Introduction

This section of the Syntax Reference provides general information about the syntax in the Tables procedure. Following this introduction, a detailed section describes the subcommands individually in alphabetical order.

Syntax Notation

Like other commands, a TABLES command begins on a new line, is followed by subcommands and keywords, and ends with a period. A command, including its subcommands with related specifications, can continue for as many lines as needed. Syntax can be pasted from a dialog box by clicking on Paste, or it can be entered directly into a syntax window. To get a new syntax window, from the menus choose:

File
  New ▸
    Syntax...

This opens a new window in which you can enter syntax. To execute a single command, place the cursor somewhere in the command and from the menus choose:

Run
  Current

To execute multiple commands, drag the cursor across portions of all the commands. This highlights the section over which you drag the cursor. From the menus choose:

Run
  Selection

There are two other options for executing commands. Place the cursor (as described above) for a single command or multiple commands and click the Run tool on the toolbar or click the right mouse button and choose Run Current.

Like other commands, the following simple rules apply to the TABLES command:

- Subcommands are separated by slashes. The slash before the first subcommand on a command is optional.
- Keywords are not case sensitive, and three-letter abbreviations can be used for most keywords.
- Variable names must be spelled out in full.
- You can use as many lines as you want to specify a single command. However, text included within apostrophes or quotation marks must be contained on a single line.
- You can add space or break lines at almost any point where a single blank is allowed, such as around slashes, parentheses, table structure operators, or between variable names.

- Each line of syntax cannot exceed 80 characters.
- The period must be used as the decimal indicator.

The syntax for each subcommand on the TABLES command is presented in the subcommands section. The syntax diagrams there and elsewhere in this manual use a shorthand style that follows these rules:

- Elements printed in upper case are subcommands or keywords.
- Elements in lower case describe items you should provide.
- Subcommands and keywords can be truncated up to the first three characters; however, the statistical format names, such as COMMA, must be spelled out in full.
- Equals signs are optional.
- Special delimiters (:, (), =, ', /, and ") must be entered exactly as shown.
- Slashes are required between subcommands.
- Blanks or commas must separate keywords, names, labels, and numbers.
- Elements enclosed in square brackets [ ] are optional. Brackets are not part of the syntax.
- Braces { } are not part of the syntax either. They indicate a choice among the elements they enclose. Use only one alternative from a list in braces.
- Ellipses ... also are not part of the syntax. They indicate the option of repeating an element or an entire sequence of elements.
- Default options are in boldface type. A default is the option that TABLES assumes is in effect if you do not explicitly request an alternative.

Subcommand Types

The Tables procedure has two types of subcommands: global and local. **Global subcommands** affect all of the tables that a TABLES command produces. These subcommands can appear in any order but must precede the first TABLE subcommand. Global subcommands do things such as define how a variable is going to be used or define the format of the subsequent tables. **Local subcommands** include the TABLE subcommand itself and the subcommands that follow TABLE. They apply only to the table specified by the preceding TABLE subcommand. Local subcommands define what is going to be in a table and how it's going to be structured. Each TABLE subcommand on a TABLES command can have its own local subcommands. No matter how many TABLE subcommands are issued, the Tables procedure reads the data just once for the entire procedure.

Summary of TABLES Subcommands

Global subcommands (see Table 1) must precede the first TABLE subcommand. They apply to all tables created by a single TABLES command.

Table 1 Global subcommands

Subcommand	Explanation
AUTOLABEL	Creates labels for all tables
BASE	Determines percentage base for category variables
FTOTAL, PTOTAL	Creates stand-in variables for totals within tables
GBASE	Determines percentage base for multiple-response variables
MDGROUP, MRGROUP	Creates multiple-response variables
MISSING	Determines treatment of missing values
OBSERVATION	Declares variables to be summarized rather than counted

Local subcommands (see Table 2) except STATISTICS can be repeated once for each table created. STATISTICS can be repeated within each table. TABLE must come first. Local subcommands apply only to the preceding TABLE subcommand.

Table 2 Local subcommands

Subcommand	Explanation
CAPTION	Creates the caption cell for individual tables
CORNER	Creates labels for the corner area in a table
SORT	Sorts cells of individual tables
STATISTICS	Indicates statistics to be calculated
TABLE	Determines structure of individual tables
TITLE	Creates a title for individual tables

Usage of some subcommands changed as of SPSS 7.0 (see Table 3). Continued use of any of the following subcommands will result in a warning message or failure of the request. In all cases, the subcommand was replaced by a function, as noted.

Table 3 Subcommands that changed as of SPSS 7.0

Subcommand	Explanation
BOXCHARS	Subcommand is ignored, function replaced by metafile rendering.
CONTINUED	Subcommand is ignored, function replaced by Page Setup on the File menu of the Viewer.
FOOTNOTE	Subcommand is aliased to CAPTION.
FORMAT	With the exception of BLANK|ZERO and the MISSING keyword, the FORMAT subcommand is ignored. Function replaced by TableLook and Table Properties on the Format menu in the Pivot Table Editor.
INDEX	Subcommand is ignored, function replaced by the outline side of the Viewer.
)PAGE	When keyword is used within titles and captions it is ignored with a warning. Function replaced by Page Setup on the File menu of the Viewer.
PFOOTNOTE	Subcommand is ignored, function replaced by Page Setup on the File Menu of the Viewer.
PTITLE	Subcommand is ignored, function replaced by Page Setup on the File menu of the Viewer.
TFOOTNOTE	Subcommand is aliased to CAPTION. The CENTER, LEFT, and RIGHT options are eliminated with a warning. Function replaced by Table Properties on the Format menu in the Pivot Table Editor.
TTITLE	Subcommand is aliased to TITLE. The CENTER, LEFT, and RIGHT options are eliminated with a warning. The TITLE subcommand produces the title cell.
WRITE	Subcommand is eliminated.

Variables

Five types of variables and “stand-in” variables are available in the Tables procedure, as listed below. Category variables are the default. The rest are declared or created with global subcommands.

Category *Classification of data*. Any variable that is in the active file and not named on an OBSERVATION subcommand is treated as a category variable whose values classify the data.

Observation *Variables for summary statistics.* The OBSERVATION subcommand identifies variables in the active file whose values are used to compute summary statistics.

Multiple response	*Summary variables for multiple-response items*. The MDGROUP and MRGROUP subcommands create multiple-response variables from elementary variables in the active file. When MDGROUP is used to create the multiple-response variable, it tallies occurrences of a specified value in each of the elementary variables. When MRGROUP is used to create the multiple-response variable, it tallies occurrences of each distinct value in all the elementary variables.
Following total	*Totals that follow the items they summarize*. The FTOTAL subcommand creates a total, a variable-like syntax device that reserves a row, column, or layer for summary statistics. The statistics are for the item that the total follows on the TABLE subcommand, and they appear following that item in the table that is produced.
Preceding total	*Totals that precede the items they summarize*. Like FTOTAL, the PTOTAL subcommand creates a total. However, the associated summary statistics are for the item that the total precedes on the TABLE subcommand, and they appear preceding that item in the table that is produced.

Relations among Variables

Variables (and totals) on a TABLE subcommand can be combined in several ways to specify the design of a table. All the types of variables can enter into such combinations, but some restrictions apply (see the section on the TABLE subcommand). The relations that can combine variables are:

BY	*Dimensions*. The keyword BY separates the variables or combinations of variables that are assigned to different table dimensions. The first BY on a TABLE subcommand separates the row expression from the column expression, and the second BY separates the column expression from the layer expression.
+	*Stacking*. A variable that is stacked with another variable is added next to it along the same dimension, in effect creating a new table that is an extension of the original.
>	*Nesting*. When a variable is nested within another (the controlling variable), the resulting table displays all values of the nested variable after each value of the controlling variable. The control variable is specified before the nesting operator, and the nested variable, after.
()	*Changed order of joining with nesting*. In an expression containing both joining and nesting, the default order of operations is nesting first. Parentheses can change the order in either of two ways. The expression SEX > (STORE + REGION) specifies that the joining of *store* and *region* will be nested within *sex*. The expression (SEX + STORE) > REGION is a shorthand expression that is expanded into SEX > REGION joined with STORE > REGION.

Statistics

Four subcommands affect the statistics that appear in the tables. The first three listed below are global, and the last is local.

BASE	*Percentage base*. Defines missing-value handling for the variables that define the base used in calculating percentages.
GBASE	*Multiple-response variable percentage base*. Determines the type of count (cases or responses) for the percentage base used with multiple-response variables.
MISSING	*Missing values*. Determines the treatment of cases with user-missing values for variables named on the VARIABLES subcommand.
STATISTICS	*Cell statistics*. Specifies the functions used to compute counts, percentages, and other statistics that form the content of table cells. This subcommand also determines the display dimension, controlling whether the functions create rows, columns, or layers.

TableLook Setting

The TLOOK subcommand on the SET command instructs the TABLES command to use a specific TableLook, which is a set of properties that defines the appearance of a table. Each TableLook consists of a collection of table properties, including general appearance, footnote properties, cell properties, and borders. The default is NONE, which uses the TableLook provided as the system default. You can define a TableLook in the Pivot Table Editor.

Table Layout

The layout of a table is predetermined by the specified or default TableLook. Once the table is created, it can be customized by using the Pivot Table Editor. Three subcommands on the TABLES command affect table layout. The first two subcommands listed below are global, and the last one is local.

AUTOLABEL	*Level of automatic labeling*. Instructs the TABLES command to use the level of automatic labeling and titling that you want, which can include variable names when no labels exist, values when no value labels are found, statistical function names when no labels are provided, a default header and the page number as the page title, and the TABLE subcommand as the default table title.
FORMAT	*Printing characteristics*. Dictates the appearance of data within a table by specifying BLANK|ZERO or MISSING characters. Complete format defaults are available through the default TableLook on the Pivot Table Editor.

CORNER	*Corner label*. Lines of text in the box that is above the row titles and next to the column titles. Note that to view text in the CORNER you must set the row dimension labels as nested in Table Properties on the Format menu in the Pivot Table Editor.

Table Size

The size of a table depends on the number of row, column and layer categories specified. The output generated may be reorganized in the Pivot Table Editor prior to printing. Tables may be scaled to fit the page when printed in Table Properties on the Format menu in the Pivot Table Editor. The number of printed pages depends on the size of the table.

Length

In addition to the default TableLook, the length of a table varies according to:

- The number of row variables and the number of values for each (see the TABLE subcommand).
- The number of statistics requested for row variables (see the STATISTICS subcommand).
- The number of lines for labels, including the number of additional lines for labels that wrapped because they exceeded the column width (see “Labels” on p. 8).
- The number of lines for titles and footnotes (see the TITLE and FOOTNOTE subcommands).

The length of a table can be reduced by:

- Null variable and value labels that remove lines allotted for explicit labels.
- A null statistic label in the rows for the first function requested.
- Null labels for all the statistics in a dimension.
- Smaller fonts and margins in the TableLook.

Width

In addition to the default TableLook the width of a table varies according to:

- The number of column variables and the number of values for each (see the TABLE subcommand).
- The number of statistics requested for column variables (see the STATISTICS subcommand).

The width of a table can be reduced by:

- Specifying the row dimension for statistics so they are stacked in the cells (see the STATISTICS subcommand).
- Smaller fonts and margins in the TableLook.

Layers

The number of layers depends on:

- The number of layer variables and the number of values for each (see the TABLE subcommand).
- The number of statistics requested across the layers (see the STATISTICS subcommand).

The number of layers can be reduced by moving layer variables to the rows or columns as control variables for nestings.

Pages

The number of pages that is generated depends on:

- The number of TABLE subcommands.
- The number of layers in each table.
- Paper size and margins are specified in the Page Setup dialog box.

Labels

The Tables procedure uses labels defined by the VARIABLE LABELS and VALUE LABELS commands. The Tables procedure also lets you create variable labels for group variables specified on the MDGROUP and MRGROUP subcommands, labels for totals specified on the FTOTAL and PTOTAL subcommands, and labels for statistics whose functions are named on the STATISTICS subcommand. See the individual sections on these subcommands for the complete syntax to use in specifying the labels. The settings on the Output Labels tab of the Options dialog box determine whether the Variable Labels or Value Labels will display in the Viewer.

The syntax for defining labels on a VARIABLE LABELS command or on a VALUE LABELS command is:

```
VARIABLE LABELS varname 'label' [/varname...]
VALUE LABELS varlist value 'label' value 'label'...[/varlist...]
```

The following general rules apply to labels on the TABLES command:

- A variable label can be up to 120 characters long. (When MDGROUP uses elementary variable labels as value labels for a group variable, it retains up to 120 characters for each label.)
- A statistic label can be up to 120 characters long.
- When string values are used instead of labels, they are truncated to short strings.
- A value label can be up to 60 characters long.
- A label that is longer than the established column width is wrapped onto one or more additional lines until the entire label is printed.

- A value label assigned to a value that does not exist never appears in a table.
- Title, footnote, caption, and corner text lines are not labels. Each can be up to 10 lines of text. Corner text not fitting inside the corner will not increase the size of the corner area.

Example

```
VARIABLE LABELS VAR1
  'This is an example of a variable label that is ' +
  'very long.  It shows how the long ' +
  'label appears in the rows.'.
VALUE LABELS VAR1
   1 'This is how Tables handles a value label with 60 characters.'
   2 'Another example of a long value label in the procedure.'
   3 'This value does not exist.'.
TABLES FORMAT =
      /TABLE = VAR1+VAR2 BY VAR3.
```

		VAR3	
		1.00	2.00
This is an example of a variable label that is very long. It shows how the long label appears in the rows.	This is how Tables handles a value label with 60 characters.	2	2
	Another example of a long value label in the procedure.	2	2
VAR2	1.00	2	2
	2.00	2	2

- Variable *var1* has long variable and value labels. Since the length exceeds the width of the row title column, the labels occupy multiple lines.
- The value label for value 3 of variable *var1* does not appear in the table, since that value does not exist in the data.
- Variables *var2* and *var3* have no variable labels, so the variable names are printed.
- Variables *var2* and *var3* have no value labels, so the values are printed.

Blank and Null Labels

Blank labels and null labels are different from each other and have different effects:

- A blank label for a variable, value, or statistic (' ') is treated as a valid label, and the Tables procedure provides a blank line where a text label would normally print.
- A null variable or value label ('') removes the line allotted for the label.

- A null statistic label (’ ’) for the first function specified for the stub causes the cell contents to rise one line, thus beginning on the same line as the value label.
- A null statistic label (’ ’) for a function after the first specified for the stub produces no label but leaves a blank line.

Example

```
VARIABLE LABELS VAR1 ‘ ‘VAR3’ ‘.
VALUE LABELS VAR2 1 ‘ ‘/VAR4 1 ’ ‘.
TABLES
       /TABLE = VAR1 + VAR2 BY VAR3 + VAR4 + VAR5 + VAR6
       /STATISTICS = CPCT(VAR3) CPCT(VAR4’ ‘) CPCT(VAR5‘%’).
```

				VAR4		VAR5		VAR6	
		1.00	2.00			1.00	2.00		
		Count Percent	Count Percent		2.00	%	%	1.00	2.00
	1.00	25.0%	25.0%	25.0%	25.0%	25.0%	25.0%	2	2
	2.00	25.0%	25.0%	25.0%	25.0%	25.0%	25.0%	2	2
VAR2		25.0%	25.0%	25.0%	25.0%	25.0%	25.0%	2	2
	2.00	25.0%	25.0%	25.0%	25.0%	25.0%	25.0%	2	2

- The variable *var1* has a blank label, and *var3* has a null label. The row title for *var1* shows a blank line, while the column title for *var3* doesn’t show any space for the variable name.
- The value 1 has a null label for *var2* and a blank label for *var4*. This moves the row title for *var2* down into the row for value 1 of that variable and leaves a blank space for value 1 of *var4*.
- Statistics (percentages) are explicitly assigned to *var3*, *var4*, and *var5*. The variable *var3* has the default label, *var4* has a null label, and *var5* has a specified label.
- The variable *var6* gets the default statistic (count) with no label, because no statistics were requested.

Wrapping Labels

The following general rules for the TABLES command apply to label wrapping:

- Labels wrap to the next line by default when they are too long for the column width.
- The table formats uses blanks as the points for wrapping. It removes the blank where a label is wrapped and adjusts the continuation of the label appropriately within the column.
- A word, defined by surrounding blanks, is broken when the entire word does not fit within the column width.

Errors

A syntax error of any kind in the Tables procedure prevents the procedure from executing. Syntax errors include title, footnote, caption, and corner text lines that are too wide for the space allotted by the TableLook. When an error is detected, the procedure continues to check for other errors but does not read the data.

General Limitations

The upper limits set for various aspects of the Tables procedure are:

- 100 observation variables.
- 100 category variables.
- 100 elementary variables on all group variable subcommands.
- 20 group variables for MDGROUP and MRGROUP.
- 20 totals for FTOTAL.
- 20 totals for PTOTAL.
- 10 separate text lines for each multiple-line title in the TITLE, FOOTNOTE, and CORNER subcommands.
- 120 characters for variable labels, statistics labels, and MDGROUP value labels.
- 60 characters for value labels.
- 255 characters for each title line.

AUTOLABEL

```
/AUTOLABEL {DEFAULT}
           {ON     }
           {OFF    }
```

Overview

AUTOLABEL is a global subcommand. It instructs the Tables procedure to use one of the following levels of labeling:

DEFAULT	*Variable names, variable values, and function names.* Prints a variable name when no variable label is found, a value when no value label is found, and a statistical function name when a function is explicitly requested but no label is provided.
ON	*Title defaults and other defaults.* Supplies the automatic labels provided with the DEFAULT keyword, a default table title consisting of the TABLE subcommand, and a default page title.
OFF	*No defaults.* Turns off all default labeling and uses only the titles and the variable, value, and statistics labels that are explicitly given.

Operations

- When testing several different table designs, use AUTOLABEL=ON to provide table titles that clearly identify the alternatives.
- Use OFF when running tables that have unusual values or obvious variable names that are not required, such as *sex* when you have the explicit value labels *Male* and *Female*.

BASE

```
/BASE = {ANSWERING}
        {QUALIFIED}
        {ALL      }
```

Example:

```
TABLES MISSING = INCLUDE
       /BASE = QUALIFIED
       /TABLE = USINTL BY SEX
       /STATISTICS = CPCT (USINTL:SEX).
```

			Respondent's Sex	
			Male	Female
Take Active Part in World Affairs	NAP	Count Percent	33.5%	31.8%
	Active Part	Count Percent	52.5%	47.4%
	Stay Out	Count Percent	13.1%	18.6%
	DK	Count Percent	.9%	2.2%

Overview

BASE is a global subcommand. It determines how missing values are handled for the variables that define a percentage base.

ANSWERING *Excludes missing values.* This is the default. When a percentage base is calculated, this option excludes cases with missing values.

QUALIFIED *Includes user-missing values.* When a percentage base is calculated, this option includes cases with user-missing values.

ALL *Includes all missing values.* When a percentage base is calculated, this option includes cases with system- and user-missing values.

Operations

- The handling of missing values for the percentage base should normally correspond to the specification on the MISSING subcommand. However, use BASE=ALL when you want the percentage base to include all the cases regardless of the MISSING specification.
- If you have more than one BASE subcommand, only the last one is in effect.
- To clean the appearance of the table, the redundant labels (*Respondent's Sex* and *Count Percent*) could be deleted in the Pivot Table Editor.

BREAK

```
/BREAK BY expression
```

Overview

BREAK is a global subcommand. It allows you to specify an expression that is appended to every TABLE subcommand in the TABLES command, thereby repeating the expression without reentering it.

Operations

- The argument to BREAK BY can be any expression that yields valid syntax when appended to every TABLE subcommand in the TABLES command.
- You can break the TABLE subcommand at any symbol, except in the middle of a variable name.

Example:

In the following TABLES command, each TABLE subcommand ends with the same expression.

```
TABLES
 / TABLE = (a+b)   BY d>(e+f)
 / TABLE =       c BY d>(e+f)
 / TABLE = (a+b)>c BY d>(e+f)
 / TABLE = (a+b)>c BY d>(e+f)
 / TABLE =  a+b+c  BY d>(e+f).
```

Using BREAK, the command can be stated as:

```
TABLES/ BREAK BY d>(e+f)
 / TABLE = (a+b)
 / TABLE =       c
 / TABLE = (a+b)>c
 / TABLE = (a+b)>c
 / TABLE =  a+b+c.
```

CAPTION and TITLE

```
/{TITLE   } = [.........]
 {CAPTION } [..........]
```

Overview

TITLE and CAPTION are local subcommands. They provide lines of text for a title and a caption for the table specified on the preceding TABLE subcommand.

Operations

- If AUTOLABEL=ON, the default for TITLE is the TABLE subcommand. If AUTOLABEL=DEFAULT or OFF, no default table title is supplied.
- Table title lines print at the top of the table.
- Table caption lines print at the bottom of the table and before any footnotes.
- If multiple TITLE or CAPTION subcommands follow the same TABLE subcommand, only the last one is in effect.

Limitations

- Caption and title subcommands accept no more than 10 lines of text.

Functions

Both TITLE and CAPTION recognize the)DATE function for use in the lines of text.)DATE must be entered in upper case.

- The)DATE function prints numbers for the day, month, and year separated by spaces.

Example

```
SORT CASES BY SEX.
SPLIT FILE BY SEX.
TABLES
       /TABLE = USINTL BY RACE
       /TITLE =    ')DATE' '' '' 'Title'
```

12 Oct 95

Title

				Race of Respondent		
				White	Black	Other
Respondent's Sex	Male	Take Active Part in World Affairs	Active Part	295	29	10
			Stay Out	70	11	2
	Female	Take Active Part in World Affairs	Active Part	359	49	10
			Stay Out	124	36	4

CORNER

```
/CORNER = ['line' 'line'...]
```

Overview

CORNER is a local subcommand. It places lines of text in the box that is above the row titles and next to the column titles. You can specify multiple lines of text, but each must fit within the row title column width. There is no default text for the corner box.

Operations

- If you specify more lines than are available in the corner, the TABLES command prints as many lines as will fit. Use the Cell Properties Alignment tab on the Format menu in the Pivot Table Editor to set the alignment of the text.
- If the default TableLook uses the Corner format, which reserves the corner for dimension labels in the rows, text specified in syntax will not appear in the Output Navigator. To view text in the corner you must have the Row Dimension Labels set as Nested in Table Properties on the Format menu in the Pivot Table Editor. Note this option may be preset in the default TableLook.

Limitations

- The CORNER subcommand accepts no more than 10 lines of text.

FORMAT

```
/FORMAT = [{BLANK}]      [MISSING ({'.'     })]
           {ZERO }                 {'chars'}
```

Overview

FORMAT is a global subcommand. It controls the appearance of data within tables by specifying BLANK, ZERO or MISSING characters. Complete format settings are available through TableLook and Table Properties in the Pivot Table Editor.

Operations

These options define characters for empty cells and for missing data in cells.

MISSING('chars') *Characters for missing data.* Defines special fill characters for cells with missing data. For example, if the data for a category variable are missing in a cell, the MISSING characters become the cell contents (see the STATISTICS subcommand).

- The default for MISSING characters is a single period.
- MISSING(' ') leaves the cell contents blank.
- Up to 255 characters can be specified.
- The specified character string is right-justified in the spaces for the statistical format.
- If there are more MISSING characters than there are spaces for the statistical format, the character string is allowed to extend into any available column spaces to the left of the format. If the additional spaces are still not enough, the character string is truncated from the right.

BLANK *Blank empty cells.* Leaves empty cells blank.

ZERO *Zeros in empty cells.* Prints zeros in empty cells according to the statistical format. For example, with an F4.1 format, a zero value prints as .0 in the cell.

FTOTAL and PTOTAL

```
/FTOTAL = varname [’label’] varname ...
/PTOTAL = varname [’label’] varname ...
```

Example:

```
TABLES
      /FTOTAL = FTOT
      /PTOTAL = PTOT
      /TABLE = USINTL + FTOT BY PTOT + REGION.
```

			Region of the United States		
		PTOT	North East	South East	West
Take Active Part in World Affairs	Active Part	**752**	327	207	218
	Stay Out	**247**	112	68	67
FTOT		**999**	**439**	**275**	**285**

- FTOTAL or PTOTAL is followed by a list of variable names, each of which is optionally followed by a variable label in apostrophes or quotation marks.
- No reserved keywords can be used as the variable names.
- The variable names cannot conflict with category, observation, or multiple-response variable names on the same TABLES command. They can, however, duplicate other variable names in the working data file.
- More than one FTOTAL or PTOTAL subcommand can be specified before the first TABLE subcommand.

Overview

Both FTOTAL and PTOTAL are global subcommands. They produce summary statistics by using **totals**, syntax devices that function like temporary variables. Totals hold summary statistics for the items they either follow or precede on a TABLE subcommand. Totals that follow the items they summarize are declared on FTOTAL, and totals that precede their summarized items are declared on PTOTAL. The position of the summary statistics in the table that is produced corresponds to the type of total. The statistics for following totals are displayed in the column, row, or layer following the summarized items, and the statistics for preceding totals precede the summarized items. The function of totals in the table structure is discussed with the TABLE subcommand. The statistics available for totals are discussed with the STATISTICS subcommand.

Operations

- If no optional label is specified for a total, the variable name for the total is used as the label.
- The same variable name for a total can be used more than once on a TABLE subcommand, because the total serves as a place holder. However, using the same name more than once may result in failure of the total to pick up the correct statistic and may result in an error when statistics are explicitly applied to variables in the table.

Limitations

- A total variable name can be no more than 8 characters long, and its label can be no more than 120 characters long.
- You can declare no more than 20 following totals and 20 preceding totals.

GBASE

```
/GBASE = {CASES    }
         {RESPONSES}
```

Overview

GBASE is a global subcommand. It specifies the count used for the percentage base with multiple-response variables.

CASES	*Cases as the denominator*. Bases percentages on the number of respondents (cases). This option is the default.
RESPONSES	*Responses as the denominator*. Bases percentages on the number of responses.

The GBASE subcommand does not alter the missing-value treatment specified on the MISSING and BASE subcommands. It is used in addition to those subcommands to determine the desired count for the denominator when percentages are requested for multiple-response variables. (See "STATISTICS: Percentage Bases" on p. 44.)

Operations

- Conventionally, percentages are calculated with the same type of count for both the numerator and the denominator—either cases or responses for both.
- You can request cases for the numerator and responses for the denominator by specifying COUNT or CASES on the STATISTICS subcommand and RESPONSES on the GBASE subcommand.
- You can request responses for the numerator and cases for the denominator by specifying RESPONSES on the STATISTICS subcommand and by leaving cases (the default) for GBASE.

MDGROUP and MRGROUP

```
/MDGROUP = varname ['label'] varlist ({value  })
                                      {'chars'}
/MRGROUP = varname ['label'] varlist
```

Example:

```
TABLES
       /MDGROUP = MDVAR 'Health and Work Problems (dichotomy vars)'
         HLTH1 TO WORK9 (1)
       /MRGROUP = MRVAR 'Problems (multi-category vars)'
         PROB1 TO PROB4
       /TABLE = MRVAR + MDVAR BY SEX.
```

		Respondent's Sex	
		Male	Female
Problems (multi-category vars)	Health	48	90
	Finances	83	125
	Lack of Basic Services	4	3
	Family	23	53
	Personal	15	26
	Legal	1	1
	Miscellaneous	23	48
Health and Work Problems (dichotomy vars)	Ill Enough to Go to a Doctor	185	374
	Counselling for Mental Problems	18	40
	Infertility, Unable to Have a Baby	9	26
	Drinking Problem	9	8
	Illegal Drugs (Marijuana, Cocaine)	23	7
	Partner (Husband, Wife) In Hospital	37	36
	Child in Hospital	25	53
	Child on Drugs, Drinking Problem	6	22
	Death of a Close Friend	99	131
	Unemployed and Looking for Work a Month +	36	23
	Being Demoted or Move to Worse Positior	10	16
	Cut in Pay or Reduced Hours	36	25
	Being Passed Over for Promotion	18	20
	Having Trouble with One's Boss	21	21
	Own Business Losing Money or Failing	11	9
	Partner (Husband, Wife) Being Fired	8	17
	Partner (Husband, Wife) Cut in Pay	18	34
	One's Spouse Being Unemployed	24	31

- The variable name following the subcommand name identifies a multiple-response set. The variable list after the multiple-response set specifies elementary variables.
- Elementary variables can be numeric or short string. The type of the first elementary variable determines the type of its multiple-response set.
- Multiple-response sets function like category variables and can be nested or joined with any type of variable.

Overview

MDGROUP and MRGROUP are global subcommands. They are usually used for tabulating multiple-response questions, each of which allows more than one answer from a respondent. Each subcommand creates a multiple-response set that combines elementary variables.

MDGROUP combines elementary variables that are dichotomous. Typically, each dichotomous elementary variable records the presence of a different response. Any elementary variables with a particular coded value show the presence of a response. All other values show the absence of the response. For all of the elementary variables, the coded value is the same, and this value must be specified on the MDGROUP subcommand. The multiple-response set holds a count of the value's occurrences for each elementary variable.

The typical use of the MRGROUP subcommand is to combine elementary variables that contain a different value for each of the possible responses. Summing across all cases and all elementary variables, the multiple-response set holds counts for each occurrence of a different value in the elementary variables.

Operations

- An error results if you mix numeric and string elementary variables for one multiple-response set.
- If you do not specify a label for a multiple-response variable, its name is used instead.
- Each MDGROUP or MRGROUP subcommand creates only one multiple-response set. Multiple subcommands must be used to create several such variables when you tabulate more than one multiple-response question.

MDGROUP Subcommand

- The value to be tallied by MDGROUP must be specified. There is no default.
- The specified value should match the type of the elementary variables, numeric or string.
- An MDGROUP elementary variable can have more than two values. The variable is still effectively dichotomous, because only the specified value is used and the others are ignored.
- The names or labels of the elementary variables become the value labels for the multiple-response variable.

MRGROUP Subcommand

- MRGROUP variables have no limit on the number of distinct values allowed. A value or value label for the multiple-response variable is displayed for every distinct value found in the elementary variables.
- The value labels for the multiple-response variable are taken from the first elementary variables that have labels for the individual values.
- If a value label is not available, the Tables procedure uses the value itself as a label and displays it using the print format from the first elementary variable. If the print formats differ across variables, name the elementary variable with the desired print format first.

Limitations

- A multiple-response variable name can be no more than 8 characters long, and its label can be no more than 120 characters long.
- No more than 20 multiple-response variables can be created with any combination of MDGROUP and MRGROUP subcommands.
- All MDGROUP or MRGROUP subcommands on a table can combine up to 100 elementary variables into multiple-response variables.

MISSING

```
/MISSING = {EXCLUDE}
           {INCLUDE}
```

Overview

MISSING is a global subcommand. It determines the treatment of cases with user-missing values. System-missing values are never considered valid.

EXCLUDE	*User-missing values are invalid.* This is the default.
INCLUDE	*User-missing values are valid.*

INCLUDE on the MISSING subcommand should ordinarily correspond to QUALIFIED on the BASE subcommand, and EXCLUDE should correspond to ANSWERING. For example, it is typical that when MISSING=INCLUDE, BASE=QUALIFIED.

Operations

- User-missing values that are included in a table are treated like any other values.
- If you have more than one MISSING subcommand, the last one overrides the others preceding it.

The effect of MISSING=EXCLUDE differs according to the types of variables and statistical functions that are requested:

- If a case has a missing value for a category variable, the case is excluded from a table using that variable.
- If a case has a missing value for an elementary variable of a multiple-response variable, the elementary variable is excluded from the multiple-response variable. The case itself is not excluded, however, because it may have valid values for other elementary variables.
- If a case has a missing value for an observation variable, the case is excluded from summary statistics (such as the mean, sum, and standard deviation) for that variable. The case is also excluded from the VALIDN and VPCT functions.
- If a case has a missing value for an observation variable, the case is not excluded from COUNT or CPCT. These functions are affected only by missing values for category or group variables.

MRGROUP

See “MDGROUP and MRGROUP” on p. 22.

OBSERVATION

```
/OBSERVATION = varlist
```

Example:

```
TABLES /OBSERVATION = AGE EDUC
       /TABLE = (AGE + EDUC) > REGION BY SEX.
```

			Respondent's Sex	
			Male	Female
Age of Respondent	Region of the United States	North East	44	47
		South East	46	49
		West	43	44
Highest Year of School Completed	Region of the United States	North East	13	13
		South East	13	12
		West	13	13

- Observation variables must be numeric.
- More than one OBSERVATION subcommand can be used before the first TABLE subcommand to add observation variables to the list.

Overview

OBSERVATION is a global subcommand. It identifies variables in the working data file whose values are used to compute summary statistics such as the sum, mean, and standard deviation. The function of observation variables in the table structure is discussed with the TABLE subcommand. The statistics available for observation variables are discussed with the STATISTICS subcommand.

Operations

- Specified variable labels are used as labels for observation variables. If no variable labels are specified, variable names are used instead.
- An observation variable can be used as an elementary variable for an MDGROUP or MRGROUP subcommand.
- To clean the appearance of the table, the redundant labels (*Respondent's Sex* and *Region of the United States*) could be deleted in the Pivot Table Editor.

Limitations

- The Tables procedure allows a maximum of 100 observation variables.

PTOTAL

See “FTOTAL and PTOTAL” on p. 19.

SORT

```
/SORT [{DESCENDING}] VAR1[(FUNCTION1[OBS1])] [VAR2 ...]
       {ASCENDING }
```

Example:

```
TABLES
  /FTOTAL = TOTAL
  /OBSERVATION = AGE
  /TABLE = HAPPY BY AGE + REGION + TOTAL
  /SORT = HAPPY
  /TABLE = HAPPY BY AGE + REGION + TOTAL
  /SORT = HAPPY(MEAN(AGE)).
```

			Region of the United States			
		Age of Respondent	North East	South East	West	**TOTAL**
General Happiness	Pretty Happy	45	412	215	245	**872**
	Very Happy	47	185	149	133	**467**
	Not Too Happy	46	76	47	42	**165**

.

			Region of the United States			
		Age of Respondent	North East	South East	West	**TOTAL**
General Happiness	Very Happy	47	185	149	133	**467**
	Not Too Happy	46	76	47	42	**165**
	Pretty Happy	45	412	215	245	**872**

- In the first table, the cells of *happy* are sorted so that the row with the largest total comes first. The observation variable does not affect the sort.
- In the second table, the cells of *happy* are sorted so that the row with the largest mean comes first. The values in the columns produced by *region* do not affect the sort.

Overview

SORT is a local subcommand. It sorts the cells of a table by their contents.

DESCENDING *Sorts in descending order.* Sorts the cells of the requested variables so that the row, column, or layer with the highest total comes first. This is the default.

ASCENDING *Sorts in ascending order.* Sorts the cells of the requested variables so that the row, column, or layer with the lowest total comes first.

- At least one variable must be specified on the SORT subcommand. Additional variables are optional.
- Only category and MRGROUP variables may be SORT variables.
- A function may be specified along with the variable to identify the cells controlling the sort. If no function is mentioned, the variable's COUNT marginals are used.
- Functions explicitly specified for the sort must be defined for the table. That is, those functions must be shown somewhere in the table.
- PTILE, MODE, or MEDIAN may not be sort functions.
- A function may be accompanied by a variable in parentheses. The variable specified must use the base statistic somewhere in the table.

Operations

When you specify multiple variables on the SORT subcommand, the order in which variables are sorted is determined by their order in a table dimension. If they are nested, variables listed first will have their cells sorted before later variables. Variables in different dimensions control only the dimension in which they appear.

When the cells of a sorted variable contain different statistics, the sort function will be determined according to the following rules:

- If you explicitly specify a function (and a variable), that function controls the sort.
- If there is no explicit function, the sort will be on the marginals (totals on the variable sorted).

STATISTICS

```
/STATISTICS = [{UNWEIGHTED}]
               {U         }

    function[([varname] [(fmt)] ['label'] varname...)] function...
```

Functions for all variable types:

COUNT
CPCT

Functions for totals and observation variables:

APTILE	MEDIAN	RPTILE	VALIDN
EPTILE	MINIMUM	SEMEAN	VARIANCE
HPTILE	MODE	SPCT	VPCT
MAXIMUM	PTILE value	STDDEV	WPTILE
MEAN	RANGE	SUM	

Functions for group variables and their totals:

CASES
CSPCT
RESPONSES
RPCT

Formats:

COMMAw.d	NEQUALw.d
DOLLARw.d	PARENw.d
DOTw.d	PCTw.d
Fw.d	PCTPARENw.d
NEGPARENw.d	CCAw.d—CCEw.d

Example:

```
TABLES
    /TABLE = USINTL BY SEX
    /STATISTICS = CPCT (USINTL (PCT7) '').
```

		Respondent's Sex	
		Male	Female
Take Active Part ir World Affairs	Active Part	33%	42%
	Stay Out	8%	16%

- Variables named on the STATISTICS subcommand must also be named on the preceding TABLE subcommand and the variables must both be in the same dimension.
- All of the optional items following a function on the STATISTICS subcommand must be enclosed in a set of parentheses.
- If specified, a variable name must precede its format and label, which apply only to that variable.
- More than one variable and label can be named for a single function.
- Functions can be repeated.

Overview

STATISTICS is a local subcommand, applying only to the preceding TABLE subcommand. The STATISTICS subcommand determines the functions used to compute statistics. It also determines whether the statistics labels are above the columns, beside the rows, or above the layers. Functions available to STATISTICS include counts, percentages, and means. The STATISTICS subcommand is optional. When it is not present, default statistics are applied.

For information on bases for percentage statistics, see “STATISTICS: Percentage Bases” on p. 44. For information on the treatment of missing values in the calculation of statistics, see the MISSING subcommand.

Operations

- The position of statistics labels (above the columns, beside the rows, or above the layers) is the same as the position of the variable(s) named on the STATISTICS subcommand. All such variables must be in the same dimension.
- Functions can be applied only to category and group variables that are at the lowest level of a nesting, but they can be applied to observation variables at any level of a nesting.

- If a category variable or a group variable is in a nesting relationship with an observation variable, and statistical functions are named for both the nested variable and the control variable, the Tables procedure uses only the functions for the observation variable and ignores the others.
- For category variables, the default function is COUNT; for observation variables, MEAN; and for multiple-response variables, CASES. For totals, the default is the statistical function for the item that is summarized.
- A function uses weighted cases unless the function name is preceded by U or UNWEIGHTED.
- A function label appears on the line below a variable name or label. However, when the function label applies to an observation variable with a category variable nested within it, the function label follows the value label for the category variable. For more information on labels, see “Labels” on p. 8.

Position of Statistics

The default position of statistics labels is determined by the position of variables named on the STATISTICS subcommand. Only one dimension can be used for statistics. The dimension in which statistics are requested is called the **statistics dimension**. When there is no STATISTICS subcommand, statistics are applied to column variables.

- If a variable is in the statistics dimension but is not assigned a function on the STATISTICS subcommand, that variable receives its normal default function with a null function label.
- Any variable assigned a function receives only that function and not the default function.
- If a function is requested but a variable is not specified, that function becomes assigned to all variables in the statistics dimension to which it can be applied. The function is said to be implicitly applied to those variables.

Example

```
TABLES
       /TABLE = REGION BY SEX + RACE
       /STATISTICS = CPCT(SEX)
       /TABLE = REGION BY SEX + RACE
       /STATISTICS = CPCT COUNT(SEX).
```

		Respondent's Sex		Race of Respondent		
		Male	Female			
		Count Percent	Count Percent	White	Black	Other
Region of the United States	North East	18.5%	26.2%	582	82	15
	South East	11.7%	15.7%	307	94	14
	West	11.7%	16.2%	375	28	20

		Respondent's Sex				Race of Respondent		
		Male		Female		White	Black	Other
		Count Percent	Count	Count Percent	Count	Count Percent	Count Percent	Count Percent
Region of the United States	North East	18.5%	281	26.2%	398	38.4%	5.4%	1.0%
	South East	11.7%	177	15.7%	238	20.2%	6.2%	.9%
	West	11.7%	178	16.2%	245	24.7%	1.8%	1.3%

- The statistics dimension in the first table is determined by the variable specified on the STATISTICS subcommand. Since *sex* is specified for CPCT, the statistics dimension is the columns.
- The default statistic for categorical variables is COUNT. Since no statistics are assigned to *race*, counts are shown. Count percentages are explicitly assigned to *sex*. Therefore, count percentages are shown instead of the default.

- The statistics dimension in the second table is also determined by the variable specified on the STATISTICS subcommand. Since *sex* is specified for COUNT, the statistics dimension is the columns.
- In the second table, no variable is specified for CPCT. Count percentages, thus, are implicitly assigned to all variables in the statistics dimension. Since no statistics are explicitly assigned to *race*, only the implicitly assigned statistic (count percentage) is shown. Since counts are explicitly assigned to *sex*, both the implicitly and explicitly assigned statistics are shown for *sex*.

Default Statistics Dimension

When a STATISTICS subcommand does not name a variable, the following rules determine the statistics dimension:

- If there is an observation variable on the TABLE subcommand, and statistics are not explicitly assigned to a variable, then statistics are shown in the dimension with the observation variable.
- If there is no observation variable, the statistics are shown in the columns.

Example

```
TABLES
       /OBSERVATION = AGE
       /TABLE = USINTL BY RACE
       /STATISTICS = COUNT CPCT
       /TABLE = USINTL>AGE BY RACE
       /STATISTICS = MEAN STDDEV.
```

		Race of Respondent					
		White		Black		Other	
		Count	Count Percent	Count	Count Percent	Count	Count Percent
Take Active Part in World Affairs	Active Part	654	65.5%	78	7.8%	20	2.0%
	Stay Out	194	19.4%	47	4.7%	6	.6%

				Race of Respondent		
				White	Black	Other
Take Active Part in World Affairs	Active Part	Age of Respondent	Mean	45	42	41
			Standard Deviation	17	16	15
	Stay Out	Age of Respondent	Mean	50	44	39
			Standard Deviation	21	19	17

- Because the first STATISTICS subcommand does not name a variable and no observation variable is declared, the statistics for the first table appear in the columns.
- The second table contains the observation variable *age* in the rows. The statistics thus appear in the rows, not the columns.

Totals

The default statistics for a total are the functions specified for the cells that are summarized. Because a total's main purpose is to provide these summary statistics, the defaults should generally be used. However, if you need a different statistic for a total, you may need to name the total on the statistical function explicitly.

Example

```
TABLES
       /FTOTAL = FTOT
       /TABLE = SEX BY REGION + FTOT
       /STATISTICS = COUNT(FTOT) CPCT(FTOT)
       /TABLE = SEX BY REGION + FTOT
       /STATISTICS = COUNT(REGION) CPCT(REGION).
```

		Region of the United States			**FTOT**	
		North East	South East	West	**Count**	**Count Percent**
Respondent's Sex	Male	281	177	178	**636**	**41.9%**
	Female	398	238	245	**881**	**58.1%**

		Region of the United States						**FTOT**	
		North East		South East		West			
		Count	Count Percent	Count	Count Percent	Count	Count Percent	**Count**	**Count Percent**
Respondent's Sex	Male	281	18.5%	177	11.7%	178	11.7%	**636**	**41.9%**
	Female	398	26.2%	238	15.7%	245	16.2%	**881**	**58.1%**

- When statistics are explicitly assigned to a total, they apply only to the total. In the first table, count and count percentages are assigned to *ftot*. Only the default statistic (COUNT) is shown for *region*.
- The second table shows the same statistics explicitly assigned to *region*. Since *ftot* totals *region*, the default statistic for *ftot* is the same as the statistics explicitly assigned to *region*.

Example

A function requested without a variable becomes an implicit function for any eligible variable (or total) in the statistics dimension. In this way, the default function for a total can be superceded.

```
TABLES
       /FTOTAL = FTOT
       /TABLE = SEX BY REGION + FTOT
       /STATISTICS = COUNT CPCT(REGION)
       /TABLE = SEX BY REGION + FTOT
       /STATISTICS = COUNT(REGION) CPCT.
```

		Region of the United States						**FTOT**
		North East		South East		West		
		Count	Count Percent	Count	Count Percent	Count	Count Percent	**Count**
Respondent's Sex	Male	281	18.5%	177	11.7%	178	11.7%	**636**
	Female	398	26.2%	238	15.7%	245	16.2%	**881**

		Region of the United States						**FTOT**
		North East		South East		West		
		Count	Count Percent	Count	Count Percent	Count	Count Percent	**Count Percent**
Respondent's Sex	Male	281	18.5%	177	11.7%	178	11.7%	**41.9%**
	Female	398	26.2%	238	15.7%	245	16.2%	**58.1%**

- In the first table, COUNT is implicitly assigned to all the variables, while CPCT is explicitly assigned to *region*. Thus, counts and count percentages are shown for *region*, while only counts (the implicitly assigned statistic) are shown for the total.
- In the second table, CPCT is implicitly assigned to all the variables, while COUNT is explicitly assigned to *region*. Thus, counts and count percentages are shown for *region*, while only count percentages (the implicitly assigned statistic) are shown for the total.

Example

Like a variable, a total that is explicitly named on the STATISTICS subcommand must be in the statistics dimension. If the total is the only variable explicitly named on the STATISTICS subcommand, the dimension it is in becomes the statistics dimension.

```
TABLES
       /FTOTAL = FTOT1 FTOT2
       /TABLE = SEX + FTOT1 BY REGION + FTOT2
       /STATISTICS = COUNT CPCT(FTOT1).
```

			Region of the United States			
			North East	South East	West	**FTOT2**
Respondent's Sex	Male	Count	281	177	178	**636**
	Female	Count	398	238	245	**881**
FTOT1	**Count**		**679**	**415**	**423**	**1517**
	Count Percent		**44.8%**	**27.4%**	**27.9%**	**100.0%**

- The STATISTICS subcommand specifies CPCT for *ftot1*, thereby changing the statistics dimension to the rows.
- The variable *ftot1* now gets the CPCT and COUNT functions, and *ftot2* summarizes the rows by using the function assigned to each.
- Because the formats assigned to each row are different, the numbers may not line up in the columns. You can specify the number format on the Value tab of the Cell Properties dialog box on the Format menu in the Pivot Table Editor. To activate the Cell Properties, click on the cell you want to format.

Functions

Each statistical function can be applied only to particular types of variables. It receives a default format and label if explicit ones are not given. Functions preceded by the keyword UNWEIGHTED (or U) use unweighted cases, and the word "Unweighted" precedes their function labels.

Functions for All Variable Types

The following functions are available for all the variable types in the Tables procedure:

COUNT *Number of occurrences of values.* This is the default function for category variables. If requested for group variables or totals of group variables, COUNT produces the same results as CASES. The default label is *Count* when the function is requested explicitly and null when the function is a default. The default format is F5.

CPCT *Count percentage.* The cell count as a percentage of a specified base (see "STATISTICS: Percentage Bases" on p. 44). For group variables, default percentages are calculated for cases rather than for cell counts, because GBASE=CASES is the default. The default format is PCT5.1, and the default label is *Count (Dimension) %*.

Functions for Totals and Observation Variables

Additional functions are available for numeric observation variables and numeric totals:

MAXIMUM	*Largest value found.* The default format is the variable print format, and the default label is *Maximum*.
MEAN	*Arithmetic mean.* This is the default statistic for an observation variable. The default format is the variable print format. The default label is *Mean* when the function is requested explicitly or implicitly and null when it is the default for an observation variable.
MEDIAN	*The value below which one half of the observations fall.* If the number of observations is even, the mean of the two middle observations is displayed. The default format is the variable print format, and the default label is *Median*.
MINIMUM	*Smallest value found.* The default format is the variable print format, and the default label is *Minimum*.
MODE	*The value that is the most frequent.* If two or more values tie for the most frequent, only the smallest of them is shown. The default format is the variable print format, and the default label is *Mode*.
PTILE value	*Percentile.* The specified value for the percentage may be any number between 0 and 100. The default specification is 50. The default format is the variable print format, and the default label is *Percentile for ###.##*, where ###.## is the specified value expressed to two decimal places. Five types of percentiles are available via the keywords discussed in "Types of Percentiles" on p. 40.
RANGE	*Difference between the maximum and minimum values.* The default format is the variable print format, and the default label is *Range*.
SEMEAN	*Standard error of the mean.* The default format is the variable print format, and the default label is *Std Err of Mean*.
SPCT	*Sum percentage.* The cell sum as a percentage of a specified base (see "STATISTICS: Percentage Bases" on p. 44). The default format is PCT5.1, and the default label is *(Dimension) Sum %*.
STDDEV	*Standard deviation.* The default format is the variable print format, and the default label is *Std Deviation*.
SUM	*Sum of the values.* The default format is the variable print format, and the default label is *Sum*.
VALIDN	*Count of the nonmissing values of an observation variable.* The default format is F5, and the default label is *Valid N*.

VARIANCE *Variance of the values.* The default format is the variable print format, and the default label is *Variance*.

VPCT *Valid count percentage.* The valid cell count as a percentage of a specified base (see “STATISTICS: Percentage Bases” on p. 44). The default format is PCT5.1, and the default label is *Valid N %*.

Types of Percentiles

The following percentiles are available. You can specify as many as you wish, each with a value indicating the percentile you wish to display. In the following formulas, cases are assumed to be ranked in ascending order. The following notation is used: *w* is the sum of the weights for all nonmissing cases, *p* is the specified percentile divided by 100, *i* is the rank of each case, and X_i is the value of the *i*th case.

HPTILE value *Weighted average at* $X_{(w+1)p}$. The percentile value is the weighted average of X_i and X_{i+1} using the formula $(1-f)X_i + fX_{i+1}$, where $(w+1)p$ is decomposed into an integer part *i* and a fractional part *f*. This is the default calculation for PTILE.

WPTILE value *Weighted average at* X_{wp}. The percentile value is the weighted average of X_i and $X_{(i+1)}$ using the formula $(1-f)X_i + fX_{i+1}$, where *i* is the integer part of *wp* and *f* is the fractional part of *wp*.

RPTILE value *Observation closest to* wp. The percentile value is X_i, where *i* is the integer part of $(wp + 0.5)$.

EPTILE value *Empirical distribution function.* The percentile value is X_i when the fractional part of *wp* is equal to 0. The percentile value is X_{i+1} when the fractional part of *wp* is greater than 0.

APTILE value *Empirical distribution with averaging.* The percentile value is $(X_i + X_{i+1})/2$ when the fractional part of *wp* equals 0. The percentile value is X_{i+1} when the fractional part of *wp* is greater than 0.

Functions for Group Variables

The following functions are available for group variables and for totals applied to group variables:

CASES *Count of cases (respondents) for a group variable.* This is the default statistic for group variables. The default format is F5, and the default label is *Cases*.

CSPCT *Case percentage.* The cell count of cases as a percentage of the specified base. (See “STATISTICS: Percentage Bases” on p. 44. See also the GBASE subcommand.) The default format is PCT5.1, and the default label is *Cases (Dimension) %*.

RESPONSES *Count of responses for a group variable.* The values shown for each category of the group variable are the same as those shown for CASES. A total of responses, however, shows responses, not cases. The default format is F5, and the default label is *Responses*.

RPCT *Response percentage.* The cell count of responses as a percentage of the specified base. (See "STATISTICS: Percentage Bases" on p. 44. See also the GBASE subcommand.) The values shown for each category of the group variable are the same as those shown for CSPCT. A total of response percentages shows responses, not cases. The default format is PCT5.1, and the default label is *Responses (Dimension) %*.

Formats

In all of the formats below, *w* is the overall width and *d* is the number of digits after the decimal point. In all cases, the specification *w* is the same as the specification *w.0*. The following formats are available for use with functions on the STATISTICS subcommand:

COMMAw.d *Commas between sets of digits.* Width *w* includes a comma before every three digits, a decimal point, and *d* decimal places. For example, 8,210.50 results from a COMMA8.2 format.

DOLLARw.d *Dollar sign, commas, and decimal point.* Width *w* includes a preceding dollar sign, a comma before every three digits, a decimal point, and *d* decimal places. For example, $8,210.50 results from a DOLLAR9.2 format.

Fw.d *Standard numeric.* Width *w* includes the decimal point and *d* decimal places. For example, 8210.50 results from an F7.2 format.

NEGPARENw.d *Parentheses around negative numbers.* Width *w* includes parentheses around a number when it is negative, appropriate commas, a decimal point, and *d* decimal places. For example, (8,210.50) results from a NEGPAREN10.2 format.

NEQUALw.d N= *preceding the number.* Width *w* includes *N*= preceding the number, a decimal point, and *d* decimal places. For example, *N*=8210.50 results from an NEQUAL9.2 format.

PARENw.d *Parentheses around the number.* Width *w* includes parentheses, a decimal point, and *d* decimal places. For example, (8210.50) results from a PAREN9.2 format.

PCTw.d *Percentage.* Width *w* includes a trailing percent sign, a decimal point, and *d* decimal places. For example, 10.50% results from a PCT6.2 format.

PCTPARENw.d *Parentheses around percentages.* Width *w* includes parentheses around the number, a decimal point, *d* decimal places, and a percent sign. For example, (10.50%) results from a PCTPAREN8.2 format.

- A format that specifies only a *w* value for the width but no *d* for decimal places produces integer results (the default *d* value is 0). For example, 100% results from a PCT4 format.
- If a format width is smaller than a number, but there is a space to the left of the format within the column width, the Tables procedure uses that space to print the number anyway. This rule applies to both default and explicit formats.

- If a column width is too narrow for a number, the Tables procedure first removes dollar signs, commas, parentheses, and percent signs. If the column width is still too narrow, decimal places are dropped until only the integer portion of the number is left. Next, the Tables procedure tries exponential notation. Finally, if exponential notation doesn't fit, the Tables procedure prints asterisks in the cells.

Defaults

- Defaults may be preset in the default TableLook on the Format menu of the Pivot Table Editor.
- For some functions, the default format used for cell contents is the print format of the variable. Other functions have their own specific defaults.
- The default format for a total is ordinarily the format of the function of the item that is summarized. However, a different format can be assigned to a total by using the STATISTICS subcommand.

The following table summarizes the default formats as well as the default labels for statistical functions.

Table 1 Default formats and labels

Function	**Default format**	**Default label**
CASES	F5.0	Cases
COUNT	F5.0	Count
CPCT	PCT5.1	Count (Dimension) %
CSPCT	PCT5.1	Cases (Dimension) %
MAXIMUM	variable print format	Maximum
MEAN	variable print format	Mean
MEDIAN	variable print format	Median
MINIMUM	variable print format	Minimum
MODE	variable print format	Mode
PTILE value	variable print format	Percentile for ###.##
RANGE	variable print format	Range
RESPONSES	F5.0	Responses
RPCT	PCT5.1	Responses (Dimension)%
SEMEAN	variable print format	Std Err of Mean
SPCT	PCT5.1	Sum (Dimension) %
STDEV	variable print format	Std Deviation
SUM	variable print format	Sum
VALIDN	F5.0	Valid *N*
VARIANCE	variable print format	Variance
VPCT	PCT5.1	Valid *N* %

Example

```
/STATISTICS = COUNT (SEX (F7) STORE (F6) BUY).
```

- Two of the three variables have specific formats assigned for the COUNT function. The third variable takes the default format, F5.
- No explicit label is assigned to the COUNT function, so the default label *Count* appears under the value labels for each of the variables named.

Example

```
/STATISTICS = MEAN (OBS1 (COMMA9.2)).
```

- A COMMA format applies to the MEAN function for the variable *obs1*.
- No label is specified, so the default label *Mean* is used.

Example

```
/STATISTICS = SUM (OBS1 (DOLLAR10) '').
```

- A DOLLAR format applies to the sum of the variable *obs1*.
- A null label is specified for SUM.

STATISTICS: Percentage Bases

```
/STATISTICS = [{UNWEIGHTED}]
               {UNW       }
               {U         }
  {CPCT} [(([varname] [({Fw.d       })] ['label']...[:base varlist])]
  {RPCT}                {PCTw.d     }
  {SPCT}                {PCTPARENw.d}
  {VPCT}
```

Example:

```
TABLES
       /TABLE = USINTL BY SEX > REGION
       /STATISTICS = CPCT (REGION (PCT4) '':SEX REGION).
```

		Respondent's Sex					
		Male			Female		
		Region of the United States			Region of the United States		
		North East	South East	West	North East	South East	West
Take Active Part in World Affairs	Active Part	78%	82%	82%	72%	70%	73%
	Stay Out	22%	18%	18%	28%	30%	28%

- A variable before the colon defines the numerator for calculating percentages.
- When multiple variables are named as numerators, a specified format and/or label apply only to the variable they follow.
- Variables after the colon (:base varlist) define a **base** (the denominator) for calculating percentages. This base applies to all variables specified for the function.

Overview

Four functions for calculating percentages are available with the STATISTICS subcommand. Each uses a different statistic as the numerator, and each has a different default for the base.

CPCT *Count percentage*. The numerator is the cell count or, for group variables, a count of cases. The default base is the total number of cases in the table or, for group variables, the total number of cases or responses in the table (the GBASE subcommand specifies cases or responses). The default label is *Count (Dimension) %*. This function is valid for any variable.

CSPCT *Case percentage*. The numerator is a count of cases. The default base is the total number of cases or responses depending upon the specified GBASE subcommand (see the GBASE subcommand). The default label is *Cases (Dimension) %*. This function is valid only for group variables and their totals.

RPCT	*Response percentage.* The numerator is the cell count of responses. The default base is the total number of cases or responses in the table, depending upon the specified GBASE subcommand (see the GBASE subcommand). The default label is *Responses (Dimension) %*. This function is valid only for group variables and their totals. The values shown for each category of the multiple-response variable are the same as those shown for case percentage; only the totals show different values.
SPCT	*Sum percentage.* The numerator is the cell sum. The default base is the sum for the table. The default label is *Sum (Dimension) %*. This function is valid only for totals and observation variables.
VPCT	*Valid* N *percentage.* The numerator is the valid cell count. The default base is the total number of valid cases in the table. The default label is *Valid N %*. This function is valid only for totals and observation variables.

For general information about the use of STATISTICS, see the STATISTICS subcommand. For information about the treatment of missing values for percentage numerators, see the MISSING subcommand. For information about missing values in the base, see the BASE subcommand.

Operations

- By default, a specified percentage function is calculated for each variable in the statistics dimension (see the STATISTICS subcommand) if the function is valid for the variable.
- The default format for percentages is PCT5.1.
- A percentage function uses weighted cases unless the function name is preceded by the keyword UNWEIGHTED (or U).

Bases and Percentage Types

Percentages are classified according to the table component that defines the base. **Table percentages** use the count or sum for a whole table as the base. The percentages for all the cells add to 100%. Table percentages for multiple stacked variables add to 100% across all the cells for each of the stacked variables. For example, six tables within the same display result when two variables are joined in the rows and three are joined in the columns. The percentages for each table add to 100%. See “Example” on p. 48.

If the percentage base is the count or sum for each column within a table, the percentages are **column percentages**, and they add to 100% within each column. Similarly, if the base is the count or sum for each row within a table, the percentages are **row percentages**, and they add to 100% within each row. When a TABLE subcommand specifies a nesting, the resulting table contains subtables, one for each value of the control variable. **Subtable percentages** use the count or sum for all the cells in a subtable as the base. (Separate layers are also referred to as subtables.)

The types of percentages that are produced depend on where the variables named on base variable lists are located in the table. Base variables displayed in the rows create row percentages. Base variables displayed in the columns create column percentages. Layer variables create percentages based on each layer of the table. Base variables that are control variables create subtable percentages. A percentage without an explicit base variable is a table percentage.

- The base variable list can name only category or group variables that are also named on the TABLE subcommand. This base applies to all variables specified for the function. Base has no effect on statistics other than percentages.
- If there is no base variable list, table percentages are produced.
- If the control variable for a nesting is specified as the base, subtable percentages are produced. If both the control variable and the nested variable are specified for the base, subtable, column, row, or layer percentages are produced according to the dimension of the nesting.
- If variables are stacked, you may include one or more of the stacked variables on the base variable list. Cells for stacked variables that are not on the base variable list use the default base rather than the specified base.
- To get percentages calculated on more than one base, specify multiple percentage functions with a different base for each.

Example

This example illustrates table, column, and row percentages.

```
TABLES
       /FTOTAL = T 'Total'
       /TABLE = SEX + T BY REGION + T
       /STATISTICS = COUNT(SEX'')
                     CPCT(SEX'Table %')
                     CPCT(SEX'Column %':REGION)
                     CPCT(SEX'Row %':SEX).
```

			Region of the United States			
			North East	South East	West	**Total**
Respondent's Sex	Male		281	177	178	**636**
		Table %	18.5%	11.7%	11.7%	**41.9%**
		Column %	41.4%	42.7%	42.1%	**41.9%**
		Row %	44.2%	27.8%	28.0%	**100.0%**
	Female		398	238	245	**881**
		Table %	26.2%	15.7%	16.2%	**58.1%**
		Column %	58.6%	57.3%	57.9%	**58.1%**
		Row %	45.2%	27.0%	27.8%	**100.0%**
Total			**679**	**415**	**423**	**1517**
	Table %		**44.8%**	**27.4%**	**27.9%**	**100.0%**
	Column %		**100.0%**	**100.0%**	**100.0%**	**100.0%**
	Row %		**44.8%**	**27.4%**	**27.9%**	**100.0%**

- Specifying the variable *sex* within parentheses for the first function, COUNT, causes rows to become the statistics dimension.

- No base variable is specified for the first CPCT function, so table percentages are produced by default. The appropriate label is assigned.
- For the second CPCT function, the column variable *region* is specified for the base. Accordingly, column percentages are produced.
- For the third CPCT function, the row variable *sex* is specified, producing row percentages.

Example

This example illustrates subtable percentages with nesting in the columns.

```
TABLES
       /FTOTAL = T 'Total'
       /TABLE = USINTL + T BY SEX > (REGION + T)
       /STATISTICS = COUNT(USINTL'')
                     CPCT(USINTL'Table %')
                     CPCT(USINTL'Subtable %':SEX).
```

			Respondent's Sex							
			Male				Female			
			Region of the United States				Region of the United States			
			North East	South East	West	**Total**	North East	South East	West	**Total**
Take Active Part in World Affairs	Active Part		138	94	102	**334**	189	113	116	**418**
		Table %	13.8%	9.4%	10.2%	**33.4%**	18.9%	11.3%	11.6%	**41.8%**
		Subtable %	33.1%	22.5%	24.5%	**80.1%**	32.5%	19.4%	19.9%	**71.8%**
	Stay Out		40	20	23	**83**	72	48	44	**164**
		Table %	4.0%	2.0%	2.3%	**8.3%**	7.2%	4.8%	4.4%	**16.4%**
		Subtable %	9.6%	4.8%	5.5%	**19.9%**	12.4%	8.2%	7.6%	**28.2%**
Total			**178**	**114**	**125**	**417**	**261**	**161**	**160**	**582**
	Table %		**17.8%**	**11.4%**	**12.5%**	**41.7%**	**26.1%**	**16.1%**	**16.0%**	**58.3%**
	Subtable %		**42.7%**	**27.3%**	**30.0%**	**100%**	**44.8%**	**27.7%**	**27.5%**	**100%**

- Because no base is specified on the first CPCT, table percentages are produced by default.
- The second CPCT function specifies the control variable *sex* as the base, so subtable percentages are produced.

Example

This example illustrates table, column, and row percentages with stacking.

```
TABLES
       /FTOTAL = T 'Total'
       /TABLE = USINTL + T BY REGION + T + SEX + T
       /STATISTICS = COUNT(USINTL'')
                     CPCT(USINTL'Table %')
                     CPCT(USINTL'Column %':REGION SEX)
                     CPCT(USINTL'Row %':USINTL).
```

			Region of the United States				Respondent's Sex		
			North East	South East	West	**Total**	Male	Female	**Total**
Take Active Part in World Affairs	Active Part		327	207	218	**752**	334	418	**752**
		Table %	32.7%	20.7%	21.8%	**75.3%**	33.4%	41.8%	**75.3%**
		Column %	74.5%	75.3%	76.5%	**75.3%**	80.1%	71.8%	**75.3%**
		Row %	43.5%	27.5%	29.0%	**100.0%**	44.4%	55.6%	**100.0%**
	Stay Out		112	68	67	**247**	83	164	**247**
		Table %	11.2%	6.8%	6.7%	**24.7%**	8.3%	16.4%	**24.7%**
		Column %	25.5%	24.7%	23.5%	**24.7%**	19.9%	28.2%	**24.7%**
		Row %	45.3%	27.5%	27.1%	**100.0%**	33.6%	66.4%	**100.0%**
Total			439	275	285	**999**	417	582	**999**
	Table %		**43.9%**	**27.5%**	**28.5%**	**100.0%**	**41.7%**	**58.3%**	**100.0%**
	Column %		**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**
	Row %		**43.9%**	**27.5%**	**28.5%**	**100.0%**	**41.7%**	**58.3%**	**100.0%**

- Because the variable *region* is joined with the variable *sex*, there are two tables (*usintl* by *region* and *usintl* by *sex*) within the same display. Each table within the display adds up to 100% for table percentages.
- Because the second CPCT function specifies both *region* and *sex* on the base variable list, column percentages are produced for each table. (If only one variable were named, only columns for that variable would add up to 100%.)
- For the third CPCT function, the variable *usintl* is sufficient to produce row percentages for each table, because each row contains that variable.

Example

This example illustrates column percentages with nesting and stacking.

```
TABLES
  /FTOTAL = T 'Total'
  /TABLE = SEX > (USINTL + T + TAX + T) BY REGION + T
  /STATISTICS = CPCT(USINTL'' TAX'':REGION).
```

				Region of the United States			
				North East	South East	West	**Total**
Respondent's Sex	Male	Take Active Part in World Affairs	Active Part	31.4%	34.2%	35.8%	**33.4%**
			Stay Out	9.1%	7.3%	8.1%	**8.3%**
		Total		**40.5%**	**41.5%**	**43.9%**	**41.7%**
		R's Federal Income Tax	Too High	23.6%	27.0%	25.3%	**25.0%**
			About Right	20.0%	18.4%	17.8%	**19.0%**
			Too Low	.7%	.4%	.7%	**.6%**
		Total		**44.4%**	**45.9%**	**43.9%**	**44.6%**
	Female	Take Active Part in World Affairs	Active Part	43.1%	41.1%	40.7%	**41.8%**
			Stay Out	16.4%	17.5%	15.4%	**16.4%**
		Total		**59.5%**	**58.5%**	**56.1%**	**58.3%**
		R's Federal Income Tax	Too High	33.4%	31.1%	34.2%	**33.0%**
			About Right	22.0%	22.1%	21.2%	**21.8%**
			Too Low	.2%	.8%	.7%	**.5%**
		Total		**55.6%**	**54.1%**	**56.1%**	**55.4%**

- The variables *usintl* and *tax* are explicitly specified for the percentage, making rows the statistics dimension. (Remember that if a variable in a nesting is specified for statistics, it must be at the lowest level of nesting.)
- The variable *region* is named as the base for the percentage. Since *region* is the only variable in the columns, it produces column percentages. Notice that the totals for *usintl* add up to 100% in each column and that the totals for *tax* add up to 100% in each column.
- A null label is specified for the percentage. This forces the percentage to print in the same row as that of the category labels, which conserves space.

Percentage Types

There are four subtly different types of percentages: count, sum, response, and valid number percentages. For categorical variables, count percentages are always used. For observation variables, sum, count, or valid number percentages are used. For multiple-response variables, case and response percentages are used.

Count Percentage

CPCT gives counts as a percentage of the base.

- Category or group variables can be named as the base.
- The default format is PCT5.1, and the default label is *Count (Dimension) %*.
- If the CPCT function is applied to an observation variable, the function includes the missing values for that variable.
- CPCT applied to a multiple-response variable or to a total of a multiple-response variable uses the count of cases, not responses, as the numerator. This makes a difference only for the total of the multiple-response variable.
- With GBASE=CASES (the default), CPCT uses total cases as the denominator. With GBASE=RESPONSES, CPCT uses total responses as the denominator.

Case Percentage

CSPCT gives cases as a percentage of the base.

- CSPCT can be requested only for multiple-response variables or for totals of multiple-response variables.
- The default format is PCT5.1, and the default label is *Cases (Dimension) %*.
- CSPCT uses the count of cases, not responses, as the numerator. This makes a difference only for the total of the multiple-response variable.
- With GBASE=CASES (the default), CSPCT uses total cases as the denominator. With GBASE=RESPONSES, CSPCT uses total responses as the denominator.

Sum Percentage

SPCT gives the sum as a percentage of the base.

- SPCT can apply to an observation variable or to a total for a categorical or observation variable.
- The default format is PCT5.1, and the default label is *Sum (Dimension) %*.

Example

```
TABLES
       /OBSERVATION = CHILDS
       /FTOTAL = T 'Total'
       /TABLE = RACE > CHILDS + T BY REGION + T
       /STATISTICS = SUM('')
              SPCT('Table %')
              SPCT('Column %':REGION)
              SPCT('Row %':RACE).
```

				Region of the United States			
				North East	South East	West	**Total**
Race of Respondent	White	Number of Children		1086	558	660	**2304**
			Table %	37.9%	19.4%	23.0%	**80.3%**
			Column %	83.9%	70.1%	84.7%	**80.3%**
			Row %	47.1%	24.2%	28.6%	**100.0%**
	Black	Number of Children		185	209	63	**457**
			Table %	6.4%	7.3%	2.2%	**15.9%**
			Column %	14.3%	26.3%	8.1%	**15.9%**
			Row %	40.5%	45.7%	13.8%	**100.0%**
	Other	Number of Children		23	29	56	**108**
			Table %	.8%	1.0%	2.0%	**3.8%**
			Column %	1.8%	3.6%	7.2%	**3.8%**
			Row %	21.3%	26.9%	51.9%	**100.0%**
Total				**1294**	**796**	**779**	**2869**
	Table %			**45.1%**	**27.7%**	**27.2%**	**100.0%**
	Column %			**100.0%**	**100.0%**	**100.0%**	**100.0%**
	Row %			**45.1%**	**27.7%**	**27.2%**	**100.0%**

- SUM requests the cell sum.
- By default, SPCT shows table percentages.
- *Region* is in columns, so if *region* is named as the base, it produces column percentages.
- *Race* is in rows, so if *race* is named as the base, it produces row percentages. Note that because *childs* is an observation variable, it cannot be named as the base. Normally, a controlling variable (like *race*) used as a base produces subtable percentages. Because the variable nested under it is an observation variable and each subtable is a separate row, the base (*race*) produces row percentages (which, in this case, are the same as subtable percentages).

Response Percentage

RPCT gives responses as a percentage of the base. RPCT can be requested only for multiple-response variables or for totals of multiple-response variables.

- The default format is PCT5.1, and the default label is *Responses (Dimension) %*.
- With GBASE=CASES (the default), RPCT uses total cases as the denominator. With GBASE=RESPONSES, RPCT uses total responses as the denominator.

Example

The following TABLES command computes the percentage of cases using CPCT for a multiple-response variable.

```
TABLES
       /MRGROUP = PROB_C 'Most Significant Problems in the’+
        ‘Last 12 Months' PROB1 TO PROB4
       /FTOTAL = T ‘Total’
       /TABLE = PROB_C + T BY SEX + T
       /STATISTICS = CASES(PROB_C'')
          CPCT(PROB_C'Table %')
          CPCT(PROB_C'Column %':SEX)
          CPCT(PROB_C'Row %':PROB_C).
```

			Respondent's Sex		
			Male	Female	**Total**
Most Significant Problems in the Last 12 Months	Health		48	90	**138**
		Table %	14.3%	26.8%	**41.1%**
		Column %	36.4%	44.1%	**41.1%**
		Row %	34.8%	65.2%	**100.0%**
	Finances		83	125	**208**
		Table %	24.7%	37.2%	**61.9%**
		Column %	62.9%	61.3%	**61.9%**
		Row %	39.9%	60.1%	**100.0%**
	Lack of Basic Services		4	3	**7**
		Table %	1.2%	.9%	**2.1%**
		Column %	3.0%	1.5%	**2.1%**
		Row %	57.1%	42.9%	**100.0%**
	Family		23	53	**76**
		Table %	6.8%	15.8%	**22.6%**
		Column %	17.4%	26.0%	**22.6%**
		Row %	30.3%	69.7%	**100.0%**
	Personal		15	26	**41**
		Table %	4.5%	7.7%	**12.2%**
		Column %	11.4%	12.7%	**12.2%**
		Row %	36.6%	63.4%	**100.0%**
	Legal		1	1	**2**
		Table %	.3%	.3%	**.6%**
		Column %	.8%	.5%	**.6%**
		Row %	50.0%	50.0%	**100.0%**
	Miscellaneous		23	48	**71**
		Table %	6.8%	14.3%	**21.1%**
		Column %	17.4%	23.5%	**21.1%**
		Row %	32.4%	67.6%	**100.0%**
Total			**132**	**204**	**336**
	Table %		**39.3%**	**60.7%**	**100.0%**
	Column %		**100.0%**	**100.0%**	**100.0%**
	Row %		**39.3%**	**60.7%**	**100.0%**

- The total counts for the table and the columns are less than the sums of the respective statistics in the cells, because *prob_c* is a multiple-response variable and CASES has been specified as the counting function.
- Because GBASE=CASES is the default, the total cases are used for the percentage bases. Note that total column percentages and table percentages are each 100%.
- Row variables add up as you would expect them to, because *sex* is a normal category variable.

Example

The following TABLES command computes the percentage of responses using RPCT for the same multiple-response variable.

```
TABLES
       /MRGROUP = PROB_C 'Most Significant Problems in the’+
        ‘Last 12 Months' PROB1 TO PROB4
       /GBASE = RESPONSES
       /FTOTAL = T ‘Total’
       /TABLE = PROB_C + T BY SEX + T
       /STATISTICS = RESPONSES(PROB_C'')
                     RPCT(PROB_C'Table %')
                     RPCT(PROB_C'Column %':SEX)
                     RPCT(PROB_C'Row %':PROB_C).
```

			Respondent's Sex		
			Male	Female	**Total**
Most Significant Problems in the Last 12 Months	Health		48	90	**138**
		Table %	8.8%	16.6%	**25.4%**
		Column %	24.4%	26.0%	**25.4%**
		Row %	34.8%	65.2%	**100.0%**
	Finances		83	125	**208**
		Table %	15.3%	23.0%	**38.3%**
		Column %	42.1%	36.1%	**38.3%**
		Row %	39.9%	60.1%	**100.0%**
	Lack of Basic Services		4	3	**7**
		Table %	.7%	.6%	**1.3%**
		Column %	2.0%	.9%	**1.3%**
		Row %	57.1%	42.9%	**100.0%**
	Family		23	53	**76**
		Table %	4.2%	9.8%	**14.0%**
		Column %	11.7%	15.3%	**14.0%**
		Row %	30.3%	69.7%	**100.0%**
	Personal		15	26	**41**
		Table %	2.8%	4.8%	**7.6%**
		Column %	7.6%	7.5%	**7.6%**
		Row %	36.6%	63.4%	**100.0%**
	Legal		1	1	**2**
		Table %	.2%	.2%	**.4%**
		Column %	.5%	.3%	**.4%**
		Row %	50.0%	50.0%	**100.0%**
	Miscellaneous		23	48	**71**
		Table %	4.2%	8.8%	**13.1%**
		Column %	11.7%	13.9%	**13.1%**
		Row %	32.4%	67.6%	**100.0%**
Total			**197**	**346**	**543**
	Table %		**36.3%**	**63.7%**	**100.0%**
	Column %		**100.0%**	**100.0%**	**100.0%**
	Row %		**36.3%**	**63.7%**	**100.0%**

- The GBASE subcommand requests responses, not counts, for the percentage base.
- The RESPONSES function on the STATISTICS subcommand shows the number of responses in each cell.
- RPCT shows the percentage of responses in each cell.
- Now the totals for table percentages and column percentages are equal to the sums of the respective statistics in the cells. Since GBASE=RESPONSES was specified, total column and table percentages are 100%.

Valid N Percentage

VPCT gives the valid number of cases as a percentage of the base.

- VPCT is valid only for observation variables.
- VPCT gives results identical to those of CPCT, except that missing cases are not included in the count.
- The default format is PCT5.1, and the default label is *Valid N %*.

TABLE

```
/TABLE = rows [BY columns [BY layers ]]
```

Example:

```
TABLES
      /TABLE = USINTL BY REGION BY SEX.
```

Respondent's Sex Male

		Region of the United States		
		North East	South East	West
Take Active Part in World Affairs	Active Part	138	94	102
	Stay Out	40	20	23

Respondent's Sex Female

		Region of the United States		
		North East	South East	West
Take Active Part in World Affairs	Active Part	189	113	116
	Stay Out	72	48	44

- The TABLE subcommand is the first local subcommand after all the global subcommands. Any number of TABLE subcommands can be used. For each one, any other local subcommands that apply to it must come immediately after the TABLE subcommand.
- The row expression on the TABLE subcommand must precede the optional column expression, and both must precede the optional layer expression. The keyword BY must separate one expression from another.
- Expressions are constructed from working data file variables, group variables, totals, and the keywords (LABELS) and (STATISTICS). When more than one of these elements are specified for one dimension, they are separated by + and/or > operators. The + operator stacks variables one above the other. The > operator nests the variable following it beneath the variable preceding it. Parentheses control the order of the operations.
- Before the first TABLE subcommand, the subcommands OBSERVATION, MDGROUP or MRGROUP, and FTOTAL or PTOTAL must be used to declare any observation variables, multiple-response variables, and totals.

Overview

TABLE is a local subcommand, the only one required in the procedure. The TABLE subcommand defines a table's structure. A table can have one, two, or three dimensions, each defined by a single variable or by an expression that combines multiple variables in the same dimension.

Two keywords can be specified on TABLE:

(LABELS)	*Prints a dimension with value labels only and, optionally, totals.*
(STATISTICS)	*Moves the statistics to another dimension.*

These keywords are discussed below. Note that the parentheses are part of the specification.

Operations

- One row, column, or layer is created for each statistic requested.
- Empty rows and columns are retained only when a value exists for a row or column variable in one layer but not in another. This may also occur with stacked variables when a variable is present in one stacked variable in a row or column but not the other.
- Only the row dimension can be requested without other dimensions.
- By default, variables from the working data file are treated as category variables.

Multivariable Dimensions

Stacking and nesting combine items within a single dimension on a TABLE subcommand. In this context, the term “item” refers to a variable or a combination of variables. Each stacking or nesting combines two items at a time, although more than one operator can appear within the same dimension.

Stacking

The + sign stacks an item in the same dimension as the item before the + sign. This produces a display in which the combined items simply appear next to each other along one dimension. Otherwise, the two items have no direct connection or effect on each other. For example, when one item is stacked with another in the column dimension, the typical result is a single display with two tables that have common row titles but different column titles. The two tables are stacked along the horizontal dimension, as if completely separate tables had simply been pasted together. If table percentages are computed, the base for the percentages is defined by the counts or sums for the individual tables rather than by the count or sum for the whole display (see “STATISTICS: Percentage Bases” on p. 44).

Example

```
TABLES
      /TABLE = USINTL + TAX BY SEX + RACE.
```

		Respondent's Sex		Race of Respondent		
		Male	Female	White	Black	Other
Take Active Part in World Affairs	Active Part	334	418	654	78	20
	Stay Out	83	164	194	47	6
R's Federal Income Tax	Too High	233	308	445	71	25
	About Right	177	203	332	37	11
	Too Low	6	5	11		

- Stacking occurs in both dimensions. *Usintl* and *tax* are stacked in the rows, and *sex* and *race* are stacked in the columns.
- Effectively, four different tables are displayed together: *usintl* by *sex*, *tax* by *sex*, *usintl* by *race*, and *tax* by *race*.

Nesting

The > sign nests an item beneath the preceding item. An item preceding the > sign is a **control item**, and an item following the operator is a **nested item**. All the values of the nested item are tabulated within each value of the control item. The effect is the same as a crosstabulation, although nesting "crosstabulates" in a single dimension. Individual values of the control item define different **subtables**.

- Multiple levels of nesting are permitted.
- An observation variable cannot nest within another observation variable. The expressions OBS1 > OBS2 and (OBS1 + VAR2) > OBS2 are both illegal (see "Observation Variables" on p. 63).
- A total cannot nest within another total, and a total cannot be applied to a nesting that includes a total. The expressions (VAR1 + FTOT) > (VAR2 + FTOT) and PTOT + VAR1 > (VAR2 + FTOT) are both illegal (see "Totals" on p. 68).

Example

```
/TABLE = USINTL > TAX BY SEX > RACE.
```

				Respondent's Sex					
				Male			Female		
				Race of Respondent			Race of Respondent		
				White	Black	Other	White	Black	Other
Take Active Part in World Affairs	Active Part	R's Federal Income Tax	Too High	88	5	4	86	13	6
			About Right	60	9		71	4	1
			Too Low	3			2		
	Stay Out	R's Federal Income Tax	Too High	22	2	1	32	12	1
			About Right	12	2	1	22	3	1
			Too Low				1		

- Nesting occurs in both dimensions.
- Each nesting is crosstabulated with the other nesting, creating four subtables within one table.
- The TABLE subcommand in this example has nesting operators in place of the stacking operators in the previous example. A key difference between the resulting tables is that the variables in each dimension of the nested table are in effect crosstabulated, whereas the variables in each dimension of the stacked table remain separate.

Order of Operations

Normally, nesting is performed before stacking.

Example

```
/TABLE = SEX > USINTL + TAX BY REGION.
```

				Region of the United States		
				North East	South East	West
Respondent's Sex	Male	Take Active Part in World Affairs	Active Part	138	94	102
			Stay Out	40	20	23
	Female	Take Active Part in World Affairs	Active Part	189	113	116
			Stay Out	72	48	44
R's Federal Income Tax	Too High			239	142	160
	About Right			176	99	105
	Too Low			4	3	4

- In the row expression, *usintl* is nested within *sex*, and *tax* is stacked with the nested variables. *Tax* is not nested within *sex*.

Changed Order of Operations

You can change the normal order of operations by using parentheses. An expression within parentheses is evaluated first.

Example

```
/TABLE = SEX > (USINTL + TAX) BY REGION.
```

				Region of the United States		
				North East	South East	West
Respondent's Sex	Male	Take Active Part in World Affairs	Active Part	138	94	102
			Stay Out	40	20	23
		R's Federal Income Tax	Too High	99	66	68
			About Right	84	45	48
			Too Low	3	1	2
	Female	Take Active Part in World Affairs	Active Part	189	113	116
			Stay Out	72	48	44
		R's Federal Income Tax	Too High	140	76	92
			About Right	92	54	57
			Too Low	1	2	2

- In the row expression, *usintl* is stacked with *tax*, and both are nested under *sex*.

Example

```
/TABLE = SEX + USINTL > TAX BY REGION.
```

				Region of the United States		
				North East	South East	West
Respondent's Sex	Male			281	177	178
	Female			398	238	245
Take Active Part in World Affairs	Active Part	R's Federal Income Tax	Too High	91	56	55
			About Right	61	39	45
			Too Low	2	1	2
	Stay Out	R's Federal Income Tax	Too High	30	20	20
			About Right	17	7	17
			Too Low		1	

- Nesting occurs before stacking. Thus, *tax* is nested under *usintl*, which is stacked with *sex*. *Tax* is not nested under *sex*.

Example

```
/TABLE = (SEX + USINTL) > TAX BY REGION.
```

				Region of the United States		
				North East	South East	West
Respondent's Sex	Male	R's Federal Income Tax	Too High	99	66	68
			About Right	84	45	48
			Too Low	3	1	2
	Female	R's Federal Income Tax	Too High	140	76	92
			About Right	92	54	57
			Too Low	1	2	2
Take Active Part in World Affairs	Active Part	R's Federal Income Tax	Too High	91	56	55
			About Right	61	39	45
			Too Low	2	1	2
	Stay Out	R's Federal Income Tax	Too High	30	20	20
			About Right	17	7	17
			Too Low		1	

- Expressions within parentheses are evaluated first. Thus, *tax* is nested under *sex* and *usintl*, which are stacked below each other.

Category Variables

Variables named on the TABLE subcommand that come from the working data file are assumed to be category variables, unless specifically declared otherwise. The Tables procedure creates a row, column, or layer for each value of a category variable (or for each combination of the values in a nesting).

- Multiple-response variables created on the MDGROUP or MRGROUP subcommand follow the same rules as category variables.
- Category variables can have alphanumeric values, but the values are truncated to the limit set for short strings.

Limitations

- The Tables procedure allows a maximum of 100 category variables.
- There is no limit on the number of unique values in a category variable.
- To limit the range for a category variable, select cases with the transformation language before using the Tables procedure.

Observation Variables

Observation variables must be declared on the OBSERVATION subcommand before being named on the TABLE subcommand.

- All observation variables come from the working data file and must be numeric.
- All observation variables in a table must be specified for the same dimension.
- Separate summary statistics requested for an observation variable in the rows, columns, or layers dimension produce separate rows, columns, or layers, respectively.

Example

```
TABLES
       /OBSERVATION = AGE EDUC
       /TABLE = SEX BY AGE + EDUC
       /STATISTICS = MEDIAN MODE.
```

		Age of Respondent		Highest Year of School Completed	
		Median	Mode	Median	Mode
Respondent's Sex	Male	41	35	13	12
	Female	42	35	12	12

- The observation variables are in the columns dimension.
- Two summary statistics (MEDIAN and MODE) are requested, producing two columns for each observation variable.

Nesting Observation Variables

Observation variables may be nested within a category variable, or a category variable may be nested within an observation variable.

- An observation variable cannot nest within another observation variable. The following two expressions are both illegal:

  ```
  OBS1 > OBS2
  (OBS1 + VAR2) > OBS2
  ```

- If an observation variable is nested within a category or group variable, the order of nesting does not affect the numbers produced or the structure of the table; it affects only the labeling. Regardless of how the nesting was specified, the Tables procedure creates a row, column, or layer for each observation statistic.
- If a category or group variable is specified as nested within an observation variable, a label for the observation variable precedes the category or group variable in the display.
- If an observation variable is specified as nested within a category or group variable, a label for the observation variable follows each value label for the category or group variable.

Example

```
TABLES
       /OBSERVATION = AGE
       /TABLE = REGION BY AGE > SEX
       /TABLE = REGION BY SEX > AGE.
```

		Age of Respondent	
		Respondent's Sex	
		Male	Female
Region of the United States	North East	44	47
	South East	46	49
	West	43	44

		Respondent's Sex	
		Male	Female
		Age of Respondent	Age of Respondent
Region of the United States	North East	44	47
	South East	46	49
	West	43	44

- The observation variable is in the columns.
- On the first TABLE subcommand, *sex* is nested within *age*.
- On the second TABLE subcommand, *age* is nested within *sex*.
- Both tables show the same statistics, but the column titles are organized differently.

Creating a Labels Dimension

Use the keyword (LABELS) on the TABLE subcommand to create a table with the value labels in one dimension and variables in another dimension. The keyword is placed on a TABLE subcommand as if it were a variable. Note that the parentheses are part of the keyword. (LABELS) has the following capabilities and restrictions:

- Only category variables can be used with the (LABELS) keyword. Multiple-response variables, totals, and the (STATISTICS) keyword are illegal in the dimension from which (LABELS) takes the variable labels. Observation variables may be used only in this dimension if they are nested within a category variable. Observation variables in this dimension may not be stacked with category variables (but may be stacked with other observation variables).
- If multiple variables are used, they should all have the same value labels.

- (LABELS) can be used only once on a TABLE subcommand.
- (LABELS) cannot be used in the layer dimension.
- (LABELS) can be stacked only with PTOTAL and FTOTAL variables.
- (LABELS) may be nested within category or observation variables. The (STATISTICS) keyword may be nested within the (LABELS) keyword.
- Explicit statistics can apply only to variables, not (LABELS).
- When the STATISTICS subcommand is used and multiple statistics are requested, the statistics labels are placed in the same dimension as each variable label, not each value label. If you want to nest statistics within the value labels, nest the (STATISTICS) keyword within the (LABELS) keyword.
- The default statistic with the (LABELS) option is COUNT.

Example

```
TABLES
    /TABLE=OBEY+POPULAR+THNKSELF BY (LABELS).
```

	Most Important	2nd Important	3rd Important	4th Important	Least Important
To Obey	195	123	142	343	179
To Be Well Liked or Popular	4	27	57	185	709
To Think for Oneself	510	161	130	135	46

- The row variables all have the same value labels.
- Each value is shown as a separate column. Each variable is a separate row.

Example

```
TABLES
        /TABLE=OBEY+POPULAR+THNKSELF BY SEX > (LABELS).
```

	Respondent's Sex									
	Male					Female				
	Most Important	2nd Important	3rd Important	4th Important	Least Important	Most Important	2nd Important	3rd Important	4th Important	Least Important
To Obey	87	53	62	123	83	108	70	80	220	96
To Be Well Liked or Popular	2	15	33	83	275	2	12	24	102	434
To Think for Oneself	193	79	52	61	23	317	82	78	74	23

- Again, the row variables all have the same value labels.
- Each value of the row variables is a column nested under *sex.*
- The FORMAT subcommand makes the columns narrower, so the table fits on the page.

Creating a Statistics Dimension

Use the (STATISTICS) keyword with the TABLE subcommand to create a table with statistics in a separate dimension. Like (LABELS), (STATISTICS) is placed on a TABLE subcommand as if it were a variable. Again, the parentheses are part of the keyword. The (STATISTICS) keyword has capabilities and restrictions similar to those of (LABELS).

- (STATISTICS) can be used in any dimension.
- (STATISTICS) can be stacked only with PTOTAL and FTOTAL variables. If (STATISTICS) is nested beneath a variable, that group may be stacked with another group composed of a variable with (STATISTICS) nested beneath.
- (STATISTICS) can be nested beneath observation and category variables and the (LABELS) keyword. (STATISTICS) cannot be nested beneath a total or multiple-response set.
- (STATISTICS) must be at the lowest level of nesting.
- All variables in the statistics dimension should be assigned the same statistics. If they are not, the (STATISTICS) keyword cannot be applied correctly. In this case, the statistics will be taken from the first (nontotal) variable and a warning will be issued.
- Unlike (LABELS), (STATISTICS) may be used more than once. However, all instances of the (STATISTICS) keyword must be in the same dimension. If (STATISTICS) occurs more than once, each instance must be nested beneath variables that are concatenated.

Example

```
TABLES
    /TABLE=OBEY+POPULAR+THNKSELF BY (STATISTICS)
    /STATISTICS=COUNT CPCT.
```

		Count	Count Percent
To Obey	Most Important	195	19.9%
	2nd Important	123	12.5%
	3rd Important	142	14.5%
	4th Important	343	34.9%
	Least Important	179	18.2%
To Be Well Liked or Popular	Most Important	4	.4%
	2nd Important	27	2.7%
	3rd Important	57	5.8%
	4th Important	185	18.8%
	Least Important	709	72.2%
To Think for Oneself	Most Important	510	51.9%
	2nd Important	161	16.4%
	3rd Important	130	13.2%
	4th Important	135	13.7%
	Least Important	46	4.7%

- The row variables are all assigned the same statistics.
- Each statistic is shown as a separate column. Each variable label is a separate row.

Example

```
TABLES
    /TABLE=OBEY+POPULAR+THNKSELF BY SEX > (STATISTICS)
    /STATISTICS=COUNT CPCT.
```

		Respondent's Sex			
		Male		Female	
		Count	Count Percent	Count	Count Percent
To Obey	Most Important	87	8.9%	108	11.0%
	2nd Important	53	5.4%	70	7.1%
	3rd Important	62	6.3%	80	8.1%
	4th Important	123	12.5%	220	22.4%
	Least Important	83	8.5%	96	9.8%
To Be Well Liked or Popular	Most Important	2	.2%	2	.2%
	2nd Important	15	1.5%	12	1.2%
	3rd Important	33	3.4%	24	2.4%
	4th Important	83	8.5%	102	10.4%
	Least Important	275	28.0%	434	44.2%
To Think for Oneself	Most Important	193	19.7%	317	32.3%
	2nd Important	79	8.0%	82	8.4%
	3rd Important	52	5.3%	78	7.9%
	4th Important	61	6.2%	74	7.5%
	Least Important	23	2.3%	23	2.3%

- Again, the row variables all have the same statistics.
- Each statistic for the row variables is a column nested under *sex*.

Example

```
TABLES
 /FTOTAL=TOTAL
 /TABLE=THNKSELF BY RACE>(STATISTICS)+SEX>(STATISTICS)+TOTAL
 /STATISTICS=CPCT.
```

		Race of Respondent			Respondent's Sex		**TOTAL**
		White	Black	Other	Male	Female	
		Count Percent	Count Percent	Count Percent	Count Percent	Count Percent	**Count Percent**
To Think for Oneself	Most Important	45.8%	5.3%	.8%	19.7%	32.3%	**51.9%**
	2nd Important	13.3%	2.4%	.6%	8.0%	8.4%	**16.4%**
	3rd Important	10.3%	2.3%	.6%	5.3%	7.9%	**13.2%**
	4th Important	10.3%	2.7%	.7%	6.2%	7.5%	**13.7%**
	Least Important	3.5%	.9%	.3%	2.3%	2.3%	**4.7%**

- This is an alternative to the syntax:

  ```
  TABLES
   /FTOTAL=TOTAL
   /TABLE=THNKSELF BY RACE+SEX+TOTAL
   /STATISTICS=CPCT(RACE) CPCT(SEX).
  ```

  Both produce the same table.

Totals

Totals must be defined on a TOTALS subcommand before they can be used on the TABLE subcommand.

- A total reserves a row, column, or layer for summary statistics.
- A total is stacked with the item it summarizes. A following total (defined on FTOTAL) must follow a stacking operator on the TABLE subcommand. A preceding total (defined on PTOTAL) must precede a stacking operator. A TABLE specification, such as the following example, clarifies the distinction between FTOTAL and PTOTAL:

  ```
  /TABLE = VAR1 + TOTAL + VAR2
  ```

  Here, if the total is a following total, it totals *var1*. If the total is a preceding total, it totals *var2*.
- In a table display, the row, column, or layer reserved by a following total follows the item that the total summarizes. Similarly, a preceding total precedes its summarized item.
- Totals specified for the rows produce rows that contain column summary statistics. Totals for the columns produce columns containing row summaries. Totals for the layer dimension produce layers containing layer summaries.

- An item that is summarized cannot include another total. The following expression is illegal because it calls for an FTOTAL to summarize a PTOTAL:

  ```
  VAR1 > (PTOT + VAR3) + FTOT
  ```

Example

```
TABLES
       /OBSERVATION = age
       /FTOTAL = FTOT 'Following total'
       /PTOTAL = PTOT 'Preceding total'
       /TABLE = REGION > USINTL + FTOT BY PTOT + AGE > SEX
       /STATISTICS = RANGE.
```

				Preceding total	Age of Respondent	
					Respondent's Sex	
					Male	Female
				Range	Range	Range
Region of the United States	North East	Take Active Part in World Affairs	Active Part	68	62	68
			Stay Out	70	67	70
	South East	Take Active Part in World Affairs	Active Part	65	61	65
			Stay Out	61	55	61
	West	Take Active Part in World Affairs	Active Part	69	62	69
			Stay Out	67	67	62
Following total				**71**	**69**	**70**

- Each of the two totals summarizes a nesting.
- Where the total row and the total column cross, the statistic summarizes the whole table.

Totals and Adjacent Symbols

On a TABLE subcommand, only certain symbols are valid at the immediate left or right of a following or preceding total. The eligible symbols for each total are:

```
         ...+  ftotal { BY }
                      { )  }
                      { +  }
                      { /  }

        { =  } ptotal  +...
        { BY }
        { (  }
        { +  }
```

For *following* totals, the valid symbols are used in these ways:

+ When the stack operator is to the *left* of a following total, the operator links the total to the item it summarizes. The total shown below summarizes everything before it:

```
VAR3 > (VAR1 + VAR2) + FTOT
```

When the stack operator is to the *right* of a following total, the operator places the total before the next item. The total does not summarize the next item or have any other effect on it. Here, the total summarizes *var1*.

```
VAR1 + FTOT + (VAR3 + VAR4) > VAR2
```

BY This keyword appears to the right of a following total to separate the total from the next dimension (the columns or layers dimension).

) The right parenthesis appears to the right of a following total when the total is stacked after a variable that is nested within another item. Here, the total summarizes *var3*>*var2* and then *var1*>*var2*.

```
(VAR3 + VAR1) > (VAR2 + FTOT)
```

/ The terminator for a TABLE subcommand appears to the right of a following total if the total is the last item named on the subcommand.

For *preceding* totals, the valid adjacent symbols are used in these ways:

+ When the stack operator is to the *right* of a preceding total, the operator links the total to the item it summarizes. For example, the total shown below summarizes everything after it:

```
PTOT + VAR3 > (VAR1 + VAR2)
```

When the stack operator is to the *left* of a preceding total, the operator joins the total after the prior item. The total does not summarize the prior item or have any other effect on it.

```
VAR3 > VAR1 + PTOT + VAR2
```

= An equals sign to the left of a preceding total means that the total is the first item named on the TABLE subcommand. For example, the total summarizes *var1*>*var2*.

```
/TABLE = PTOT + VAR1 > VAR2 BY VAR3
```

BY This keyword appears to the left of a preceding total to separate the total from the prior dimension (the rows or columns dimension).

(The left parenthesis appears to the left of a preceding total when the total is stacked before an item that is nested within another item. For example, the total summarizes *var3*>*var2* and *var1*>*var2*.

```
(VAR3 + VAR1) > (PTOT + VAR2)
```

A nesting operator cannot come before or after a total because a total cannot be included directly in a nesting. However, a total can be nested indirectly within another item if the total is part of a stacked item. For example, these expressions are valid:

```
VAR1 > (VAR2 + FTOT)
VAR1 > (PTOT + VAR2)
```

Subject Index

Syntax Index