Chapter 9 PROC TABULATE

- Introduction and syntax
- Constructing basic tables
- Improving table appearance
- More complex tables and percentages, including use of indicator variables and dummy variables
- Miscellaneous topics
    - Controlling the order of formatted CLASS variables
    - Indenting CLASS variable values in the ROW dimension

Introduction and Syntax

- PROC TABULATE displays descriptive statistics in tabular format. The procedure computes many of the same statistics as PROC MEANS, but in addition you can specify exactly how you want those statistics displayed.

- The procedure produces tables in one, two, or three dimensions – columns, rows, and pages.

- CLASS statements define which variables are to be used as categories. CLASS variable values determine the table dimensions. The MISSING option tells SAS to treat missing values like other CLASS variable values.

- VAR statements define which variables are to be analyzed (the analysis variables).

- The TABLE statement defines the structure and appearance of the resulting table.

- The contents of a cell of the table is some statistic computed for the subgroup of observations in that cell (as determined by the CLASS variables), computed for one of the VAR (analysis) variables.

TABLE Statement Components

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

Operators:

- * for nesting
- space for concatenation

Operands:

- class variables (from CLASS statement) or ALL
- analysis variables (from VAR statement)
- ( ) for grouping
- statistics keywords such as N, NMISS, MEAN, STD, MIN, MAX, SUM, MEDIAN
- label specifications (='text')
- format specifications (F=w.d)
- < > for specifying percentage denominators

General forms of the TABLE statement:

TABLE *column*;
or
TABLE *row*, *column*;
or
TABLE *page*, *row*, *column*;

where *page*, *row*, and *column* are expressions which consist of CLASS variables, analysis (VAR) variables, statistic names, and label and format specifications, connected by operators (* or space).

Notes

- All analysis variables must be used in one dimension.
- All statistics must also appear in one dimension, which may be the same or different from the analysis variable dimension.
- Class variables can appear in any combination of dimensions.
- When only class variables are specified (no analysis variables), TABULATE gives you counts (N) by default.
- When only analysis variables are specified (no class variables), TABULATE gives you sums of those variables by default.
- ALL is a universal class variable that you can concatenate whenever you want a summary over a whole dimension.
- **When using PROC TABULATE, always design (sketch) the table you want before starting to code.**

Introductory Example

To get output similar to PROC MEANS, we can specify analysis variables in the row dimension and a list of statistics in the column dimension.

```
PROC TABULATE DATA=bios511.sales;
     VAR cost price;
     TABLE cost price, N MEAN STD;
RUN;
```

	N	Mean	Std
COST	50	55.27	56.00
PRICE	50	143.47	192.40

But PROC TABULATE gives us much more flexibility than PROC MEANS. We can transpose the previous table by simply changing the TABLE statement to

```
TABLE N MEAN STD, cost price;
```

	COST	PRICE
N	50	50
Mean	55.27	143.47
Std	56.00	192.40

Constructing Basic Tables

```
TITLE 'PROC TABULATE';
PROC TABULATE DATA=bios511.sales;
     CLASS dept;
     TABLE dept;          /* TABLE dept*N */
     TITLE2 'One-Dimensional Table';
RUN;
```

PROC TABULATE
One-Dimensional Table

DEPT	
FURS	SHOES
N	N
6	44

```
PROC TABULATE DATA=bios511.sales;
     CLASS dept clerk;
     TABLE dept, clerk;  /* dept, clerk*n */
     TITLE2 'Two-Dimensional Table';
RUN;
```

PROC TABULATE
Two-Dimensional Table

	CLERK		
	AGILE	BURLEY	CLEVER
	N	N	N
DEPT			
FURS	2	4	.
SHOES	11	5	28

```
PROC TABULATE DATA=bios511.sales;
     CLASS dept clerk;
     TABLE dept*clerk;
     TITLE2 'Nested Variables';
RUN;
```

PROC TABULATE
Nested Variables

DEPT				
FURS		SHOES		
CLERK		CLERK		
AGILE	BURLEY	AGILE	BURLEY	CLEVER
N	N	N	N	N
2	4	11	5	28

```
PROC TABULATE DATA=bios511.sales;
     VAR cost;
     TABLE cost*MEAN;
     TABLE cost, MEAN;
     TITLE2 'Analysis Variables';
RUN;
```

PROC TABULATE
Analysis Variables

COST
Mean
55.27

PROC TABULATE
Analysis Variables

	Mean
COST	55.27

```
PROC TABULATE DATA=bios511.sales;
     VAR cost price;
     TABLE cost price, N MEAN STD;
     TABLE N MEAN STD, cost price;
     TABLE cost*(N SUM) price*(MEAN STD);
     TITLE2 'Analysis Variables';
RUN;
```

PROC TABULATE
Analysis Variables

	N	Mean	Std
COST	50	55.27	56.00
PRICE	50	143.47	192.40

PROC TABULATE
Analysis Variables

	COST	PRICE
N	50	50
Mean	55.27	143.47
Std	56.00	192.40

PROC TABULATE
Analysis Variables

COST		PRICE	
N	Sum	Mean	Std
50	2763.62	143.47	192.40

```
PROC TABULATE DATA=bios511.sales;
     VAR cost price;
     CLASS dept;
     TABLE dept, cost*(N SUM) price*(MEAN STD);
     TABLE dept ALL, cost*(N SUM) price*(MEAN STD);
     TABLE dept*price*MEAN;
     TITLE2 'Crossed Class and Analysis Variables';
RUN;
```

PROC TABULATE
Crossed Class and Analysis Variables

	COST		PRICE	
	N	Sum	Mean	Std
DEPT				
FURS	6	1222.02	648.32	105.93
SHOES	44	1541.60	74.62	21.58

PROC TABULATE
Crossed Class and Analysis Variables

	COST		PRICE	
	N	Sum	Mean	Std
DEPT				
FURS	6	1222.02	648.32	105.93
SHOES	44	1541.60	74.62	21.58
All	50	2763.62	143.47	192.40

PROC TABULATE
Crossed Class and Analysis Variables

DEPT	
FURS	SHOES
PRICE	PRICE
Mean	Mean
648.32	74.62

```
PROC TABULATE DATA=bios511.sales2;
     VAR cost;
     CLASS dept / MISSING;
     TABLE dept*cost*MEAN;
     TITLE3 'With MISSING option on CLASS statement';
RUN;
```

PROC TABULATE
Crossed Class and Analysis Variables
With MISSING option on CLASS statement

DEPT		
	FURS	SHOES
COST	COST	COST
Mean	Mean	Mean
240.00	196.40	35.01

```
PROC TABULATE DATA=bios511.sales;
     VAR cost price;
     CLASS dept clerk;
     TABLE dept clerk, cost*(N SUM) price*(MEAN STD);
     TABLE dept*clerk, cost*(N SUM) price*(MEAN STD);
     TITLE2 'Crossed Class and Analysis Variables';
RUN;
```

PROC TABULATE
Crossed Class and Analysis Variables

	COST		PRICE	
	N	Sum	Mean	Std
DEPT				
FURS	6	1222.02	648.32	105.93
SHOES	44	1541.60	74.62	21.58
CLERK				
AGILE	13	782.08	161.89	216.06
BURLEY	9	993.06	323.86	319.39
CLEVER	28	988.48	76.93	24.62

PROC TABULATE
Crossed Class and Analysis Variables

		COST		PRICE	
		N	Sum	Mean	Std
DEPT	CLERK				
FURS	AGILE	2	392.00	645.00	77.78
	BURLEY	4	830.02	649.98	129.13
SHOES	AGILE	11	390.08	74.05	15.80
	BURLEY	5	163.04	62.97	9.09
	CLEVER	28	988.48	76.93	24.62

```
PROC TABULATE DATA=bios511.sales;
     VAR cost price;
     CLASS dept clerk;
     TABLE dept*(clerk ALL) clerk ALL,
           cost*(N SUM) price*(MEAN STD);
     TITLE2 'Crossed Class and Analysis Variables';
RUN;
```

PROC TABULATE
Crossed Class and Analysis Variables

		COST		PRICE	
		N	Sum	Mean	Std
DEPT	CLERK				
FURS	AGILE	2	392.00	645.00	77.78
	BURLEY	4	830.02	649.98	129.13
	All	6	1222.02	648.32	105.93
SHOES	CLERK				
	AGILE	11	390.08	74.05	15.80
	BURLEY	5	163.04	62.97	9.09
	CLEVER	28	988.48	76.93	24.62
	All	44	1541.60	74.62	21.58
CLERK					
AGILE		13	782.08	161.89	216.06
BURLEY		9	993.06	323.86	319.39
CLEVER		28	988.48	76.93	24.62
All		50	2763.62	143.47	192.40

```
PROC TABULATE DATA=bios511.sales;
     CLASS dept clerk;
     VAR cost price;
     TABLE clerk ALL, (dept ALL)*(cost*(N SUM) price*SUM);
     TITLE2 'Crossed Class and Analysis Variables -- Wide
Table';
RUN;
```

PROC TABULATE
Crossed Class and Analysis Variables -- Wide Table

	DEPT						All		
	FURS			SHOES					
	COST		PRICE	COST		PRICE	COST		PRICE
	N	Sum	Sum	N	Sum	Sum	N	Sum	Sum
CLERK									
AGILE	2	392.00	1290.00	11	390.08	814.60	13	782.08	2104.60
BURLEY	4	830.02	2599.90	5	163.04	314.85	9	993.06	2914.75
CLEVER	.	.	.	28	988.48	2154.00	28	988.48	2154.00
All	6	1222.02	3889.90	44	1541.60	3283.45	50	2763.62	7173.35

If you were using ODS to produce RTF output and wanted to include a very wide table, you could submit OPTIONS ORIENTATION=LANDSCAPE; to request a change from the default PORTRAIT paper orientation. To return to the default you could of course submit OPTIONS ORIENTATION=PORTRAIT;

```
PROC TABULATE DATA=bios511.sales;
     CLASS dept clerk;
     VAR cost price;
     TABLE dept*clerk dept clerk ALL,
          cost*(N SUM) price*SUM;
     TITLE2 'Crossed Class and Analysis Variables';
RUN;
```

PROC TABULATE
Crossed Class and Analysis Variables

		COST		PRICE
		N	Sum	Sum
DEPT	CLERK			
FURS	AGILE	2	392.00	1290.00
	BURLEY	4	830.02	2599.90
SHOES	AGILE	11	390.08	814.60
	BURLEY	5	163.04	314.85
	CLEVER	28	988.48	2154.00
DEPT				
FURS		6	1222.02	3889.90
SHOES		44	1541.60	3283.45
CLERK				
AGILE		13	782.08	2104.60
BURLEY		9	993.06	2914.75
CLEVER		28	988.48	2154.00
All		50	2763.62	7173.35

Improving Table Appearance

Besides enabling you to display a lot of information compactly, PROC TABULATE provides features for displaying the information attractively. In this example, we will start with a basic table and then use labels, formats, and other PROC TABULATE options to make it more appealing.

1. The basic table

```
PROC TABULATE DATA=bios511.sales;
     CLASS dept clerk;
     VAR price cost;
     TABLE clerk ALL,
          (dept ALL)*(cost*N (cost price)*SUM);
RUN;
```

PROC TABULATE
Improving Table Appearance
No improvements yet

	DEPT						All		
	FURS			SHOES					
	COST	COST	PRICE	COST	COST	PRICE	COST	COST	PRICE
	N	Sum	Sum	N	Sum	Sum	N	Sum	Sum
CLERK									
AGILE	2	392.00	1290.00	11	390.08	814.60	13	782.08	2104.60
BURLEY	4	830.02	2599.90	5	163.04	314.85	9	993.06	2914.75
CLEVER	.	.	.	28	988.48	2154.00	28	988.48	2154.00
All	6	1222.02	3889.90	44	1541.60	3283.45	50	2763.62	7173.35

2. Specify a default format, width of the row title space, and variable labels

```
PROC TABULATE DATA=bios511.sales FORMAT=7.2;
     CLASS dept clerk;
     CLASSLEV clerk / STYLE={cellwidth=1in};
     VAR price cost;
     LABEL dept  = 'Department'
          clerk = 'Clerk';
     TABLE clerk ALL,
          (dept ALL)*(cost*N (cost price)*SUM) ;
RUN;
```

	Department						All		
	FURS			SHOES					
	COST	COST	PRICE	COST	COST	PRICE	COST	COST	PRICE
	N	Sum	Sum	N	Sum	Sum	N	Sum	Sum
Clerk									
AGILE	2	392.00	1290.00	11	390.08	814.60	13	782.08	2104.60
BURLEY	4	830.02	2599.90	5	163.04	314.85	9	993.06	2914.75
CLEVER	.	.	.	28	988.48	2154.00	28	988.48	2154.00
All	6	1222.02	3889.90	44	1541.60	3283.45	50	2763.62	7173.35

3. Improve labeling of "N" columns, apply different column widths

```
PROC TABULATE DATA=bios511.sales FORMAT=7.2;
     CLASS dept clerk;
     VAR price cost;
     LABEL dept  = 'Department'
          clerk = 'Clerk';
     KEYLABEL N='# of Sales';
     TABLE clerk ALL,
          (dept ALL)*(cost=' '*N*F=2.0*{STYLE={CELLWIDTH=.5in}}
                    (cost price)*SUM*{STYLE={CELLWIDTH=.75in}}) ;
RUN;
```

	Department						All		
	FURS			SHOES					
		COST	PRICE		COST	PRICE		COST	PRICE
	# of Sales	Sum	Sum	# of Sales	Sum	Sum	# of Sales	Sum	Sum
Clerk									
AGILE	2	392.00	1290.00	11	390.08	814.60	13	782.08	2104.60
BURLEY	4	830.02	2599.90	5	163.04	314.85	9	993.06	2914.75
CLEVER	.	.	.	28	988.48	2154.00	28	988.48	2154.00
All	6	1222.02	3889.90	44	1541.60	3283.45	50	2763.62	7173.35

4. Improve labeling of "Sum" columns and add dollar format

```
PROC TABULATE DATA=bios511.sales FORMAT=7.2;
     CLASS dept clerk;
     CLASSLEV clerk / STYLE={cellwidth=1in};
     VAR price cost;
     LABEL dept  = 'Department'
          clerk = 'Clerk';
     KEYLABEL N='# of Sales';
     TABLE clerk ALL,
          (dept all)*
          (cost=' '*N*F=2.0*{STYLE={CELLWIDTH=.5in}}
           cost=' '*SUM='Wholesale Cost Total'
                 *{STYLE={CELLWIDTH=.75in}}*F=DOLLAR9.2
           price=' '*SUM='Retail Price Total'
                 *{STYLE={CELLWIDTH=.75in}}*F=DOLLAR9.2);
RUN;
```

	Department						All		
	FURS			SHOES					
	# of Sales	Wholesale Cost Total	Retail Price Total	# of Sales	Wholesale Cost Total	Retail Price Total	# of Sales	Wholesale Cost Total	Retail Price Total
Clerk									
AGILE	2	$392.00	$1,290.00	11	$390.08	$814.60	13	$782.08	$2,104.60
BURLEY	4	$830.02	$2,599.90	5	$163.04	$314.85	9	$993.06	$2,914.75
CLEVER	.	.	.	28	$988.48	$2,154.00	28	$988.48	$2,154.00
All	6	$1,222.02	$3,889.90	44	$1,541.60	$3,283.45	50	$2,763.62	$7,173.35

5. Apply user-defined formats, label the "All" rows and columns, and add a table description in the upper-left box

```
PROC FORMAT;
     VALUE $cft 'AGILE'='Joe Agile'
                'BURLEY'='Carol Burley'
                'CLEVER'='Amos Clever';
     value $dft 'FURS'='Furs'
                'SHOES'='Shoes';
RUN;

PROC TABULATE DATA=bios511.sales FORMAT=DOLLAR9.2;
     CLASS dept clerk;
     CLASSLEV clerk / STYLE={CELLWIDTH=1in};
     VAR price cost;
     LABEL dept  = 'Department'
           clerk = 'Clerk';
     KEYLABEL N='# of Sales';
     TABLE clerk ALL='Over All Clerks',
           (dept ALL='Over All Departments')*
           (cost=' '*N*F=2.0*{STYLE={CELLWIDTH=.5in}}
            cost=' '*SUM='Wholesale Cost Total'
                  *{STYLE={CELLWIDTH=.75in}}*F=DOLLAR9.2
            price=' '*SUM='Retail Price Total'
                  *{STYLE={CELLWIDTH=.75in}}*F=DOLLAR9.2)
             / BOX='Sales for the Month';
     FORMAT dept $dft. clerk $cft.;
RUN;
```

Sales for the Month	Department						Over All Departments		
	Furs			Shoes					
	# of Sales	Wholesale Cost Total	Retail Price Total	# of Sales	Wholesale Cost Total	Retail Price Total	# of Sales	Wholesale Cost Total	Retail Price Total
Clerk									
Joe Agile	2	$392.00	$1,290.00	11	$390.08	$814.60	13	$782.08	$2,104.60
Carol Burley	4	$830.02	$2,599.90	5	$163.04	$314.85	9	$993.06	$2,914.75
Amos Clever	.	.	.	28	$988.48	$2,154.00	28	$988.48	$2,154.00
Over All Clerks	6	$1,222.02	$3,889.90	44	$1,541.60	$3,283.45	50	$2,763.62	$7,173.35

More Complex Tables and Percentages

- PROC TABULATE has certain built-in percentage statistics (REPPCTN, REPPCTSUM, COLPCTN, COLPCTSUM, ROWPCTN, and ROWPCTSUM) that enable you to easily add percentages to your tables.

- Additional percentage statistics PCTN and PCTSUM can be used with denominator specifications to compute non-standard percentages.

- An indicator variable is a variable with values 0 and 1, with 1 signaling a certain state and 0 the lack of that state (for example, coronary heart disease and its absence, or drug group and placebo group). Indicator variables can be used to compute certain special types of percentages in PROC TABULATE.

Using PROC TABULATE's Built-in Percentages

- PROC TABULATE can compute two general kinds of percentages: PCTN and PCTSUM.

- Use some type of PCTN when you want to know what % of the total number of observations is in each cell of the table.

- Use some type of PCTSUM when you want to know what % of the total sum of an analysis variable is in each cell of the table.

- Using PCTN and PCTSUM can be tricky, as will be shown later, but PROC TABULATE provides six built-in types of percentages for the following purposes:

    - If all count %'s in your table should add up to 100, use REPPCTN. (This is the same behavior as a simple PCTN.)
    - If all sum %'s in your table should add up to 100, use REPPCTSUM. (This is the same behavior as a simple PCTSUM.)

    - If all count %'s in a column should add up to 100, use COLPCTN.
    - If all sum %'s in a column should add up to 100, use COLPCTSUM.

    - If all count %'s in a row should add up to 100, use ROWPCTN.
    - If all sum %'s in a row should add up to 100, use ROWPCTSUM.

- Note that if CLASS variables are stacked rather than crossed (stacked=sex age, crossed=sex*age), PROC TABULATE considers each breakdown a separate logical table even though they appear in the same physical table.

- The NOSEPS option used below removes all horizontal lines from the body of the table.

```
PROC TABULATE DATA=bios511.hw4 FORMAT=6.2
                STYLE={CELLWIDTH=.65in};
     CLASS cvd trt;
     TABLE trt, cvd*(N*F=3.0 REPPCTN);
     TABLE trt, cvd*(N*F=3.0 COLPCTN);
     TABLE trt, cvd*(N*F=3.0 ROWPCTN);
     TITLE 'Seeing Behavior of REPPCTN, COLPCTN, and ROWPCTN';
RUN;
```

Seeing Behavior of REPPCTN, COLPCTN, and ROWPCTN

	cardiovascular disease(0=no 1=yes)			
	0		1	
	N	RepPctN	N	RepPctN
treatment group(0=placebo 1=drug)				
0	118	59.00	17	8.50
1	55	27.50	10	5.00

Seeing Behavior of REPPCTN, COLPCTN, and ROWPCTN

	cardiovascular disease(0=no 1=yes)			
	0		1	
	N	ColPctN	N	ColPctN
treatment group(0=placebo 1=drug)				
0	118	68.21	17	62.96
1	55	31.79	10	37.04

Seeing Behavior of REPPCTN, COLPCTN, and ROWPCTN

	cardiovascular disease(0=no 1=yes)			
	0		1	
	N	RowPctN	N	RowPctN
treatment group(0=placebo 1=drug)				
0	118	87.41	17	12.59
1	55	84.62	10	15.38

```
PROC TABULATE DATA=bios511.sales FORMAT=7.2
           STYLE={CELLWIDTH=.8in};
     CLASS dept clerk;
     VAR price;
     TABLE clerk, dept*price*(SUM REPPCTSUM);
     TABLE clerk, dept*price*(SUM COLPCTSUM);
     TABLE clerk, dept*price*(SUM ROWPCTSUM);
     TITLE 'Seeing Behavior of REPPCTSUM, COLPCTSUM, and
             ROWPCTSUM';
RUN;
```

Seeing Behavior of REPPCTSUM, COLPCTSUM, and ROWPCTSUM

	DEPT			
	FURS		SHOES	
	PRICE		PRICE	
	Sum	RepPctSum	Sum	RepPctSum
CLERK				
AGILE	1290.00	17.98	814.60	11.36
BURLEY	2599.90	36.24	314.85	4.39
CLEVER	.	.	2154.00	30.03

Seeing Behavior of REPPCTSUM, COLPCTSUM, and ROWPCTSUM

	DEPT			
	FURS		SHOES	
	PRICE		PRICE	
	Sum	ColPctSum	Sum	ColPctSum
CLERK				
AGILE	1290.00	33.16	814.60	24.81
BURLEY	2599.90	66.84	314.85	9.59
CLEVER	.	.	2154.00	65.60

Seeing Behavior of REPPCTSUM, COLPCTSUM, and ROWPCTSUM

	DEPT			
	FURS		SHOES	
	PRICE		PRICE	
	Sum	RowPctSum	Sum	RowPctSum
CLERK				
AGILE	1290.00	61.29	814.60	38.71
BURLEY	2599.90	89.20	314.85	10.80
CLEVER	.	.	2154.00	100.00

```
PROC TABULATE DATA=bios511.hw4 FORMAT=6.2
       STYLE={CELLWIDTH=.65in};
     CLASS cvd trt;
     TABLE trt ALL, cvd*(N*F=3.0 REPPCTN) ALL*(N*F=3.0 REPPCTN);
     /* or  TABLE trt ALL, (cvd ALL) * (N*F=3.0 REPPCTN); */
     TABLE trt ALL, cvd*(N*F=3.0 COLPCTN) ALL*(N*F=3.0 COLPCTN);
     TABLE trt ALL, cvd*(N*F=3.0 ROWPCTN) ALL*(N*F=3.0 ROWPCTN);
     TITLE 'Adding Rows and Columns of Totals with REPPCTN,
              COLPCTN, and ROWPCTN';
RUN;
```

Adding Rows and Columns of Totals with REPPCTN, COLPCTN, and ROWPCTN

	cardiovascular disease(0=no 1=yes)				All	
	0		1			
	N	RepPctN	N	RepPctN	N	RepPctN
treatment group(0=placebo 1=drug)						
0	118	59.00	17	8.50	135	67.50
1	55	27.50	10	5.00	65	32.50
All	173	86.50	27	13.50	200	100.00

Adding Rows and Columns of Totals with REPPCTN, COLPCTN, and ROWPCTN

	cardiovascular disease(0=no 1=yes)				All	
	0		1			
	N	ColPctN	N	ColPctN	N	ColPctN
treatment group(0=placebo 1=drug)						
0	118	68.21	17	62.96	135	67.50
1	55	31.79	10	37.04	65	32.50
All	173	100.00	27	100.00	200	100.00

Adding Rows and Columns of Totals with REPPCTN, COLPCTN, and ROWPCTN

	cardiovascular disease(0=no 1=yes)				All	
	0		1			
	N	RowPctN	N	RowPctN	N	RowPctN
treatment group(0=placebo 1=drug)						
0	118	87.41	17	12.59	135	100.00
1	55	84.62	10	15.38	65	100.00
All	173	86.50	27	13.50	200	100.00

Percentages in a Table within a Table

Tables within a table occur when you have stacked CLASS variables. For example, consider this example.

```
PROC FORMAT;
     VALUE agecat 20-<30='20-29'
                  30-<40='30-39'
                  40-<50='40-49';
RUN;

PROC TABULATE DATA=bios511.fitness STYLE={CELLWIDTH=.5in};
     CLASS sex age;
     CLASSLEV sex / STYLE={CELLWIDTH=1in};
     TABLE sex age, N;
     FORMAT age agecat.;
RUN;
```

Sex and age tables stacked

	N
Sex	
F	24
M	19
Age	
20-29	23
30-39	11
40-49	9

Here, REPPCTN works since we want percentages in each logical table to add up to 100%. Note that PCTN or COLPCTN would produce the same result.

```
PROC TABULATE DATA=bios511.fitness FORMAT=6.2
        STYLE={CELLWIDTH=.5in};
    CLASS sex age;
    CLASSLEV sex / STYLE={CELLWIDTH=1in};
    KEYLABEL REPPCTN='%';
    TABLE sex age, N*F=2.0 REPPCTN;
    FORMAT age agecat.;
    TITLE 'Sex and age tables stacked - adding percentages';
RUN;
```

Sex and age tables stacked - adding percentages

	N	%
Sex		
F	24	55.81
M	19	44.19
Age		
20-29	23	53.49
30-39	11	25.58
40-49	9	20.93

Specifying a Denominator with PCTN

- As illustrated above, the standard denominators for PCTN are all observations in the table, all observations in a row, or all observations in a column.

- If you want to use a non-standard denominator for a PCTN calculation, you can do so by specifying PCTN<denominator>. **A signal that you should use PCTN<denominator> is that the percentages in a <u>portion</u> of a logical table should add up to 100.**

- For example, in this table with crossed variables (sex*age) the age percentages for males add up to 100, as do the age percentages for females. This is a signal that the built-in percentages will not work, and we would need to provide our own denominator specification to produce this table.

		N	%
Sex	Age		
F	20-29	10	41.67
	30-39	9	37.50
	40-49	5	20.83
M	20-29	13	68.42
	30-39	2	10.53
	40-49	4	21.05

Using the FITNESS data set, let's look at how you could produce several variations on the sex / age breakdown table shown above, including adding an additional grouping variable.

1. Sex / age breakdown table with age group percentages by sex

In this table, we want the percentages for all age groups within each sex group to add up to 100. Use PCTN<age> to get the correct denominator (which is the total number of people of all ages in the current sex group).

```
PROC FORMAT;
     VALUE agecat 20-<30='20-29'
                  30-<40='30-39'
                  40-<50='40-49';
RUN;

PROC TABULATE DATA=fitness STYLE={CELLWIDTH=.5in};
     CLASS sex age;
     CLASSLEV sex age / STYLE={CELLWIDTH=1in};
     KEYLABEL PCTN='%';
     TABLE sex*age, N*F=2.0 PCTN<age>*F=5.2;
     FORMAT age agecat.;
RUN;
```

		N	%
Sex	**Age**		
F	**20-29**	10	41.67
	30-39	9	37.50
	40-49	5	20.83
M	**20-29**	13	68.42
	30-39	2	10.53
	40-49	4	21.05

For this particular type of table, the correct denominator specification is the nested variable: that is, the second one in the crossing.

2. Sex / age breakdown table with age group percentages overall

In this table, we want the age group percentages across both sex groups to add up to 100. In the old days (before SAS provided the six built-in percentages), we would have had to figure out to use PCTN<sex*age> to get the right denominator. Now, however, note that we want the entire % column to add up to 100, which is our signal to use COLPCTN. And it works! The two TABLE statements generate identical percentages.

```
PROC TABULATE DATA=fitness STYLE={CELLWIDTH=.5in};
     CLASS sex age;
     CLASSLEV sex age / STYLE={CELLWIDTH=1in};
     KEYLABEL PCTN='%' COLPCTN='%';
     TABLE sex*age, N*F=2.0 PCTN<sex*age>*F=5.2;
     TABLE sex*age, N*F=2.0 COLPCTN*F=5.2;
     FORMAT age agecat.;
RUN;
```

		N	%
Sex	Age		
F	20-29	10	23.26
	30-39	9	20.93
	40-49	5	11.63
M	20-29	13	30.23
	30-39	2	4.65
	40-49	4	9.30

Two identical tables like the above are produced.

Sex / age breakdown table with age group pcts by sex, including subtotals

This example is like the first one except that we want to show subtotals for each sex. Since we are concatenating the universal class variable ALL with our nested age groups, we need to add ALL to the denominator specification also.

```
PROC TABULATE DATA=fitness STYLE={CELLWIDTH=.5in};
     CLASS sex age;
     CLASSLEV sex age / STYLE={CELLWIDTH=1in};
     KEYLABEL PCTN='%';
     TABLE sex*(age ALL), N*F=2.0 PCTN<age ALL>*F=6.2;
     FORMAT age agecat.;
RUN;
```

		N	%
Sex	Age		
F	20-29	10	41.67
	30-39	9	37.50
	40-49	5	20.83
	All	24	100.00
M	Age		
	20-29	13	68.42
	30-39	2	10.53
	40-49	4	21.05
	All	19	100.00

3. Adding an additional grouping variable

A very neat feature of this sex / age breakdown table is that you can easily modify the code to produce the table for each value of an additional grouping variable. If we want to see the sex / age breakdown of the students for each workout instructor, we simply need to add TEACHER as a CLASS variable and cross the column specification with TEACHER in the TABLE statement.

```
PROC TABULATE DATA=fitness STYLE={CELLWIDTH=.5in};
     CLASS sex age teacher;
     CLASSLEV sex age / STYLE={CELLWIDTH=.5in};
     KEYLABEL PCTN='%';
     TABLE sex*(age ALL), teacher*(N*F=5.0 PCTN<age ALL>*F=6.2)
     FORMAT age agecat.;
RUN;
```

		Teacher							
		Amund		Czika		Reed		Yang	
		N	%	N	%	N	%	N	%
Sex	Age								
F	20-29	4	40.00	4	57.14	2	40.00	.	.
	30-39	3	30.00	2	28.57	3	60.00	1	50.00
	40-49	3	30.00	1	14.29	.	.	1	50.00
	All	10	100.00	7	100.00	5	100.00	2	100.00
M	Age								
	20-29	5	83.33	4	100.00	1	33.33	3	50.00
	30-39	.	.	.	.	.	.	2	33.33
	40-49	1	16.67	.	.	2	66.67	1	16.67
	All	6	100.00	4	100.00	3	100.00	6	100.00

Using Indicator Variables

- The mean value of an indicator variable is the proportion of observations with the indicated state.

- For example, consider a data set of 20 people that includes an indicator variable for the disease diabetes. If 5 of the people have diabetes and therefore a 1 for the indicator variable (the other 15 having 0), the mean value of the diabetes indicator is ((5*1)+(15*0))/20 = 5/20 = .25, which is the proportion of the data set having diabetes.

This interesting example also illustrates combining statistics such as mean and standard deviation in the same table with proportions.

```
PROC FORMAT;
      VALUE trtft 0='Placebo'
                  1='Drug';
      VALUE yn    0='No'
                  1='Yes';
RUN;

PROC TABULATE DATA=bios511.hw4;
      CLASS trt;
      VAR cvd hdl;
      TABLE (trt=' ' ALL='All Subjects'), (hdl='HDL'*(MEAN STD)
            (cvd=' '*MEAN='% CVD'*F=6.4)) / BOX='Treatment';
      FORMAT trt trtft.;
      TITLE 'Illustrate that the mean of a 0/1 variable (CVD) is
             the proportion having 1';
      TITLE2 'Also show combining statistics and proportions in
              the same table';
RUN;

PROC FREQ DATA=bios511.hw4;
      TABLES trt*cvd / NOPCT NOCOL;
      LABEL trt='Treatment' CVD='CVD';
      FORMAT trt trtft. cvd yn.;
      TITLE 'PROC FREQ output to confirm PROC TABULATE output';
RUN;
```

Illustrate that the mean of a 0/1 variable (CVD) is the proportion having 1
Also show combining statistics and proportions in the same table

Treatment	HDL		
	Mean	Std	% CVD
Placebo	43.66	8.32	0.1259
Drug	42.40	7.81	0.1538
All Subjects	43.25	8.16	0.1350

PROC FREQ output to confirm PROC TABULATE output

The FREQ Procedure

Frequency Row Pct	Table of TRT by CVD		
	CVD(CVD)		
TRT(Treatment)	No	Yes	Total
Placebo	118 87.41	17 12.59	135
Drug	55 84.62	10 15.38	65
Total	173	27	200

- If you first multiply the indicator variable by 100, then the value computed is a percentage (between 0 and 100) rather than a proportion (between 0 and 1).

```
DATA hw4mult;
SET bios511.hw4;
cvd100 = cvd*100;
RUN;

PROC TABULATE NOSEPS DATA=hw4mult;
      CLASS trt;
      VAR cvd100 hdl;
      TABLE (trt=' ' ALL='All Subjects'), (hdl='HDL'*(MEAN STD)
            (cvd100=' '*MEAN='% CVD'*F=5.2)) / BOX='Treatment';
      FORMAT trt trtft.;
      TITLE 'Illustrate that the mean of a 0/100 variable (CVD100) created
               from';
TITLE2 'an indicator variable is a percentage';
RUN;
```

Illustrate that the mean of a 0/100 variable (CVD100) created from an indicator variable is a percentage

Treatment	HDL		
	Mean	Std	% CVD
Placebo	43.66	8.32	12.59
Drug	42.40	7.81	15.38
All Subjects	43.25	8.16	13.50

- Another way to create the same percentages is to add a column of 1's to the data set, and use the sum of that column as a denominator for the sum of the indicator variable (which is a combination of 0's and 1's).

```
DATA hw4dummy;
      SET bios511.hw4;
      dummy=1;
RUN;

PROC TABULATE DATA=hw4dummy STYLE={CELLWIDTH=.75in};
      CLASS trt;
      CLASSLEV trt / STYLE={CELLWIDTH=.75in};
      VAR cvd dummy;
      TABLE trt='' ALL, cvd=''*(SUM='# with CVD'*F=2.0
                                PCTSUM<dummy>='% with CVD');
      TITLE "Illustrate use of a column of 1's in computing percentages";
      FORMAT trt trtft.;
RUN;
```

Illustrate use of a column of 1's in computing percentages

	# with CVD	% with CVD
Placebo	17	12.59
Drug	10	15.38
All	27	13.50

Controlling the Order of Formatted CLASS Values

Say you want to group values of a variable for use as categories in a table. For example, maybe you want to look at sales in your store according to whether the clerks are experienced or inexperienced, or you want to look at sales by week of the month. You could do this grouping with formats, as below.

```
PROC FORMAT;
   VALUE $exp  'BURLEY'='Experienced'
               'AGILE','CLEVER'='Inexperienced';
   VALUE da    1-7  ='First week'
               8-14 ='Second week'
               15-21='Third week'
               22-28='Fourth week';
RUN;
```

Let's apply these formats in producing some simple tables.

```
PROC TABULATE DATA=bios511.sales;
    VAR price;
    CLASS clerk day;
    TABLE clerk,price*MEAN;
    TABLE day='',price*MEAN;
    FORMAT clerk $exp. day da.;
    TITLE 'Using default order of formatted CLERK and DAY';
RUN;
```

Using default order of formatted CLERK and DAY

	PRICE
	Mean
CLERK	
Inexperienced	103.87
Experienced	323.86

Using default order of formatted CLERK and DAY

	PRICE
	Mean
First week	122.70
Second week	172.32
Third week	155.59
Fourth week	117.82

Notice that in each case, the categories are ordered by the underlying values, which in the case of the second table is good. But in the first table, the most logical order would be alphabetical. To obtain this order, we can use the ORDER= option on the CLASS statement.

```
PROC TABULATE DATA=bios511.sales;
    VAR price;
    CLASS clerk / ORDER=FORMATTED;
    CLASS day;
    TABLE clerk,price*MEAN;
    TABLE day='',price*MEAN;
    FORMAT clerk $exp. day da.;
    TITLE 'Using default order of DAY but formatted order of
               CLERK';
RUN;
```

Using default order of DAY but formatted order of CLERK

	PRICE
	Mean
CLERK	
Experienced	323.86
Inexperienced	103.87

Using default order of DAY but formatted order of CLERK

	PRICE
	Mean
First week	122.70
Second week	172.32
Third week	155.59
Fourth week	117.82

The default value of the ORDER= option is UNFORMATTED, which produces the same order as sorting the original values by PROC SORT.

In general, if you are creating a new variable and want controlling the order of display to be simple, make the variable numeric with values 1, 2, 3, etc. Then use a format to assign character value labels. SAS will by default display the "values" in the order of the underlying numbers. If you had instead created a character variable, SAS would display the values in alphabetical order rather than your preferred order, and the preferred order would be difficult to achieve.

Indenting CLASS Variable Values in the Row Dimension

```
PROC FORMAT;
    VALUE $cft  'AGILE' ="Joe Agile"
                'BURLEY'="Carol Burley"
                'CLEVER'="Amos Clever";
    VALUE $dft  'FURS'  ="Furs"
                'SHOES' ="Shoes";
RUN;

PROC TABULATE DATA=bios511.sales;
    VAR price;
    CLASS dept clerk;
    LABEL dept='Department'  clerk='Clerk';
    TABLES dept clerk,price=' '*MEAN='Average Price of Items
                                      Sold';
    FORMAT dept $dft. clerk $cft.;
RUN;
```

	Average Price of Items Sold
Department	
Furs	648.32
Shoes	74.62
Clerk	
Joe Agile	161.89
Carol Burley	323.86
Amos Clever	76.93

This table looks pretty nice, but it would be even clearer if the CLASS variable values – Furs and Shoes for DEPT, and the clerk names for CLERK – were indented a few spaces.

As a first attempt to do this, you might try adding some blanks to the beginning of the value labels in your format definitions, like this:

```
PROC FORMAT;
    VALUE $cft  'AGILE' ="   Joe Agile"
                'BURLEY'="   Carol Burley"
                'CLEVER'="   Amos Clever";
    VALUE $dft  'FURS'  ="   Furs"
                'SHOES' ="   Shoes";
RUN;
```

However, you find that this has no effect when you rerun the table – PROC TABULATE gets rid of the leading blanks.

A way to insert blanks that PROC TABULATE does NOT get rid of is the following (additions bolded):

```
%LET nbs=%sysfunc(byte(160)) ; *Non-breaking space;
%LET space=&nbs&nbs&nbs;

PROC FORMAT;
    VALUE $cft  'AGILE' ="&space.Joe Agile"
                'BURLEY'="&space.Carol Burley"
                'CLEVER'="&space.Amos Clever";
    VALUE $dft  'FURS'  ="&space.Furs"
                'SHOES' ="&space.Shoes";
RUN;


PROC TABULATE DATA=bios511.sales NOSEPS;
    VAR price;
    CLASS dept clerk;
    LABEL dept='Department'  clerk='Clerk';
    TABLES dept clerk,price=' '*MEAN='Average Price of Items
                                      Sold';
    FORMAT dept $dft. clerk $cft.;
RUN;
```

Note that this solution works only for Windows SAS! The code byte(160) does not stand for a non-breaking space in University Edition SAS.

Note also that some versions of Windows may not recognize the byte(160) code either. See the third solution on the next page in that case.

For University Edition SAS, use:

```
PROC FORMAT;
    VALUE $cft  'AGILE' ="   Joe Agile"
                'BURLEY'="   Carol Burley"
                'CLEVER'="   Amos Clever";
    VALUE $dft  'FURS'  ="   Furs"
                'SHOES' ="   Shoes";
RUN;

PROC TABULATE DATA=bios511.sales NOSEPS;
    VAR price;
    CLASS dept clerk;
    CLASSLEV dept clerk / style=[asis=on];
    LABEL dept='Department'  clerk='Clerk';
    TABLES dept clerk,price=' '*MEAN='Average Price of Items
                                      Sold';
    FORMAT dept $dft. clerk $cft.;
RUN;
```

Note that while this will not produce an error with Windows SAS, this solution does not work perfectly on Windows SAS. Extra unwanted lines are added.

The following method should work on either Windows SAS or University Edition SAS:

```
PROC FORMAT;
    VALUE $cft  'AGILE' ="   Joe Agile"
                'BURLEY'="   Carol Burley"
                'CLEVER'="   Amos Clever";
    VALUE $dft  'FURS'  ="   Furs"
                'SHOES' ="   Shoes";
RUN;

PROC TABULATE DATA=bios511.sales NOSEPS;
    VAR price;
    CLASS dept clerk;
    CLASSLEV dept clerk /
        style={nobreakspace=on htmlstyle="white-space:nowrap"};
    LABEL dept='Department'  clerk='Clerk';
    TABLES dept clerk,price=' '*MEAN='Average Price of Items
                                      Sold';
    FORMAT dept $dft. clerk $cft.;
RUN;
```

The resulting table from each of these methods is:

	Average Price of Items Sold
Department	
Furs	648.32
Shoes	74.62
Clerk	
Joe Agile	161.89
Carol Burley	323.86
Amos Clever	76.93