
</tr>
<tr>
<td>Blank</td>
<td>(</td>
<td>table concatenator</td>
</tr>
<tr>
<td>Parenthesis</td>
<td>(</td>
<td>grouping agent</td>
</tr>
<tr>
<td>Brackets</td>
<td>&lt;&gt;</td>
<td>specifies denominator definitions</td>
</tr>
<tr>
<td>Equal</td>
<td>=</td>
<td>assigns labels or formats</td>
</tr>
</tbody>
</table>
One of the most important TABLE statement operators to initially focus on is the comma. When there is not a comma present, the report has only one dimension (column). When there is only one comma in the TABLE statement, there are two dimensions (row, column). When there are two commas in the TABLE statement, there are three dimensions (page, row, column).

GETTING STARTED
This paper presents several examples of PROC TABULATE ranging from fairly simple to more complex that illustrate the power and flexibility of this procedure. The data that is used is derived from the SASHELP.PM dataset. Only the first 18 rows are shown here.

<table>
<thead>
<tr>
<th>Obs</th>
<th>Hub</th>
<th>Country</th>
<th>TYPE</th>
<th>City</th>
<th>INCOME</th>
<th>OVERHEAD</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>London</td>
<td>Australia</td>
<td>MD11</td>
<td>Acton</td>
<td>288.24</td>
<td>230.59</td>
<td>Last</td>
</tr>
<tr>
<td>2</td>
<td>London</td>
<td>Australia</td>
<td>MD11</td>
<td>Acton</td>
<td>523.24</td>
<td>415.59</td>
<td>This</td>
</tr>
<tr>
<td>3</td>
<td>London</td>
<td>Australia</td>
<td>MD11</td>
<td>Acton</td>
<td>1,500.24</td>
<td>1,200.19</td>
<td>This</td>
</tr>
<tr>
<td>4</td>
<td>London</td>
<td>Australia</td>
<td>MD11</td>
<td>Acton</td>
<td>1,660.57</td>
<td>1,328.45</td>
<td>This</td>
</tr>
<tr>
<td>5</td>
<td>London</td>
<td>Australia</td>
<td>DC10</td>
<td>Acton</td>
<td>499.24</td>
<td>394.40</td>
<td>Last</td>
</tr>
<tr>
<td>6</td>
<td>London</td>
<td>Australia</td>
<td>DC10</td>
<td>Acton</td>
<td>523.24</td>
<td>413.36</td>
<td>This</td>
</tr>
<tr>
<td>7</td>
<td>London</td>
<td>Australia</td>
<td>DC10</td>
<td>Acton</td>
<td>804.24</td>
<td>695.36</td>
<td>This</td>
</tr>
<tr>
<td>8</td>
<td>London</td>
<td>Australia</td>
<td>DC10</td>
<td>Acton</td>
<td>874.62</td>
<td>590.95</td>
<td>This</td>
</tr>
<tr>
<td>9</td>
<td>San Fran</td>
<td>Australia</td>
<td>A300</td>
<td>Acton</td>
<td>198.24</td>
<td>152.64</td>
<td>Last</td>
</tr>
<tr>
<td>10</td>
<td>San Fran</td>
<td>Australia</td>
<td>A300</td>
<td>Acton</td>
<td>523.24</td>
<td>402.89</td>
<td>This</td>
</tr>
<tr>
<td>11</td>
<td>San Fran</td>
<td>Australia</td>
<td>A300</td>
<td>Acton</td>
<td>1,308.24</td>
<td>1,007.34</td>
<td>This</td>
</tr>
<tr>
<td>12</td>
<td>San Fran</td>
<td>Australia</td>
<td>A300</td>
<td>Acton</td>
<td>1,340.82</td>
<td>1,032.43</td>
<td>This</td>
</tr>
<tr>
<td>13</td>
<td>New York</td>
<td>Australia</td>
<td>MD11</td>
<td>Melbourne</td>
<td>529.00</td>
<td>322.69</td>
<td>Last</td>
</tr>
<tr>
<td>14</td>
<td>New York</td>
<td>Australia</td>
<td>MD11</td>
<td>Melbourne</td>
<td>1,170.00</td>
<td>713.70</td>
<td>This</td>
</tr>
<tr>
<td>15</td>
<td>New York</td>
<td>Australia</td>
<td>MD11</td>
<td>Melbourne</td>
<td>1,596.00</td>
<td>973.56</td>
<td>This</td>
</tr>
<tr>
<td>16</td>
<td>New York</td>
<td>Australia</td>
<td>MD11</td>
<td>Melbourne</td>
<td>1,876.61</td>
<td>1,144.73</td>
<td>This</td>
</tr>
<tr>
<td>17</td>
<td>Sydney</td>
<td>Australia</td>
<td>DC10</td>
<td>Melbourne</td>
<td>628.71</td>
<td>421.24</td>
<td>Last</td>
</tr>
<tr>
<td>18</td>
<td>Sydney</td>
<td>Australia</td>
<td>DC10</td>
<td>Melbourne</td>
<td>251.29</td>
<td>168.36</td>
<td>This</td>
</tr>
</tbody>
</table>

Figure 1.

EXAMPLE 1:
Use multiple TABLE statements in one step to generate multiple reports.

```
PROC TABULATE data = SASHELP.pm;
   class hub;
   var income;
   table hub * n;
   table hub * income * sum;
run;
```

Example 1 program.

<table>
<thead>
<tr>
<th>Hub</th>
<th>Frankfurt</th>
<th>London</th>
<th>New York</th>
<th>San Fran</th>
<th>Sydney</th>
<th>Tokyo</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>92.00</td>
<td>76.00</td>
<td>96.00</td>
<td>76.00</td>
<td>112.00</td>
<td>60.00</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hub</th>
<th>Income</th>
<th>Income</th>
<th>Income</th>
<th>Income</th>
<th>Income</th>
<th>Income</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum</td>
<td>Sum</td>
<td>Sum</td>
<td>Sum</td>
<td>Sum</td>
<td>Sum</td>
<td>Sum</td>
</tr>
<tr>
<td>99128.02</td>
<td>161560.84</td>
<td>73164.91</td>
<td>102212.65</td>
<td>238586.14</td>
<td>133330.27</td>
<td></td>
</tr>
</tbody>
</table>
Example 1 Output.
Because there are NO commas in either TABLE statement, both reports are only one dimensional (column). There is a column for each value of HUB. The first TABLE statement (Table HUB * N ;) generates a report that shows a frequency count (N statistic) per HUB. The default format, 12.2, controls the number of decimal places. The second TABLE statement (Table HUB * INCOME * SUM ;) generates a report that has a column for each HUB and displays the total INCOME for that HUB.

EXAMPLE 2:
Use multiple CLASS variables to generate a two dimensional report.

```sas
PROC TABULATE data = SASHELP.pm format=comma12.2;
  class year type ;
  var income;
  where type in ('777', '747');
  table year, type * income * sum ;
run;
```

Example 2 program.

<table>
<thead>
<tr>
<th></th>
<th>TYPE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>747</td>
<td>777</td>
</tr>
<tr>
<td>INCOME</td>
<td>INCOME</td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td>Sum</td>
<td></td>
</tr>
</tbody>
</table>

Example 2 output

The FORMAT = option on the PROC statement controls the appearance of the whole report. On the TABLE statement, there is one comma, which means the report has 2 dimensions. Starting with the keyword TABLE and moving to the left, everything up to the comma goes in the row dimension. Everything after the comma goes in the column dimension. You can have a third, or page, dimension in your report, but this paper only shows one and two dimensional reports.

EXAMPLE 3:
Display multiple statistics and use an OUT = option to create a SAS dataset,

```sas
PROC TABULATE data = SASHELP.pm format=comma12.2 out = pm;
  class hub ;
  var income;
  table hub, income * (n sum mean max mode);
run;
```

Example 3 program.
This step generates two forms of output. The first is the report that is generated from the procedure, and the second is the output dataset created from the OUT= option on the PROC statement. Both types of output are shown in this paper.

Example 3 output – the WORK.PM dataset

Example 3 output – the report.

**EXAMPLE 4:**

Use an ‘=' operator to create blank labels for TYPE and SUM.

```
PROC TABULATE data = SASHELP.pm format=comma12.2 ;
  class hub type;
  var income;
  where type in('777', '747');
  table hub, type = ' ' * income * sum = ' ';
run;
```

Example 4 program.

The '=' can be used to display text in the report. On the TABLE statement, it is used twice: once after the TYPE variable, and once after the SUM statistic. Both times, the text is set to a blank. This has the effect of 'blanking' out the words TYPE and SUM. Notice that they are in Example 2 output. TYPE is at the top of the report in line one, and SUM is on line 3 of that report. In this example, those words are not displayed. .
Example 5:
Enhance the program by adding the 'f=' and the rts option. Include the special ALL variable.

PROC TABULATE data = SASHELP.pm format=comma12.2 ;
  class hub type;
  var income;
  where type in('777', '747');
  table hub all, type * income * sum = ' ' 
       all = 'Total' * income * sum = ' ' f=dollar12.2 / rts = 12;
run;

Example 5 program
Example 5 output.

Column and row totals were added to this example by using the special variable ALL in both dimensions. ALL creates totals in either the row or the column dimension. In the column dimension, the "Total" option was used so that ALL is not displayed like it is in the row dimension. Also, the f= option is used in the column dimension to control the format for the total column. Notice the ALL column is the only column where dollar signs appear.

Output Delivery System (ODS) components can be used to enhance this report. Although there are many components to ODS, this paper only looks at the STYLE= option. The screen capture below illustrates the type of things that can be done to add color to the last report.

This chart serves as the 'gameplan' for the next report.

**EXAMPLE 6:**

Use the **STYLE = options** in the PROC TABULATE step to produce a 'colorful' report.
Example 6 program.

```sas
ods pdf file='mypdf.pdf';

proc tabulate data=sas_1.pm format = comma12.2;
   class hub type / style={font_face=arial background=red};
   classlev hub / style={background=pink};
   classlev type / style={background=orange};
   var income   / style={background=yellow};
   keyword all   / style={background=cfffafa};
   where type in('747','777');
   table hub all, type * income *sum=' ' 
      all='Total' * income * sum='  *f=dollar12.2 
      / rts=12 box={label='New Flights' 
                     style={background=ccbfffbb font_face=arial font_size=4}});
run;

ods pdf close;
```

Example 6 output.

**EXAMPLE 7:**
Modify the program by using different style attributes. Define a URL to be used as a hyperlink in the BOX= option. Link the PROC TABULATE output to a spreadsheet. Note: The ODS statements directing this output to a PDF are not included in this screen capture.
Example 7 program.

```
proc tabulate data=sas_1.pm f=15.2 s={font_face=arial} ;
  var income;
  class hub type;
  where type in('747', '777');
  keyword all;
  table hub all, type * income * sum='' all='Total' * income * sum = '' * f=dollar12.2 /
    rts=12 box={label='New Flights' style=
    { url='c:\newflights.xls' background=light yellow} } ;
run;
```

Notice the URL= option within the BOX=option. This creates a link for the label 'New Flights'. When this label is selected in the report, the newflights.xls spreadsheet opens.

Example 7 output.

The text 'New Flights' has been defined as the link to the spreadsheet. When you click on 'New Flights', the spreadsheet below opens.
DOING MORE WITH STATISTICS

The TABULATE procedure can calculate the following descriptive statistics:

- COLPCTN  COLPCTSUM  NMISS  MIN  MAX  VAR  CV  MODE
- KURTOSIS  ROWPCTN  ROWPCTSUM  SUMWGT  CSS  USS  RANGE  STD
- SKEWNESS  REPPCTN  REPPCTSUM  STDERR  SUM  MEAN  STDERR  N
- LCLM  UCLM  PAGEPCTN  PAGEPCTSUM  PCTN  PCTSUM  STD  STDDEV

Starting with example 8, the SASHELP.CLASS dataset is used to generate PROC TABULATE reports.

EXAMPLE 8:
Illustrate the N, SUM, PCTN and the PCTSUM statistics. Note: the following program was run in Enterprise Guide. The STYLE = option on the TABLE statement gives the TOTAL row a yellow background.

```
proc tabulate data = sashelp.class format=comma12.2;
   class sex age;
   var   height;
   table age all='Total' * ( [style = [ background=yellow] ] ),
       height * (n sum pctn pctsum);
   keyword all / style = {background=yellow};
run;
```

Example 8 program
**PCTN** and **PCTSUM** are calculated as follows:
The TOTAL number of observations (N) is 19. For each cell in the PCTN column the formula is:
\[
PCTN (cell) = \frac{N}{19} \times 100.
\]
The **SUM** of all the student's HEIGHT is 1,184.40.
\[
PCTSUM (cell) = \frac{SUM}{1184} \times 100.
\]

**EXAMPLE 8B:**
Illustrate the N, SUM, **ROWPCTN** and the **ROWPCTSUM** statistics. A second class variable (SEX) is added to the TABLE statement to give the report a two dimensional appearance. Because the new class variable is in the column dimension, there will be a group of columns for each gender.

```
proc tabulate data = sashelp.class format=comma12.2;
  class sex age;
  var height ;
  table age, sex * height * (n sum rowpctn rowpctsum)
run;
```

Example 8B program.
When the two RowPctN columns are added together, they total 100. Also, when the two RowPctSum columns are added together, they also total 100. For RowPct statistics, the formula is as follows:

RowPctN = N (single cell) / Row Total for N.

RowPctSum = SUM (single cell) / Row total for Sum.

EXAMPLE 8C:
Use the denominator definitions as an alternative way to create percentages that add to 100 across the row. Prior to the development of RowPctN and RowPctSum, denominator definitions ‘<’ were used to control the calculation of the statistics.

Example 8C program.

Notice the use of Proc FORMAT to create a Percent ‘picture’. It is applied only to the PctSum column.

Example 8C output.

EXAMPLE 8D:
Modify the previous program by adding STYLE = options and the Keyword ALL to the Column dimension. Note that STYLE = can be abbreviated as S=.
Also notice that since both **RowPctN** and **RowPctSum** add up to 100, only one statistical column is used in the ALL section of the column dimension.

**Example 9 program.**

```
proc tabulate data = sashelp.class format=comma12.2;
    class sex age;
    var height;
    table age, sex * height * (n sum rowpctn*[s=[background=pink] ]
                          rowpctsum*[style=[background=yellow] ]
                          all * height * (n sum rowpctn='Percent'*[s=[background=orange] ] )
                          keyword rowpctn  / style=[background=red];
run;
```

Notice that the keyword **ALL** has been added to the ROW dimension. The **ColPctN** and **ColPctSum** columns add up to 100 percent within each value of the SEX variable.
CONCLUSION
TABULATE is a very useful procedure in the SAS report writing arsenal. It can deploy many statistics to crunch the numbers, then display them in a number of ways. The addition of ODS as an enhancement with version 8 SAS makes this procedure even more valuable.

ACKNOWLEDGMENTS
As always, I would like to thank the people in the Technical Support department of SAS Institute for their kind and helpful assistance for this past year. Their knowledge helped make this presentation possible.

CONTACT INFORMATION
If you have any questions or comments, the author can be reached at:
   Ben Cochran
   The Bedford Group
   3224 Bedford Avenue
   Raleigh, NC 27607
   Work Phone:  919.741.0370
   Email: bencochran@nc.rr.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration.

Other brand and product names are trademarks of their respective companies.