CHAPTER

37

The TABULATE Procedure

Overview

The TABULATE procedure displays descriptive statistics in tabular format, using some or all of the variables in a data set. You can create a variety of tables ranging from simple to highly customized.

PROC TABULATE computes many of the same statistics that are computed by other descriptive statistical procedures such as MEANS, FREQ, and REPORT. PROC TABULATE provides

- simple but powerful methods to create tabular reports
- flexibility in classifying the values of variables and establishing hierarchical relationships between the variables
- mechanisms for labeling and formatting variables and procedure-generated statistics.

Output 37.1 on page 1152 shows a simple table that was produced by PROC TABULATE. The data set on page 1199 contains data on expenditures of energy by two types of customers, residential and business, in individual states in the Northeast (1) and West (4) regions of the United States. The table sums expenditures for states within a geographic division. (The RTS option provides enough space to display the column headers without hyphenating them.)

```
options nodate pageno=1 linesize=64
        pagesize=40;

proc tabulate data=energy;
   class region division type;
   var expenditures;
   table region*division, type*expenditures /
         rts=20;
run;
```

Output 37.1 Simple Table Produced by PROC TABULATE

The SAS System 1

		Type	
		1	2
		Expenditures	Expenditures
		Sum	Sum
Region	Division		
1	1	7477.00	5129.00
	2	19379.00	15078.00
4	3	5476.00	4729.00
	4	13959.00	12619.00

Output 37.2 on page 1153 is a more complicated table using the same data set that was used to create Output 37.1 on page 1152. The statements that create this report

- customize column and row headers
- apply a format to all table cells
- sum expenditures for residential and business customers
- compute subtotals for each division
- compute totals for all regions.

For an explanation of the program that produces this report, see Example 6 on page 1212.

Output 37.2 Complex Table Produced by PROC TABULATE

Energy Expenditures for Each Region 1
(millions of dollars)

		Customer Base		
		Residential Customers	Business Customers	All Customers
Region	Division			
Northeast	New England	7,477	5,129	12,606
	Middle Atlantic	19,379	15,078	34,457
	Subtotal	26,856	20,207	47,063
West	Division			
	Mountain	5,476	4,729	10,205
	Pacific	13,959	12,619	26,578
	Subtotal	19,435	17,348	36,783
Total for All Regions		$46,291	$37,555	$83,846

Display 37.1 on page 1154 shows a table created with HTML. Beginning with Version 7 of the SAS System, you can use the Output Delivery System to create customized HTML files from PROC TABULATE. For an explanation of the program that produces this table, see Example 14 on page 1243.

Display 37.1 HTML Table Produced by PROC TABULATE

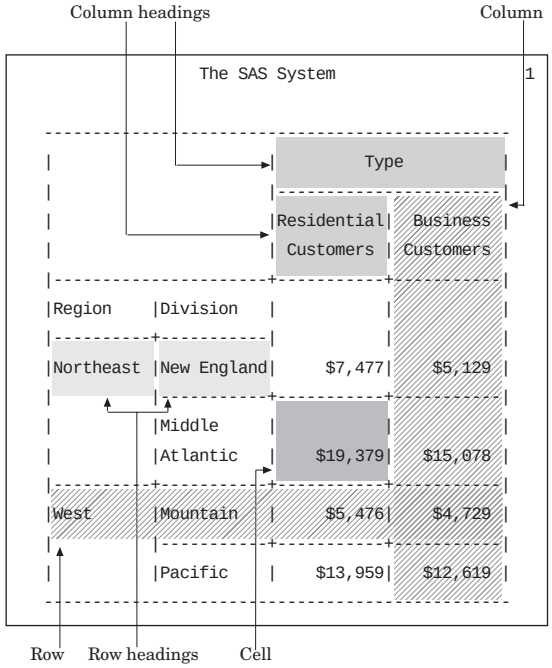
Energy Expenditures
(millions of dollars)

Region by Division by Type		Type		Total
		Residential Customers	Business Customers	
		Expenditures	Expenditures	Expenditures
		Sum	Sum	Sum
Region	Division			
Northeast	New England	**$7,477**	**$5,129**	**$12,606**
	Middle Atlantic	**$19,379**	**$15,078**	**$34,457**
	Total	$26,856	$20,207	$47,063
West	Division			
	Mountain	**$5,476**	**$4,729**	**$10,205**
	Pacific	**$13,959**	**$12,619**	**$26,578**
	Total	$19,435	$17,348	$36,783
Total	Division			
	New England	**$7,477**	**$5,129**	**$12,606**
	Middle Atlantic	**$19,379**	**$15,078**	**$34,457**
	Mountain	**$5,476**	**$4,729**	**$10,205**
	Pacific	**$13,959**	**$12,619**	**$26,578**
	Total	$46,291	$37,555	$83,846

Terminology

Figure 37.1 on page 1155 illustrates some of the terms that are commonly used in discussions of PROC TABULATE.

Figure 37.1 Illustration of Terms Used to Discuss PROC TABULATE



In addition, the following terms frequently appear in discussions of PROC TABULATE:

category
: the combination of unique values of class variables. The TABULATE procedure creates a separate category for each unique combination of values that exists in the observations of the data set. Each category that is created by PROC TABULATE is represented by one or more cells in the table where the pages, rows, and columns that describe the category intersect.

  The table in Figure 37.1 on page 1155 contains three class variables: Region, Division, and Type. These class variables form the eight categories listed in Table 37.1 on page 1156. (For convenience, the categories are described in terms of their formatted values.)

Table 37.1 Categories Created from Three Class Variables

Region	Division	Type
Northeast	New England	Residential Customers
Northeast	New England	Business Customers
Northeast	Middle Atlantic	Residential Customers
Northeast	Middle Atlantic	Business Customers
West	Mountain	Residential Customers
West	Mountain	Business Customers
West	Pacific	Residential Customers
West	Pacific	Business Customers

continuation message
is the text that appears below the table if it spans multiple physical pages.

A continuation message has a style. The default style is *Aftercaption*. For more information about using styles, see STYLE= on page 1164 in the PROC TABULATE statement and "Using Style Elements in PROC TABULATE" on page 1188.

nested variable
a variable whose values appear in the table with each value of another variable.

In Figure 37.1 on page 1155, Division is nested under Region.

page dimension text
is the text that appears above the table if the table has a page dimension. However, if you specify BOX=_PAGE_ in the TABLE statement, the text that would appear above the table appears in the box.

Page dimension text has a style. The default style is *Beforecaption*. For more information about using styles, see STYLE= on page 1164 in the PROC TABULATE statement and "Using Style Elements in PROC TABULATE" on page 1188.

subtable
the group of cells that is produced by crossing a single element from each dimension of the TABLE statement when one or more dimensions contain concatenated elements.

Figure 37.1 on page 1155 contains no subtables. For an illustration of a table that is composed of multiple subtables, see Figure 37.17 on page 1238.

Procedure Syntax

Requirements: At least one TABLE statement is required.

Requirements: Depending on the variables that appear in the TABLE statement, a CLASS statement, a VAR statement, or both are required.

Tip: Supports the Output Delivery System (see Chapter 2, "Fundamental Concepts for Using Base SAS Procedures")

Reminder: You can use the ATTRIB, FORMAT, LABEL, and WHERE statements. See Chapter 3, "Statements with the Same Function in Multiple Procedures," for details. You can also use any global statements as well. See Chapter 2, "Fundamental Concepts for Using Base SAS Procedures," for a list.

PROC TABULATE <*option(s)*>;

BY <DESCENDING> *variable-1*
<…<DESCENDING> *variable-n*>
<NOTSORTED>;

CLASS *variable(s)* </ *options*>;

CLASSLEV *variable(s)* / *style* =<*style-element-name* | <PARENT>>
<[*style-attribute-specification(s)*]>;

FREQ *variable*;

KEYLABEL *keyword-1='description-1'*
<…*keyword-n='description-n'*>;

KEYWORD *keyword(s)* / *style* =<*style-element-name* | <PARENT>>
<[*style-attribute-specification(s)*]>;

TABLE <<*page-expression*,> *row-expression*,> *column-expression* </ *table-option(s)*>;

VAR *analysis-variable(s)*</ *options*>;

WEIGHT *variable*;

To do this	Use this statement
Create a separate table for each BY group	BY
Identify variables in the input data set as class variables	CLASS
Specify a style for class variable level value headings	CLASSLEV
Identify a variable in the input data set whose values represent the frequency of each observation	FREQ
Specify a label for a keyword	KEYLABEL
Specify a style for keyword headings	KEYWORD
Describe the table to create	TABLE

To do this	Use this statement
Identify variables in the input data set as analysis variables	VAR
Identify a variable in the input data set whose values weight each observation in the statistical calculations	WEIGHT

PROC TABULATE Statement

PROC TABULATE <*option(s)*>;

To do this	Use this option
Customize the HTML contents link to the output	CONTENTS=
Specify the input data set	DATA=
Disable floating point exception recovery	NOTRAP
Specify the output data set	OUT=
Enable floating point exception recovery	TRAP
Identify categories of data that are of interest	
Specify a secondary data set that contains the combinations of values of class variables to include in tables and output data sets	CLASSDATA=
Exclude from tables and output data sets all combinations of class variable values that are not in the CLASSDATA= data set	EXCLUSIVE
Consider missing values as valid values for class variables	MISSING
Control the statistical analysis	
Exclude observations with nonpositive weights	EXCLNPWGTS
Specify the sample size to use for the P2 quantile estimation method	QMARKERS=
Specify the quantile estimation method	QMETHOD=
Specify the mathematical definition to calculate quantiles	QNTLDEF=
Specify the variance divisor	VARDEF=
Customize the appearance of the table	
Specify a default format for each cell in the table	FORMAT=
Define the characters to use to construct the table outlines and dividers	FORMCHAR=
Eliminate horizontal separator lines from the row titles and the body of the table	NOSEPS

To do this	Use this option
Order the values of a class variable according to the specified order	ORDER=
Specify the default style element or style elements (for the Output Delivery System) to use for each cell of the table	STYLE=

Options

CLASSDATA=*SAS-data-set*
specifies a data set that contains the combinations of values of the class variables that must be present in the output. Any combinations of values of the class variables that occur in the CLASSDATA= data set but not in the input data set appear in each table or output data set and have a frequency of zero.

Restriction: The CLASSDATA= data set must contain all class variables. Their data type and format must match the corresponding class variables in the input data set.

Interaction: If you use the EXCLUSIVE option, PROC TABULATE excludes any observations in the input data set whose combinations of class variables is not in the CLASSDATA= data set.

Tip: Use the CLASSDATA= data set to filter or supplement the input data set.

Featured in: Example 2 on page 1201

CONTENTS=*link-name*
allows you to name the link in the HTML table of contents that points to the ODS output of the first table that was produced using the TABULATE procedure.

Restrictions: CONTENTS= has no effect on TABULATE procedure reports.

DATA=*SAS-data-set*
specifies the input data set.

Main Discussion: "Input Data Sets" on page 18

EXCLNPWGTS
excludes observations with nonpositive weight values (zero or negative) from the analysis. By default, PROC TABULATE treats observations with negative weights like those with zero weights and counts them in the total number of observations.

Alias: EXCLNPWGT

See also: WEIGHT= on page 1180 and "WEIGHT Statement" on page 1181

EXCLUSIVE
excludes from the tables and the output data sets all combinations of the class variable that are not found in the CLASSDATA= data set.

Requirement: If a CLASSDATA= data set is not specified, this option is ignored.

Featured in: Example 2 on page 1201

FORMAT=*format-name*
specifies a default format for the value in each table cell. You can use any SAS or user-defined format.

Default: If you omit FORMAT=, PROC TABULATE uses BEST12.2 as the default format.

Interaction: Formats that are specified in a TABLE statement override the format that is specified with FORMAT=.

Tip: This option is especially useful for controlling the number of print positions that are used to print a table.

Featured in: Example 1 on page 1199 and Example 6 on page 1212

FORMCHAR <(*position(s)*)>=*'formatting-character(s)'*

defines the characters to use for constructing the table outlines and dividers.

position(s)

identifies the position of one or more characters in the SAS formatting-character string. A space or a comma separates the positions.

Default: Omitting *position(s)* is the same as specifying all 20 possible SAS formatting characters, in order.

Range: PROC TABULATE uses 11 of the 20 formatting characters that SAS provides. Table 37.2 on page 1160 shows the formatting characters that PROC TABULATE uses. Figure 37.2 on page 1161 illustrates the use of each formatting character in the output from PROC TABULATE.

formatting-character(s)

lists the characters to use for the specified positions. PROC TABULATE assigns characters in *formatting-character(s)* to *position(s)*, in the order that they are listed. For example, the following option assigns the asterisk (*) to the third formatting character, the pound sign (#) to the seventh character, and does not alter the remaining characters:

```
formchar(3,7)='*#'
```

Interaction: The SAS system option FORMCHAR= specifies the default formatting characters. The system option defines the entire string of formatting characters. The FORMCHAR= option in a procedure can redefine selected characters.

Tip: You can use any character in *formatting-characters*, including hexadecimal characters. If you use hexadecimal characters, you must put an **x** after the closing quote. For instance, the following option assigns the hexadecimal character 2D to the third formatting character, the hexadecimal character 7C to the seventh character, and does not alter the remaining characters:

```
formchar(3,7)='2D7C'x
```

Tip: Specifying all blanks for *formatting-character(s)* produces tables with no outlines or dividers.

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

See also: For more information on formatting output, see Chapter 5 "Controlling the Table's Appearance" in the *SAS Guide to TABULATE Processing*.

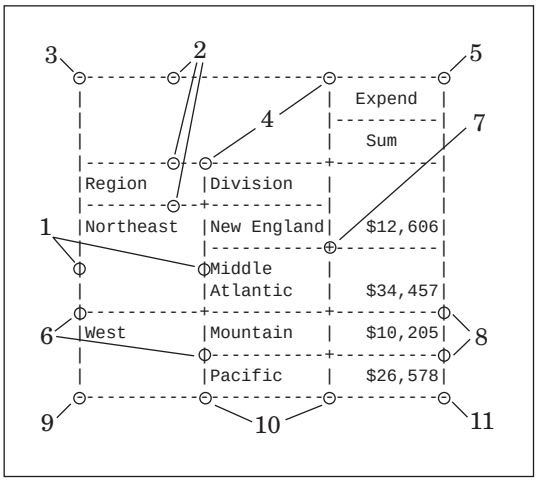
For information on which hexadecimal codes to use for which characters, consult the documentation for your hardware.

Table 37.2 Formatting Characters Used by PROC TABULATE

Position	Default	Used to draw
1	|	the right and left borders and the vertical separators between columns
2	-	the top and bottom borders and the horizontal separators between rows

Position	Default	Used to draw
3	-	the top character in the left border
4	-	the top character in a line of characters that separate columns
5	-	the top character in the right border
6	|	the leftmost character in a row of horizontal separators
7	+	the intersection of a column of vertical characters and a row of horizontal characters
8	|	the rightmost character in a row of horizontal separators
9	-	the bottom character in the left border
10	-	the bottom character in a line of characters that separate columns
11	-	the bottom character in the right border

Figure 37.2 Formatting Characters in PROC TABULATE Output



MISSING

considers missing values as valid values to create the combinations of class variables. Special missing values that are used to represent numeric values (the letters A through Z and the underscore (_) character) are each considered as a separate value. A heading for each missing value appears in the table.

Default: If you omit MISSING, PROC TABULATE does not include observations with a missing value for any class variable in the report.

Main Discussion: "Including Observations with Missing Class Variables" on page 1192

See also: *SAS Language Reference: Concepts* for a discussion of missing values that have special meaning.

NOSEPS
eliminates horizontal separator lines from the row titles and the body of the table. Horizontal separator lines remain between nested column headers.

Tip: If you want to replace the separator lines with blanks rather than remove them, use the FORMCHAR= option on page 1160.

Featured in: Example 8 on page 1216

NOTRAP
disables floating point exception (FPE) recovery during data processing. Note that normal SAS System FPE handling is still in effect so that PROC TABULATE terminates in the case of math exceptions.

Default: FPE recovery is disabled.

Tip: In operating environments where the overhead of FPE recovery is significant, NOTRAP can improve performance.

See also: TRAP on page 1165

ORDER=DATA | FORMATTED | FREQ | UNFORMATTED
specifies the sort order to create the unique combinations of the values of the class variables, which form the headings of the table, according to the specified order.

DATA
orders values according to their order in the input data set.

Interaction: If you use PRELOADFMT in the CLASS statement, the order for the values of each class variable matches the order that PROC FORMAT uses to store the values of the associated user-defined format. If you use the CLASSDATA= option, PROC TABULATE uses the order of the unique values of each class variable in the CLASSDATA= data set to order the output levels. If you use both options, PROC TABULATE first uses the user-defined formats to order the output. If you omit EXCLUSIVE, PROC TABULATE appends after the user-defined format and the CLASSDATA= values the unique values of the class variables in the input data set based on the order that they are encountered.

Tip: By default, PROC FORMAT stores a format definition in sorted order. Use the NOTSORTED option to store the values or ranges of a user defined format in the order that you define them.

FORMATTED
orders values by their ascending formatted values. This order depends on your operating environment.

Alias: FMT | EXTERNAL

FREQ
orders values by descending frequency count.

Interaction: Use the ASCENDING option in the CLASS statement to order values by ascending frequency count.

UNFORMATTED
orders values by their unformatted values, which yields the same order as PROC SORT. This order depends on your operating environment. This sort sequence is particularly useful for displaying dates chronologically.

Alias: UNFMT | INTERNAL

Default: UNFORMATTED

Interaction: If you use the PRELOADFMT option in the CLASS statement, PROC TABULATE orders the levels by the order of the values in the user-defined format.

Featured in: "Understanding the Order of Headings with ORDER=DATA" on page 1198

OUT=*SAS-data-set*
names the output data set. If *SAS-data-set* doesn't exist, PROC TABULATE creates it.

The number of observations in the output data set depends on the number of categories of data that are used in the tables and the number of subtables that are generated. The output data set contains these variables (in this order):

by variables
variables listed in the BY statement.

class variables
variables listed in the CLASS statement.

TYPE
a character variable that shows which combination of class variables produced the summary statistics in that observation. Each position in _TYPE_ represents one variable in the CLASS statement. If that variable is in the category that produced the statistic, the position contains a 1; if it is not, the position contains a 0. In simple PROC TABULATE steps that do not use the universal class variable ALL, all values of _TYPE_ contain only 1's because the only categories that are being considered involve all class variables. If you use the variable ALL, your tables will contain data for categories that do not include all the class variables, and values of _TYPE_ will, therefore, include both 1's and 0's.

PAGE
The logical page that contains the observation.

TABLE
The number of the table that contains the observation.

statistics
statistics calculated for each observation in the data set.

Featured in: Example 3 on page 1203

QMARKERS=*number*
specifies the default number of markers to use for the P^2 quantile estimation method. The number of markers controls the size of fixed memory space.

Default: The default value depends on which quantiles you request. For the median (P50), *number* is 7. For the quartiles (P25 and P75), *number* is 25. For the quantiles P1, P5, P10, P90, P95, or P99, *number* is 107. If you request several quantiles, PROC TABULATE uses the largest default value of *number*.

Range: an odd integer greater than 3

Tip: Increase the number of markers above the default settings to improve the accuracy of the estimates; reduce the number of markers to conserve memory and computing time.

Main Discussion: "Quantiles" on page 653

QMETHOD=OS|P2
specifies the method PROC TABULATE uses to process the input data when it computes quantiles. If the number of observations is less than or equal to the QMARKERS= value and QNTLDEF=5, both methods produce the same results.

OS
uses order statistics. This is the technique that PROC UNIVARIATE uses.

Note: This technique can be very memory-intensive. △

P2
uses the P^2 method to approximate the quantile.

Default: OS

Restriction: When QMETHOD=P2, PROC TABULATE does not compute weighted quantiles.

Tip: When QMETHOD=P2, reliable estimates of some quantiles (P1,P5,P95,P99) may not be possible for some types of data.

Main Discussion: "Quantiles" on page 653

QNTLDEF=1|2|3|4|5
specifies the mathematical definition that the procedure uses to calculate quantiles when QMETHOD=OS is specified. When QMETHOD=P2, you must use QNTLDEF=5.

Default: 5

Alias: PCTLDEF=

Main discussion: "Percentile and Related Statistics" on page 1463

STYLE=<*style-element-name* | <PARENT>><[*style-attribute-specification(s)*]>
specifies the style element to use for the data cells of a table when it is used in the PROC TABULATE statement. For example, the following statement specifies that the background color for data cells be red:

```
proc tabulate data=one style=[background=red];
```

Note: This option can be used in other statements, or in dimension expressions, to specify style elements for other parts of a table. △

Note: You can use braces ({ and }) instead of square brackets ([and]). △

style-element-name
is the name of a style element that is part of a style definition that is registered with the Output Delivery System. SAS Institute provides some style definitions. Users can create their own style definitions with PROC TEMPLATE.

Default: If you do not specify a style element, PROC TABULATE uses Data.

See also: For information about Institute-supplied style definitions, see "What Style Definitions Are Shipped with the Software?" on page 43. For information about PROC TEMPLATE and the Output Delivery System, see *The Complete Guide to the SAS Output Delivery System*.

PARENT
specifies that the data cell use the style element of its parent heading. The parent style element of a data cell is one of the following:

- the style element of the leaf heading above the column that contains the data cell, if the table specifies no row dimension, or if the table specifies the style element in the column dimension expression.
- the style element of the leaf heading above the row that contains the cell, if the table specifies the style element in the row dimension expression.
- the Beforecaption style element, if the table specifies the style element in the page dimension expression.
- undefined, otherwise.

Note: The parent of a heading (not applicable to STYLE= in the PROC TABULATE statement) is the heading under which the current heading is nested. △

style-attribute-specification(s)
describes the attribute to change. Each *style-attribute-specification* has this general form:

style-attribute-name=style-attribute-value

You can set or change the following attributes with the STYLE= option in the PROC TABULATE statement (or in any other statement that uses STYLE=, except for the TABLE statement):

ASIS=	FONT_WIDTH=
BACKGROUND=	HREFTARGET=
BACKGROUNDIMAGE=	HTMLCLASS=
BORDERCOLOR=	JUST=
BORDERCOLORDARK=	NOBREAKSPACE=
BORDERCOLORLIGHT=	POSTHTML=
BORDERWIDTH=	POSTIMAGE=
CELLHEIGHT=	POSTTEXT=
CELLWIDTH=	PREHTML=
FLYOVER=	PREIMAGE=
FONT=	PRETEXT=
FONT_FACE=	PROTECTSPECIALCHARS=
FONT_SIZE=	TAGATTR=
FONT_STYLE=	URL=
FONT_WEIGHT=	VJUST=

For more information about style attributes, see "What Style Attributes Can Base Procedures Specify?" on page 43.

Alias: S=

Restriction: This option affects only the HTML and Printer output.

Tip: To specify a style element for data cells with missing values, use STYLE= in the TABLE statement MISSTEXT= option.

See also: "Using Style Elements in PROC TABULATE" on page 1188

Featured in: Example 14 on page 1243

TRAP
enables floating point exception (FPE) recovery during data processing beyond that provided by normal SAS System FPE handling, which terminates PROC TABULATE in the case of math exceptions.

Default: FPE recovery is disabled.

Tip: Remove TRAP or use NOTRAP to improve performance in operating environments where the overhead of FPE recovery is significant.

See also: NOTRAP on page 1162

VARDEF=*divisor*
specifies the divisor to use in the calculation of the variance and standard deviation. Table 37.3 on page 1166 shows the possible values for *divisor* and the associated divisors.

Table 37.3 Possible Values for VARDEF=

Value	Divisor	Formula for Divisor
DF	degrees of freedom	$n - 1$
N	number of observations	n
WDF	sum of weights minus one	$(\Sigma_i\, w_i) - 1$
WEIGHT | WGT	sum of weights	$\Sigma_i\, w_i$

The procedure computes the variance as $CSS/divisor$, where CSS is the corrected sums of squares and equals $\sum (x_i - \overline{x})^2$. When you weight the analysis variables, CSS equals $\sum w_i (x_i - \overline{x}_w)^2$ where $\overline{x}_w$ is the weighted mean.

Default: DF

Requirement: To compute standard error of the mean, use the default value of VARDEF=.

Tip: When you use the WEIGHT statement and VARDEF=DF, the variance is an estimate of σ^2, where the variance of the *i*th observation is $var\,(x_i) = \sigma^2/w_i$, and w_i is the weight for the *i*th observation. This yields an estimate of the variance of an observation with unit weight.

Tip: When you use the WEIGHT statement and VARDEF=WGT, the computed variance is asymptotically (for large n) an estimate of $\sigma^2/\overline{w}$, where $\overline{w}$ is the average weight. This yields an asymptotic estimate of the variance of an observation with average weight.

See also: the example of weighted statistics"WEIGHT" on page 73.

BY Statement

Creates a separate table on a separate page for each BY group.

Main discussion: "BY" on page 68

BY <DESCENDING> *variable-1*
<...<DESCENDING> *variable-n*>
<NOTSORTED>;

Required Arguments

variable
specifies the variable that the procedure uses to form BY groups. You can specify more than one variable. If you do not use the NOTSORTED option in the BY statement, the observations in the data set must either be sorted by all the variables that you specify, or they must be indexed appropriately. Variables in a BY statement are called *BY variables*.

Options

DESCENDING
specifies that the observations are sorted in descending order by the variable that immediately follows the word DESCENDING in the BY statement.

NOTSORTED
specifies that observations are not necessarily sorted in alphabetic or numeric order. The observations are grouped in another way, for example, chronological order.

The requirement for ordering or indexing observations according to the values of BY variables is suspended for BY-group processing when you use the NOTSORTED option. In fact, the procedure does not use an index if you specify NOTSORTED. The procedure defines a BY group as a set of contiguous observations that have the same values for all BY variables. If observations with the same values for the BY variables are not contiguous, the procedure treats each contiguous set as a separate BY group.

CLASS Statement

Identifies class variables for the table. Class variables determine the categories that PROC TABULATE uses to calculate statistics.

Tip: You can use multiple CLASS statements.

Tip: Some CLASS statement options are also available in the PROC TABULATE statement. They affect all CLASS variables rather than just the one(s) that you specify in a CLASS statement.

CLASS *variable(s) </option(s)>*;

Required Arguments

variable(s)
specifies one or more variables that the procedure uses to group the data. Variables in a CLASS statement are referred to as *class variables*. Class variables can be numeric or character. Class variables can have continuous values, but they typically have a few discrete values that define the classifications of the variable. You do not have to sort the data by class variables.

Options

ASCENDING
specifies to sort the class variable values in ascending order.

Alias: ASCEND

Interaction: PROC TABULATE issues a warning message if you specify both ASCENDING and DESCENDING and ignores both options.

DESCENDING
specifies to sort the class variable values in descending order.

Alias: DESCEND

Default: ASCENDING

Interaction: PROC TABULATE issues a warning message if you specify both ASCENDING and DESCENDING and ignores both options.

EXCLUSIVE
excludes from tables and output data sets all combinations of class variables that are not found in the preloaded range of user-defined formations.

Requirement: You must specify the PRELOADFMT option in the CLASS statement to preload the class variable formats.

Featured in: Example 3 on page 1203

GROUPINTERNAL
specifies not to apply formats to the class variables when PROC TABULATE groups the values to create combinations of class variables.

Interaction: If you specify the PRELOADFMT option in the CLASS statement, PROC TABULATE ignores the GROUPINTERNAL option and uses the formatted values.

Tip: This option saves computer resources when the class variables contain discrete numeric values.

MISSING
considers missing values as valid class variable levels. Special missing values that represent numeric values (the letters A through Z and the underscore (_) character) are each considered as a separate value.

Default: If you omit MISSING, PROC TABULATE excludes the observations with any missing CLASS variable values from tables and output data sets.

See also: *SAS Language Reference: Concepts* for a discussion of missing values with special meanings.

MLF
enables PROC TABULATE to use the primary and secondary format labels for a given range or overlapping ranges to create subgroup combinations when a multilabel format is assigned to a class variable.

Requirement: You must use PROC FORMAT and the MULTILABEL option in the VALUE statement to create a multilabel format.

Interaction: Using MLF with ORDER=FREQ may not produce the order that you expect for the formatted values.

Tip: If you omit MLF, PROC TABULATE uses the primary format labels, which corresponds to the first external format value, to determine the subgroup combinations.

See also: The MULTILABEL option on page 451 in the VALUE statement of the FORMAT procedure.

Featured in: Example 4 on page 1208

Note: When the formatted values overlap, one internal class variable value maps to more than one class variable subgroup combination. Therefore, the sum of the N statistics for all subgroups is greater than the number of observations in the data set (the overall N statistic). △

ORDER=DATA | FORMATTED | FREQ | UNFORMATTED
specifies the order to group the levels of the class variables in the output, where

DATA
orders values according to their order in the input data set.

Interaction: If you use PRELOADFMT, the order for the values of each class variable matches the order that PROC FORMAT uses to store the values of the associated user-defined format. If you use the CLASSDATA= option in the PROC statement, PROC MEANS uses the order of the unique values of each class variable in the CLASSDATA= data set to order the output levels. If you use both options, PROC TABULATE first uses the user-defined formats to order the output. If you omit EXCLUSIVE in the PROC statement, PROC TABULATE appends after the user-defined format and the CLASSDATA= values the unique values of the class variables in the input data set based on the order that they are encountered.

Tip: By default, PROC FORMAT stores a format definition in sorted order. Use the NOTSORTED option to store the values or ranges of a user-defined format in the order that you define them.

FORMATTED
orders values by their ascending formatted values. This order depends on your operating environment.

Alias: FMT | EXTERNAL

FREQ
orders values by descending frequency count.

Interaction: Use the ASCENDING option to order values by ascending frequency count.

UNFORMATTED
orders values by their unformatted values, which yields the same order as PROC SORT. This order depends on your operating environment. This sort sequence is particularly useful for displaying dates chronologically.

Alias: UNFMT | INTERNAL

Default: UNFORMATTED

Interaction: If you use the PRELOADFMT option in the CLASS statement, PROC TABULATE orders the levels by the order of the values in the user-defined format.

Tip: By default, all orders except FREQ are ascending. For descending orders, use the DESCENDING option.

Featured in: "Understanding the Order of Headings with ORDER=DATA" on page 1198

PRELOADFMT
specifies that all formats are preloaded for the class variables.

Requirement: PRELOADFMT has no effect unless you specify either EXCLUSIVE, ORDER=DATA, or PRINTMISS and you assign formats to the class variables.

Note: If you specify PRELOADFMT without also specifying either EXCLUSIVE or PRINTMISS, SAS writes a warning message to the SAS log. △

Interaction: To limit PROC TABULATE output to the combinations of formatted class variable values present in the input data set, use the EXCLUSIVE option in the CLASS statement.

Interaction: To include all ranges and values of the user-defined formats in the output, use the PRINTMISS option in the TABLE statement.

CAUTION:
Use care when you use PRELOADFMT with PRINTMISS. This feature creates all possible combinations of formatted class variables. Some of these combinations may not make sense. △

Featured in: Example 3 on page 1203

STYLE=<*style-element-name* | <PARENT>><[*style-attribute-specification(s)*]>
specifies the style element to use for page dimension text, continuation messages, and class variable name headings. For information about the arguments of this option, and how it is used, see STYLE= on page 1164 in the PROC TABULATE statement.

Note: When you use STYLE= in the CLASS statement, it differs slightly from its use in the PROC TABULATE statement. In the CLASS statement, the parent of the heading is the page dimension text or heading under which the current heading is nested. △

Note: If a page dimension expression contains multiple nested elements, the Beforecaption style element is the style element of the first element in the nesting. △

Alias: S=

Restriction: This option affects only the HTML and Printer output.

Tip: To override a style element that is specified for page dimension text in the CLASS statement, you can specify a style element in the TABLE statement page dimension expression.

Tip: To override a style element that is specified for a class variable name heading in the CLASS statement, you can specify a style element in the related TABLE statement dimension expression.

Featured in: Example 14 on page 1243

How PROC TABULATE Handles Missing Values for Class Variables

By default, if an observation contains a missing value for any class variable, PROC TABULATE excludes that observation from all tables that it creates. CLASS statements apply to all TABLE statements in the PROC TABULATE step. Therefore, if you define a variable as a class variable, PROC TABULATE omits observations that have missing values for that variable from every table even if the variable does not appear in the TABLE statement for one or more tables.

If you specify the MISSING option in the PROC TABULATE statement, the procedure considers missing values as valid levels for all class variables. If you specify the MISSING option in a CLASS statement, PROC TABULATE considers missing values as valid levels for the class variable(s) that are specified in that CLASS statement.

CLASSLEV Statement

Specifies a style element for class variable level value headings.

Restriction: This statement affects only the HTML and Printer output.

CLASSLEV *variable(s)* / *style* =<*style-element-name* | <PARENT>>
<[*style-attribute-specification(s)*] >;

Required Arguments

variable(s)
specifies one or more class variables from the CLASS statement for which you want to specify a style element.

Options

STYLE=<*style-element-name* | <PARENT>><[*style-attribute-specification(s)*]>
specifies a style element for class variable level value headings. For information on the arguments of this option and how it is used, see STYLE= on page 1164 in the PROC TABULATE statement.

Note: When you use STYLE= in the CLASSLEV statement, it differs slightly from its use in the PROC TABULATE statement. In the CLASSLEV statement, the parent of the heading is the heading under which the current heading is nested. △

Alias: S=

Restriction: This option affects only the HTML and Printer output.

Tip: To override a style element that is specified in the CLASSLEV statement, you can specify a style element in the related TABLE statement dimension expression.

Featured in: Example 14 on page 1243

FREQ Statement

Specifies a numeric variable that contains the frequency of each observation.

Tip: The effects of the FREQ and WEIGHT statements are similar except when calculating degrees of freedom.

See also: For an example that uses the FREQ statement, see "FREQ" on page 70.

FREQ *variable*;

Required Arguments

variable
specifies a numeric variable whose value represents the frequency of the observation. If you use the FREQ statement, the procedure assumes that each observation represents *n* observations, where *n* is the value of *variable*. If *n* is not an integer, the SAS System truncates it. If *n* is less than 1 or is missing, the procedure does not use that observation to calculate statistics.

The sum of the frequency variable represents the total number of observations.

KEYLABEL Statement

Labels a keyword for the duration of the PROC TABULATE step. PROC TABULATE uses the label anywhere that the specified keyword would otherwise appear.

KEYLABEL *keyword-1='description-1'*

<…keyword-n='description-n'>;

Required Arguments

keyword
is one of the keywords for statistics that is discussed in "Statistics Available in PROC TABULATE" on page 1181 or is the universal class variable ALL (see "Elements That You Can Use in a Dimension Expression" on page 1177).

description
is up to 256 characters to use as a label. As the syntax shows, you must enclose *description* in quotes.

Restriction: Each keyword can have only one label in a particular PROC TABULATE step; if you request multiple labels for the same keyword, PROC TABULATE uses the last one that is specified in the step.

KEYWORD Statement

Specifies a style element for keyword headings.

Restriction: This statement affects only the HTML and Printer output.

KEYWORD *keyword(s) / style =<style-element-name* | <PARENT>>
<[style-attribute-specification(s)] >;

Required Arguments

keyword
is one of the keywords for statistics that is discussed in "Statistics Available in PROC TABULATE" on page 1181 or is the universal class variable ALL (see "Elements That You Can Use in a Dimension Expression" on page 1177).

Options

STYLE=<*style-element-name* | <PARENT>><[*style-attribute-specification(s)*]>
specifies a style element for the keyword headings. For information on the arguments of this option and how it is used, see STYLE= on page 1164 in the PROC TABULATE statement.

Note: When you use STYLE= in the KEYWORD statement, it differs slightly from its use in the PROC TABULATE statement. In the KEYWORD statement, the parent of the heading is the heading under which the current heading is nested. △

Alias: S=

Restriction: This option affects only the HTML and Printer output.

Tip: To override a style element that is specified in the KEYWORD statement, you can specify a style element in the related TABLE statement dimension expression.

Featured in: Example 14 on page 1243

TABLE Statement

Describes a table to print.

Requirement: All variables in the TABLE statement must appear in either the VAR statement or the CLASS statement.

Tip: Use multiple TABLE statements to create several tables.

TABLE <<*page-expression*,> *row-expression*,>
column-expression </ *table-option(s)*>;

Required Arguments

column-expression
defines the columns in the table. For information on constructing dimension expressions, see "Constructing Dimension Expressions" on page 1177.

Restriction: A column dimension is the last dimension in a TABLE statement. A row dimension or a row dimension and a page dimension may precede a column dimension.

Options

To do this	Use this option
Add dimensions	
Define the pages in a table	*page-expression*
Define the rows in a table	*row-expression*
Customize the HTML contents entry link to the output	CONTENTS=
Specify a style element for various parts of the table	STYLE=
Customize text in the table	
Specify the text to place in the empty box above row titles	BOX=
Supply up to 256 characters to print in table cells that contain missing values	MISSTEXT=
Suppresses the continuation message for tables that span multiple physical pages	NOCONTINUED
Modify the layout of the table	
Print as many complete logical pages as possible on a single printed page or, if possible, print multiple pages of tables that are too wide to fit on a page one below the other on a single page, instead of on separate pages.	CONDENSE

To do this	Use this option
Create the same row and column headings for all logical pages of the table	PRINTMISS
Customize row headings	
Specify the number of spaces to indent nested row headings	INDENT=
Control allocation of space for row titles within the available space	ROW=
Specify the number of print positions available for row titles	RTSPACE=

BOX=*value*
BOX={*<label=value>*
***<style=<style-element-name><[style-attribute-specification(s)]>>* }**
specifies text and a style element for the empty box above the row titles.

Value can be one of the following:

PAGE
writes the page-dimension text in the box. If the page-dimension text does not fit, it is placed in its default position above the box, and the box remains empty.

'*string*'
writes the quoted string in the box. Any string that does not fit in the box is truncated.

variable
writes the name (or label, if the variable has one) of a variable in the box. Any name or label that does not fit in the box is truncated.

For details about the arguments of the STYLE= option and how it is used, see STYLE= on page 1164 in the PROC TABULATE statement.

Featured in: Example 9 on page 1218 and Example 14 on page 1243

CONDENSE
prints as many complete logical pages as possible on a single printed page or, if possible, prints multiple pages of tables that are too wide to fit on a page one below the other on a single page, instead of on separate pages. A *logical page* is all the rows and columns that fall within one of the following:

- a page-dimension category (with no BY-group processing)
- a BY group with no page dimension
- a page-dimension category within a single BY group.

Restrictions: CONDENSE has no effect on the pages that are generated by the BY statement. The first table for a BY group always begins on a new page.

CONDENSE is ignored by the HTML destination but supported by the printer.

Featured in: Example 9 on page 1218

CONTENTS=*link-name*
allows you to name the link in the HTML table of contents that points to the ODS output of the table that is produced by using the TABLE statement.

Restrictions: CONTENTS= has no effect on TABULATE procedure reports.

FUZZ=*number*
supplies a numeric value against which analysis variable values and table cell values other than frequency counts are compared to eliminate trivial values (absolute values less than the FUZZ= value) from computation and printing. A number whose

absolute value is less than the FUZZ= value is treated as zero in computations and printing. The default value is the smallest representable floating-point number on the computer that you are using.

INDENT=*number-of-spaces*
specifies the number of spaces to indent nested row headings, and suppresses the row headings for class variables.

Tip: When there are no crossings in the row dimension, there is nothing to indent, so the value of *number-of-spaces* has no effect. However, in such cases INDENT= still suppresses the row headings for class variables.

Featured in: Example 8 on page 1216 (with crossings) and Example 9 on page 1218 (without crossings)

page-expression
defines the pages in a table. For information on constructing dimension expressions, see "Constructing Dimension Expressions" on page 1177.

Restriction: A page dimension is the first dimension in a table statement. Both a row dimension and a column dimension must follow a page dimension.

Featured in: Example 9 on page 1218

MISSTEXT='*text*'
MISSTEXT={<*label*= '*text*'
><*style*=<*style-element-name*><[*style-attribute-specification(s)*]>> }
supplies up to 256 characters of text to print and specifies a style element for table cells that contain missing values. For details on the arguments of the STYLE= option and how it is used, see STYLE= on page 1164 in the PROC TABULATE statement.

Interaction: A style element that is specified in a dimension expression overrides a style element that is specified in the MISSTEXT= option for any given cell(s).

Featured in: "Providing Text for Cells That Contain Missing Values" on page 1195 and Example 14 on page 1243

NOCONTINUED
suppresses the continuation message, **`continued`**, that is displayed at the bottom of tables that span multiple pages. The text is rendered with the Aftercaption style element.

Restrictions: NOCONTINUED is ignored by the HTML destination but supported by the printer.

PRINTMISS
prints all values that occur for a class variable each time headings for that variable are printed, even if there are no data for some of the cells that these headings create. Consequently, PRINTMISS creates row and column headings that are the same for all logical pages of the table, within a single BY group.

Default: If you omit PRINTMISS, PROC TABULATE suppresses a row or column for which there are no data, unless you use the CLASSDATA= option in the PROC TABULATE statement.

Restrictions: If an entire logical page contains only missing values, that page does not print regardless of the PRINTMISS option.

See also: CLASSDATA= option on page 1159

Featured in: "Providing Headings for All Categories" on page 1194

ROW=*spacing*
specifies whether all title elements in a row crossing are allotted space even when they are blank. The possible values for *spacing* are as follows:

CONSTANT
allots space to all row titles even if the title has been blanked out (for example, N=' ').

Alias: CONST

FLOAT
divides the row title space equally among the nonblank row titles in the crossing.

Default: CONSTANT

Featured in: Example 7 on page 1214

row-expression
defines the rows in the table. For information on constructing dimension expressions, see "Constructing Dimension Expressions" on page 1177.

Restriction: A row dimension is the next to last dimension in a table statement. A column dimension must follow a row dimension. A page dimension may precede a row dimension.

RTSPACE=*number*
specifies the number of print positions to allot to all of the headings in the row dimension, including spaces that are used to print outlining characters for the row headings. PROC TABULATE divides this space equally among all levels of row headings.

Alias: RTS=

Default: one-fourth of the value of the SAS system option LINESIZE=

Interaction: By default, PROC TABULATE allots space to row titles that are blank. Use ROW=FLOAT on page 1175 to divide the space among only nonblank titles.

See also: For more information about controlling the space for row titles, see Chapter 5, "Controlling the Table's Appearance" in *SAS Guide to TABULATE Processing*.

Featured in: Example 1 on page 1199

STYLE=<*style-element-name*><[*style-attribute-specification(s)*]>
specifies a style element to use for the entire table. For information about the arguments of this option and how it is used, see STYLE= on page 1164 in the PROC TABULATE statement.

Note: The list of attributes that you can set or change with the STYLE= option in the TABLE statement differs from that of the PROC TABULATE statement. △

You can set or change the following attributes with the STYLE= option in the TABLE statement. These attributes apply to the table as a whole. Attributes that you apply in the PROC TABULATE statement and in other locations in the PROC TABULATE step apply to cells within the table.

BACKGROUND=
BACKGROUNDIMAGE=
BORDERCOLOR=
BORDERCOLORDARK=
BORDERCOLORLIGHT=
BORDERWIDTH=
CELLPADDING=
CELLSPACING=
FONT_WIDTH=*
FOREGROUND=*
FRAME=
HTMLCLASS=
JUST=
OUTPUTWIDTH=
POSTHTML=
POSTIMAGE=

FONT=*	POSTTEXT=
FONT_FACE=*	PREHTML=
FONT_SIZE=*	PREIMAGE=
FONT_STYLE=*	PRETEXT=
FONT_WEIGHT=*	RULES=

* When you use these attributes in this location, they affect only the text that is specified with the PRETEXT=, POSTTEXT=, PREHTML=, and POSTHTML= attributes. To alter the foreground color or the font for the text that appears in the table, you must set the corresponding attribute in a location that affects the cells rather than the table.

For more information about style attributes, see "What Style Attributes Can Base Procedures Specify?" on page 43.

Note: You can use braces ({ and }) instead of square brackets ([and]). △

Alias: S=

Restriction: This option affects only the HTML and Printer output.

Tip: To override a style element specification that is made as an option in the TABLE statement, specify STYLE= in a dimension expression of the TABLE statement.

Featured in: Example 14 on page 1243

Constructing Dimension Expressions

A TABLE statement consists of from one to three dimension expressions separated by commas. Options can follow the dimension expressions. If all three dimensions are specified, the leftmost dimension defines pages, the middle dimension defines rows, and the rightmost dimension defines columns. If two dimensions are specified, the left defines rows, and the right defines columns. If a single dimension is specified, it defines columns.

A dimension expression is composed of elements and operators.

Elements That You Can Use in a Dimension Expression

analysis variables
(see "VAR Statement" on page 1179).

class variables
(see "CLASS Statement" on page 1167).

the universal class variable ALL
summarizes all of the categories for class variables in the same parenthetical group or dimension (if the variable ALL is not contained in a parenthetical group).

Featured in: Example 6 on page 1212, Example 9 on page 1218, and Example 13 on page 1233

Note: If the input data set contains a variable named ALL, enclose the name of the universal class variable in quotes. △

keywords for statistics

Requirement: To compute standard error or a *t*-test, you must use the default value of VARDEF=, which is DF.

Featured in: Example 10 on page 1220 and Example 13 on page 1233

format modifiers
define how to format values in cells. Cross a format modifier with the elements that produce the cells that you want to format. Format modifiers have the form

```
f=format
```

Tip: Format modifiers have no effect on CLASS variables.

See also: For more information on specifying formats in tables, see "Formatting Values in Tables" on page 1183.

Featured in: Example 6 on page 1212

labels
temporarily replace the names of variables and statistics. Labels affect only the variable or statistic that immediately precedes the label. Labels have the form

```
stat-or-variable-name='label-text'
```

Tip: PROC TABULATE eliminates the space for blank column headings from a table but by default does not eliminate the space for blank row headings. Use ROW=FLOAT in the TABLE statement to remove the space for blank row headings.

Featured in: Example 5 on page 1210 and Example 7 on page 1214

style—element specifications
specify style elements for page dimension text, continuation messages, headings, or data cells. For details, see "Specifying Style Elements in Dimension Expressions" on page 1178.

You can also form dimension expressions by combining any of these elements.

Operators That You Can Use in a Dimension Expression

asterisk *
creates categories from the combination of values of the class variables and constructs the appropriate headers for the dimension. If one of the elements is an analysis variable, the statistics for the analysis variable are calculated for the categories that are created by the class variables. This process is called *crossing*.

Featured in: Example 1 on page 1199

(blank)
places the output for each element immediately after the output for the preceding element. This process is called *concatenation*.

Featured in: Example 6 on page 1212

parentheses ()
group elements and associate an operator with each concatenated element in the group.

Featured in: Example 6 on page 1212

angle brackets <>
specify denominator definitions, which determine the value of the denominator in the calculation of a percentage. For a discussion of how to construct denominator definitions, see "Calculating Percentages" on page 1184.

Featured in: Example 10 on page 1220 and Example 13 on page 1233

Specifying Style Elements in Dimension Expressions

You can specify a style element in a dimension expression to control the appearance in HTML and Printer output of the following table elements:

analysis variable name headings

class variable name headings

class variable level value headings

data cells

keyword headings

page dimension text

Specifying a style element in a dimension expression is useful when you want to override a style element that you have specified in another statement, such as the PROC TABULATE, CLASS, CLASSLEV, KEYWORD, TABLE, or VAR statements.

The syntax for specifying a style element in a dimension expression is

[STYLE<(CLASSLEV)>=<*style-element-name* |
<PARENT>><[*style-attribute-specification(s)*]>]

Some examples of style elements in dimension expressions are

```
dept={label='Department'
      style=[foreground=red]}, N

dept*[style=MyDataStyle], N

dept*[format=12.2 style=MyDataStyle], N
```

Note: When used in a dimension expression, the STYLE= option must be enclosed within square brackets ([and]) or braces ({ and }). △

With the exception of (CLASSLEV), all arguments are described in STYLE= on page 1164 in the PROC TABULATE statement.

(CLASSLEV)
assigns a style element to a class variable level value heading. For example, the following TABLE statement specifies that the level value heading for the class variable, DEPT, has a foreground color of yellow:

```
table dept=[style(classlev)=
           [foreground=yellow]]*sales;
```

Note: This option is used only in dimension expressions. △

For an example that shows how to specify style elements within dimension expressions, see Example 14 on page 1243.

VAR Statement

Identifies numeric variables to use as analysis variables.

Alias: VARIABLES

Tip: You can use multiple VAR statements.

VAR *analysis-variable(s)* </ *option(s)*>;

Required Arguments

analysis-variable(s);
identifies the analysis variables in the table. Analysis variables are numeric variables for which PROC TABULATE calculates statistics. The values of an analysis variable can be continuous or discrete.

If an observation contains a missing value for an analysis variable, PROC TABULATE omits that value from calculations of all statistics except N (the number of observations with nonmissing variable values) and NMISS (the number of observations with missing variable values). For example, the missing value does not increase the SUM, and it is not counted when you are calculating statistics such as the MEAN.

Options

STYLE=<*style-element-name* | <PARENT>><[*style-attribute-specification(s)*>]
specifies a style element for analysis variable name headings. For information on the arguments of this option and how it is used, see STYLE= on page 1164 in the PROC TABULATE statement.

Note: When you use STYLE= in the VAR statement, it differs slightly from its use in the PROC TABULATE statement. In the VAR statement, the parent of the heading is the heading under which the current heading is nested. △

Alias: S=

Restriction: This option affects only the HTML and Printer output.

Tip: To override a style element that is specified in the VAR statement, you can specify a style element in the related TABLE statement dimension expression.

Featured in: Example 14 on page 1243

WEIGHT=*weight-variable*
specifies a numeric variable whose values weight the values of the variables that are specified in the VAR statement. The variable does not have to be an integer. If the value of the weight variable is

Weight value...	PROC TABULATE...
0	counts the observation in the total number of observations
less than 0	converts the value to zero and counts the observation in the total number of observations
missing	excludes the observation

To exclude observations that contain negative and zero weights from the analysis, use EXCLNPWGT. Note that most SAS/STAT procedures, such as PROC GLM, exclude negative and zero weights by default.

Restriction: To compute weighted quantiles, use QMETHOD=OS in the PROC statement.

Tip: When you use the WEIGHT= option, consider which value of the VARDEF= option is appropriate (see the discussion of VARDEF= on page 1165).

Tip: Use the WEIGHT option in multiple VAR statements to specify different weights for the analysis variables.

Note: Prior to Version 7 of the SAS System, the procedure did not exclude the observations with missing weights from the count of observations. △

WEIGHT Statement

Specifies weights for analysis variables in the statistical calculations.

See also: For information on calculating weighted statistics and for an example that uses the WEIGHT statement, see "Calculating Weighted Statistics" on page 74

WEIGHT *variable*;

Required Arguments

variable
specifies a numeric variable whose values weight the values of the analysis variables. The values of the variable do not have to be integers. If the value of the weight variable is

Weight value ...	PROC TABULATE ...
0	counts the observation in the total number of observations
less than 0	converts the value to zero and counts the observation in the total number of observations
missing	excludes the observation

To exclude observations that contain negative and zero weights from the analysis, use EXCLNPWGT. Note that most SAS/STAT procedures, such as PROC GLM, exclude negative and zero weights by default.

Restriction: To compute weighted quantiles, use QMETHOD=OS in the PROC statement.

Interaction: If you use the WEIGHT= option in a VAR statement to specify a weight variable, PROC TABULATE uses this variable instead to weight those VAR statement variables.

Tip: When you use the WEIGHT statement, consider which value of the VARDEF= option is appropriate. See the discussion of VARDEF= on page 1165 and the calculation of weighted statistics in "Keywords and Formulas" on page 1458 for more information.

Note: Prior to Version 7 of the SAS System, the procedure did not exclude the observations with missing weights from the count of observations. △

Concepts

Statistics Available in PROC TABULATE

Use the following keywords to request statistics in the TABLE statement. If a variable name (class or analysis) and a statistic name are the same, enclose the statistic name in single quotes.

Descriptive statistic keywords	
COLPCTN	PCTSUM
COLPCTSUM	RANGE
CSS	REPPCTN
CV	REPPCTSUM
MAX	ROWPCTN
MEAN	ROWPCTSUM
MIN	STDDEV|STD
N	STDERR
NMISS	SUM
PAGEPCTN	SUMWGT
PAGEPCTSUM	USS
PCTN	VAR
Quantile statistic keywords	
MEDIAN|P50	Q3|P75
P1	P90
P5	P95
P10	P99
Q1|P25	QRANGE
Hypothesis testing keyword	
PROBT	T

To compute standard error (STD), you must use VARDEF=DF in the PROC statement. To compute weighted quantiles, you must use QMETHOD=OS in the PROC statement.

Explanations of the keywords, the formulas that are used to calculate them, and the data requirements are discussed in “Keywords and Formulas” on page 1458.

Formatting Class Variables

Use the FORMAT statement to assign a format to a class variable for the duration of a PROC TABULATE step. When you assign a format to a class variable, PROC TABULATE uses the formatted values to create categories, and it uses the formatted values in headings.

User-defined formats are particularly useful for grouping values into fewer categories. For example, if you have a class variable, Age, with values ranging from 1 to 99, you could create a user-defined format that groups the ages so that your tables contain a manageable number of categories. The following PROC FORMAT step creates a format that condenses all possible values of age into six groups of values.

```
proc format;
   value agefmt  0-29='Under 30'
                 30-39='30-39'
                 40-49='40-49'
                 50-59='50-59'
                 60-69='60-69'
                 other='70 or over';
```

```
run;
```

For information on creating user-defined formats, see Chapter 19, "The FORMAT Procedure," on page 433.

By default, PROC TABULATE includes in a table only those formats for which the frequency count is not zero and for which values are not missing. To include missing values for all class variables in the output, use the MISSING option in the PROC TABULATE statement, and to include missing values for selected class variables, use the MISSING option in a CLASS statement. To include formats for which the frequency count is zero, use the PRELOADFMT option in a CLASS statement and the PRINTMISS option in the TABLE statement, or use the CLASSDATA= option in the PROC TABULATE statement.

Formatting Values in Tables

The formats for data in table cells serve two purposes. They determine how PROC TABULATE displays the values, and they determine the width of the columns. The default format for values in table cells is 12.2. You can modify the format for printing values in table cells by

- changing the default format with the FORMAT= option in the PROC TABULATE statement
- crossing elements in the TABLE statement with the F= format modifier.

PROC TABULATE determines the format to use for a particular cell based on the following order of precedence for formats:

1. If no other formats are specified, PROC TABULATE uses the default format (12.2).
2. The FORMAT= option in the PROC TABULATE statement changes the default format. If no format modifiers affect a cell, PROC TABULATE uses this format for the value in that cell.
3. A format modifier in the page dimension applies to the values in all the table cells on the page unless you specify another format modifier for a cell in the row or column dimension.
4. A format modifier in the row dimension applies to the values in all the table cells in the row unless you specify another format modifier for a cell in the column dimension.
5. A format modifier in the column dimension applies to the values in all the table cells in the column.

For more information about formatting table cells, see "Formatting Values in Table Cells" in Chapter 5, "Controlling the Table's Appearance" in *SAS Guide to TABULATE Processing*.

How Using BY-group Processing Differs from Using the Page Dimension

Using the page-dimension expression in a TABLE statement can have an effect similar to using a BY statement.

Table 37.4 on page 1184 contrasts the two methods.

Table 37.4 Contrasting the BY Statement and the Page Dimension

Issue	PROC TABULATE with a BY statement	PROC TABULATE with a page dimension in the TABLE statement
Order of observations in the input data set	The observations in the input data set must be sorted by the BY variables. [1]	Sorting is unnecessary.
One report summarizing all BY groups	You cannot create one report for all the BY groups.	Use ALL in the page dimension to create a report for all classes. (See Example 6 on page 1212.)
Percentages	The percentages in the tables are percentages of the total for that BY group. You cannot calculate percentages for a BY group compared to the totals for all BY groups because PROC TABULATE prepares the individual reports separately. Data for the report for one BY group are not available to the report for another BY group.	You can use denominator definitions to control the meaning of PCTN (see "Calculating Percentages" on page 1184.)
Titles	You can use the #BYVAL, #BYVAR, and #BYLINE specifications in TITLE statements to customize the titles for each BY group (see "Creating Titles That Contain BY-Group Information" on page 54).	The BOX= option in the TABLE statement customizes the page headers, but you must use the same title on each page.
Ordering class variables	ORDER=DATA and ORDER=FREQ order each BY group independently.	The order of class variables is the same on every page.
Obtaining uniform headings	You may need to insert dummy observations into BY groups that do not have all classes represented.	The PRINTMISS option ensures that each page of the table has uniform headings.
Multiple ranges with the same format	PROC TABULATE produces a table for each range.	PROC TABULATE combines observations from the two ranges.

1 You can use the BY statement without sorting the data set if the data set has an index for the BY variable.

Calculating Percentages

The following statistics print the percentage of the value in a single table cell in relation to the total of the values in a group of cells. No denominator definitions are required; however, an analysis variable may be used as a denominator definition for percentage sum statistics.

REPPCTN and REPPCTSUM statistics—print the percentage of the value in a single table cell in relation to the total of the values in the report.

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.

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.

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.

These statistics calculate the most commonly used percentages. See Example 12 on page 1230 for an example.

PCTN and PCTSUM statistics can be used to calculate these same percentages. They allow you to manually define denominators. PCTN and PCTSUM statistics print the percentage of the value in a single table cell in relation to the value (used in the denominator of the calculation of the percentage) in another table cell or to the total of the values in a group of cells. By default, PROC TABULATE summarizes the values in all N cells (for PCTN) or all SUM cells (for PCTSUM) and uses the summarized value for the denominator. You can control the value that PROC TABULATE uses for the denominator with a denominator definition.

You place a denominator definition in angle brackets (< and >) next to the N or PCTN statistic. The denominator definition specifies which categories to sum for the denominator.

This section illustrates how to specify denominator definitions in a simple table. Example 13 on page 1233 illustrates how to specify denominator definitions in a table that is composed of multiple subtables. For more examples of denominator definitions, see "How Percentages Are Calculated" in Chapter 3, "Details of TABULATE Processing" in *SAS Guide to TABULATE Processing*.

Specifying a Denominator for the PCTN Statistic

The following PROC TABULATE step calculates the N statistic and three different versions of PCTN using the data set ENERGY on page 1199.

```
proc tabulate data=energy;
   class division type;
   table division*
           (n='Number of customers'
            pctn<type>='% of row' ❶
            pctn<division>='% of column'  ❷
            pctn='% of all customers'), ❸
          type/rts=50;
   title 'Number of Users in Each Division';
run;
```

The TABLE statement creates a row for each value of Division and a column for each value of Type. Within each row, the TABLE statement nests four statistics: N and three different calculations of PCTN (see Figure 37.3 on page 1186). Each occurrence of PCTN uses a different denominator definition.

Figure 37.3 Three Different Uses of the PCTN Statistic with Frequency Counts Highlighted

Number of Users in Each Division

1

Division		Type 1	Type 2
1	Number of customers	6.00	6.00
	% of row ❶	50.00	50.00
	% of column ❷	27.27	27.27
	% of all customers ❸	13.64	13.64
2	Number of customers	3.00	3.00
	% of row	50.00	50.00
	% of column	13.64	13.64
	% of all customers	6.82	6.82
3	Number of customers	8.00	8.00
	% of row	50.00	50.00
	% of column	36.36	36.36
	% of all customers	18.18	18.18
4	Number of customers	5.00	5.00
	% of row	50.00	50.00
	% of column	22.73	22.73
	% of all customers	11.36	11.36

1 **<type>** sums the frequency counts for all occurrences of Type within the same value of Division. Thus, for Division=1, the denominator is 6 + 6, or 12.

2 **<division>** sums the frequency counts for all occurrences of Division within the same value of Type. Thus, for Type=1, the denominator is 6 + 3 + 8 + 5, or 22.

3 The third use of PCTN has no denominator definition. Omitting a denominator definition is the same as including all class variables in the denominator definition. Thus, for all cells, the denominator is 6 + 3 + 8 + 5 + 6 + 3 + 8 + 5, or 44.

Specifying a Denominator for the PCTSUM Statistic

The following PROC TABULATE step sums expenditures for each combination of Type and Division and calculates three different versions of PCTSUM.

```
proc tabulate data=energy format=8.2;
   class division type;
   var expenditures;
   table division*
           (sum='Expenditures'*f=dollar10.2
            pctsum<type>='% of row' ❶
            pctsum<division>='% of column' ❷
```

```
            pctsum='% of all customers'), ❸
          type*expenditures/rts=40;
   title 'Expenditures in Each Division';
run;
```

The TABLE statement creates a row for each value of Division and a column for each value of Type. Because Type is crossed with Expenditures, the value in each cell is the sum of the values of Expenditures for all observations that contribute to the cell. Within each row, the TABLE statement nests four statistics: SUM and three different calculations of PCTSUM (see Figure 37.4 on page 1187). Each occurrence of PCTSUM uses a different denominator definition.

Figure 37.4 Three Different Uses of the PCTSUM Statistic with Sums Highlighted

Expenditures in Each Division 1

		Type	
		1	2
		Expend	Expend
Division			
1	Expenditures	$7,477.00	$5,129.00
	% of row ❶	59.31	40.69
	% of column ❷	16.15	13.66
	% of all customers ❸	8.92	6.12
2	Expenditures	$19,379.00	$15,078.00
	% of row	56.24	43.76
	% of column	41.86	40.15
	% of all customers	23.11	17.98
3	Expenditures	$5,476.00	$4,729.00
	% of row	53.66	46.34
	% of column	11.83	12.59
	% of all customers	6.53	5.64
4	Expenditures	$13,959.00	$12,619.00
	% of row	52.52	47.48
	% of column	30.15	33.60
	% of all customers	16.65	15.05

1 **<type>** sums the values of Expenditures for all occurrences of Type within the same value of Division. Thus, for Division=1, the denominator is $7,477 + $5,129.

2 **<division>** sums the frequency counts for all occurrences of Division within the same value of Type. Thus, for Type=1, the denominator is $7,477 + $19,379 + $5,476 + $13,959.

3 The third use of PCTN has no denominator definition. Omitting a denominator definition is the same as including all class variables in the denominator definition. Thus, for all cells, the denominator is $7,477 + $19,379 + $5,476 + $13,959 + $5,129 + $15,078 + $4,729 + $12,619.

Using Style Elements in PROC TABULATE

If you use the Output Delivery System to create both HTML and Printer output from PROC TABULATE, you can set the style element that the procedure uses for various parts of the table. Style elements determine presentation attributes, such as font face, font weight, color, and so forth. Information about the style attributes that you can set for a style element is in "Customizing the Style Definition That ODS Uses" on page 42. lists the default styles for various regions of a table.

Table 37.5 Default Styles for Table Regions

Region	Style
column headings	Header
continuation message	Aftercaption
box	Header
page dimension text	Beforecaption
row headings	Rowheader
data cells	Data
table	Table

You specify style elements for PROC TABULATE with the STYLE= option. The following shows where you can use this option. Specifications in the TABLE statement override the same specification in the PROC TABULATE statement. However, any style attributes that you specify in the PROC TABULATE statement and that you do not override in the TABLE statement are inherited. For instance, if you specify a blue background and a white foreground for all data cells in the PROC TABULATE statement, and you specify a gray background for the data cells of a particular crossing in the TABLE statement, the background for those data cells is gray, and the foreground is white (as specified in the PROC TABULATE statement).

Detailed information on STYLE= is provided in the documentation for individual statements.

Table 37.6 Using the STYLE= Option in PROC TABULATE

To set the style element for	Use STYLE in this statement
data cells	PROC TABULATE
page dimension text, continuation messages, and class variable name headings	CLASS
class level value headings	CLASSLEV
keyword headings	KEYWORD
the entire table	TABLE
analysis variable name headings	VAR

Results

Missing Values

How a missing value for a variable in the input data set affects your output depends on how you use the variable in the PROC TABULATE step. Table 37.7 on page 1189 summarizes how the procedure treats missing values.

Table 37.7 Summary of How PROC TABULATE Treats Missing Values

If . . .	PROC TABULATE, by default, . . .	To override the default . . .
an observation contains a missing value for an analysis variable	excludes that observation from the calculation of statistics (except N and NMISS) for that particular variable	no alternative
an observation contains a missing value for a class variable	excludes that observation from the table[1]	use MISSING in the PROC TABULATE statement, or MISSING in the CLASS statement
there are no data for a category	does not show the category in the table	use PRINTMISS in the TABLE statement, or use CLASSDATA= in the PROC TABULATE statement
every observation that contributes to a table cell contains a missing value for an analysis variable	displays a missing value for any statistics (except N and NMISS) in that cell	use MISSTEXT= in the TABLE statement
there are no data for a formatted value	does not display that formatted value in the table	use PRELOADFMT in the CLASS statement with PRINTMISS in the TABLE statement, or use CLASSDATA= in the PROC TABULATE statement, or add dummy observations to the input data set so that it contains data for each formatted value
a FREQ variable value is missing or is less than 1	does not use that observation to calculate statistics	no alternative
a WEIGHT variable value is missing or 0	uses a value of 0	no alternative

1 The CLASS statement applies to all TABLE statements in a PROC TABULATE step. Therefore, if you define a variable as a class variable, PROC TABULATE omits observations that have missing values for that variable even if you do not use the variable in a TABLE statement.

This section presents a series of PROC TABULATE steps that illustrate how PROC TABULATE treats missing values. The following program creates the data set and formats that are used in this section and prints the data set. The data set COMPREV contains no missing values (see Figure 37.5 on page 1190).

```
proc format;
   value cntryfmt 1='United States'
                  2='Japan';
```

```
   value compfmt  1='Supercomputer'
                  2='Mainframe'
                  3='Midrange'
                  4='Workstation'
                  5='Personal Computer'
                  6='Laptop';
run;


data comprev;
   input Country Computer Rev90 Rev91 Rev92;
   datalines;
1 1 788.8 877.6 944.9
1 2 12538.1 9855.6 8527.9
1 3 9815.8 6340.3 8680.3
1 4 3147.2 3474.1 3722.4
1 5 18660.9 18428.0 23531.1
2 1 469.9 495.6 448.4
2 2 5697.6 6242.4 5382.3
2 3 5392.1 5668.3 4845.9
2 4 1511.6 1875.5 1924.5
2 5 4746.0 4600.8 4363.7
;


proc print data=comprev noobs;
   format country cntryfmt. computer compfmt.;
   title 'The Data Set COMPREV';
run;
```

Figure 37.5 The Data Set COMPREV

The Data Set COMPREV 1

Country	Computer	Rev90	Rev91	Rev92
United States	Supercomputer	788.8	877.6	944.9
United States	Mainframe	12538.1	9855.6	8527.9
United States	Midrange	9815.8	6340.3	8680.3
United States	Workstation	3147.2	3474.1	3722.4
United States	Personal Computer	18660.9	18428.0	23531.1
Japan	Supercomputer	469.9	495.6	448.4
Japan	Mainframe	5697.6	6242.4	5382.3
Japan	Midrange	5392.1	5668.3	4845.9
Japan	Workstation	1511.6	1875.5	1924.5
Japan	Personal Computer	4746.0	4600.8	4363.7

No Missing Values

The following PROC TABULATE step produces Figure 37.6 on page 1191:

```
proc tabulate data=comprev;
   class country computer;
   var rev90 rev91 rev92;
   table computer*country,rev90 rev91 rev92 /
         rts=32;
```

```
   format country cntryfmt. computer compfmt.;
   title 'Revenues from Computer Sales';
   title2 'for 1990 to 1992';
run;
```

Figure 37.6 Computer Sales Data: No Missing Values

Because the data set contains no missing values, the table includes all observations. All headers and cells contain nonmissing values.

```
               Revenues from Computer Sales                    1
                       for 1990 to 1992

------------------------------------------------------------------
|                             | Rev90    | Rev91    | Rev92  |
|                             |----------+----------+--------|
|                             |  Sum     |  Sum     |  Sum   |
|-----------------------------+----------+----------+--------|
|Computer      |Country       |          |          |        |
|--------------+--------------|          |          |        |
|Supercomputer |United States |    788.80|    877.60|  944.90|
|              |--------------+----------+----------+--------|
|              |Japan         |    469.90|    495.60|  448.40|
|--------------+--------------+----------+----------+--------|
|Mainframe     |United States |  12538.10|   9855.60| 8527.90|
|              |--------------+----------+----------+--------|
|              |Japan         |   5697.60|   6242.40| 5382.30|
|--------------+--------------+----------+----------+--------|
|Midrange      |United States |   9815.80|   6340.30| 8680.30|
|              |--------------+----------+----------+--------|
|              |Japan         |   5392.10|   5668.30| 4845.90|
|--------------+--------------+----------+----------+--------|
|Workstation   |United States |   3147.20|   3474.10| 3722.40|
|              |--------------+----------+----------+--------|
|              |Japan         |   1511.60|   1875.50| 1924.50|
|--------------+--------------+----------+----------+--------|
|Personal      |United States |  18660.90|  18428.00|23531.10|
|Computer      |--------------+----------+----------+--------|
|              |Japan         |   4746.00|   4600.80| 4363.70|
------------------------------------------------------------------
```

A Missing Class Variable

The next program copies COMPREV and alters the data so that the eighth observation has a missing value for Computer. Except for specifying this new data set, the program that produces Figure 37.7 on page 1192 is the same as the program that produces Figure 37.6 on page 1191. By default, PROC TABULATE ignores observations with missing values for a class variable.

```
data compmiss;
   set comprev;
   if _n_=8 then computer=.;
run;


proc tabulate data=compmiss;
   class country computer;
   var rev90 rev91 rev92;
   table computer*country,rev90 rev91 rev92 /
```

```
          rts=32;
    format country cntryfmt. computer compfmt.;
    title 'Revenues from Computer Sales';
    title2 'for 1990 to 1992';
run;
```

Figure 37.7 Computer Sales Data: Midrange, Japan, Deleted

The observation with a missing value for Computer was the category **`Midrange, Japan`**. This category no longer exists. By default, PROC TABULATE ignores observations with missing values for a class variable, so this table contains one fewer row than Figure 37.6 on page 1191.

```
                Revenues from Computer Sales                1
                       for 1990 to 1992

------------------------------------------------------------
|                            | Rev90   | Rev91    | Rev92   |
|                            |---------+----------+---------|
|                            |  Sum    |  Sum     |  Sum    |
|----------------------------+---------+----------+---------|
|Computer      |Country      |         |          |         |
|--------------+-------------|         |          |         |
|Supercomputer |United States |   788.80|    877.60|   944.90|
|              |-------------+---------+----------+---------|
|              |Japan        |   469.90|    495.60|   448.40|
|--------------+-------------+---------+----------+---------|
|Mainframe     |United States | 12538.10|   9855.60|  8527.90|
|              |-------------+---------+----------+---------|
|              |Japan        |  5697.60|   6242.40|  5382.30|
|--------------+-------------+---------+----------+---------|
|Midrange      |United States |  9815.80|   6340.30|  8680.30|
|--------------+-------------+---------+----------+---------|
|Workstation   |United States |  3147.20|   3474.10|  3722.40|
|              |-------------+---------+----------+---------|
|              |Japan        |  1511.60|   1875.50|  1924.50|
|--------------+-------------+---------+----------+---------|
|Personal      |United States | 18660.90|  18428.00| 23531.10|
|Computer      |-------------+---------+----------+---------|
|              |Japan        |  4746.00|   4600.80|  4363.70|
------------------------------------------------------------
```

Including Observations with Missing Class Variables

This program adds the MISSING option to the previous program. MISSING is available either in the PROC TABULATE statement or in the CLASS statement. If you want MISSING to apply only to selected class variables, but not to others, specify MISSING in a separate CLASS statement with the selected variable(s). The MISSING option includes observations with missing values of a class variable in the report (see Figure 37.8 on page 1193).

```
proc tabulate data=compmiss missing;
   class country computer;
   var rev90 rev91 rev92;
   table computer*country,rev90 rev91 rev92 /
         rts=32;
   format country cntryfmt. computer compfmt.;
   title 'Revenues from Computer Sales';
   title2 'for 1990 to 1992';
```

```
run;
```

Figure 37.8 Computer Sales Data: Missing Value for COMP

This table includes a category with missing values of COMP. This category makes up the first row of data in the table.

```
               Revenues from Computer Sales                1
                     for 1990 to 1992

------------------------------------------------------------
|                          |  Rev90   | Rev91   | Rev92   |
|                          |----------+---------+---------|
|                          |   Sum    |  Sum    |  Sum    |
|--------------------------+----------+---------+---------|
|Computer     |Country     |          |         |         |
|-------------+------------|          |         |         |
|.            |Japan       |   5392.10|  5668.30|  4845.90|
|-------------+------------+----------+---------+---------|
|Supercomputer|United States|    788.80|   877.60|   944.90|
|             |------------+----------+---------+---------|
|             |Japan       |    469.90|   495.60|   448.40|
|-------------+------------+----------+---------+---------|
|Mainframe    |United States|  12538.10|  9855.60|  8527.90|
|             |------------+----------+---------+---------|
|             |Japan       |   5697.60|  6242.40|  5382.30|
|-------------+------------+----------+---------+---------|
|Midrange     |United States|   9815.80|  6340.30|  8680.30|
|-------------+------------+----------+---------+---------|
|Workstation  |United States|   3147.20|  3474.10|  3722.40|
|             |------------+----------+---------+---------|
|             |Japan       |   1511.60|  1875.50|  1924.50|
|-------------+------------+----------+---------+---------|
|Personal     |United States|  18660.90| 18428.00| 23531.10|
|Computer     |------------+----------+---------+---------|
|             |Japan       |   4746.00|  4600.80|  4363.70|
------------------------------------------------------------
```

Formatting Headings for Observations with Missing Class Variables

By default, as shown in Figure 37.8 on page 1193, PROC TABULATE displays missing values of a class variable as one of the standard SAS characters for missing values (a period, a blank, an underscore, or one of the letters A through Z). If you want to display something else instead, you must assign a format to the class variable that has missing values, as shown in the following program (see Figure 37.9 on page 1194):

```
proc format;
   value misscomp 1='Supercomputer'
                  2='Mainframe'
                  3='Midrange'
                  4='Workstation'
                  5='Personal Computer'
                  6='Laptop'
                  .='No type given';
run;


proc tabulate data=compmiss missing;
   class country computer;
   var rev90 rev91 rev92;
```

```
   table computer*country,rev90 rev91 rev92 /
         rts=32;
   format country cntryfmt. computer misscomp.;
   title 'Revenues for Computer Sales';
   title2 'for 1990 to 1992';
run;
```

Figure 37.9 Computer Sales Data: Text Supplied for Missing COMP Value

In this table, the missing value appears as the text that the MISSCOMP. format specifies.

```
               Revenues for Computer Sales                1
                     for 1990 to 1992

-----------------------------------------------------------
|                            | Rev90  | Rev91  | Rev92  |
|                            |--------+--------+--------|
|                            |  Sum   |  Sum   |  Sum   |
|----------------------------+--------+--------+--------|
|Computer      |Country      |        |        |        |
|--------------+-------------|        |        |        |
|No type given |Japan        | 5392.10| 5668.30| 4845.90|
|--------------+-------------+--------+--------+--------|
|Supercomputer |United States |  788.80|  877.60|  944.90|
|              |-------------+--------+--------+--------|
|              |Japan        |  469.90|  495.60|  448.40|
|--------------+-------------+--------+--------+--------|
|Mainframe     |United States |12538.10| 9855.60| 8527.90|
|              |-------------+--------+--------+--------|
|              |Japan        | 5697.60| 6242.40| 5382.30|
|--------------+-------------+--------+--------+--------|
|Midrange      |United States | 9815.80| 6340.30| 8680.30|
|--------------+-------------+--------+--------+--------|
|Workstation   |United States | 3147.20| 3474.10| 3722.40|
|              |-------------+--------+--------+--------|
|              |Japan        | 1511.60| 1875.50| 1924.50|
|--------------+-------------+--------+--------+--------|
|Personal      |United States |18660.90|18428.00|23531.10|
|Computer      |-------------+--------+--------+--------|
|              |Japan        | 4746.00| 4600.80| 4363.70|
-----------------------------------------------------------
```

Providing Headings for All Categories

By default, PROC TABULATE evaluates each page that it prints and omits columns and rows for categories that do not exist. For example, Figure 37.9 on page 1194 does not include a row for **`No type given`** and for **`United States`** or for **`Midrange`** and for **`Japan`** because there are no data in these categories. If you want the table to represent all possible categories, use the PRINTMISS option in the TABLE statement, as shown in the following program (see Figure 37.10 on page 1195):

```
proc tabulate data=compmiss missing;
   class country computer;
   var rev90 rev91 rev92;
   table computer*country,rev90 rev91 rev92 /
         rts=32 printmiss;
   format country cntryfmt. computer misscomp.;
   title 'Revenues for Computer Sales';
   title2 'for 1990 to 1992';
```

```
run;
```

Figure 37.10 Computer Sales Data: Missing Statistics Values

This table contains a row for the categories **No type given**, **United States** and **Midrange, Japan**. Because there are no data in these categories, the values for the statistics are all missing.

```
                Revenues for Computer Sales                1
                       for 1990 to 1992

-------------------------------------------------------------
|                            | Rev90   |  Rev91  |  Rev92  |
|                            |---------+---------+---------|
|                            |  Sum    |   Sum   |   Sum   |
|----------------------------+---------+---------+---------|
|Computer      |Country      |         |         |         |
|--------------+-------------|         |         |         |
|No type given |United States|        .|        .|        .|
|              |-------------+---------+---------+---------|
|              |Japan        |  5392.10|  5668.30|  4845.90|
|--------------+-------------+---------+---------+---------|
|Supercomputer |United States|   788.80|   877.60|   944.90|
|              |-------------+---------+---------+---------|
|              |Japan        |   469.90|   495.60|   448.40|
|--------------+-------------+---------+---------+---------|
|Mainframe     |United States| 12538.10|  9855.60|  8527.90|
|              |-------------+---------+---------+---------|
|              |Japan        |  5697.60|  6242.40|  5382.30|
|--------------+-------------+---------+---------+---------|
|Midrange      |United States|  9815.80|  6340.30|  8680.30|
|              |-------------+---------+---------+---------|
|              |Japan        |        .|        .|        .|
|--------------+-------------+---------+---------+---------|
|Workstation   |United States|  3147.20|  3474.10|  3722.40|
|              |-------------+---------+---------+---------|
|              |Japan        |  1511.60|  1875.50|  1924.50|
|--------------+-------------+---------+---------+---------|
|Personal      |United States| 18660.90| 18428.00| 23531.10|
|Computer      |-------------+---------+---------+---------|
|              |Japan        |  4746.00|  4600.80|  4363.70|
-------------------------------------------------------------
```

Providing Text for Cells That Contain Missing Values

If some observations in a category contain missing values for analysis variables, PROC TABULATE does not use those observations to calculate statistics (except N and NMISS). However, if each observation in a category contains a missing value, PROC TABULATE displays a missing value for the value of the statistic. To replace missing values for analysis variables with text, use the MISSTEXT= option in the TABLE statement to specify the text to use, as shown in the following program (see Figure 37.11 on page 1196).

```
proc tabulate data=compmiss missing;
   class country computer;
   var rev90 rev91 rev92;
   table computer*country,rev90 rev91 rev92 /
         rts=32 printmiss misstext='NO DATA!';
   format country cntryfmt. computer misscomp.;
   title 'Revenues for Computer Sales';
```

```
   title2 'for 1990 to 1992';
run;
```

Figure 37.11 Computer Sales Data: Text Supplied for Missing Statistics Values

This table replaces the period normally used to display missing values with the text of the MISSTEXT= option.

```
             Revenues for Computer Sales                    1
                   for 1990 to 1992

 ------------------------------------------------------------
 |                             | Rev90  | Rev91  | Rev92  |
 |                             |--------+--------+--------|
 |                             |   Sum  |  Sum   |  Sum   |
 |-----------------------------+--------+--------+--------|
 |Computer      |Country       |        |        |        |
 |--------------+--------------|        |        |        |
 |No type given |United States |NO DATA!|NO DATA!|NO DATA!|
 |              |--------------+--------+--------+--------|
 |              |Japan         | 5392.10| 5668.30| 4845.90|
 |--------------+--------------+--------+--------+--------|
 |Supercomputer |United States |  788.80|  877.60|  944.90|
 |              |--------------+--------+--------+--------|
 |              |Japan         |  469.90|  495.60|  448.40|
 |--------------+--------------+--------+--------+--------|
 |Mainframe     |United States |12538.10| 9855.60| 8527.90|
 |              |--------------+--------+--------+--------|
 |              |Japan         | 5697.60| 6242.40| 5382.30|
 |--------------+--------------+--------+--------+--------|
 |Midrange      |United States | 9815.80| 6340.30| 8680.30|
 |              |--------------+--------+--------+--------|
 |              |Japan         |NO DATA!|NO DATA!|NO DATA!|
 |--------------+--------------+--------+--------+--------|
 |Workstation   |United States | 3147.20| 3474.10| 3722.40|
 |              |--------------+--------+--------+--------|
 |              |Japan         | 1511.60| 1875.50| 1924.50|
 |--------------+--------------+--------+--------+--------|
 |Personal      |United States |18660.90|18428.00|23531.10|
 |Computer      |--------------+--------+--------+--------|
 |              |Japan         | 4746.00| 4600.80| 4363.70|
 ------------------------------------------------------------
```

Providing Headings for All Values of a Format

PROC TABULATE prints headings only for values that appear in the input data set. For example, the format COMPFMT. provides for six possible values of COMP. Only five of these values occur in the data set COMPREV. The data set contains no data for laptop computers.

If you want to include headings for all possible values of COMP (perhaps to make it easier to compare the output with tables that are created later when you do have data for laptops), you have three different ways to create such a table:

- Use the PRELOADFMT option in the CLASS statement with the PRINTMISS option in the TABLE statement. See Example 3 on page 1203 for another example that uses PRELOADFMT.
- Use the CLASSDATA= option in the PROC TABULATE statement. See Example 2 on page 1201 for an example that uses the CLASSDATA= option.
- Add dummy values to the input data set so that each value that the format handles appears at least once in the data set.

The following program adds the PRELOADFMT option to a CLASS statement that contains the relevant variable.

The results are shown in Figure 37.12 on page 1197.

```
proc tabulate data=compmiss missing;
   class country;
   class computer / preloadfmt;
   var rev90 rev91 rev92;
   table computer*country,rev90 rev91 rev92 /
         rts=32 printmiss misstext='NO DATA!';
   format country cntryfmt. computer compfmt.;
   title 'Revenues for Computer Sales';
   title2 'for 1990 to 1992';
run;
```

Figure 37.12 Computer Sales Data: All Possible COMP Valued Included

This table contains a heading for each possible value of COMP.

```
               Revenues for Computer Sales              1
                     for 1990 to 1992

------------------------------------------------------------
|                           | Rev90  | Rev91  |  Rev92 |
|                           |--------+--------+--------|
|                           |  Sum   |   Sum  |   Sum  |
|---------------------------+--------+--------+--------|
|Computer      |Country     |        |        |        |
|--------------+------------|        |        |        |
|.             |United States|NO DATA!|NO DATA!|NO DATA!|
|              |------------+--------+--------+--------|
|              |Japan       | 5392.10| 5668.30| 4845.90|
|--------------+------------+--------+--------+--------|
|Supercomputer |United States|  788.80|  877.60|  944.90|
|              |------------+--------+--------+--------|
|              |Japan       |  469.90|  495.60|  448.40|
|--------------+------------+--------+--------+--------|
|Mainframe     |United States|12538.10| 9855.60| 8527.90|
|              |------------+--------+--------+--------|
|              |Japan       | 5697.60| 6242.40| 5382.30|
|--------------+------------+--------+--------+--------|
|Midrange      |United States| 9815.80| 6340.30| 8680.30|
|              |------------+--------+------------+----|
|              |Japan       |NO DATA!|NO DATA!|NO DATA!|
|--------------+------------+--------+------------+----|
|Workstation   |United States| 3147.20| 3474.10| 3722.40|
|              |------------+--------+------------+----|
|              |Japan       | 1511.60| 1875.50| 1924.50|
|--------------+------------+--------+--------+--------|
|Personal      |United States|18660.90|18428.00|23531.10|
|Computer      |------------+--------+--------+--------|
|              |Japan       | 4746.00| 4600.80| 4363.70|
|--------------+------------+--------+--------+--------|
|Laptop        |United States|NO DATA!|NO DATA!|NO DATA!|
|              |------------+--------+--------+--------|
|              |Japan       |NO DATA!|NO DATA!|NO DATA!|
------------------------------------------------------------
```

Understanding the Order of Headings with ORDER=DATA

The ORDER= option applies to all class variables. Occasionally, you want to order the headings for different variables differently. One method for doing this is to group the data as you want them to appear and to specify ORDER=DATA.

For this technique to work, the first value of the first class variable must occur in the data with all possible values of all the other class variables. If this criterion is not met, the order of the headings may surprise you.

The following program creates a simple data set in which the observations are ordered first by the values of Animal, then by the values of Food. The ORDER= option in the PROC TABULATE statement orders the heading for the class variables by the order of their appearance in the data set (see Figure 37.13 on page 1198). Although **bones** is the first value for Food in the group of observations where Animal= **dog**, all other values for Food appear before **bones** in the data set because **bones** never appears when Animal= **cat**. Therefore, the header for **bones** in the table in Figure 37.13 on page 1198 is not in alphabetic order.

In other words, PROC TABULATE maintains for subsequent categories the order that was established by earlier categories. If you want to reestablish the order of Food for each value of Animal, use BY-group processing. PROC TABULATE creates a separate table for each BY group, so that the ordering can differ from one BY group to the next.

```
data foodpref;
   input Animal $ Food $;
   datalines;
cat fish
cat meat
cat milk
dog bones
dog fish
dog meat
;

proc tabulate data=foodpref format=9.
              order=data;
   class animal food;
   table animal*food;
run;
```

Figure 37.13 Ordering the Headings of Class Variables

						1
	Animal					
cat			dog			
Food			Food			
fish	meat	milk	fish	meat	bones	
N	N	N	N	N	N	
1	1	1	1	1	1	

Examples

Example 1: Creating a Basic Two-Dimensional Table

Procedure features:

PROC TABULATE statement options:

FORMAT=

TABLE statement

crossing (* operator)

TABLE statement options:

RTS=

Other features: FORMAT statement

This example

- creates a category for each type of user (residential or business) in each division of each region
- applies the same format to all cells in the table
- applies a format to each class variable
- extends the space for row headings.

Program

The data set ENERGY contains data on expenditures of energy for business and residential customers in individual states in the Northeast and West regions of the United States. A DATA step on page 1503 creates the data set.

```
data energy;
   length State $2;
   input Region Division state $ Type Expenditures;
   datalines;
1 1 ME 1 708
1 1 ME 2 379

. . . more lines of data . . .

4 4 HI 1 273
4 4 HI 2 298
;
```

PROC FORMAT creates formats for Region, Division, and Type.

```
proc format;
   value regfmt 1='Northeast'
                2='South'
                3='Midwest'
                4='West';
   value divfmt 1='New England'
                2='Middle Atlantic'
                3='Mountain'
                4='Pacific';
   value usetype 1='Residential Customers'
                 2='Business Customers';
run;
```

The FORMAT= option specifies DOLLAR12. as the default format for the value in each table cell.

```
options nodate pageno=1 linesize=80 pagesize=60;
proc tabulate data=energy format=dollar12.;
```

The CLASS statement identifies Region, Division, and Type as class variables. The VAR statement identifies Expenditures as an analysis variable.

```
   class region division type;
   var expenditures;
```

The TABLE statement creates a row for each formatted value of Region. Nested within each row are rows for each formatted value of Division. The TABLE statement also creates a column for each formatted value of Type. Each cell created by these rows and columns contains the sum of the analysis variable Expenditures for all observations that contribute to that cell.

```
    table region*division,
          type*expenditures
```

RTS= provides 25 characters per line for row headings.

```
          / rts=25;
```

The FORMAT statement assigns formats to Region, Division, and Type. The TITLE statements specify the titles.

```
   format region regfmt. division divfmt. type usetype.;
   title 'Energy Expenditures for Each Region';
   title2 '(millions of dollars)';
run;
```

Output

Energy Expenditures for Each Region 1
(millions of dollars)

		Type	
		Residential Customers	Business Customers
		Expenditures	Expenditures
		Sum	Sum
Region	Division		
Northeast	New England	$7,477	$5,129
	Middle Atlantic	$19,379	$15,078
West	Mountain	$5,476	$4,729
	Pacific	$13,959	$12,619

Example 2: Specifying Class Variable Combinations to Appear in a Table

Procedure features:

PROC TABULATE Statement options:

CLASSDATA=
EXCLUSIVE

Data set: ENERGY on page 1199

Formats: REGFMT., DIVFMT., and USETYPE. on page 1200

This example

- uses the CLASSDATA= option to specify combinations of class variables to appear in a table
- uses the EXCLUSIVE option to restrict the output to only the combinations specified in the CLASSDATA= data set. Without the EXCLUSIVE option, the output would be the same as in Example 1 on page 1199.

Program

The data set CLASSES contains the combinations of class variable values that PROC TABULATE uses to create the table.

```
data classes;
   input region division type;
   datalines;
```

```
1 1 1
1 1 2
4 4 1
4 4 2
;
```

CLASSDATA= and EXCLUSIVE restrict the class level combinations to those specified in the CLASSES data set.

```
options nodate pageno=1 linesize=80 pagesize=60;
proc tabulate data=energy format=dollar12.
              classdata=classes exclusive;
```

The CLASS statement identifies Region, Division, and Type as class variables. The VAR statement identifies Expenditures as an analysis variable.

```
    class region division type;
    var expenditures;
```

The TABLE statement creates a row for each formatted value of Region. Nested within each row are rows for each formatted value of Division. The TABLE statement also creates a column for each formatted value of Type. Each cell created by these rows and columns contains the sum of the analysis variable Expenditures for all observations that contribute to that cell.

```
    table region*division,
          type*expenditures
```

RTS= provides 25 characters per line for row headings.

```
         / rts=25;
```

The FORMAT statement assigns formats to Region, Division, and Type. The TITLE statements specify the titles.

```
    format region regfmt. division divfmt. type usetype.;
    title 'Energy Expenditures for Each Region';
    title2 '(millions of dollars)';
run;
```

Output

Energy Expenditures for Each Region 1
(millions of dollars)

		Type	
		Residential Customers	Business Customers
		Expenditures	Expenditures
		Sum	Sum
Region	Division		
Northeast	New England	$7,477	$5,129
West	Pacific	$13,959	$12,619

Example 3: Using Preloaded Formats with Class Variables

Procedure features:

PROC TABULATE statement option:

OUT=

CLASS statement options:

EXCLUSIVE
PRELOADFMT

TABLE statement option:

PRINTMISS

Other features: PRINT procedure

Data set: ENERGY on page 1199

Formats: REGFMT., DIVFMT., and USETYPE. on page 1200

This example

- creates a table that includes all possible combinations of formatted class variable values (PRELOADFMT with PRINTMISS), even if those combinations have a zero frequency and even if they do not make sense
- restricts the data in the table to combinations of formatted class variable values that appear in the input data set (PRELOADFMT with EXCLUSIVE).
- writes the output to an output data set, and prints that data set.

Program

The FORMAT= option specifies DOLLAR12. as the default format for the value in each table cell.

```
options nodate pageno=1 linesize=80 pagesize=60;
proc tabulate data=energy format=dollar12.;
```

PRELOADFMT specifies that PROC TABULATE use the preloaded values of the user-defined formats for the class variables.

```
    class region division type / preloadfmt;
    var expenditures;
```

PRINTMISS specifies that all possible combinations of user-defined formats be used as the levels of the class variables.

```
    table region*division,
          type*expenditures / rts=25 printmiss;
```

The FORMAT statement assigns formats to Region, Division, and Type. The TITLE statements specify the titles.

```
    format region regfmt. division divfmt. type usetype.;
    title 'Energy Expenditures for Each Region';
    title2 '(millions of dollars)';
run;
```

The OUT= option specifies the name of the output data set to which PROC TABULATE writes the data.

```
proc tabulate data=energy format=dollar12. out=tabdata;
```

The EXCLUSIVE option (used with PRELOADFMT) restricts the output to only the combinations of formatted class variable values that appear in the input data set.

```
    class region division type / preloadfmt exclusive;
    var expenditures;
```

The PRINTMISS option is not specified in this case. If it were, it would override the EXCLUSIVE option in the CLASS statement.

```
    table region*division,
          type*expenditures / rts=25;
```

The FORMAT statement assigns formats to Region, Division, and Type. The TITLE statements specify the titles.

```
   format region regfmt. division divfmt. type usetype.;
   title 'Energy Expenditures for Each Region';
   title2 '(millions of dollars)';
run;
```

The PRINT procedure lists the output data set from PROC TABULATE.

```
proc print data=tabdata;
run;
```

Output

This output, created with the PRELOADFMT and PRINTMISS options, contains all possible combinations of preloaded user-defined formats for the class variable values. It includes combinations with zero frequencies, and combinations that make no sense, such as **`Northeast`** and **`Pacific`**.

Energy Expenditures for Each Region 1
(millions of dollars)

		Type	
		Residential Customers	Business Customers
		Expenditures	Expenditures
		Sum	Sum
Region	Division		
Northeast	New England	$7,477	$5,129
	Middle Atlantic	$19,379	$15,078
	Mountain	.	.
	Pacific	.	.
South	New England	.	.
	Middle Atlantic	.	.
	Mountain	.	.
	Pacific	.	.
Midwest	New England	.	.
	Middle Atlantic	.	.
	Mountain	.	.
	Pacific	.	.
West	New England	.	.
	Middle Atlantic	.	.
	Mountain	$5,476	$4,729
	Pacific	$13,959	$12,619

This output, created with the PRELOADFMT and EXCLUSIVE options, contains only those combinations of preloaded user-defined formats for the class variable values that appear in the input data set. This output is identical to the output from Example 1 on page 1199.

Energy Expenditures for Each Region 1
(millions of dollars)

		Type	
		Residential Customers	Business Customers
		Expenditures	Expenditures
		Sum	Sum
Region	Division		
Northeast	New England	$7,477	$5,129
	Middle Atlantic	$19,379	$15,078
West	Mountain	$5,476	$4,729
	Pacific	$13,959	$12,619

This output is a listing of the output data set from PROC TABULATE. It contains the data created with the PRELOADFMT and EXCLUSIVE options specified.

Energy Expenditures for Each Region
(millions of dollars)

Obs	Region	Division	Type	_TYPE_	_PAGE_	_TABLE_	Expenditures_Sum
1	Northeast	New England	Residential Customers	111	1	1	7477
2	Northeast	New England	Business Customers	111	1	1	5129
3	Northeast	Middle Atlantic	Residential Customers	111	1	1	19379
4	Northeast	Middle Atlantic	Business Customers	111	1	1	15078
5	West	Mountain	Residential Customers	111	1	1	5476
6	West	Mountain	Business Customers	111	1	1	4729
7	West	Pacific	Residential Customers	111	1	1	13959
8	West	Pacific	Business Customers	111	1	1	12619

Example 4: Using Multilabel Formats

Procedure features:

CLASS statement options:
- MLF

PROC TABULATE statement options:
- FORMAT=

TABLE statement
- ALL class variable
- concatenation (blank operator)
- crossing (* operator)
- grouping elements (parentheses operator)
- label
- variable list

Other features:

FORMAT procedure

FORMAT statement

VALUE statement options:
- MULTILABEL

This example

- shows how to specify a multilabel format in the VALUE statement of PROC FORMAT
- shows how to activate multilabel format processing using the MLF option with the CLASS statement
- demonstrates the behavior of the N statistic when multilabel format processing is activated.

Program

The CARSURVEY data set contains data from a survey distributed by a car manufacturer to a focus group of potential customers brought together to evaluate new car names. Each observation in the data set contains an id, the participant's age, and the participant's ratings of four car names. A DATA step creates the data set.

```
options nodate pageno=1 linesize=80 pagesize=64;

data carsurvey;
   input Rater Age Progressa Remark Jupiter Dynamo;
   datalines;
1   38  94  98  84  80
2   49  96  84  80  77
3   16  64  78  76  73
4   27  89  73  90  92

. . . more lines of data . . .

77   61  92  88  77  85
78   24  87  88  88  91
```

```
79   18  54  50  62  74
80   62  90  91  90  86
;
```

The FORMAT procedure creates a multilabel format for ages using the MULTILABEL option on page 451.

```
proc format;
   value agefmt (multilabel notsorted)
         15 - 29 = 'Below 30 years'
         30 - 50 = 'Between 30 and 50'
       51 - high = 'Over 50 years'
         15 - 19 = '15 to 19'
         20 - 25 = '20 to 25'
         25 - 39 = '25 to 39'
         40 - 55 = '40 to 55'
       56 - high = '56 and above';
run;
```

The FORMAT= option specifies up to ten digits as the default format for the value in each table cell.

```
proc tabulate data=carsurvey format=10.;
```

The CLASS statement identifies Age as the class variable and uses the MLF option to activate multilabel format processing. The VAR statement identifies Progressa, Remark, Jupiter, and Dynamo as the analysis variables.

```
   class age /mlf;
   var progressa remark jupiter dynamo;
```

The row dimension of the TABLE statement creates a row for each formatted value of Age. Multilabel formatting allows an observation to be included in multiple rows or age categories. The row dimension uses the ALL class variable to sumarize information for all rows. The column dimension uses the N statistic to calculate the number of observations for each age group. Notice the result of the N statistic crossed with the ALL class variable in the row dimension is the total number of observations instead of the sum of the N statistics for the rows. The column dimension uses the ALL class variable at the beginning of a crossing to assign a label, **`Potential Car Names`** , instead of calculating statistics. The four nested columns calculate the mean ratings for the car names for each age group.

```
   table age all, n all='Potential Car Names'*(progressa remark
   jupiter dynamo)*mean;
```

The TITLE1 and TITLE2 statements specify the first and second titles.

```
   title1 "Rating Four Potential Car Names";
   title2 "Rating Scale 0-100 (100 is the highest rating)";
```

The FORMAT statement assigns the user-defined format. **`agefmt.`**, to Age for this analysis.

```
   format age agefmt.;
run;
```

Output

Output 37.3

Rating Four Potential Car Names 1
Rating Scale 0-100 (100 is the highest rating)

		Potential Car Names			
		Progressa	Remark	Jupiter	Dynamo
	N	Mean	Mean	Mean	Mean
Age					
15 to 19	14	75	78	81	73
20 to 25	11	89	88	84	89
25 to 39	26	84	90	82	72
40 to 55	14	85	87	80	68
56 and above	15	84	82	81	75
Below 30 years	36	82	84	82	75
Between 30 and 50	25	86	89	81	73
Over 50 years	19	82	84	80	76
All	80	83	86	81	74

Example 5: Customizing Row and Column Headings

Procedure features:

TABLE statement

labels

Data set: ENERGY on page 1199

Formats: REGFMT., DIVFMT., and USETYPE on page 1200

This example shows how to customize row and column headings. A label specifies text for a heading. A blank label creates a blank heading. PROC TABULATE removes the space for blank column headings from the table.

Program

The FORMAT= option specifies DOLLAR12. as the default format for the value in each table cell.

```
options nodate pageno=1 linesize=80 pagesize=60;
proc tabulate data=energy format=dollar12.;
```

The CLASS statement identifies Region, Division, and Type as class variables. The VAR statement identifies Expenditures as an analysis variable.

```
    class region division type;
    var expenditures;
```

The TABLE statement creates a row for each formatted value of Region. Nested within each row are rows for each formatted value of Division. The TABLE statement also creates a column for each formatted value of Type. Each cell created by these rows and columns contains the sum of the analysis variable Expenditures for all observations that contribute to that cell. Text in quotation marks specifies headings for the corresponding variable or statistic. Although Sum is the default statistic, it is specified here so that you can remove the heading.

```
     table region*division,
          type='Customer Base'*expenditures=' '*sum=' '
```

RTS= provides 25 characters per line for row headings.

```
          / rts=25;
```

The FORMAT statement assigns formats to Region, Division, and Type. The TITLE statements specify the titles.

```
    format region regfmt. division divfmt. type usetype.;
    title 'Energy Expenditures for Each Region';
    title2 '(millions of dollars)';
run;
```

Output

The headings for Region, Division, and Type contain text specified in the TABLE statement. The TABLE statement eliminated the headings for Expenditures and Sum.

```
                Energy Expenditures for Each Region                    1
                       (millions of dollars)

-------------------------------------------------------
|                       |      Customer Base      |
|                       |-------------------------|
|                       |Residential |  Business  |
|                       | Customers  | Customers  |
|-----------------------+------------+------------|
|Region     |Division   |            |            |
|-----------+-----------|            |            |
|Northeast  |New England|      $7,477|      $5,129|
|           |-----------+------------+------------|
|           |Middle     |            |            |
|           |Atlantic   |     $19,379|     $15,078|
|-----------+-----------+------------+------------|
|West       |Mountain   |      $5,476|      $4,729|
|           |-----------+------------+------------|
|           |Pacific    |     $13,959|     $12,619|
-------------------------------------------------------
```

Example 6: Summarizing Information with the Universal Class Variable ALL

Procedure features:

PROC TABULATE statement options:

FORMAT=

TABLE statement:

ALL class variable
concatenation (blank operator)
format modifiers
grouping elements (parentheses operator)

Data set: ENERGY on page 1199

Formats: REGFMT., DIVFMT., and USETYPE on page 1200

This example shows how to use the universal class variable ALL to summarize information from multiple categories.

Program

The FORMAT= option specifies COMMA12. as the default format for the value in each table cell.

```
options nodate pageno=1 linesize=64 pagesize=60;
proc tabulate data=energy format=comma12.;
```

The CLASS statement identifies Region, Division, and Type as class variables. The VAR statement identifies Expenditures as an analysis variable.

```
    class region division type;
    var expenditures;
```

The row dimension of the TABLE statement creates a row for each formatted value of Region. Nested within each row are rows for each formatted value of Division and a row (labeled **Subtotal**) that summarizes all divisions in the region. The last row of the report (labeled **Total for All Regions**) summarizes all regions. The format modifier f=DOLLAR12. assigns the DOLLAR12. format to the cells in this row.

```
table region*(division all='Subtotal')
          all='Total for All Regions'*f=dollar12.,
```

The column dimension of the TABLE statement creates a column for each formatted value of Type and a column labeled **All customers** that shows expenditures for all customers in a row of the table. Each cell created by these rows and columns contains the sum of the analysis variable Expenditures for all observations that contribute to that cell. Text in quotation marks specifies headings for the corresponding variable or statistic. Although Sum is the default statistic, it is specified here so that you can remove the heading.

```
          type='Customer Base'*expenditures=' '*sum=' '
          all='All Customers'*expenditures=' '*sum=' '
```

RTS= provides 25 characters per line for row headings.

```
        / rts=25;
```

The FORMAT statement assigns formats to Region, Division, and Type. The TITLE statements specify the titles.

```
    format region regfmt. division divfmt. type usetype.;
    title 'Energy Expenditures for Each Region';
    title2 '(millions of dollars)';
run;
```

Output

The universal class variable ALL provides subtotals and totals in this table.

Energy Expenditures for Each Region 1
(millions of dollars)

		Customer Base		
		Residential Customers	Business Customers	All Customers
Region	Division			
Northeast	New England	7,477	5,129	12,606
	Middle Atlantic	19,379	15,078	34,457
	Subtotal	26,856	20,207	47,063
West	Division			
	Mountain	5,476	4,729	10,205
	Pacific	13,959	12,619	26,578
	Subtotal	19,435	17,348	36,783
Total for All Regions		$46,291	$37,555	$83,846

Example 7: Eliminating Row Headings

Procedure features:
TABLE statement:
labels
ROW=FLOAT

Data set: ENERGY on page 1199

Formats: REGFMT., DIVFMT., and USETYPE on page 1200

This example shows how to eliminate blank row headings from a table. To do so, you must both provide blank labels for the row headings and specify ROW=FLOAT in the TABLE statement.

Program

The FORMAT= option specifies DOLLAR12. as the default format for the value in each table cell.

```
options nodate pageno=1 linesize=80 pagesize=60;
proc tabulate data=energy format=dollar12.;
```

The CLASS statement identifies Region, Division, and Type as class variables. The VAR statement identifies Expenditures as an analysis variable.

```
    class region division type;
    var expenditures;
```

The row dimension of the TABLE statement creates a row for each formatted value of Region. Nested within these rows is a row for each formatted value of Division. The analysis variable Expenditures and the Sum statistic are also included in the row dimension, so PROC TABULATE creates row headings for them as well. The text in quotation marks specifies the headings. In the case of Expenditures and Sum, the headings are blank.

```
    table region*division*expenditures=' '*sum=' ',
```

The column dimension of the TABLE statement creates a column for each formatted value of Type.

```
          type='Customer Base'
```

RTS= provides 25 characters per line for row headings. ROW=FLOAT eliminates blank row headings.

```
          / rts=25 row=float;
```

The FORMAT statement assigns formats to Region, Division, and Type. The TITLE statements specify the titles.

```
    format region regfmt. division divfmt. type usetype.;
    title 'Energy Expenditures for Each Region';
    title2 '(millions of dollars)';
run;
```

Output

Compare this table with the table in Example 5 on page 1210. The two tables are identical, but the program that creates the table uses Expenditures and Sum in the column dimension. PROC TABULATE automatically eliminates blank headings from the column dimension, whereas you must specify ROW=FLOAT to eliminate blank headings from the row dimension.

```
                Energy Expenditures for Each Region                    1
                       (millions of dollars)

-------------------------------------------------------
|                       |      Customer Base          |
|                       |-----------------------------|
|                       |Residential |  Business      |
|                       | Customers  | Customers      |
|-----------------------+------------+----------------|
|Region     |Division   |            |                |
|-----------+-----------|            |                |
|Northeast  |New England|      $7,477|         $5,129|
|           |-----------+------------+----------------|
|           |Middle     |            |                |
|           |Atlantic   |     $19,379|        $15,078|
|-----------+-----------+------------+----------------|
|West       |Mountain   |      $5,476|         $4,729|
|           |-----------+------------+----------------|
|           |Pacific    |     $13,959|        $12,619|
-------------------------------------------------------
```

Example 8: Indenting Row Headings and Eliminating Horizontal Separators

Procedure features:

PROC TABULATE statement options:

NOSEPS

TABLE statement options:

INDENT=

Data set: ENERGY on page 1199

Formats: REGFMT., DIVFMT., and USETYPE on page 1200

This example shows how to condense the structure of a table by

- removing row headings for class variables
- indenting nested rows underneath parent rows instead of placing them next to each other
- eliminating horizontal separator lines from the row titles and the body of the table.

Program

The FORMAT= option specifies DOLLAR12. as the default format for the value in each table cell. NOSEPS eliminates horizontal separator lines from row titles and from the body of the table.

```
options nodate pageno=1 linesize=80 pagesize=60;
proc tabulate data=energy format=dollar12. noseps;
```

The CLASS statement identifies Region, Division, and Type as class variables. The VAR statement identifies Expenditures as an analysis variable.

```
    class region division type;
    var expenditures;
```

The TABLE statement creates a row for each formatted value of Region. Nested within each row are rows for each formatted value of Division. The TABLE statement also creates a column for each formatted value of Type. Each cell created by these rows and columns contains the sum of the analysis variable Expenditures for all observations that contribute to that cell. Text in quotation marks in all dimensions specifies headings for the corresponding variable or statistic. Although Sum is the default statistic, it is specified here so that you can remove the heading.

```
     table region*division,
           type='Customer Base'*expenditures=' '*sum=' '
```

RTS= provides 25 characters per line for row headings. INDENT= removes row headings for class variables, places values for Division beneath values for Region rather than beside them, and indents values for Division 4 spaces.

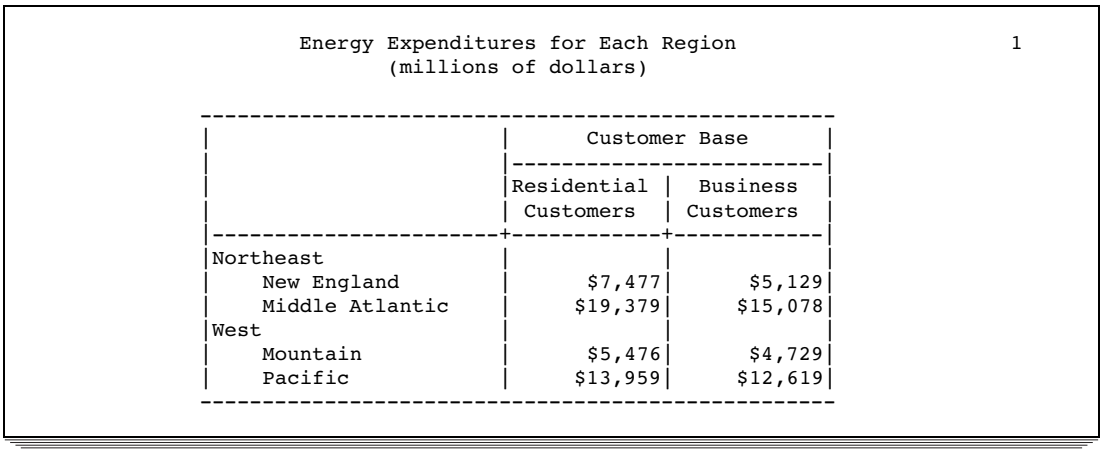
```
          / rts=25 indent=4;
```

The FORMAT statement assigns formats to Region, Division, and Type. The TITLE statements specify the titles.

```
    format region regfmt. division divfmt. type usetype.;
    title 'Energy Expenditures for Each Region';
    title2 '(millions of dollars)';
run;
```

Output

NOSEPS removes the separator lines from the row titles and the body of the table. INDENT= eliminates the row headings for Region and Division, and indents values for Division underneath values for Region.



Energy Expenditures for Each Region 1
(millions of dollars)

	Customer Base	
	Residential Customers	Business Customers
Northeast		
New England	$7,477	$5,129
Middle Atlantic	$19,379	$15,078
West		
Mountain	$5,476	$4,729
Pacific	$13,959	$12,619

Example 9: Creating Multipage Tables

Procedure features:

TABLE statement

ALL class variable
BOX=
CONDENSE
INDENT=
page expression

Data set: ENERGY on page 1199

Formats: REGFMT., DIVFMT., and USETYPE. on page 1200

This example creates a separate table for each region and one table for all regions. By default, PROC TABULATE creates each table on a separate page, but the CONDENSE option places them all on the same page.

Program

The FORMAT= option specifies DOLLAR12. as the default format for the value in each table cell.

```
options nodate pageno=1 linesize=80 pagesize=60;
proc tabulate data=energy format=dollar12.;
```

The CLASS statement identifies Region, Division, and Type as class variables. The VAR statement identifies Expenditures as an analysis variable.

```
    class region division type;
    var expenditures;
```

The page dimension of the TABLE statement creates one table for each formatted value of Region and one table for all regions. Text in quotation marks provides the heading for each page.

```
    table region='Region: ' all='All Regions',
```

The row dimension creates a row for each formatted value of Division and a row for all divisions. Text in quotation marks provides the row headings.

```
    division all='All Divisions',
```

The column dimension of the TABLE statement creates a column for each formatted value of Type. Each cell created by these pages, rows, and columns contains the sum of the analysis variable Expenditures for all observations that contribute to that cell. Text in quotation marks specifies headings for the corresponding variable or statistic. Although Sum is the default statistic, it is specified here so that you can remove the heading.

```
          type='Customer Base'*expenditures=' '*sum=' '
```

RTS= provides 25 characters per line for row headings. BOX= places the page heading inside the box above the row headings. CONDENSE places as many tables as possible on one physical page. INDENT= eliminates the row heading for Division. (Because there is no nesting in the row dimension, there is nothing to indent.)

```
          / rts=25 box=_page_ condense indent=1;
```

The FORMAT statement assigns formats to Region, Division, and Type. The TITLE statements specify the titles.

```
    format region regfmt. division divfmt. type usetype.;
    title 'Energy Expenditures for Each Region and All Regions';
    title2 '(millions of dollars)';
run;
```

Output

Energy Expenditures for Each Region and All Regions 1
(millions of dollars)

Region: Northeast	Customer Base	
	Residential Customers	Business Customers
New England	$7,477	$5,129
Middle Atlantic	$19,379	$15,078
All Divisions	$26,856	$20,207

Region: West	Customer Base	
	Residential Customers	Business Customers
Mountain	$5,476	$4,729
Pacific	$13,959	$12,619
All Divisions	$19,435	$17,348

All Regions	Customer Base	
	Residential Customers	Business Customers
New England	$7,477	$5,129
Middle Atlantic	$19,379	$15,078
Mountain	$5,476	$4,729
Pacific	$13,959	$12,619
All Divisions	$46,291	$37,555

Example 10: Reporting on Multiple-Response Survey Data

Procedure features:

TABLE statement:

- denominator definition (angle bracket operators)
- N statistic
- PCTN statistic
- variable list

Other features:

FORMAT procedure

SAS system options:
FORMDLIM=
NONUMBER
SYMPUT routine

The two tables in this example show

- which factors most influenced customers' decisions to buy products
- where customers heard of the company.

The reports appear on one physical page with only one page number. By default, they would appear on separate pages.

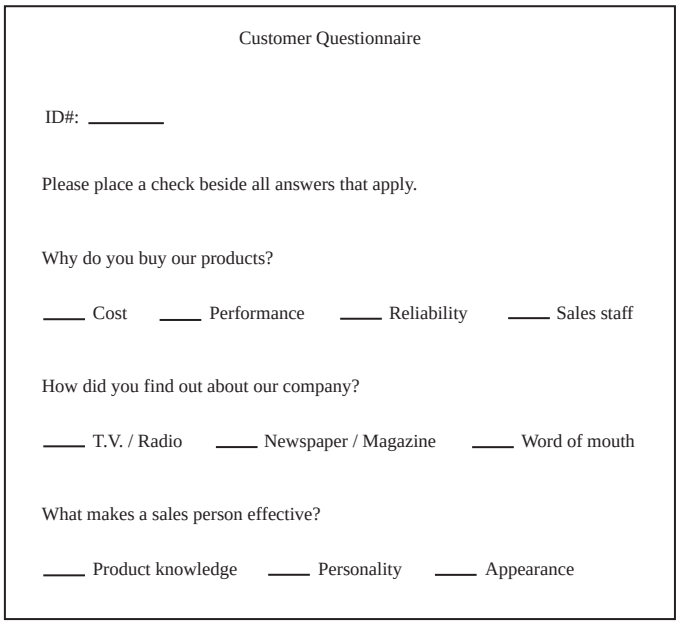
In addition to showing how to create these tables, this example shows how to

- use a DATA step to count the number of observations in a data set
- store that value in a macro variable
- access that value later in the SAS session.

Collecting the Data

Figure 37.14 on page 1221 shows the survey form used to collect data.

Figure 37.14 Completed Survey Form


Customer Questionnaire

ID#: _____

Please place a check beside all answers that apply.

Why do you buy our products?

____ Cost ____ Performance ____ Reliability ____ Sales staff

How did you find out about our company?

____ T.V. / Radio ____ Newspaper / Magazine ____ Word of mouth

What makes a sales person effective?

____ Product knowledge ____ Personality ____ Appearance

Program

The FORMDLIM= option replaces the character that delimits page breaks with a single blank. By default, a new physical page starts whenever a page break occurs.

```
options nodate pageno=1 linesize=80 pagesize=18
                          formdlim=' ';
```

The CUSTOMER_RESPONSE data set contains data from a customer survey. Each observation in the data set contains information about factors that influence one respondent's decisions to buy products. A DATA step on page 1496 creates the data set. Using missing values rather than 0's is crucial for calculating frequency counts in PROC TABULATE.

```
data customer_response;
   input Customer Factor1-Factor4 Source1-Source3
         Quality1-Quality3;
   datalines;
1 . . 1 1 1 1 . 1 . .
2 1 1 . 1 1 1 . 1 1 .
3 . . 1 1 1 1 . . . .

. . . more lines of data . . .

119 . . . 1 . . . 1 . .
120 1 1 . 1 . . . . 1 .
;
```

The SET statement reads the descriptor portion of CUSTOMER_RESPONSE at compile time and stores the number of observations (the number of respondents) in COUNT. The SYMPUT routine stores the value of COUNT in the macro variable NUM. This variable is available to the remainder of the SAS session. The IF 0 condition, which is always false, ensures that the SET statement, which reads the observations, never executes. (Reading observations is unnecessary.) The STOP statement ensures that the DATA step executes only once.

```
data _null_;
   if 0 then set customer_response nobs=count;
   call symput('num',left(put(count,4.)));
   stop;
run;
```

The FORMAT procedure creates a format for percentages. The PCTFMT. format writes all values with at least one digit to the left of the decimal point and with one digit to the right of the decimal point. A blank and a percent sign follow the digits.

```
proc format;
   picture pctfmt low-high='009.9 %';
run;
```

The VAR statement identifies Factor1, Factor2, Factor3, Factor4, and Customer as the analysis variables. Customer must be listed because it appears in the denominator definition.

```
proc tabulate data=customer_response;
   var factor1-factor4 customer;
```

The TABLE statement creates a row for each factor, a column for frequency counts, and a column for the percentages. Text in quotation marks supplies headers for the corresponding row or column. The format modifiers F=7. and F=PCTFMT9. provide formats for values in the associated cells and extend the column widths to accommodate the column headers.

```
   table factor1='Cost'
         factor2='Performance'
         factor3='Reliability'
         factor4='Sales Staff',
         (n='Count'*f=7. pctn<customer>='Percent'*f=pctfmt9.) ;
```

The TITLE statements specify titles.

```
   title 'Customer Survey Results: Spring 1996';
   title3 'Factors Influencing the Decision to Buy';
run;
```

The SAS system option NONUMBER suppresses page numbers for subsequent pages.

```
options nonumber;
```

The VAR statement specifies the analysis variables. Customer must be in the variable list because it appears in the denominator definition.

```
proc tabulate data=customer_response;
   var source1-source3 customer;
```

The TABLE statement creates a row for each source of the company name, a column for frequency counts, and a column for the percentages. Text in quotation marks supplies a header for the corresponding row or column.

```
   table source1='TV/Radio'
         source2='Newspaper'
         source3='Word of Mouth',
         (n='Count'*f=7. pctn<customer>='Percent'*f=pctfmt9.) ;
```

The TITLE and FOOTNOTE statements specify the title and footnote. The macro variable NUM resolves to the number of respondents. The FOOTNOTE statement uses double rather than single quotes so that the macro variable will resolve.

```
   title 'Source of Company Name';
   footnote "Number of Respondents: &num";
run;
```

The FORMDLIM= option resets the page delimiter to a page eject. The NUMBER option resumes the display of page numbers on subsequent pages.

```
options formdlim='' number;
```

Output

Customer Survey Results: Spring 1996 1

Factors Influencing the Decision to Buy

	Count	Percent
Cost	87	72.5 %
Performance	62	51.6 %
Reliability	30	25.0 %
Sales Staff	120	100.0 %

Source of Company Name

	Count	Percent
TV/Radio	92	76.6 %
Newspaper	69	57.5 %
Word of Mouth	26	21.6 %

Number of Respondents: 120

Example 11: Reporting on Multiple-Choice Survey Data

Procedure features:

TABLE statement:

N statistic

Other features:

FORMAT procedure

TRANSPOSE procedure

Data set options:

RENAME=

This report of listener preferences shows how many listeners select each type of programming during each of seven time periods on a typical weekday. The data were

collected by a survey, and the results were stored in a SAS data set. Although this data set contains all the information needed for this report, the information is not arranged in a way that PROC TABULATE can use.

To make this crosstabulation of time of day and choice of radio programming, you must have a data set that contains a variable for time of day and a variable for programming preference. PROC TRANSPOSE reshapes the data into a new data set that contains these variables. Once the data are in the appropriate form, PROC TABULATE creates the report.

Collecting the Data

Figure 37.15 on page 1225 shows the survey form used to collect data.

Figure 37.15 Completed Survey Form

phone_ _ _

LISTENER SURVEY

1. _____ What is your age?

2. _____ What is your gender?

3. _____ On the average WEEKDAY, how many hours do you listen to the radio?

4. _____ On the average WEEKEND-DAY, how many hours do you listen to the radio?

Use codes 1-8 for question 5. Use codes 0-8 for 6-19.
0 Do not listen at that time

1 Rock	5 Classical
2 Top 40	6 Easy Listening
3 Country	7 News/Information/Talk
4 Jazz	8 Other

5. _____ What style of music or radio programming do you most often listen to?

On a typical WEEKDAY, what kind of radio programming do you listen to	On a typical WEEKEND-DAY, what kind of radio programming do you listen to
6. _____ from 6-9 a.m.?	13. _____ from 6-9 a.m.?
7. _____ from 9 a.m. to noon?	14. _____ from 9 a.m. to noon?
8. _____ from noon to 1 p.m.?	15. _____ from noon to 1 p.m.?
9. _____ from 1-4 p.m.?	16. _____ from 1-4 p.m.?
10. _____ from 4-6 p.m.?	17. _____ from 4-6 p.m.?
11. _____ from 6-10 p.m.?	18. _____ from 6-10 p.m.?
12. _____ from 10 p.m. to 2 a.m.?	19. _____ from 10 p.m. to 2 a.m.?

An external file on page 1522 contains the raw data for the survey. Several lines from that file appear here.

```
967 32 f 5 3 5
7 5 5 5 7 0 0 0 8 7 0 0 8 0
781 30 f 2 3 5
```

```
5 0 0 0 5 0 0 0 4 7 5 0 0 0
859 39 f 1 0 5
1 0 0 0 1 0 0 0 0 0 0 0 0 0

. . . more lines of data . . .

859 32 m .25 .25 1
1 0 0 0 0 0 0 0 1 0 0 0 0 0
```

Program

```
options nodate pageno=1 linesize=132 pagesize=40;
```

The data set RADIO contains data from a survey of 336 listeners. The data set contains information about listeners and their preferences in radio programming. The INFILE statement specifies the external file that contains the data. MISSOVER prevents the input pointer from going to the next record if it doesn't find values in the current line for all variables listed in the INPUT statement. Each raw-data record contains two lines of information about each listener. The INPUT statement reads only the information that this example needs. The / line control skips the first line of information in each record. The rest of the INPUT statement reads Time1-Time7 from the beginning of the second line. These variables represent the listener's radio programming preference for each of seven time periods on weekdays (see Figure 37.15 on page 1225). Listener=_N_ assigns a unique identifier to each listener.

```
data radio;
   infile 'input-file' missover;
   input /(Time1-Time7) ($1. +1);
   listener=_n_;
run;
```

PROC FORMAT creates formats for the time of day and the choice of programming.

```
proc format;
   value $timefmt 'Time1'='6-9 a.m.'
                  'Time2'='9 a.m. to noon'
                  'Time3'='noon to 1 p.m.'
                  'Time4'='1-4 p.m.'
                  'Time5'='4-6 p.m.'
                  'Time6'='6-10 p.m.'
                  'Time7'='10 p.m. to 2 a.m.'
                    other='*** Data Entry Error ***';
   value $pgmfmt      '0'="Don't Listen"
                  '1','2'='Rock and Top 40'
                      '3'='Country'
              '4','5','6'='Jazz, Classical, and Easy Listening'
                      '7'='News/ Information /Talk'
                      '8'='Other'
                    other='*** Data Entry Error ***';
run;
```

PROC TRANSPOSE creates RADIO_TRANSPOSED. This data set contains the variable Listener from the original data set. It also contains two transposed variables: Timespan and Choice. Timespan contains the names of the variables (Time1-Time7) from the input data set that are transposed to form observations in the output data set. Choice contains the values of these variables. (See "A Closer Look" on page 1228 for a complete explanation of the PROC TRANSPOSE step.)

```
proc transpose data=radio
               out=radio_transposed(rename=(col1=Choice))
               name=Timespan;
   by listener;
   var time1-time7;
```

The FORMAT statement permanently associates these formats with the variables in the output data set.

```
   format timespan $timefmt. choice $pgmfmt.;
run;
```

The FORMAT= option specifies the default format for the values in each table cell.

```
proc tabulate data=radio_transposed format=12.;
```

The CLASS statement identifies Timespan and Choice as class variables.

```
   class timespan choice;
```

The TABLE statement creates a row for each formatted value of Timespan and a column for each formatted value of Choice. In each column are values for the N statistic. Text in quotation marks supplies headers for the corresponding rows or columns.

```
   table timespan='Time of Day',
         choice='Choice of Radio Program'*n='Number of Listeners';
```

The TITLE statement specifies the title.

```
   title 'Listening Preferences on Weekdays';
run;
```

Output

Listening Preferences on Weekdays 1

	Choice of Radio Program					
	Don't Listen	Rock and Top 40	Country	Jazz, Classical, and Easy Listening	News/ Information /Talk	Other
Time of Day	Number of Listeners	Number of Listeners	Number of Listeners	Number of Listeners	Number of Listeners	Number of Listeners
6-9 a.m.	34	143	7	39	96	17
9 a.m. to noon	214	59	5	51	3	4
noon to 1 p.m.	238	55	3	27	9	4
1-4 p.m.	216	60	5	50	2	3
4-6 p.m.	56	130	6	57	69	18
6-10 p.m.	202	54	9	44	20	7
10 p.m. to 2 a.m.	264	29	3	36	2	2

A Closer Look

Reshape the data

The original input data set has all the information that you need to make the crosstabular report, but PROC TABULATE cannot use the information in that form. PROC TRANSPOSE rearranges the data so that each observation in the new data set contains the variable Listener, a variable for time of day, and a variable for programming preference. PROC TABULATE uses this new data set to create the crosstabular report.

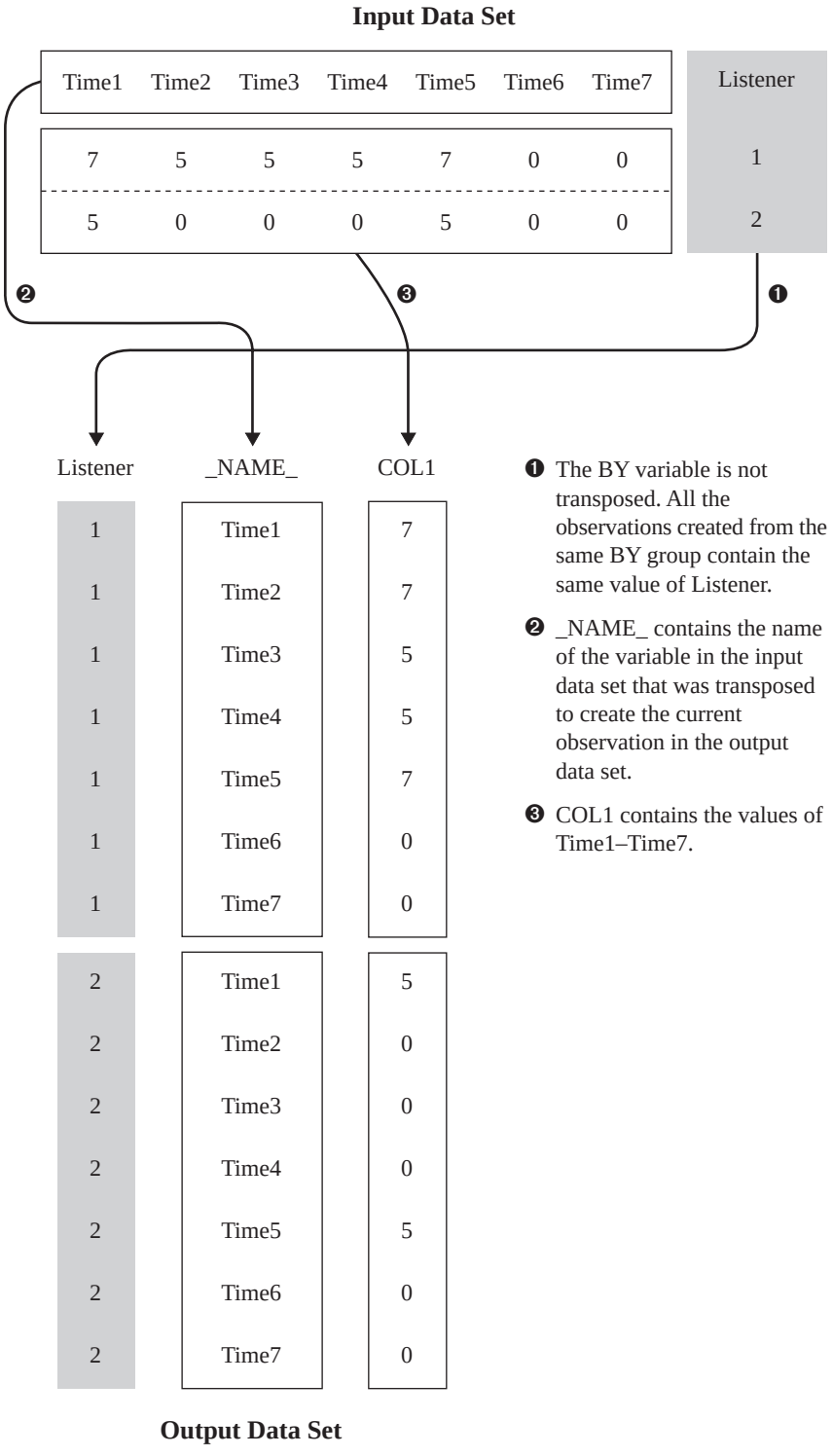
PROC TRANSPOSE restructures data so that values that were stored in one observation are written to one variable. You can specify which variables you want to transpose. This section illustrates how PROC TRANSPOSE reshapes the data. The following section explains the PROC TRANSPOSE step in this example.

When you transpose with BY processing, as this example does, you create from each BY group one observation for each variable that you transpose. In this example, Listener is the BY variable. Each observation in the input data set is a BY group because the value of Listener is unique for each observation.

This example transposes seven variables, Time1 through Time7. Therefore, the output data set has seven observations from each BY group (each observation) in the input data set.

Figure 37.16 on page 1229 uses the first two observations in the input data set to illustrate the transposition.

Figure 37.16 Transposing Two Observations



Understanding the PROC TRANSPOSE Step

This is the PROC TRANSPOSE step that reshapes the data:

```
proc transpose data=radio  ❶
               out=radio_transposed(rename=(col1=Choice))    ❷
```

```
            name=Timespan;    ❸
   by listener;    ❹
   var time1-time7;    ❺
   format timespan $timefmt. choice $pgmfmt.;    ❻
run;
```

1. The DATA= option in the PROC TRANSPOSE statement specifies the input data set.
2. The OUT= option in the PROC TRANSPOSE statement specifies the output data set. The RENAME= data set option renames the transposed variable from COL1 (the default name) to Choice.
3. The NAME= option in the PROC TRANSPOSE statement specifies the name for the variable in the output data set that contains the name of the variable that is being transposed to create the current observation. By default, the name of this variable is _NAME_.
4. The BY statement identifies Listener as the BY variable.
5. The VAR statement identifies Time1 through Time7 as the variables to transpose.
6. The FORMAT statement assigns formats to Timespan and Choice. The PROC TABULATE step that creates the report does not need to format Timespan and Choice because the formats are stored with these variables.

Example 12: Calculating Various Percentage Statistics

Procedure features:

PROC TABULATE statement options:
- FORMAT=

TABLE statement:
- ALL class variable
- COLPCTSUM statistic
- concatenation (blank operator)
- crossing (* operator)
- format modifiers
- grouping elements (parentheses operator)
- labels
- REPPCTSUM statistic
- ROWPCTSUM statistic
- variable list

TABLE statement options:
- ROW=FLOAT
- RTS=

Other features: FORMAT procedure

This example shows how to use three percentage sum statistics: COLPCTSUM, REPPCTSUM, and ROWPCTSUM.

The data set FUNDRAIS contains data on student sales during a school fundraiser. A DATA step creates the data set.

```
options nodate pageno=1 linesize=105 pagesize=60;
data fundrais;
   length name $ 8 classrm $ 1;
   input @1 team $ @8 classrm $ @10 name $
         @19 pencils @23 tablets;
   sales=pencils + tablets;
   cards;
BLUE   A ANN       4   8
RED    A MARY      5  10
GREEN  A JOHN      6   4
RED    A BOB       2   3
BLUE   B FRED      6   8
GREEN  B LOUISE   12   2
BLUE   B ANNETTE   .   9
RED    B HENRY     8  10
GREEN  A ANDREW    3   5
RED    A SAMUEL   12  10
BLUE   A LINDA     7  12
GREEN  A SARA      4   .
BLUE   B MARTIN    9  13
RED    B MATTHEW   7   6
GREEN  B BETH     15  10
RED    B LAURA     4   3
;
```

The FORMAT procedure creates a format for percentages. The PCTFMT. format writes all values with at least one digit, a blank, and a percent sign.

```
proc format;
   picture pctfmt low-high='009 %';
run;
```

The TITLE statement specifies the title.

```
title "Fundraiser Sales";
```

The FORMAT= option specifies up to seven digits as the default format for the value in each table cell.

```
proc tabulate format=7.;
```

The CLASS statement identifies Team and Classrm as class variables. The VAR statement identifies Sales as the analysis variable.

```
   class team classrm;
   var sales;
```

The row dimension of the TABLE statement creates a row for each formatted value of Team. Nested within each row is a row (labeled **sales**) that summarizes sales for the team. The last row of the report summarizes sales for all teams.

```
   table (team all)*sales=' ',
```

The column dimension of the TABLE statement creates a column for each formatted value of Classrm. Nested within each column are columns that summarize sales for the class. The first nested column, labeled **sum**, is the sum of sales for the row for the classroom. The second nested column, labeled **ColPctSum**, is the percentage of the sum of sales for the row for the classroom in relation to the sum of sales for all teams in the classroom. The third nested column, labeled **RowPctSum**, is the percentage of the sum of sales for the row for the classroom in relation to the sum of sales for the row for all classrooms. The fourth nested column, labeled **RepPctSum**, is the percentage of the sum of sales for the row for the classroom in relation to the sum of sales for all teams for all classrooms. The last column of the report summarizes sales for the row for all classrooms.

```
            classrm='Classroom'*(sum
            colpctsum*f=pctfmt9.
            rowpctsum*f=pctfmt9.
            reppctsum*f=pctfmt9.)
            all
```

RTS= provides 20 characters per line for row headings. ROW=FLOAT eliminates blank row headings.

```
            /rts=20 row=float;
  run;
```

Output

```
                                  Fundraiser Sales                                          1

-------------------------------------------------------------------------------------------------
|                  |                              Classroom                              |       |
|                  |-------------------------------------------------------------------------|       |
|                  |                A                |                B                |  All  |
|                  |---------------------------------+---------------------------------+-------|
|                  |  Sum  |ColPctSum|RowPctSum|RepPctSum|  Sum  |ColPctSum|RowPctSum|RepPctSum|  Sum  |
|------------------+-------+---------+---------+---------+-------+---------+---------+---------+-------|
|team              |       |         |         |         |       |         |         |         |       |
|------------------|       |         |         |         |       |         |         |         |       |
|BLUE              |    31 |    34 % |    46 % |    15 % |    36 |    31 % |    53 % |    17 % |    67 |
|------------------+-------+---------+---------+---------+-------+---------+---------+---------+-------|
|GREEN             |    18 |    19 % |    31 % |     8 % |    39 |    34 % |    68 % |    19 % |    57 |
|------------------+-------+---------+---------+---------+-------+---------+---------+---------+-------|
|RED               |    42 |    46 % |    52 % |    20 % |    38 |    33 % |    47 % |    18 % |    80 |
|------------------+-------+---------+---------+---------+-------+---------+---------+---------+-------|
|All               |    91 |   100 % |    44 % |    44 % |   113 |   100 % |    55 % |    55 % |   204 |
-------------------------------------------------------------------------------------------------
```

A Closer Look

Here are the percentage sum statistic calculations used to produce the output for the Blue Team in Classroom A:

COLPCTSUM=31/91*100=34%

ROWPCTSUM=31/67*100=46%

REPPCTSUM=31/204*100=15%

Similar calculations were used to produce the output for the remaining teams and classroom.

Example 13: Using Denominator Definitions to Display Basic Frequency Counts and Percentages

Procedure features:

TABLE statement:

ALL class variable
denominator definitions (angle bracket operators)
N statistic
PCTN statistic

Other features:

FORMAT procedure

Crosstabulation tables (also called *contingency tables* and *stub-and-banner reports*) show combined frequency distributions for two or more variables. This table shows frequency counts for females and males within each of four job classes. The table also shows the percentage that each frequency count represents of

- the total women and men in that job class (row percentage)
- the total for that gender in all job classes (column percentage)
- the total for all employees.

Program

```
options nodate pageno=1 linesize=80 pagesize=60;
```

The JOBCLASS data set contains encoded information about the gender and job class of employees in a fictitious company.

```
data jobclass;
   input Gender Occupation @@;
   datalines;
1 1  1 1  1 1  1 1  1 1  1 1  1 1
1 2  1 2  1 2  1 2  1 2  1 2  1 2
1 3  1 3  1 3  1 3  1 3  1 3  1 3
1 1  1 1  1 1  1 2  1 2  1 2  1 2
1 2  1 2  1 3  1 3  1 4  1 4  1 4
1 4  1 4  1 4  1 1  1 1  1 1  1 1
1 1  1 2  1 2  1 2  1 2  1 2  1 2
1 2  1 3  1 3  1 3  1 3  1 4  1 4
1 4  1 4  1 4  1 1  1 3  2 1  2 1
2 1  2 1  2 1  2 1  2 1  2 2  2 2
2 2  2 2  2 2  2 3  2 3  2 3  2 4
2 4  2 4  2 4  2 4  2 4  2 1  2 3
2 3  2 3  2 3  2 3  2 4  2 4  2 4
2 4  2 4  2 1  2 1  2 1  2 1  2 1
2 2  2 2  2 2  2 2  2 2  2 2  2 2
2 3  2 3  2 4  2 4  2 4  2 1  2 1
```

```
2 1  2 1  2 1  2 2  2 2  2 2  2 3
2 3  2 3  2 3  2 4
;
```

PROC FORMAT creates formats for Gender and Occupation.

```
proc format;
   value gendfmt 1='Female'
                 2='Male'
              other='*** Data Entry Error ***';
   value occupfmt 1='Technical'
                  2='Manager/Supervisor'
                  3='Clerical'
                  4='Administrative'
               other='*** Data Entry Error ***';
run;
```

The FORMAT= option specifies the 8.2 format as the default format for the value in each table cell.

```
proc tabulate data=jobclass format=8.2;
```

The CLASS statement identifies Gender and Occupation as class variables.

```
   class gender occupation;
```

The TABLE statement creates a set of rows for each formatted value of Occupation and for all jobs together. Text in quotation marks supplies a header for the corresponding row.

```
    table (occupation='Job Class' all='All Jobs')
```

For detailed explanations of the structure of this table and of the use of denominator definitions, see "A Closer Look" on page 1236. The asterisk in the row dimension indicates that the statistics that follow in parentheses are nested within the values of Occupation and All to form sets of rows. Each set of rows includes four statistics:

1. N, the frequency count. The format modifier (F=9.) writes the values of N without the decimal places that the default format would use. It also extends the column width to nine characters so that the word **Employees** fits on one line.
2. the percentage of the row total (row percent).
3. the percentage of the column total (column percent).
4. the overall percent. Text in quotation marks supplies the header for the corresponding row. A comma separates the row definition from the column definition.

```
          *(n='Number of employees'*f=9.
            pctn<gender all>='Percent of row total'
            pctn<occupation all>='Percent of column total'
            pctn='Percent of total'),
```

The column dimension creates a column for each formatted value of Gender and for all employees. Text in quotation marks supplies the header for the corresponding column. The RTS= option provides 50 characters per line for row headings.

```
     gender='Gender' all='All Employees'/ rts=50;
```

The FORMAT statement assigns formats to Gender and Occupation. The TITLE statements specify the titles.

```
   format gender gendfmt. occupation occupfmt.;
   title 'Gender Distribution';
   title2 'within Job Classes';
run;
```

Output

Gender Distribution
within Job Classes

		Gender		
		Female	Male	All Employees
Job Class				
Technical	Number of employees	16	18	34
	Percent of row total	47.06	52.94	100.00
	Percent of column total	26.23	29.03	27.64
	Percent of total	13.01	14.63	27.64
Manager/Supervisor	Number of employees	20	15	35
	Percent of row total	57.14	42.86	100.00
	Percent of column total	32.79	24.19	28.46
	Percent of total	16.26	12.20	28.46
Clerical	Number of employees	14	14	28
	Percent of row total	50.00	50.00	100.00
	Percent of column total	22.95	22.58	22.76
	Percent of total	11.38	11.38	22.76
Administrative	Number of employees	11	15	26
	Percent of row total	42.31	57.69	100.00
	Percent of column total	18.03	24.19	21.14
	Percent of total	8.94	12.20	21.14
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59	50.41	100.00
	Percent of column total	100.00	100.00	100.00
	Percent of total	49.59	50.41	100.00

A Closer Look

The part of the TABLE statement that defines the rows of the table uses the PCTN statistic to calculate three different percentages.

In all calculations of PCTN, the numerator is N, the frequency count for one cell of the table. The denominator for each occurrence of PCTN is determined by the *denominator definition*. The denominator definition appears in angle brackets after the keyword PCTN. It is a list of one or more expressions. The list tells PROC TABULATE which frequency counts to sum for the denominator.

Analyzing the Structure of the Table

Taking a close look at the structure of the table helps you understand how PROC TABULATE uses the denominator definitions. The following simplified version of the TABLE statement clarifies the basic structure of the table:

```
table occupation='Job Class' all='All Jobs',
      gender='Gender' all='All Employees';
```

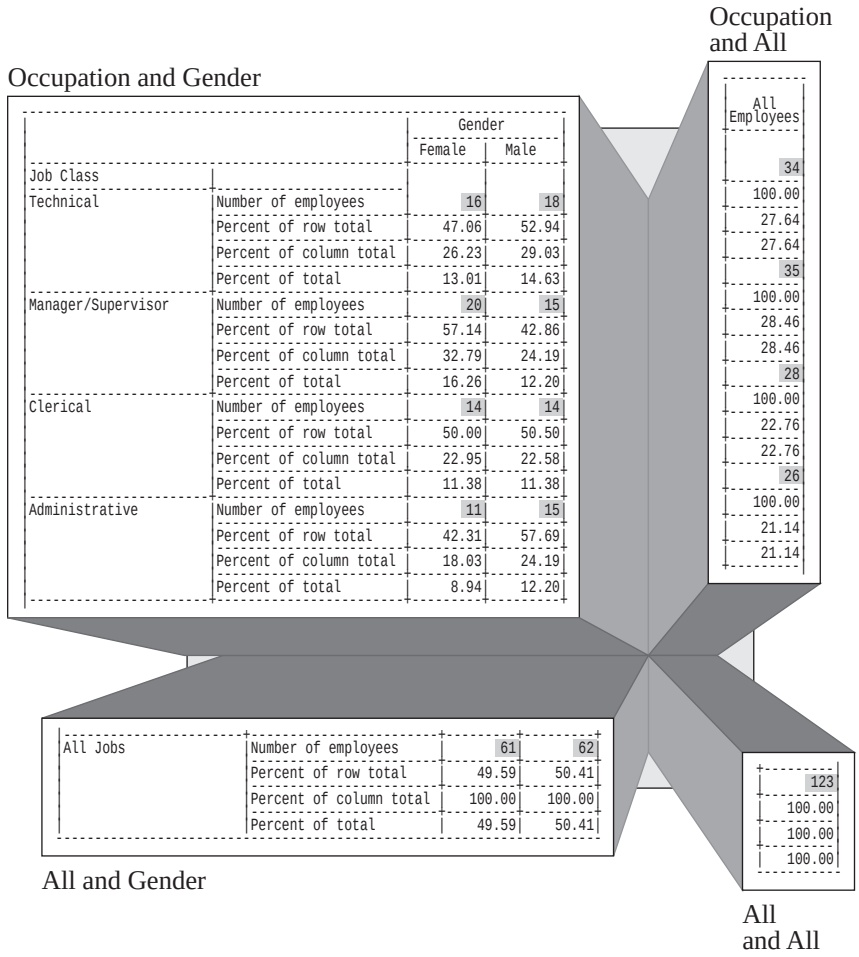
The table is a concatenation of four subtables. In this report, each subtable is a crossing of one class variable in the row dimension and one class variable in the column dimension. Each crossing establishes one or more categories. A *category* is a combination of unique values of class variables, such as **`female, technical`** or **`all, clerical`**. Table 37.8 on page 1237 describes each subtable.

Table 37.8 Contents of Subtables

Class variables contributing to the subtable	Description of frequency counts	Number of categories
Occupation and Gender	number of females in each job or number of males in each job	8
All and Gender	number of females or number of males	2
Occupation and All	number of people in each job	4
All and All	number of people in all jobs	1

Figure 37.17 on page 1238 highlights these subtables and the frequency counts for each category.

Figure 37.17 Illustration of the Four Subtables



		Gender	
		Female	Male
Job Class			
Technical	Number of employees	16	18
	Percent of row total	47.06	52.94
	Percent of column total	26.23	29.03
	Percent of total	13.01	14.63
Manager/Supervisor	Number of employees	20	15
	Percent of row total	57.14	42.86
	Percent of column total	32.79	24.19
	Percent of total	16.26	12.20
Clerical	Number of employees	14	14
	Percent of row total	50.00	50.50
	Percent of column total	22.95	22.58
	Percent of total	11.38	11.38
Administrative	Number of employees	11	15
	Percent of row total	42.31	57.69
	Percent of column total	18.03	24.19
	Percent of total	8.94	12.20

All Employees
34
100.00
27.64
27.64
35
100.00
28.46
28.46
28
100.00
22.76
22.76
26
100.00
21.14
21.14

All Jobs	Number of employees	61	62
	Percent of row total	49.59	50.41
	Percent of column total	100.00	100.00
	Percent of total	49.59	50.41

123
100.00
100.00
100.00

Interpreting Denominator Definitions

The following fragment of the TABLE statement defines the denominator definitions for this report. The PCTN keyword and the denominator definitions are underlined.

```
table (occupation='Job Class' all='All Jobs')
          *(n='Number of employees'*f=5.
            pctn<gender all>='Row percent'
            pctn<occupation all>='Column percent'
            pctn='Percent of total'),
```

Each use of PCTN nests a row of statistics within each value of Occupation and All. Each denominator definition tells PROC TABULATE which frequency counts to sum for the denominators in that row. This section explains how PROC TABULATE interprets these denominator definitions.

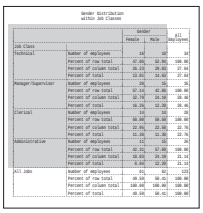
Row Percentages

The part of the TABLE statement that calculates the row percentages and that labels the row is

```
pctn<gender all>='Row percent'
```

Consider how PROC TABULATE interprets this denominator definition for each subtable.

Subtable 1: Occupation and Gender



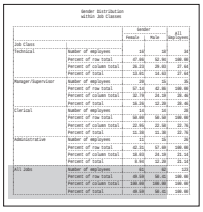
Gender Distribution
within Job Classes

		Gender		All
		Female	Male	Employees
Job Class				
Technical	Number of employees	16	18	34
	Percent of row total	47.06%	52.94%	100.00%
	Percent of column total	26.23%	29.03%	27.64%
	Percent of total	13.01%	14.63%	27.64%
Manager/Supervisor	Number of employees	24	11	35
	Percent of row total	57.14%	42.86%	100.00%
	Percent of column total	32.79%	24.19%	28.46%
	Percent of total	19.51%	12.20%	31.71%
Clerical	Number of employees	14	14	28
	Percent of row total	50.00%	50.00%	100.00%
	Percent of column total	22.95%	22.58%	22.76%
	Percent of total	11.38%	11.38%	22.76%
Administrative	Number of employees	11	13	24
	Percent of row total	45.83%	57.69%	100.00%
	Percent of column total	18.03%	29.03%	21.14%
	Percent of total	8.94%	12.20%	21.14%
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59%	50.41%	100.00%
	Percent of column total	100.00%	100.00%	100.00%
	Percent of total	49.59%	50.41%	100.00%

PROC TABULATE looks at the first element in the denominator definition, Gender, and asks if Gender contributes to the subtable. Because Gender does contribute to the subtable, PROC TABULATE uses it as the denominator definition. This denominator definition tells PROC TABULATE to sum the frequency counts for all occurrences of Gender within the same value of Occupation.

For example, the denominator for the category **female, technical** is the sum of all frequency counts for all categories in this subtable for which the value of Occupation is **technical**. There are two such categories: **female, technical** and **male, technical**. The corresponding frequency counts are 16 and 18. Therefore, the denominator for this category is 16+18, or 34.

Subtable 2: All and Gender



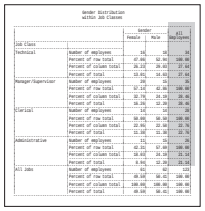
Gender Distribution
within Job Classes

		Gender		All
		Female	Male	Employees
Job Class				
Technical	Number of employees	16	18	34
	Percent of row total	47.06%	52.94%	100.00%
	Percent of column total	26.23%	29.03%	27.64%
	Percent of total	13.01%	14.63%	27.64%
Manager/Supervisor	Number of employees	24	11	35
	Percent of row total	57.14%	42.86%	100.00%
	Percent of column total	32.79%	24.19%	28.46%
	Percent of total	19.51%	12.20%	31.71%
Clerical	Number of employees	14	14	28
	Percent of row total	50.00%	50.00%	100.00%
	Percent of column total	22.95%	22.58%	22.76%
	Percent of total	11.38%	11.38%	22.76%
Administrative	Number of employees	11	13	24
	Percent of row total	45.83%	57.69%	100.00%
	Percent of column total	18.03%	29.03%	21.14%
	Percent of total	8.94%	12.20%	21.14%
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59%	50.41%	100.00%
	Percent of column total	100.00%	100.00%	100.00%
	Percent of total	49.59%	50.41%	100.00%

PROC TABULATE looks at the first element in the denominator definition, Gender, and asks if Gender contributes to the subtable. Because Gender does contribute to the subtable, PROC TABULATE uses it as the denominator definition. This denominator definition tells PROC TABULATE to sum the frequency counts for all occurrences of Gender in the subtable.

For example, the denominator for the category **all, female** is the sum of the frequency counts for **all, female** and **all, male**. The corresponding frequency counts are 61 and 62. Therefore, the denominator for cells in this subtable is 61+62, or 123.

Subtable 3: Occupation and All



Gender Distribution within Job Classes

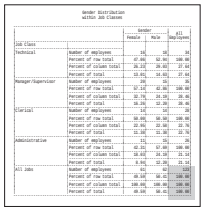
		Gender		All
		Female	Male	Employees
Job Class				
Technical	Number of employees	34	66	34
	Percent of row total	47.06	52.94	100.00
	Percent of column total	26.23	29.60	27.64
	Percent of total	13.01	14.63	27.64
Manager/Supervisor	Number of employees	25	11	25
	Percent of row total	57.14	42.86	100.00
	Percent of column total	32.79	24.19	25.40
	Percent of total	16.26	12.20	25.40
Clerical	Number of employees	14	14	28
	Percent of row total	50.00	50.00	100.00
	Percent of column total	22.95	22.58	22.76
	Percent of total	11.38	11.38	22.76
Administrative	Number of employees	31	31	29
	Percent of row total	42.31	57.69	100.00
	Percent of column total	18.03	24.19	23.12
	Percent of total	8.94	12.20	23.10
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59	50.41	100.00
	Percent of column total	100.00	100.00	100.00
	Percent of total	49.59	50.41	100.00

PROC TABULATE looks at the first element in the denominator definition, Gender, and asks if Gender contributes to the subtable. Because Gender does not contribute to the subtable, PROC TABULATE looks at the next element in the denominator definition, which is All. The variable All does contribute to this subtable, so PROC TABULATE uses it as the denominator definition. All is a reserved class variable with only one category. Therefore, this denominator definition tells PROC TABULATE to use the frequency count of All as the denominator.

For example, the denominator for the category **`clerical, all`** is the frequency count for that category, 28.

Note: In these table cells, because the numerator and the denominator are the same, the row percentages in this subtable are all 100. △

Subtable 4: All and All



Gender Distribution within Job Classes

		Gender		All
		Female	Male	Employees
Job Class				
Technical	Number of employees	34	66	34
	Percent of row total	47.06	52.94	100.00
	Percent of column total	26.23	29.60	27.64
	Percent of total	13.01	14.63	27.64
Manager/Supervisor	Number of employees	25	11	25
	Percent of row total	57.14	42.86	100.00
	Percent of column total	32.79	24.19	25.40
	Percent of total	16.26	12.20	25.40
Clerical	Number of employees	14	14	28
	Percent of row total	50.00	50.00	100.00
	Percent of column total	22.95	22.58	22.76
	Percent of total	11.38	11.38	22.76
Administrative	Number of employees	31	31	29
	Percent of row total	42.31	57.69	100.00
	Percent of column total	18.03	24.19	23.12
	Percent of total	8.94	12.20	23.10
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59	50.41	100.00
	Percent of column total	100.00	100.00	100.00
	Percent of total	49.59	50.41	100.00

PROC TABULATE looks at the first element in the denominator definition, Gender, and asks if Gender contributes to the subtable. Because Gender does not contribute to the subtable, PROC TABULATE looks at the next element in the denominator definition, which is All. The variable All does contribute to this subtable, so PROC TABULATE uses it as the denominator definition. All is a reserved class variable with only one category. Therefore, this denominator definition tells PROC TABULATE to use the frequency count of All as the denominator.

There is only one category in this subtable: **`all, all`**. The denominator for this category is 123.

Note: In this table cell, because the numerator and denominator are the same, the row percentage in this subtable is 100. △

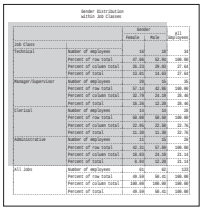
Column Percentages

The part of the TABLE statement that calculates the column percentages and labels the row is

```
pctn<occupation all>='Column percent'
```

Consider how PROC TABULATE interprets this denominator definition for each subtable.

Subtable 1: Occupation and Gender



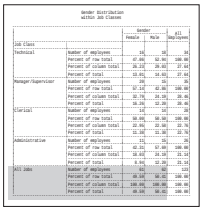
Gender Distribution
within Job Classes

		Gender		All
		Female	Male	Employees
Job Class				
Technical	Number of employees	16	18	34
	Percent of row total	47.06%	52.94%	100.00%
	Percent of column total	26.23%	29.03%	27.64%
	Percent of total	13.01%	14.63%	27.64%
Manager/Supervisor	Number of employees	20	15	35
	Percent of row total	57.14%	42.86%	100.00%
	Percent of column total	32.79%	24.19%	28.46%
	Percent of total	16.26%	12.20%	28.46%
Clerical	Number of employees	14	14	28
	Percent of row total	50.00%	50.00%	100.00%
	Percent of column total	22.95%	22.58%	22.76%
	Percent of total	11.38%	11.38%	22.76%
Administrative	Number of employees	11	15	26
	Percent of row total	42.31%	57.69%	100.00%
	Percent of column total	18.03%	24.19%	21.14%
	Percent of total	8.94%	12.20%	21.14%
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59%	50.41%	100.00%
	Percent of column total	100.00%	100.00%	100.00%
	Percent of total	49.59%	50.41%	100.00%

PROC TABULATE looks at the first element in the denominator definition, Occupation, and asks if Occupation contributes to the subtable. Because Occupation does contribute to the subtable, PROC TABULATE uses it as the denominator definition. This denominator definition tells PROC TABULATE to sum the frequency counts for all occurrences of Occupation within the same value of Gender.

For example, the denominator for the category **manager/supervisor, male** is the sum of all frequency counts for all categories in this subtable for which the value of Gender is **male**. There are four such categories: **technical, male**; **manager/supervisor, male**; **clerical, male**; and **administrative, male**. The corresponding frequency counts are 18, 15, 14, and 15. Therefore, the denominator for this category is 18+15+14+15, or 62.

Subtable 2: All and Gender



Gender Distribution
within Job Classes

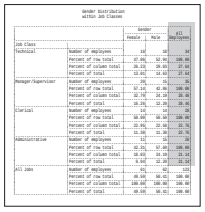
		Gender		All
		Female	Male	Employees
Job Class				
Technical	Number of employees	16	18	34
	Percent of row total	47.06%	52.94%	100.00%
	Percent of column total	26.23%	29.03%	27.64%
	Percent of total	13.01%	14.63%	27.64%
Manager/Supervisor	Number of employees	20	15	35
	Percent of row total	57.14%	42.86%	100.00%
	Percent of column total	32.79%	24.19%	28.46%
	Percent of total	16.26%	12.20%	28.46%
Clerical	Number of employees	14	14	28
	Percent of row total	50.00%	50.00%	100.00%
	Percent of column total	22.95%	22.58%	22.76%
	Percent of total	11.38%	11.38%	22.76%
Administrative	Number of employees	11	15	26
	Percent of row total	42.31%	57.69%	100.00%
	Percent of column total	18.03%	24.19%	21.14%
	Percent of total	8.94%	12.20%	21.14%
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59%	50.41%	100.00%
	Percent of column total	100.00%	100.00%	100.00%
	Percent of total	49.59%	50.41%	100.00%

PROC TABULATE looks at the first element in the denominator definition, Occupation, and asks if Occupation contributes to the subtable. Because Occupation does not contribute to the subtable, PROC TABULATE looks at the next element in the denominator definition, which is All. Because the variable All does contribute to this subtable, PROC TABULATE uses it as the denominator definition. All is a reserved class variable with only one category. Therefore, this denominator definition tells PROC TABULATE to use the frequency count for All as the denominator.

For example, the denominator for the category **all, female** is the frequency count for that category, 61.

Note: In these table cells, because the numerator and denominator are the same, the column percentages in this subtable are all 100. △

Subtable 3: Occupation and All



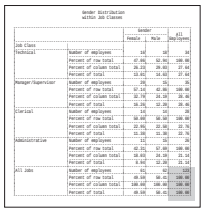
Gender Distribution within Job Classes

		Gender		
		Female	Male	All employees
Job Class				
Technical	Number of employees	18	16	34
	Percent of row total	47.06%	52.94%	100.00%
	Percent of column total	26.21%	29.63%	27.64%
	Percent of total	13.82%	14.63%	27.64%
Manager/Supervisor	Number of employees	24	11	35
	Percent of row total	57.14%	42.86%	100.00%
	Percent of column total	32.74%	24.19%	28.46%
	Percent of total	19.51%	12.20%	28.46%
Clerical	Number of employees	14	14	28
	Percent of row total	50.00%	50.00%	100.00%
	Percent of column total	22.58%	22.58%	22.76%
	Percent of total	11.38%	11.38%	22.76%
Administrative	Number of employees	12	14	26
	Percent of row total	42.31%	57.69%	100.00%
	Percent of column total	18.63%	24.19%	21.14%
	Percent of total	9.76%	12.20%	21.14%
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59%	50.41%	100.00%
	Percent of column total	100.00%	100.00%	100.00%
	Percent of total	49.59%	50.41%	100.00%

PROC TABULATE looks at the first element in the denominator definition, Occupation, and asks if Occupation contributes to the subtable. Because Occupation does contribute to the subtable, PROC TABULATE uses it as the denominator definition. This denominator definition tells PROC TABULATE to sum the frequency counts for all occurrences of Occupation in the subtable.

For example, the denominator for the category **`technical, all`** is the sum of the frequency counts for **`technical, all`**; **`manager/supervisor, all`**; **`clerical, all`**; and **`administrative, all`**. The corresponding frequency counts are 34, 35, 28, and 26. Therefore, the denominator for this category is 34+35+28+26, or 123.

Subtable 4: All and All



Gender Distribution within Job Classes

		Gender		
		Female	Male	All employees
Job Class				
Technical	Number of employees	18	16	34
	Percent of row total	47.06%	52.94%	100.00%
	Percent of column total	26.21%	29.63%	27.64%
	Percent of total	13.82%	14.63%	27.64%
Manager/Supervisor	Number of employees	24	11	35
	Percent of row total	57.14%	42.86%	100.00%
	Percent of column total	32.74%	24.19%	28.46%
	Percent of total	19.51%	12.20%	28.46%
Clerical	Number of employees	14	14	28
	Percent of row total	50.00%	50.00%	100.00%
	Percent of column total	22.58%	22.58%	22.76%
	Percent of total	11.38%	11.38%	22.76%
Administrative	Number of employees	12	14	26
	Percent of row total	42.31%	57.69%	100.00%
	Percent of column total	18.63%	24.19%	21.14%
	Percent of total	9.76%	12.20%	21.14%
All Jobs	Number of employees	61	62	123
	Percent of row total	49.59%	50.41%	100.00%
	Percent of column total	100.00%	100.00%	100.00%
	Percent of total	49.59%	50.41%	100.00%

PROC TABULATE looks at the first element in the denominator definition, Occupation, and asks if Occupation contributes to the subtable. Because Occupation does not contribute to the subtable, PROC TABULATE looks at the next element in the denominator definition, which is All. Because the variable All does contribute to this subtable, PROC TABULATE uses it as the denominator definition. All is a reserved class variable with only one category. Therefore, this denominator definition tells PROC TABULATE to use the frequency count of All as the denominator.

There is only one category in this subtable: **`all, all`**. The frequency count for this category is 123.

Note: In this calculation, because the numerator and denominator are the same, the column percentage in this subtable is 100. △

Total Percentages

The part of the TABLE statement that calculates the total percentages and labels the row is

```
pctn='Total percent'
```

If you do not specify a denominator definition, PROC TABULATE obtains the denominator for a cell by totaling all the frequency counts in the subtable. Table 37.9 on page 1243 summarizes the process for all subtables in this example.

Table 37.9 Denominators for Total Percentages

Class variables contributing to the subtable	Frequency counts	Total
Occupat and Gender	16, 18, 20, 15 14, 14, 11, 15	123
Occupat and All	34, 35, 28, 26	123
Gender and All	61, 62	123
All and All	123	123

Consequently, the denominator for total percentages is always 123.

Example 14: Specifying Style Elements for HTML Output

Procedure features:
STYLE= option in
PROC TABULATE statement
CLASSLEV statement
KEYWORD statement
TABLE statement
VAR statement
Other features: ODS HTML statement
Data set: ENERGY on page 1199
Formats: REGFMT, DIVFMT, and USETYPE on page 1200

This example creates HTML files and specifies style elements for various table regions.

Program

The ODS HTML statement produces output that is written in HTML. The output from PROC TABULATE goes to the body file.

```
ods html body='external-file';
```

The STYLE= option in the PROC TABULATE statement specifies the style element for the data cells of the table.

```
proc tabulate data=energy style=[font_weight=bold];
```

The STYLE= option in the CLASS statement specifies the style element for the class variable name headings.

```
 style=[just=center];
```

The STYLE= option in the CLASSLEV statement specifies the style element for the class variable level value headings.

```
    classlev region division type / style=[just=left];
```

The STYLE= option in the VAR statement specifies a style element for the variable name headings.

```
    var expenditures / style=[font_size=3];
```

The STYLE= option in the KEYWORD statement specifies a style element for keywords.The KEYLABEL statement assigns a label to the keyword.

```
    keyword all sum / style=[font_width=wide];
    keylabel all="Total";
```

The STYLE= option in the dimension expression overrides any other STYLE= specifications in PROC TABULATE, including the STYLE= specification after the slash in the TABLE statement.

```
    table (region all)*(division all*[style=[background=yellow]]),
          (type all)*(expenditures*f=dollar10.) /
          style=[background=red]
```

The STYLE= option in the MISSTEXT option of the TABLE statement specifies a style element to use for the text in table cells that contain missing values.

```
          misstext=[label="Missing" style=[font_weight=light]]
```

The STYLE= option in the BOX option of the TABLE statement specifies a style element to use for text in the empty box above the row titles.

```
          box=[label="Region by Division by Type"
          style=[font_style=italic]];
```

The FORMAT statement assigns formats to Region, Division, and Type. The TITLE statements specify the titles.

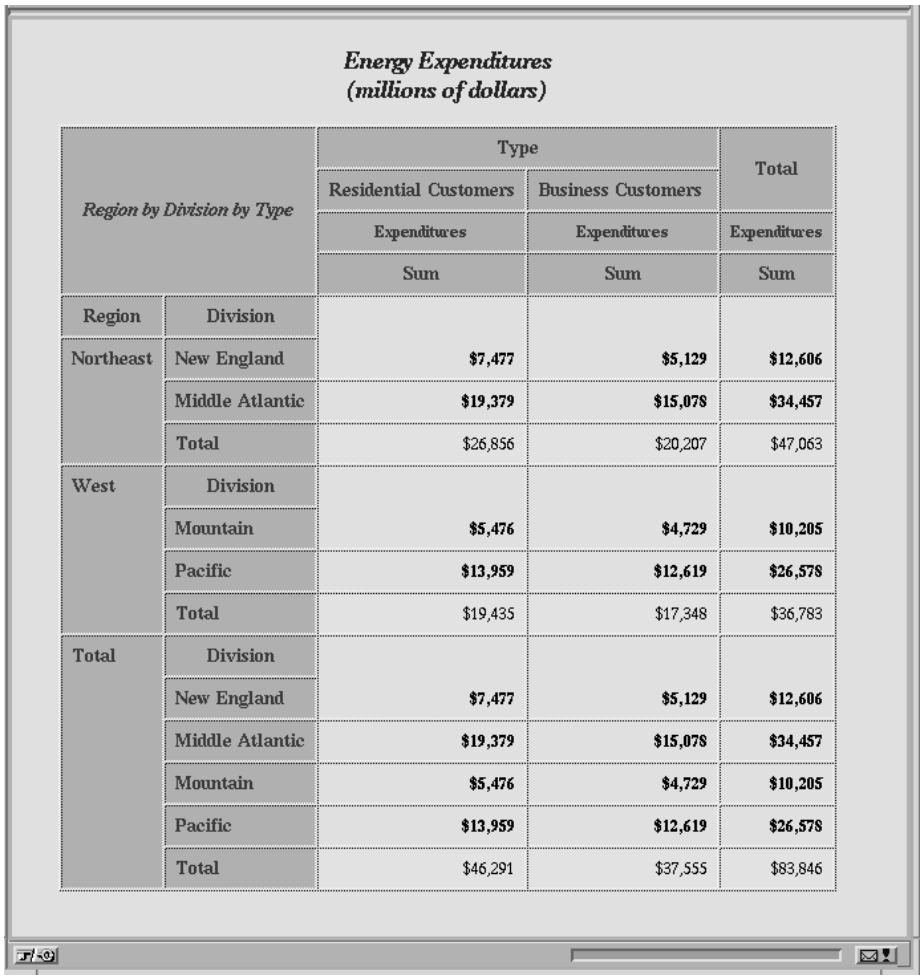
```
    format region regfmt. division divfmt. type usetype.;
    title 'Energy Expenditures';
    title2 '(millions of dollars)';
run;
```

The ODS HTML statement closes the HTML destination.

```
ods html close;
```

HTML Body File

This table uses customized style elements to control font sizes, font widths, justification, and other style attributes for the headings and data cells.



Energy Expenditures
(millions of dollars)

Region by Division by Type		Type		Total
		Residential Customers	Business Customers	
		Expenditures	Expenditures	Expenditures
		Sum	Sum	Sum
Region	Division			
Northeast	New England	**$7,477**	**$5,129**	**$12,606**
	Middle Atlantic	**$19,379**	**$15,078**	**$34,457**
	Total	$26,856	$20,207	$47,063
West	Division			
	Mountain	**$5,476**	**$4,729**	**$10,205**
	Pacific	**$13,959**	**$12,619**	**$26,578**
	Total	$19,435	$17,348	$36,783
Total	Division			
	New England	**$7,477**	**$5,129**	**$12,606**
	Middle Atlantic	**$19,379**	**$15,078**	**$34,457**
	Mountain	**$5,476**	**$4,729**	**$10,205**
	Pacific	**$13,959**	**$12,619**	**$26,578**
	Total	$46,291	$37,555	$83,846

References

Jain, Raj and Chlamtac, Imrich (1985), "The P^2 Algorithm for Dynamic Calculation of Quantiles and Historgrams without Storing Observations," *Communications of the Association of Computing Machinery*, 28:10.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS® Procedures Guide, Version 8,* Cary, NC: SAS Institute Inc., 1999. 1729 pp.

SAS® Procedures Guide, Version 8

ISBN 1–58025–482–9

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999