CHAPTER

32

The REPORT Procedure

Overview

The REPORT procedure combines features of the PRINT, MEANS, and TABULATE procedures with features of the DATA step in a single report-writing tool that can produce a variety of reports. You can use PROC REPORT in three ways:

- in a windowing environment with a prompting facility that guides you as you build a report.
- in a windowing environment without the prompting facility.
- in a nonwindowing environment. In this case, you submit a series of statements with the PROC REPORT statement, just as you do in other SAS procedures.

  You can submit these statements from the PROGRAM EDITOR window with the NOWINDOWS option in the PROC REPORT statement, or you can run SAS in batch, noninteractive, or interactive line mode (see the information on running the SAS System in *SAS Language Reference: Concepts*).

This section provides reference information about using PROC REPORT in a windowing or nonwindowing environment. Similar information is also available online through the Help facility. For task-oriented documentation for the nonwindowing environment, see SAS Technical Report P-258, *Using the REPORT Procedure in a Nonwindowing Environment, Release 6.07*.

Types of Reports

A *detail report* contains one row for every observation selected for the report.Each of these rows is a *detail row*. A *summary report* consolidates data so that each row represents multiple observations. Each of these rows is also called a detail row.

Both detail and summary reports can contain *summary lines* as well as detail rows. A summary line summarizes numerical data for a set of detail rows or for all detail rows. PROC REPORT provides both default and customized summaries (see "Using Break Lines" on page 874).

This overview illustrates the kinds of reports that PROC REPORT can produce. The statements that create the data sets and formats used in these reports are in Example 1 on page 958. The formats are stored in a permanent SAS data library. See "Examples" on page 958 for more reports and for the statements that create them.

A Sampling of Reports

The data set that these reports use contains one day's sales figures for eight stores in a chain of grocery stores.

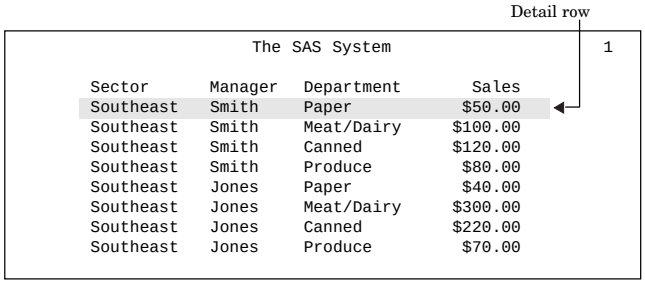
A simple PROC REPORT step produces a report similar to one produced by a simple PROC PRINT step. Figure 32.1 on page 862 illustrates the simplest kind of report that you can produce with PROC REPORT. The statements that produce the report follow. The data set and formats that the program uses are created in Example 1 on page 958. Although the WHERE and FORMAT statements are not essential, here they limit the amount of output and make the values easier to understand.

```
libname proclib 'SAS-data-library';


options nodate pageno=1 linesize=64 pagesize=60
        fmtsearch=(proclib);
```

```
proc report data=grocery nowd;
   where sector='se';
   format sector $sctrfmt.;
   format manager $mgrfmt.;
   format dept $deptfmt.;
   format sales dollar10.2;
run;
```

Figure 32.1 Simple Detail Report with a Detail Row for Each Observation

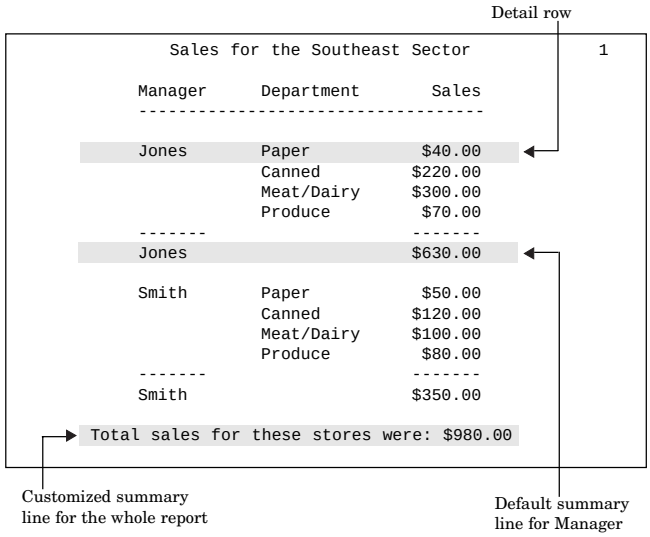


The report in Figure 32.2 on page 862 uses the same observations as those in Figure 32.1 on page 862. However, the statements that produce this report

- order the rows by the values of Manager and Department
- create a default summary line for each value of Manager
- create a customized summary line for the whole report. A customized summary lets you control the content and appearance of the summary information, but you must write additional PROC REPORT statements to create one.

For an explanation of the program that produces this report, see Example 2 on page 961.

Figure 32.2 Ordered Detail Report with Default and Customized Summaries

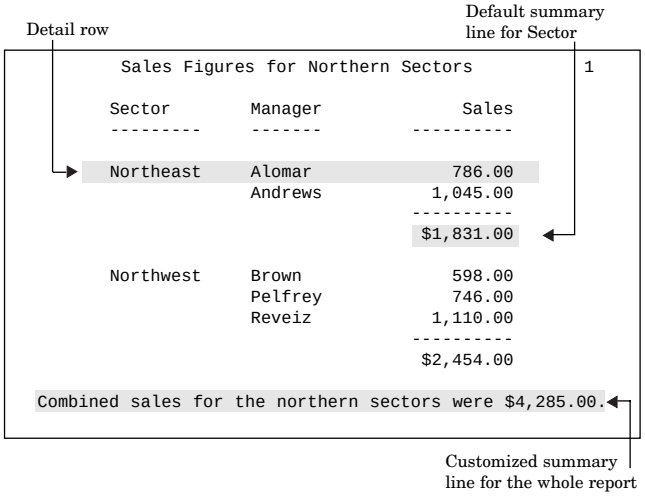


The summary report in Figure 32.3 on page 863 contains one row for each store in the northern sector. Each detail row represents four observations in the input data set–one for each department. Information about individual departments does not appear in this report. Instead, the value of Sales in each detail row is the sum of the values of Sales in all four departments. In addition to consolidating multiple observations into one row of the report, the statements that create this report

- customize the text of the column headers
- create default summaries that total the sales for each sector of the city
- create a customized summary that totals the sales for both sectors.

For an explanation of the program that produces this report, see Example 4 on page 967.

Figure 32.3 Summary Report with Default and Customized Summaries

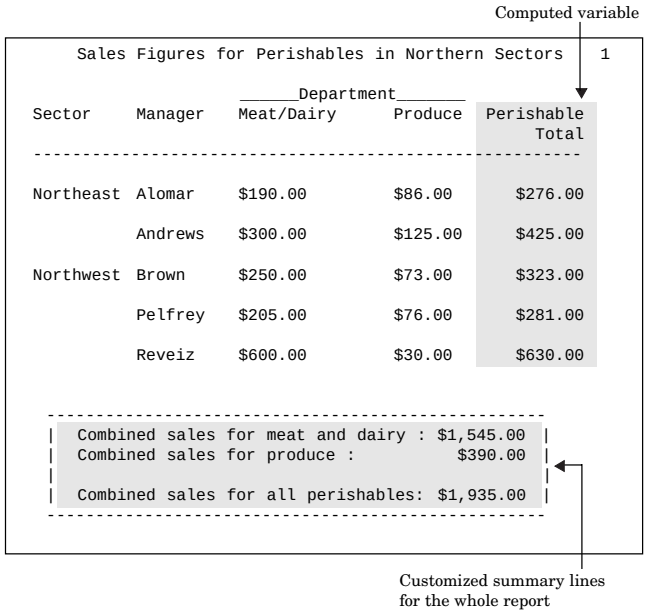


The summary report in Figure 32.4 on page 864 is similar to Figure 32.3 on page 863. The major difference is that it also includes information for individual departments. Each selected value of Department forms a column in the report. In addition, the statements that create this report

- compute and display a variable that is not in the input data set
- double-space the report
- put blank lines in some of the column headers.

For an explanation of the program that produces this report, see Example 5 on page 970.

Figure 32.4 Summary Report with a Column for Each Value of a Variable

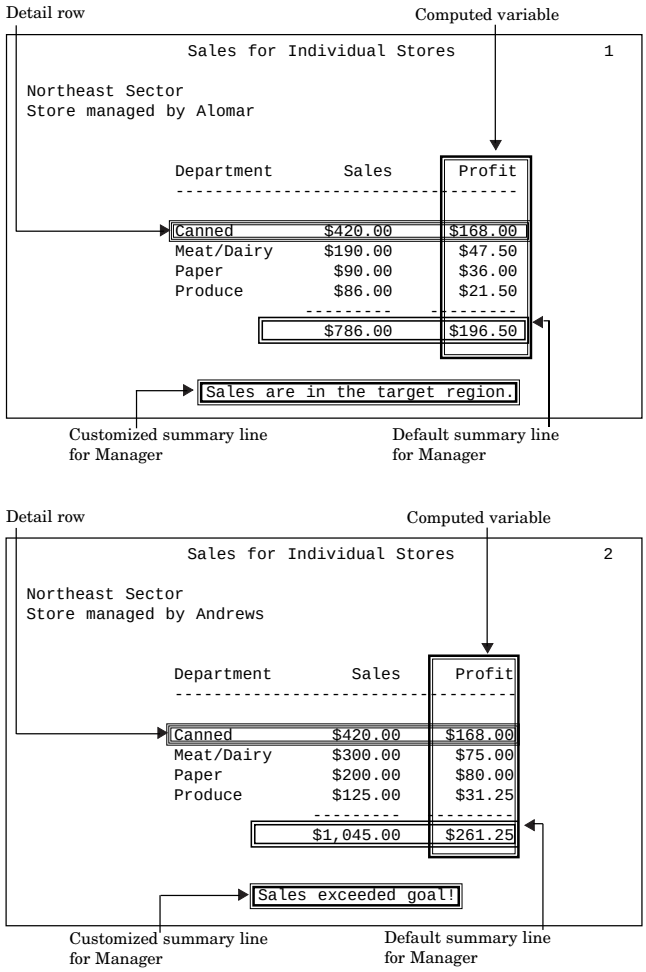


The customized report in Figure 32.5 on page 865 shows each manager's store on a separate page. Only the first two pages appear here. The statements that create this report create

- a customized header for each page of the report
- a computed variable (Profit) that is not in the input data set
- a customized summary with text that is dependent on the total sales for that manager's store.

For an explanation of the program that produces this report, see Example 9 on page 979.

Figure 32.5 Customized Summary Report



The report in Figure 32.6 on page 866uses customized style elements to control things like font faces, font sizes, and justification, as well as the width of the border of the table and the width of the spacing between cells. This report was created by using the HTML destination of the Output Delivery System (ODS) and the STYLE= option in several statements in the procedure.

For an explanation of the program that produces this report, see Example 16 on page 999. For information on ODS, see “Output Delivery System” on page 19.

Figure 32.6 HTML Output

Concepts

Laying Out a Report

Report writing is simplified if you approach it with a clear understanding of what you want the report to look like. The most important thing to figure out is the layout of the report. To determine the layout, ask yourself the following kinds of questions:

- What do I want to display in each column of the report?
- In what order do I want the columns to appear?
- Do I want to display a column for each value of a particular variable?
- Do I want a row for every observation in the report, or do I want to consolidate information for multiple observations into one row?
- In what order do I want the rows to appear?

Once you understand the layout of the report, use the COLUMN and DEFINE statements in PROC REPORT to construct the layout.

The COLUMN statement lists the items that appear in the columns of the report, describes the arrangement of the columns, and defines headers that span multiple columns. A report item can be

- a data set variable
- a statistic calculated by the procedure
- a variable that you compute based on other items in the report.

Omit the COLUMN statement if you want to include all variables in the input data set in the same order as they occur in the data set.

Note: If you start PROC REPORT in the windowing environment without the COLUMN statement, the initial report includes only as many variables as will fit on one page. △

The DEFINE statement (or, in the windowing environment, the DEFINITION window) defines the characteristics of an item in the report. These characteristics include how PROC REPORT uses the item in the report, the text of the column header, and the format to use to display values.

Usage of Variables in a Report

Much of a report's layout is determined by the usages that you specify for variables in the DEFINE statements or DEFINITION windows. For data set variables, these usages are

DISPLAY

ORDER

ACROSS

GROUP

ANALYSIS

A report can contain variables that are not in the input data set. These variables must have a usage of COMPUTED.

Display Variables

A report that contains one or more display variables has a row for every observation in the input data set. Display variables do not affect the order of the rows in the report. If no order variables appear to the left of a display variable, the order of the rows in the report reflects the order of the observations in the data set. By default, PROC REPORT treats all character variables as display variables.

Featured in: Example 1 on page 958

Order Variables

A report that contains one or more order variables has a row for every observation in the input data set. If no display variable appears to the left of an order variable, PROC REPORT orders the detail rows according to the ascending, formatted values of the order variable. You can change the default order with ORDER= and DESCENDING in the DEFINE statement or with the DEFINITION window.

If the report contains multiple order variables, PROC REPORT establishes the order of the detail rows by sorting these variables from left to right in the report. PROC REPORT does not repeat the value of an order variable from one row to the next if the value does not change.

Featured in: Example 2 on page 961

Across Variables

PROC REPORT creates a column for each value of an across variable. PROC REPORT orders the columns by the ascending, formatted values of the across variable. You can change the default order with ORDER= and DESCENDING in the DEFINE statement or with the DEFINITION window. If no other variable helps define the column (see "COLUMN Statement" on page 903), PROC REPORT displays the N statistic (the number of observations in the input data set that belong to that cell of the report).

If you are familiar with procedures that use class variables, you will see that across variables are class variables that are used in the column dimension.

Featured in: Example 5 on page 970

Group Variables

If a report contains one or more group variables, PROC REPORT tries to consolidate into one row all observations from the data set that have a unique combination of formatted values for all group variables.

When PROC REPORT creates groups, it orders the detail rows by the ascending, formatted values of the group variable. You can change the default order with ORDER= and DESCENDING in the DEFINE statement or with the DEFINITION window.

If the report contains multiple group variables, the REPORT procedure establishes the order of the detail rows by sorting these variables from left to right in the report. PROC REPORT does not repeat the values of a group variable from one row to the next if the value does not change.

If you are familiar with procedures that use class variables, you will see that group variables are class variables that are used in the row dimension.

Note: You cannot always create groups. PROC REPORT cannot consolidate observations into groups if the report contains any order variables or any display variables that do not have one or more statistics associated with them (see "COLUMN Statement" on page 903). In the windowing environment, if PROC REPORT cannot immediately create groups, it changes all display and order variables to group variables so that it can create the group variable that you requested. In the nonwindowing environment, it returns to the SAS log a message that explains why it could not create groups. Instead, it creates a detail report that displays group variables the same way as it displays order variables. Even if PROC REPORT creates a detail report, the variables that you defined as group variables retain that usage in their definitions. △

Featured in: Example 4 on page 967

Analysis Variables

An analysis variable is a numeric variable that is used to calculate a statistic for all the observations represented by a cell of the report. (Across variables, in combination with group variables or order variables, determine which observations a cell represents.) You associate a statistic with an analysis variable in the variable's definition or in the COLUMN statement. By default, PROC REPORT uses numeric variables as analysis variables that are used to calculate the Sum statistic.

The value of an analysis variable depends on where it appears in the report:

- In a detail report, the value of an analysis variable in a detail row is the value of the statistic associated with that variable calculated for a single observation. Calculating a statistic for a single observation is not practical; however, using the variable as an analysis variable enables you to create summary lines for sets of observations or for all observations.

- In a summary report, the value displayed for an analysis variable is the value of the statistic that you specify calculated for the set of observations represented by that cell of the report.
- In a summary line for any report, the value of an analysis variable is the value of the statistic that you specify calculated for all observations represented by that cell of the summary line.

See also: "BREAK Statement" on page 893 and "RBREAK Statement" on page 918

Featured in: Example 2 on page 961, Example 3 on page 964, Example 4 on page 967, and Example 5 on page 970

CAUTION:

Using dates in a report. Be careful when you use SAS dates in reports that contain summary lines. SAS dates are numeric variables. Unless you explicitly define dates as some other kind of variable, PROC REPORT summarizes them. △

Computed Variables

Computed variables are variables that you define for the report. They are not in the input data set, and PROC REPORT does not add them to the input data set. However, computed variables are included in an output data set if you create one.

In the windowing environment, you add a computed variable to a report from the COMPUTED VAR window.

In the nonwindowing environment, you add a computed variable by

- including the computed variable in the COLUMN statement
- defining the variable's usage as COMPUTED in the DEFINE statement
- computing the value of the variable in a compute block associated with the variable.

Featured in: Example 5 on page 970, Example 10 on page 983, and Example 13 on page 991

Interactions of Position and Usage

The position and usage of each variable in the report determine the report's structure and content. PROC REPORT orders the detail rows of the report according to the values of order and group variables, considered from left to right in the report. Similarly, PROC REPORT orders columns for an across variable from top to bottom, according to the values of the variable.

Several items can collectively define the contents of a column in a report. For instance, in Figure 32.7 on page 870, the values that appear in the third and fourth columns are collectively determined by Sales, an analysis variable, and by Department, an across variable. You create this kind of report with the COLUMN statement or, in the windowing environment, by placing report items above or below each other. This is called stacking items in the report because each item generates a header, and the headers are stacked one above the other.

Figure 32.7 Stacking Department and Sales

Sales Figures for Perishables in Northern Sectors

Sector	Manager	Department: Meat/Dairy	Department: Produce	Perishable Total
Northeast	Alomar	$190.00	$86.00	$276.00
	Andrews	$300.00	$125.00	$425.00
Northwest	Brown	$250.00	$73.00	$323.00
	Pelfrey	$205.00	$76.00	$281.00
	Reveiz	$600.00	$30.00	$630.00

When you use multiple items to define the contents of a column, at most one of the following can be in a column:

- a display variable with or without a statistic above or below it
- an analysis variable with or without a statistic above or below it
- an order variable
- a group variable
- a computed variable.

More than one of these items in a column creates a conflict for PROC REPORT about which values to display.

Table 32.1 on page 870 shows which report items can share a column.

Note: You cannot stack group or order variables with other report items. △

Table 32.1 Report Items That Can Share Columns

	Display	**Analysis**	**Order**	**Group**	**Computed**	**Across**	**Statistic**
Display						X*	X
Analysis						X	X
Order							
Group							
Computed variable						X	
Across	X*	X			X	X	X
Statistic	X	X				X	

*When a display variable and an across variable share a column, the report must also contain another variable that is not in the same column.

The following items can stand alone in a column:

- display variable
- analysis variable
- order variable
- group variable

- computed variable
- across variable
- N statistic.

Note: The values in a column occupied only by an across variable are frequency counts. △

Statistics Available in PROC REPORT

N	CSS
NMISS	STDERR
MEAN	CV
STD	T
MIN	PRT
MAX	VAR
RANGE	SUMWGT
SUM	PCTN
USS	PCTSUM

Every statistic except N must be associated with a variable. You associate a statistic with a variable either by placing the statistic above or below a numeric display variable or by specifying the statistic as a usage option in the DEFINE statement or in the DEFINITION window for an analysis variable.

You can place N anywhere because it is the number of observations in the input data set that contributes to the value in a cell of the report. The value of N does not depend on a particular variable.

For definitions of these statistics, see "Keywords and Formulas" on page 1458.

Note: If you use the MISSING option in the PROC REPORT statement, N includes observations with missing group, order, or across variables. △

Using Compute Blocks

A *compute block* is one or more programming statements that appear either between a COMPUTE and an ENDCOMP statement or in a COMPUTE window. PROC REPORT executes these statements as it builds the report. A compute block can be associated with a report item (a data set variable, a statistic, or a computed variable) or with a location (at the top or bottom of the report; before or after a set of observations). You create a compute block with the COMPUTE window or with the COMPUTE statement. One form of the COMPUTE statement associates the compute block with a report item. Another form associates the compute block with a location in the report (see "Using Break Lines" on page 874).

Note: When you use the COMPUTE statement, you do not have to use a corresponding BREAK or RBREAK statement. (See Example 2 on page 961, which uses COMPUTE AFTER but does not use the RBREAK statement). Use these statements only when you want to implement one or more BREAK statement or RBREAK statement options (see Example 9 on page 979, which uses both COMPUTE AFTER MANAGER and BREAK AFTER MANAGER. △

The Purpose of Compute Blocks

A compute block that is associated with a report item can

- define a variable that appears in a column of the report but is not in the input data set
- define display attributes for a report item (see "CALL DEFINE Statement" on page 899).

A compute block that is associated with a location can write a customized summary.

In addition, all compute blocks can use SAS language elements to perform calculations (see "The Contents of Compute Blocks" on page 872). A PROC REPORT step can contain multiple compute blocks.

The Contents of Compute Blocks

In the windowing environment, a compute block is in a COMPUTE window. In the nonwindowing environment, a compute block begins with a COMPUTE statement and ends with an ENDCOMP statement. Within a compute block, you can use these SAS language elements:

- DM statement
- %INCLUDE statement
- these DATA step statements:

assignment	LENGTH
CALL	LINK
DO (all forms)	RETURN
END	SELECT
GO TO	sum
IF-THEN/ELSE	

- comments
- null statements
- macro variables and macro invocations
- all DATA step functions.

For information about SAS language elements see the appropriate section in *SAS Language Reference: Dictionary*.

Within a compute block, you can also use these PROC REPORT features:

- Compute blocks for a customized summary can contain one or more LINE statements, which place customized text and formatted values in the summary. (See "LINE Statement" on page 917.)
- Compute blocks for a report item can contain one or more CALL DEFINE statements, which set attributes like color and format each time a value for the item is placed in the report. (See "CALL DEFINE Statement" on page 899.)
- Any compute block can contain the automatic variable _BREAK_ (see "The Automatic Variable _BREAK_" on page 875.

Four Ways to Reference Report Items in a Compute Block

A compute block can reference any report item that forms a column in the report (whether or not the column is visible). You reference report items in a compute block in one of four ways:

- by name.
- by a compound name that identifies both the variable and the name of the statistic that you calculate with it. A compound name has this form

  variable-name.statistic

- by an alias that you create in the COLUMN statement or in the DEFINITION window.
- by column number, in the form

  _C*n*_

  where *n* is the number of the column (from left to right) in the report.

  Note: Even though the columns that you define with NOPRINT and NOZERO do not appear in the report, you must count them when you are referencing columns by number. See the discussion of NOPRINT on page 913 and NOZERO on page 913. △

CAUTION:

Referencing variables that have missing values leads to missing values. If a compute block references a variable that has a missing value, PROC REPORT displays that variable as a blank (for character variables) or as a period (for numeric variables). △

The following table shows how to use each type of reference in a compute block.

If the variable that you reference is this type...	Then refer to it by...	For example...
group	name*	Department
order	name*	Department
computed	name*	Department
display	name*	Department
display sharing a column with a statistic	a compound name*	Sales.sum
analysis	a compound name*	Sales.mean
any type sharing a column with an across variable	column number **	_c3_

*If the variable has an alias, you must reference it with the alias.

**Even if the variable has an alias, you must reference it by column number.

Featured in: Example 3 on page 964, which references analysis variables by their aliases; Example 5 on page 970, which references variables by column number; and Example 10 on page 983, which references group variables and computed variables by name.

Compute Block Processing

PROC REPORT processes compute blocks in two different ways.

- If a compute block is associated with a location, PROC REPORT executes the compute block only at that location. Because PROC REPORT calculates statistics

for groups before it actually constructs the rows of the report, statistics for sets of detail rows are available before or after the rows are displayed, as are values for any variables based on these statistics.

- If a compute block is associated with a report item, PROC REPORT executes the compute block on every row of the report when it comes to the column for that item. The value of a computed variable in any row of a report is the last value assigned to that variable during that execution of the DATA step statements in the compute block. PROC REPORT assigns values to the columns in a row of a report from left to right. Consequently, you cannot base the calculation of a computed variable on any variable that appears to its right in the report.

Note: PROC REPORT recalculates computed variables at breaks. For details on compute block processing see "How PROC REPORT Builds a Report" on page 946. △

Using Break Lines

Break lines are lines of text (including blanks) that appear at particular locations, called *breaks*, in a report. A report can contain multiple breaks. Generally, break lines are used to visually separate parts of a report, to summarize information, or both. They can occur

- at the beginning or end of a report
- at the top or bottom of each page
- between sets of observations (whenever the value of a group or order variable changes).

Break lines can contain

- text
- values calculated for either a set of rows or for the whole report.

Creating Break Lines

There are two ways to create break lines. The first way is simpler. It produces a default summary. The second way is more flexible. It produces a customized summary and provides a way to slightly modify a default summary. Default summaries and customized summaries can appear at the same location in a report.

Default summaries are produced with the BREAK statement, the RBREAK statement, or the BREAK window. You can use default summaries to visually separate parts of the report, to summarize information for numeric variables, or both. Options provide some control over the appearance of the break lines, but if you choose to summarize numeric variables, you have no control over the content and the placement of the summary information. (A break line that summarizes information is a summary line.)

Customized summaries are produced in a compute block. You can control both the appearance and content of a customized summary, but you must write the code to do so.

Order of Break Lines

You control the order of the lines in a customized summary. However, PROC REPORT controls the order of lines in a default summary and the placement of a customized summary relative to a default summary. When a default summary contains multiple break lines, the order in which the break lines appear is

1. overlining or double overlining
2. summary line

3 underlining or double underlining

4 blank line

5 page break.

If you define a customized summary for the same location, customized break lines appear after underlining or double underlining.

The Automatic Variable _BREAK_

PROC REPORT automatically creates a variable called _BREAK_. This variable contains

- a blank if the current line is not part of a break
- the value of the break variable if the current line is part of a break between sets of observations
- the value **RBREAK** if the current line is part of a break at the beginning or end of the report.

Using Style Elements in PROC REPORT

If you use the Output Delivery System to create HTML output or Printer output from PROC REPORT, you can specify style elements for the procedure to use for various parts of the report. Style elements determine presentation attributes like font face, font weight, color, and so forth. Information about the attributes that you can set for a style is in "Customizing the Style Definition That ODS Uses" on page 42.

You specify style elements for PROC REPORT with the STYLE= option. Table 32.2 on page 875 shows where you can use this option. Specifications on a statement other than the PROC REPORT statement override the same specification in the PROC REPORT statement. However, any style attributes that you specify in the PROC REPORT statement and do not override in another statement are inherited. For instance, if you specify a blue background and a white foreground for all column headers in the PROC REPORT statement, and you specify a gray background for the column headers of a variable in the DEFINE statement, the background for that particular column header is gray, and the foreground is white (as specified in the PROC REPORT statement).

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

Table 32.2 Using the STYLE= Option in PROC REPORT

To set the style element for	Use STYLE= in this statement
The report as a whole, including attributes of the table itself (like the spacing between cells) as well as style elements for column headers, cells, default summaries, customized summaries, and individual cells defined by CALL DEFINE statements	PROC REPORT
Column headers and cells for a particular variable	DEFINE
Default summary lines	BREAK RBREAK
Customized summary lines	COMPUTE (with a location and LINE statements)
Individual cells of the report	CALL DEFINE

Printing a Report

Printing from the REPORT Window

By default, if you print from the REPORT window, the report is routed directly to your printer. If you want, you can specify a form to use for printing (see "Printing with a Form" on page 876). Forms specify things like the type of printer that you are using, text format, and page orientation.

Note: Forms are available only when you run SAS from a windowing environment. △

Operating Environment Information: Printing is implemented differently in different operating environments. For information related to printing, consult *SAS Language Reference: Concepts*. Additional information may be available in the SAS documentation for your operating environment. △

Printing with a Form

To print with a form from the REPORT window:

1. Specify a form. You can specify a form with the FORMNAME command or, in some cases, through the File menu.
2. Specify a print file if you want the output to go to a file instead of directly to the printer. You can specify a print file with the PRTFILE command or, in some cases, through the **File** pull-down menu.
3. Issue the PRINT or PRINT PAGE command from the command line or from the **File** pull-down menu.
4. If you specified a print file,
   a. Free the print file. You can free a file with the FREE command or, in some cases, through **Print utilities** in the **File** pull-down menu. You cannot view or print the file until you free it.
   b. Use operating environment commands to send the file to the printer.

Printing from the OUTPUT Window

If you are running PROC REPORT with the NOWINDOWS option, the default destination for the output is the OUTPUT window. Use the commands in the **File** pull-down menu to print the report.

Printing from Noninteractive or Batch Mode

If you use noninteractive or batch mode, SAS writes the output either to the display or to external files, depending on the operating environment and on the SAS options that you use. Refer to the SAS documentation for your operating environment for information about how these files are named and where they are stored.

You can print the output file directly or use PROC PRINTTO to redirect the output to another file. In either case, no form is used, but carriage control characters are written if the destination is a print file.

Use operating environment commands to send the file to the printer.

Printing from Interactive Line Mode

If you use interactive line mode, by default the output and log are displayed on the screen immediately following the programming statements. Use PROC PRINTTO to

redirect the output to an external file. Then use operating environment commands to send the file to the printer.

Using PROC PRINTTO

PROC PRINTTO defines destinations for the SAS output and the SAS log (see Chapter 29, "The PRINTTO Procedure," on page 819).

PROC PRINTTO does not use a form, but it does write carriage control characters if you are writing to a print file.

CAUTION:
You need two PROC PRINTTO steps. The first PROC PRINTTO step precedes the PROC REPORT step. It redirects the output to a file. The second PROC PRINTTO step follows the PROC REPORT step. It reestablishes the default destination and frees the output file. You cannot print the file until PROC PRINTTO frees it. △

Storing and Reusing a Report Definition

The OUTREPT= option in the PROC REPORT statement stores a report definition in the specified catalog entry. If you are working in the nonwindowing environment, the definition is based on the PROC REPORT step that you submit. If you are in the windowing environment, the definition is based on the report that is in the REPORT window when you end the procedure. The SAS System assigns an entry type of REPT to the entry.

In the windowing environment, you can save the definition of the current report by selecting

File ► Save Report

A report definition may differ from the SAS program that creates the report (see the discussion of OUTREPT= on page 887).

You can use a report definition to create an identically structured report for any SAS data set that contains variables with the same names as the ones used in the report definition. Use the REPORT= option in the PROC REPORT statement to load a report definition when you start PROC REPORT. In the windowing environment, load a report definition from the LOAD REPORT window by selecting

File ► Open Report

Procedure Syntax

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

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

PROC REPORT <*option(s)*>;

BREAK *location break-variable*</ *option(s)*>;

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

COLUMN *column-specification(s)*;

COMPUTE *location* <*target*>
</ STYLE=<*style-element-name*>
<[*style–attribute -specification(s)*]>>;
LINE *specification(s)*;
. . . *select SAS language elements* . . .
ENDCOMP;

COMPUTE *report-item* </ *type-specification*>;
CALL DEFINE (*column-id*, '*attribute-name*', *value*);
. . . *select SAS language elements* . . .
ENDCOMP;

DEFINE *report-item* / <*usage*>
<*attribute(s)*>
<*option(s)*>
<*justification*>
<COLOR=*color*>
<'*column-header-1*' <…'*column-header-n*'>>
<*style*>;

FREQ *variable*;

RBREAK *location* </ *option(s)*>;

WEIGHT *variable*;

To do this	Use this statement
Produce a default summary at a change in the value of a group or order variable	BREAK
Create a separate report for each BY group	BY
Set the value of an attribute for a particular column in the current row	CALL DEFINE
Describe the arrangement of all columns and of headers that span more than one column	COLUMN
Specify one or more programming statements that PROC REPORT executes as it builds the report	COMPUTE and ENDCOMP
Describe how to use and display a report item	DEFINE
Treat observations as if they appear multiple times in the input data set	FREQ
Provide a subset of features of the PUT statement for writing customized summaries	LINE

To do this	Use this statement
Produce a default summary at the beginning or end of a report or at the beginning and end of each BY group	RBREAK
Specify weights for analysis variables in the statistical calculations	WEIGHT

PROC REPORT Statement

PROC REPORT <*option(s)*>;

To do this	Use this option
Specify the input data set	DATA=
Specify the output data set	OUT=
Select the windowing or the nonwindowing environment	WINDOWS | NOWINDOWS
Specify the divisor to use in the calculation of variances	VARDEF=
Exclude observations with nonpositive weight values from the analysis.	EXCLNPWGT
Use a report that was created before compute blocks required aliases (before Release 6.11)	NOALIAS
Specify one or more style elements (for the Output Delivery System) to use for different parts of the report	STYLE=
Control the layout of the report	
Use formatting characters to add line-drawing characters to the report	BOX
Specify whether to center or left-justify the report and summary text	CENTER | NOCENTER
Specify the default number of characters for columns containing computed variables or numeric data set variables	COLWIDTH=
Define the characters to use as line-drawing characters in the report	FORMCHAR=
Specify the length of a line of the report	LS=
Consider missing values as valid values for group, order, or across variables	MISSING
Specify the number of panels on each page of the report	PANELS=

To do this	Use this option
Specify the number of lines in a page of the report	PS=
Specify the number of blank characters between panels	PSPACE=
Override options in the DEFINE statement that suppress the display of a column	SHOWALL
Specify the number of blank characters between columns	SPACING=
Display one value from each column of the report, on consecutive lines if necessary, before displaying another value from the first column	WRAP
Customize column headers	
Underline all column headers and the spaces between them	HEADLINE
Write a blank line beneath all column headers	HEADSKIP
Suppress column headers	NOHEADER
Write *name*= in front of each value in the report, where *name*= is the column header for the value	NAMED
Specify the split character	SPLIT=
Store and retrieve report definitions, PROC REPORT statements, and your report profile	
Write to the SAS log the PROC REPORT code that creates the current report	LIST
Suppress the building of the report	NOEXEC
Store in the specified catalog the report definition defined by the PROC REPORT step that you submit	OUTREPT=
Identify the report profile to use	PROFILE=
Specify the report definition to use	REPORT=
Control the windowing environment	
Display command lines rather than menu bars in all REPORT windows	COMMAND

To do this	Use this option
Identify the library and catalog containing user-defined help for the report	HELP=
Open the REPORT window and start the PROMPT facility	PROMPT

Options

BOX
uses formatting characters to add line-drawing characters to the report. These characters

- surround each page of the report
- separate column headers from the body of the report
- separate rows and columns from each other
- separate values in a summary line from other values in the same columns
- separate a customized summary from the rest of the report.

Restriction: This option has no effect on the HTML or Printer output.

Interaction: You cannot use BOX if you use WRAP in the PROC REPORT statement or in the ROPTIONS window or if you use FLOW in any item definition.

See also: the discussion of FORMCHAR= on page 882

Featured in: Example 12 on page 989

CENTER|NOCENTER
specifies whether to center or left-justify the report and summary text (customized break lines).

PROC REPORT honors the first of these centering specifications that it finds:

- the CENTER or NOCENTER option in the PROC REPORT statement or the CENTER toggle in the ROPTIONS window
- the CENTER or NOCENTER option stored in the report definition that is loaded with REPORT= in the PROC REPORT statement
- the SAS system option CENTER or NOCENTER.

Restriction: This option has no effect on the HTML or Printer output.

Interaction: When CENTER is in effect, PROC REPORT ignores spacing that precedes the leftmost variable in the report.

COLWIDTH=*column-width*
specifies the default number of characters for columns containing computed variables or numeric data set variables.

Default: 9

Range: 1 to the linesize

Restriction: This option has no effect on the HTML or Printer output.

Interaction: When setting the width for a column, PROC REPORT first looks at WIDTH= in the definition for that column. If WIDTH= is not present, PROC REPORT uses a column width large enough to accommodate the format for the item. (For information about formats see the discussion of FORMAT= on page 912.)

If no format is associated with the item, the column width depends on variable type:

If the variable is a ...	Then the column width is the ...
character variable in the input data set	length of the variable
numeric variable in the input data set	value of the COLWIDTH= option
computed variable (numeric or character)	value of the COLWIDTH= option

Featured in: Example 2 on page 961

COMMAND
displays command lines rather than menu bars in all REPORT windows.

After you have started PROC REPORT in the windowing environment, you can display the menu bars in the current window by issuing the COMMAND command. You can display the menu bars in all PROC REPORT windows by issuing the PMENU command. The PMENU command affects all the windows in your SAS session. Both of these commands are toggles.

You can store a setting of COMMAND in your report profile. PROC REPORT honors the first of these settings that it finds:

- the COMMAND option in the PROC REPORT statement
- the setting in your report profile.

Restriction: This option has no effect in the nonwindowing environment.

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

Main discussion: "Input Data Sets" on page 18

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

Alias: EXCLNPWGTS

Requirement: You must use a WEIGHT statement.

See also: "WEIGHT Statement" on page 922

FORMCHAR <(*position(s)*)>='*formatting-character(s)*'
defines the characters to use as line-drawing characters in the report.

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

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

Range: PROC REPORT uses 12 of the 20 formatting characters that SAS provides. Table 32.3 on page 883 shows the formatting characters that PROC REPORT uses. Figure 32.8 on page 884 illustrates the use of some commonly used formatting character in the output from PROC REPORT.

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

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

Restriction: This option has no effect on the HTML or Printer output.

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

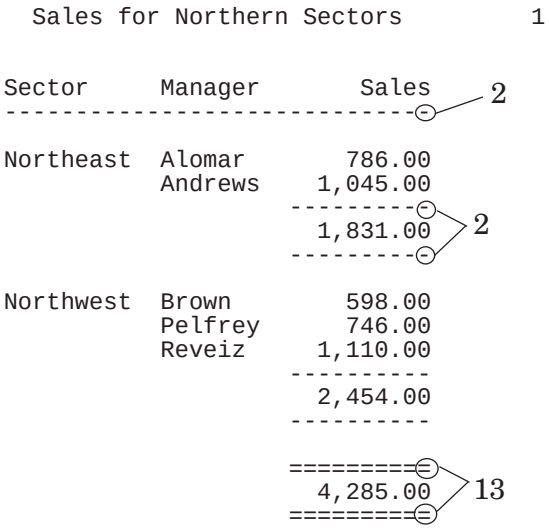
Tip: You can use any character in *formatting-characters*, including hexadecimal characters. If you use hexadecimal characters, you must put an **x** after the closing quote. For instance, the following option assigns the hexadecimal character 2D to the third formatting character, the hexadecimal character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

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

Table 32.3 Formatting Characters Used by PROC REPORT

Position	Default	Used to draw
1	|	the right and left borders and the vertical separators between columns
2	-	the top and bottom borders and the horizontal separators between rows; also underlining and overlining in break lines as well as the underlining that the HEADLINE option draws
3	-	the top character in the left border
4	-	the top character in a line of characters that separates columns
5	-	the top character in the right border
6	|	the leftmost character in a row of horizontal separators
7	+	the intersection of a column of vertical characters and a row of horizontal characters
8	|	the rightmost character in a row of horizontal separators
9	-	the bottom character in the left border
10	-	the bottom character in a line of characters that separate columns
11	-	the bottom character in the right border
13	=	double overlining and double underlining in break lines

Figure 32.8 Formatting Characters in PROC REPORT Output



HEADLINE
underlines all column headers and the spaces between them at the top of each page of the report.

The HEADLINE option underlines with the second formatting character. (See the discussion of FORMCHAR= on page 882 .)

Default: hyphen (–)

Restriction: This option has no effect on the HTML or Printer output.

Tip: In traditional (monospace) SAS output, you can underline column headers without underlining the spaces between them, by using two hyphens **('--')** as the last line of each column header instead of using HEADLINE.

Featured in: Example 2 on page 961 and Example 8 on page 977

HEADSKIP
writes a blank line beneath all column headers (or beneath the underlining that the HEADLINE option writes) at the top of each page of the report.

Restriction: This option has no effect on the HTML or Printer output.

Featured in: Example 2 on page 961

HELP=*libref.catalog*
identifies the library and catalog containing user-defined help for the report. This help can be in CBT or HELP catalog entries. You can write a CBT or HELP entry for each item in the report with the BUILD procedure in SAS/AF software. Store all such entries for a report in the same catalog.

Specify the entry name for help for a particular report item in the DEFINITION window for that report item or in a DEFINE statement.

Restriction: This option has no effect in the nonwindowing environment or on the HTML or Printer output.

LIST
writes to the SAS log the PROC REPORT code that creates the current report. This listing may differ in these ways from the statements that you submit:

- It shows some defaults that you may not have specified.
- It omits some statements that are not specific to the REPORT procedure, whether you submit them with the PROC REPORT step or had previously submitted them. These statements include

  BY

  FOOTNOTE

  FREQ

  TITLE

  WEIGHT

  WHERE
- It omits these PROC REPORT statement options:

  LIST

  OUT=

  OUTREPT=

  PROFILE=

  REPORT=

  WINDOWS|NOWINDOWS
- It omits SAS system options.
- It resolves automatic macro variables.

Restriction: This option has no effect in the windowing environment. Selecting

Tools ▶ Report Statements

serves a similar purpose. It writes the report definition for the report that is currently in the REPORT window to the SOURCE window.

LS=*line-size*
specifies the length of a line of the report.

PROC REPORT honors the first of these line size specifications that it finds:

- the LS= option in the PROC REPORT statement or Linesize= in the ROPTIONS window
- the LS= setting stored in the report definition loaded with REPORT= in the PROC REPORT statement
- the SAS system option LINESIZE=.

Range: 64-256 (integer)

Restriction: This option has no effect on the HTML or Printer output.

Featured in: Example 6 on page 973 and Example 8 on page 977

MISSING
considers missing values as valid values for group, order, or across variables. Special missing values used to represent numeric values (the letters A through Z and the underscore (_) character) are each considered as a different value. A group for each missing value appears in the report. If you omit the MISSING option, PROC REPORT does not include observations with a missing value for any group, order, or across variables in the report.

See also: For information about special missing values, see the section on missing values in *SAS Language Reference: Concepts*.

Featured in: Example 11 on page 986

NAMED
writes *name*= in front of each value in the report, where *name* is the column header for the value.

Interaction: When you use the NAMED option, PROC REPORT automatically uses the NOHEADER option.

Tip: Use NAMED in conjunction with the WRAP option to produce a report that wraps all columns for a single row of the report onto consecutive lines rather than placing columns of a wide report on separate pages.

Featured in: Example 7 on page 975

lets you use a report that was created before compute blocks required aliases (before Release 6.11). If you use NOALIAS, you cannot use aliases in compute blocks.

NOCENTER
See CENTER|NOCENTER on page 881.

NOEXEC
suppresses the building of the report. Use NOEXEC with OUTREPT= to store a report definition in a catalog entry. Use NOEXEC with LIST and REPORT= to display a listing of the specified report definition.

NOHEADER
suppresses column headers, including those that span multiple columns.

Once you suppress the display of column headers in the windowing environment, you cannot select any report items.

NOWINDOWS

Alias: NOWD

See WINDOWS|NOWINDOWS on page 893.

OUT=*SAS-data-set*
names the output data set. If this data set does not exist, PROC REPORT creates it. The data set contains one observation for each detail row of the report and one observation for each unique summary line. If you use both customized and default summaries at the same place in the report, the output data set contains only one observation because the two summaries differ only in how they present the data. Information about customization (underlining, color, text, and so forth) is not data and is not saved in the output data set.

The output data set contains one variable for each column of the report. PROC REPORT tries to use the name of the report item as the name of the corresponding variable in the output data set. However, this is not possible if a data set variable is under or over an across variable or if a data set variable appears multiple times in the COLUMN statement without aliases. In these cases, the name of the variable is based on the column number (_C1_, _C2_, and so forth).

Output data set variables that are derived from input data set variables retain the formats of their counterparts in the input data set. PROC REPORT derives labels for these variables from the corresponding column headers in the report unless the only item defining the column is an across variable. In that case, the variables have no label. If multiple items are stacked in a column, the labels of the corresponding output data set variables come from the analysis variable in the column.

The output data set also contains a variable named _BREAK_. If an observation in the output data set derives from a detail row in the report, the value of _BREAK_

is missing. If it derives from a summary line, the value of _BREAK_ is the name of the break variable associated with the summary line, or _RBREAK_.

Interaction: You cannot use OUT= in a PROC REPORT step that uses a BY statement.

Featured in: Example 12 on page 989 and Example 13 on page 991

OUTREPT=*libref.catalog.entry*

stores in the specified catalog entry the REPORT definition defined by the PROC REPORT step that you submit. PROC REPORT assigns the entry a type of REPT.

The stored report definition may differ in these ways from the statements that you submit:

- It omits some statements that are not specific to the REPORT procedure, whether you submit them with the PROC REPORT step or whether they are already in effect when you submit the step. These statements include

  BY

  FOOTNOTE

  FREQ

  TITLE

  WEIGHT

  WHERE
- It omits these PROC REPORT statement options:

  LIST

  NOALIAS

  OUT=

  OUTREPT=

  PROFILE=

  REPORT=

  WINDOWS|NOWINDOWS
- It omits SAS system options.
- It resolves automatic macro variables.

Featured in: Example 7 on page 975

PANELS=*number-of-panels*

specifies the number of panels on each page of the report. If the width of a report is less than half of the line size, you can display the data in multiple sets of columns so that rows that would otherwise appear on multiple pages appear on the same page. Each set of columns is a *panel*. A familiar example of this kind of report is a telephone book, which contains multiple panels of names and telephone numbers on a single page.

When PROC REPORT writes a multipanel report, it fills one panel before beginning the next.

The number of panels that fits on a page depends on the

- width of the panel
- space between panels
- line size.

Restriction: This option has no effect on the HTML or Printer destination.

Default: 1

Tip: If *number-of-panels* is larger than the number of panels that can fit on the page, PROC REPORT creates as many panels as it can. Let PROC REPORT put your data in the maximum number of panels that can fit on the page by specifying a large number of panels (for example, 99).

See also: For information about the space between panels and the line size, see the discussions of PSPACE= on page 889 and the discussion of LS= on page 885.

Featured in: Example 8 on page 977

PROFILE=*libref.catalog*

identifies the report profile to use. A profile

- specifies the location of menus that define alternative menu bars and pull-down menus for the REPORT and COMPUTE windows.
- sets defaults for WINDOWS, PROMPT, and COMMAND.

PROC REPORT uses the entry REPORT.PROFILE in the catalog that you specify as your profile. If no such entry exists, or if you do not specify a profile, PROC REPORT uses the entry REPORT.PROFILE in SASUSER.PROFILE. If you have no profile, PROC REPORT uses default menus and the default settings of the options.

You create a profile from the PROFILE window while using PROC REPORT in a windowing environment. To create a profile

1. Invoke PROC REPORT with the WINDOWS option.
2. Select **`Tools`** → **`Report Profile`**.
3. Fill in the fields to suit your needs.
4. Select OK to exit the PROFILE window. When you exit the window, PROC REPORT stores the profile in SASUSER.PROFILE.REPORT.PROFILE. Use the CATALOG procedure or the Explorer window to copy the profile to another location.

   Note: If you open the PROFILE window and decide not to create a profile, select CANCEL to close the window. △

PROMPT

opens the REPORT window and starts the PROMPT facility. This facility guides you through creating a new report or adding more data set variables or statistics to an existing report.

If you start PROC REPORT with prompting, the first window gives you a chance to limit the number of observations that are used during prompting. When you exit the prompter, PROC REPORT removes the limit.

Restriction: When you use the PROMPT option, you open the REPORT window. When the REPORT window is open, you cannot send procedure output to the HTML or Printer destination.

Tip: You can store a setting of PROMPT in your report profile. PROC REPORT honors the first of these settings that it finds:

- the PROMPT option in the PROC REPORT statement
- the setting in your report profile.

If you omit PROMPT from the PROC REPORT statement, the procedure uses the setting in your report profile, if you have one. If you do not have a report profile, PROC REPORT does not use the prompt facility. For information on report profiles, see “PROFILE” on page 936.

PS=*page-size*
specifies the number of lines in a page of the report.
PROC REPORT honors the first of these page size specifications that it finds:

- the PS= option in the PROC REPORT statement
- the PS= setting in the report definition specified with REPORT= in the PROC REPORT statement
- the SAS system option PAGESIZE=.

Range: 15-32,767 (integer)

Restriction: This option has no effect on the HTML or Printer output.

Featured in: Example 6 on page 973 and Example 8 on page 977

PSPACE=*space-between-panels*
specifies the number of blank characters between panels. PROC REPORT separates all panels in the report by the same number of blank characters. For each panel, the sum of its width and the number of blank characters separating it from the panel to its left cannot exceed the line size.

Default: 4

Restriction: This option has no effect on the HTML or Printer output.

Featured in: Example 8 on page 977

REPORT=*libref.catalog.entry*
specifies the report definition to use. PROC REPORT stores all report definitions as entries of type REPT in a SAS catalog.

Interaction: If you use REPORT=, you cannot use the COLUMN statement.

See also: OUTREPT= on page 887

Featured in: Example 7 on page 975

SHOWALL
overrides options in the DEFINE statement that suppress the display of a column.

See also: NOPRINT and NOZERO in "DEFINE Statement" on page 908

SPACING=*space-between-columns*
specifies the number of blank characters between columns. For each column, the sum of its width and the blank characters between it and the column to its left cannot exceed the line size.

Default: 2

Restriction: This option has no effect on the HTML or Printer output.

Interaction: PROC REPORT separates all columns in the report by the number of blank characters specified by SPACING= in the PROC REPORT statement unless you use SPACING= in the DEFINE statement to change the spacing to the left of a specific item.

Interaction: When CENTER is in effect, PROC REPORT ignores spacing that precedes the leftmost variable in the report.

Featured in: Example 2 on page 961

SPLIT='*character*'
specifies the split character. PROC REPORT breaks a column header when it reaches that character and continues the header on the next line. The split character itself is not part of the column header although each occurrence of the split character counts toward the 256-character maximum for a label.

Default: slash (/)

Interaction: The FLOW option in the DEFINE statement honors the split character.

Featured in: Example 5 on page 970

STYLE<(*location(s)*)>=<*style-element-name*><[*style-attribute-specification(s)*]>
specifies the style element to use for the specified locations in the report.

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

location
identifies the part of the report that the STYLE= option affects. Table 32.4 on page 890 shows the available locations and the other statements that you can specify them in. Specifications in a statement other than the PROC REPORT statement override the same specification in the PROC REPORT statement. However, any style attributes that you specify in the PROC REPORT statement and do not override in another statement are inherited. For instance, if you specify a blue background and a white foreground for all column headers in the PROC REPORT statement, and you specify a gray background for the column headers of a variable in the DEFINE statement, the background for that particular column header is gray, and the foreground is white (as specified in the PROC REPORT statement).

Table 32.4 Specifying Locations in the STYLE= Option

This *location*	Affects this part of the report	And can also be specified for individual items in this statement
CALLDEF	all cells that are identified by a CALL DEFINE statement	CALL DEFINE
COLUMN	the cells of all columns	DEFINE
HEADER	all column headers and all spanning headers	DEFINE (for column headers)
LINES	all LINE statements in all compute blocks	COMPUTE (with one or more LINE statements)
REPORT	the structural part of the report, that is, the underlying table. Use REPORT to set things such as the width of the border and the space between cells	none
SUMMARY	all default summary lines produced by a BREAK or an RBREAK statement	BREAK RBREAK

style-element-name
is the name of a style element that is part of a style definition that is registered with the Output Delivery System. SAS Institute provides some style definitions. Users can create their own style definitions with PROC TEMPLATE. For information about Institute-supplied style definitions, see “What Style Definitions Are Shipped with the Software?” on page 43. For information about PROC

TEMPLATE and the Output Delivery System, see *The Complete Guide to the SAS Output Delivery System*.

Default: Table 32.5 on page 891 shows the default style element for each location:

Table 32.5 The Default Style Element for Each Location in PROC REPORT

Location	Default style element
CALLDEF	Data
COLUMN	Data
HEADER	Header
LINES	NoteContent
REPORT	Table
SUMMARY	DataEmphasis

style-attribute-specification

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

style-attribute-name=style-attribute-value

You can set these style attributes in the REPORT location:

BACKGROUND=
BACKGROUNDIMAGE=
BORDERCOLOR=
BORDERCOLORDARK=
BORDERCOLORLIGHT=
BORDERWIDTH=
CELLPADDING=
CELLSPACING=
FONT=*
FONT_FACE=*
FONT_SIZE=*
FONT_STYLE=*
FONT_WEIGHT=*
FONT_WIDTH=*
FOREGROUND=*
FRAME=
HTMLCLASS=
JUST=
OUTPUTWIDTH=
POSTHTML=
POSTIMAGE=
POSTTEXT=
PREHTML=
PREIMAGE=
PRETEXT=
RULES=

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

You can set these style attributes in the CALLDEF, COLUMN, HEADER, LINES, and SUMMARY locations:

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

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

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

Featured in: Example 15 on page 996 and Example 16 on page 999

VARDEF=*divisor*

specifies the divisor to use in the calculation of the variance and standard deviation. Table 32.6 on page 892 shows the possible values for *divisor* and associated divisors.

Table 32.6 Possible Values for VARDEF=

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

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

Default: DF

Requirement: To compute the standard error of the mean and Student's *t*-test, use the default value of VARDEF=.

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

w_i is the weight for the ith observation. This yields an estimate of the variance of an observation with unit weight.

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

See also: "WEIGHT" on page 73

WINDOWS|NOWINDOWS

selects a windowing or nonwindowing environment.

When you use WINDOWS, SAS opens the REPORT window, which enables you to modify a report repeatedly and to see the modifications immediately. When you use NOWINDOWS, PROC REPORT runs without the REPORT window and sends its output to the SAS procedure output.

Alias: WD|NOWD

Restriction: When you use the WINDOWS option, you cannot send procedure output to the HTML or Printer destination.

Tip: You can store a setting of WINDOWS in your report profile, if you have one. If you do not specify WINDOWS or NOWINDOWS in the PROC REPORT statement, the procedure uses the setting in your report profile. If you do not have a report profile, PROC REPORT looks at the setting of the SAS system option DMS. If DMS is ON, PROC REPORT uses the windowing environment; if DMS is OFF, it uses the nonwindowing environment.

See also: For a discussion of the report profile see the discussion of PROFILE= on page 888.

Featured in: Example 1 on page 958

WRAP

displays one value from each column of the report, on consecutive lines if necessary, before displaying another value from the first column. By default, PROC REPORT displays values for only as many columns as it can fit on one page. It fills a page with values for these columns before starting to display values for the remaining columns on the next page.

Restriction: This option has no effect on the HTML or Printer output.

Interaction: When WRAP is in effect, PROC REPORT ignores PAGE in any item definitions.

Tip: Typically, you use WRAP in conjunction with the NAMED option in order to avoid wrapping column headers.

Featured in: Example 7 on page 975

BREAK Statement

Produces a default summary at a break (a change in the value of a group or order variable). The information in a summary applies to a set of observations. The observations share a unique combination of values for the break variable and all other group or order variables to the left of the break variable in the report.

Featured in: Example 4 on page 967 and Example 5 on page 970.

BREAK *location break-variable</ option(s)>*;

To do this	Use this option
Specify the color of the break lines in the REPORT window	COLOR=
Double overline each value	DOL
Double underline each value	DUL
Overline each value	OL
Start a new page after the last break line	PAGE
Write a blank line for the last break line	SKIP
Specify a style element for default summary lines, customized summary lines or both	STYLE=
Write a summary line in each group of break lines	SUMMARIZE
Suppress the printing of the value of the break variable in the summary line and of any underlining or overlining in the break lines in the column containing the break variable	SUPPRESS
Underline each value	UL

Required Arguments

location
controls the placement of the break lines and is either

AFTER
places the break lines immediately after the last row of each set of rows that have the same value for the break variable.

BEFORE
places the break lines immediately before the first row of each set of rows that have the same value for the break variable.

break-variable
is a group or order variable. The REPORT procedure writes break lines each time the value of this variable changes.

Options

COLOR=*color*
specifies the color of the break lines in the REPORT window. You can use the following colors:

BLACK	MAGENTA
BLUE	ORANGE
BROWN	PINK
CYAN	RED

GRAY	WHITE
GREEN	YELLOW

Default: The color of **Foreground** in the SASCOLOR window. (For more information, see the online help for the SASCOLOR window.)

Restriction: This option has no effect on the HTML or Printer output.

Note: Not all operating environments and devices support all colors, and on some operating systems and devices, one color may map to another color. For example, if the DEFINITION window displays the word BROWN in yellow characters, selecting BROWN results in a yellow item. △

DOL
(for double overlining) uses the thirteenth formatting character to overline each value

- that appears in the summary line
- that would appear in the summary line if you specified the SUMMARIZE option.

Default: equals sign (=)

Restriction: This option has no effect on the HTML or Printer output.

Interaction: If you specify both the OL and DOL options, PROC REPORT honors only OL.

See also: the discussion of FORMCHAR= on page 882.

DUL
(for double underlining) uses the thirteenth formatting character to underline each value

- that appears in the summary line
- that would appear in the summary line if you specified the SUMMARIZE option.

Default: equals sign (=)

Restriction: This option has no effect on the HTML or Printer output.

Interaction: If you specify both the UL and DUL options, PROC REPORT honors only UL.

See also: the discussion of FORMCHAR= on page 882.

OL
(for overlining) uses the second formatting character to overline each value

- that appears in the summary line
- that would appear in the summary line if you specified the SUMMARIZE option.

Default: hyphen (–)

Restriction: This option has no effect on the HTML or Printer output.

Interaction: If you specify both the OL and DOL options, PROC REPORT honors only OL.

See also: the discussion of FORMCHAR= on page 882.

Featured in: Example 2 on page 961 and Example 9 on page 979

PAGE
starts a new page after the last break line.

Interaction: If you use PAGE in the BREAK statement and you create a break at the end of the report, the summary for the whole report appears on a separate page.

Featured in: Example 9 on page 979

SKIP
writes a blank line for the last break line.

Restriction: This option has no effect on the HTML or Printer output.

Featured in: Example 2 on page 961, Example 4 on page 967, Example 5 on page 970, and Example 8 on page 977

STYLE=<*style-element-name*><[*style-attribute-specification(s)*]>
specifies the style element to use for default summary lines that are created with the BREAK statement. You can alter the default style element of the summary lines or specify another style element entirely.

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

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

Default: If you do not specify a style element, PROC REPORT uses DataEmphasis.

See also: For information about Institute-supplied style definitions, see "What Style Definitions Are Shipped with the Software?" on page 43.

For information about PROC TEMPLATE and the Output Delivery System, see *The Complete Guide to the SAS Output Delivery System*.

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

style-attribute-name=style-attribute-value

You can set these attributes:

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

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

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

SUMMARIZE
writes a summary line in each group of break lines. A summary line for a set of observations contains values for

- the break variable (which you can suppress with the SUPPRESS option)
- other group or order variables to the left of the break variable
- statistics
- analysis variables
- computed variables.

The following table shows how PROC REPORT calculates the value for each kind of report item in a summary line created by the BREAK statement:

If the report item is ...	Then its value is ...
the break variable	the current value of the variable (or a missing value if you use SUPPRESS)
a group or order variable to the left of the break variable	the current value of the variable
a group or order variable to the right of the break variable, or a display variable anywhere in the report	missing*
a statistic	the value of the statistic over all observations in the set
an analysis variable	the value of the statistic specified as the usage option in the item's definition. PROC REPORT calculates the value of the statistic over all observations in the set. The default usage is SUM.
a computed variable	the results of the calculations based on the code in the corresponding compute block (see "COMPUTE Statement" on page 905).

* If you reference a variable with a missing value in a customized summary line, PROC REPORT displays that variable as a blank (for character variables) or a period (for numeric variables).

Note: PROC REPORT cannot create groups in a report that contains order or display variables. △

Featured in: Example 2 on page 961, Example 4 on page 967, and Example 9 on page 979

SUPPRESS
suppresses printing of

- the value of the break variable in the summary line
- any underlining and overlining in the break lines in the column containing the break variable.

Interaction: If you use SUPPRESS, the value of the break variable is unavailable for use in customized break lines unless you assign a value to it in the compute block associated with the break (see "COMPUTE Statement" on page 905).

Featured in: Example 4 on page 967

UL
(for underlining) uses the second formatting character to underline each value

- that appears in the summary line
- that would appear in the summary line if you specified the SUMMARIZE option.

Default: hyphen (–)

Restriction: This option has no effect on the HTML or Printer output.

Interaction: If you specify both the UL and DUL options, PROC REPORT honors only UL.

See also: the discussion of FORMCHAR= on page 882.

Order of Break Lines

When a default summary contains more than one break line, the order in which the break lines appear is

1. overlining or double overlining (OL or DOL)
2. summary line (SUMMARIZE)
3. underlining or double underlining (UL or DUL)
4. skipped line (SKIP)
5. page break (PAGE).

Note: If you define a customized summary for the break, customized break lines appear after underlining or double underlining. For more information about customized break lines, see "COMPUTE Statement" on page 905 and "LINE Statement" on page 917. △

BY Statement

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

Restriction: If you use the BY statement, you must use the NOWINDOWS option in the PROC REPORT statement.

Restriction: You cannot use the OUT= option when you use a BY statement.

Interaction: If you use the RBREAK statement in a report that uses BY processing, PROC REPORT creates a default summary for each BY group. In this case, you cannot summarize information for the whole report.

Tip: Using the BY statement does not make the FIRST. and LAST. variables available in compute blocks.

Main discussion: "BY" on page 68

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

Required Arguments

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

Options

DESCENDING
specifies that the data set is sorted in descending order by the variable that immediately follows the word DESCENDING in the BY statement.

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

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

CALL DEFINE Statement

Sets the value of an attribute for a particular column in the current row.

Restriction: Valid only in a compute block that is attached to a report item.

Featured in: Example 4 on page 967

CALL DEFINE (*column-id*, '*attribute-name*', *value*);

The CALL DEFINE statement is often used to write report definitions that other people will use in a windowing environment. Only the FORMAT, URL, URLBP, and URLP attributes have an effect in the nonwindowing environment. In fact, URL, URLBP, and URLP are effective only in the nonwindowing environment. The STYLE= attribute is effective only when you are using the Output Delivery System to create HTML or Printer output. (See Table 32.7 on page 900 for descriptions of the available attributes.)

Required Arguments

column-id
specifies a column name or a column number. A column ID can be one of the following:

- a character literal (in quotation marks)
- a character expression
- a numeric literal
- a numeric expression
- a name of the form _C*n*_, where *n* is the column number
- the automatic variable _COL_. This variable identifies the column containing the report item that the compute block is attached to.

attribute-name
is the attribute to define. For attribute names, refer to Table 32.7 on page 900.

value
sets the value for the attribute. For values for each attribute, refer to Table 32.7 on page 900.

Table 32.7 Attribute Descriptions

Attribute	Description	Values	Affects
BLINK	Controls blinking of current value	1 turns blinking on; 0 turns it off	windowing environment
COLOR	Controls the color of the current value in the REPORT window	'blue', 'red', 'pink', 'green', 'cyan', 'yellow', 'white', 'orange', 'black', 'magenta', 'gray', 'brown'	windowing environment
COMMAND	Specifies that a series of commands follows	a quoted string of SAS commands to submit to the command line	windowing environment
FORMAT	Specifies a format for the column	a SAS format or a user-defined format	windowing and nonwindowing environments
HIGHLIGHT	Controls highlighting of the current value	1 turns highlighting on; 0 turns it off	windowing environment
RVSVIDEO	Controls display of the current value	1 turns reverse video on; 0 turns it off	windowing environment
STYLE=	Specifies the style element for the Output Delivery System	See "Using the STYLE= Attribute" on page 901	HTML and Printer output
URL	Makes the contents of each cell of the column a link to the specified Uniform Resource Locator (URL)*	a quoted URL (either single or double quotation marks can be used)	HTML output

Attribute	Description	Values	Affects
URLBP	Makes the contents of each cell of the column a link. The link points to a Uniform Resource Locator that is a concatenation of **1** the string that is specified by the BASE= option in the ODS HTML statement **2** the string that is specified by the PATH= option in the ODS HTML statement **3** the value of the URLBP attribute*,#	a quoted URL (either single or double quotation marks can be used)	HTML output
URLP	Makes the contents of each cell of the column a link. The link points to a Uniform Resource Locator that is a concatenation of **1** the string that is specified by the PATH= option in the ODS HTML statement **2** the value of the URLP attribute*,#	a quoted URL (either single or double quotation marks can be used)	HTML output

* The total length of the URL that you specify (including any characters that come from the BASE= and PATH= options) cannot exceed the line size. Use the LS= option in the PROC REPORT statement to alter the line size for the PROC REPORT step. (See the discussion of LS= on page 885.)

\# For information on the BASE= and PATH= options, see the documentation for the ODS HTML statement in *The Complete Guide to the SAS Output Delivery System*.

Note: The attributes BLINK, HIGHLIGHT, and RVSVIDEO do not work on all devices. △

Using the STYLE= Attribute

The STYLE= attribute specifies the style element to use in the cells that are affected by the CALL DEFINE statement.

The STYLE= attribute functions like the STYLE= option in other statements in PROC REPORT. However, instead of acting as an option in a statement, it becomes the value for the STYLE= attribute. For instance, the following CALL DEFINE statement sets the background color to yellow and the font size to 7:

```
call define(_col_, "style",
            "style=[background=yellow font_size=7]");
```

The general form for the value of the style attribute is

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

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

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

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

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

style-attribute-specification
describes one or more style attributes to change. Each *style-attribute-specification* has this general form:

style-attribute-name=style-attribute-value

You can set the following style attributes in the CALL DEFINE statement:

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

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

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

Interaction: If you set a style element for the CALLDEF location in the PROC REPORT statement and you want to use that exact style element in a CALL DEFINE statement, use an empty string as the value for the STYLE attribute, as shown here:

```
call define (_col_, "STYLE", "" );
```

Featured in: Example 16 on page 999

COLUMN Statement

Describes the arrangement of all columns and of headers that span more than one column.

Restriction: You cannot use the COLUMN statement if you use REPORT= in the PROC REPORT statement.

Featured in: Example 1 on page 958, Example 3 on page 964, Example 5 on page 970, Example 6 on page 973, Example 10 on page 983, and Example 11 on page 986

COLUMN *column-specification(s)*;

Required Arguments

column-specification(s)
is one or more of the following:

- *report-item(s)*
- *report-item-1*, *report-item-2* <. . . , *report-item-n*>
- (‘*header-1* ’< . . . ‘*header-n* ’> *report-item(s)*)
- *report-item=name*

where *report-item* is the name of a data set variable, a computed variable, or a statistic. Available statistics are

N	CSS
NMISS	STDERR
MEAN	CV
STD	T
MIN	PRT
MAX	VAR
RANGE	SUMWGT
SUM	PCTN
USS	PCTSUM

For definitions of these statistics, see “Keywords and Formulas” on page 1458. To compute standard error and the Student’s *t* test you must use the default value of VARDEF= which is DF.

report-item(s)
identifies items that each form a column in the report.

Featured in: Example 1 on page 958 and Example 11 on page 986

report-item-1, *report-item-2* <. . . , *report-item-n*>
identifies report items that collectively determine the contents of the column or columns. These items are said to be stacked in the report because each item generates a header, and the headers are stacked one above the other. The header for the leftmost item is on top. If one of the items is an analysis variable, a

computed variable, or a statistic, its values fill the cells in that part of the report. Otherwise, PROC REPORT fills the cells with frequency counts.

If you stack a statistic with an analysis variable, the statistic that you name in the column statement overrides the statistic in the definition of the analysis variable. For example, the following PROC REPORT step produces a report that contains the minimum value of Sales for each sector:

```
proc report data=grocery;
   column sector sales,min;
   define sector/group;
   define sales/analysis sum;
run;
```

Interaction: A series of stacked report items can include only one analysis variable or statistic. If you include more than one, PROC REPORT returns an error because it cannot determine which values to put in the cells of the report.

Tip: You can use parentheses to group report items whose headers should appear at the same level rather than stacked one above the other.

Featured in: Example 5 on page 970, Example 6 on page 973, and Example 10 on page 983

(*'header-1 '< . . . 'header-n '> report-item(s)*)
creates one or more headers that span multiple columns.

header
is a string of characters that spans one or more columns in the report. PROC REPORT prints each header on a separate line. You can use split characters in a header to split one header over multiple lines. See the discussion of SPLIT= on page 889.

In traditional (monospace) SAS output, if the first and last characters of a header are one of the following characters, PROC REPORT uses that character to expand the header to fill the space over the column or columns:

```
:- = \_ .* +
```

Similarly, if the first character of a header is < and the last character is >, or vice-versa, PROC REPORT expands the header to fill the space over the column by repeating the first character before the text of the header and the last character after it.

report-item(s)
specifies the columns to span.

Featured in: Example 10 on page 983

report-item=name
specifies an alias for a report item. You can use the same report item more than once in a COLUMN statement. However, you can use only one DEFINE statement for any given name. (The DEFINE statement designates characteristics such as formats and customized column headers. If you omit a DEFINE statement for an item, the REPORT procedure uses defaults.) Assigning an alias in the COLUMN statement does not by itself alter the report. However, it does enable you to use separate DEFINE statements for each occurrence of a variable or statistic.

Featured in: Example 3 on page 964

CAUTION:
You cannot always use an alias. When you refer in a compute block to a report item that has an alias, you must usually use the alias. However, if the report item shares

a column with an across variable, you must reference it by column number (see "Four Ways to Reference Report Items in a Compute Block" on page 872). △

COMPUTE Statement

Starts a *compute block*. A compute block contains one or more programming statements that PROC REPORT executes as it builds the report.

Interaction: An ENDCOMP statement must mark the end of the group of statements in the compute block.

Featured in: Example 2 on page 961, Example 3 on page 964, Example 4 on page 967, Example 5 on page 970, Example 9 on page 979, and Example 10 on page 983

COMPUTE *location* <*target*>
</ STYLE=<*style-element-name*>
<[*style-attribute-specification(s)*]>>;
LINE *specification(s)*;
. . . *select SAS language elements* . . .
ENDCOMP;

COMPUTE *report-item* </ *type-specification*>;
CALL DEFINE (*column-id*, '*attribute-name*', *value*);
. . . *select SAS language elements* . . .
ENDCOMP;

A compute block can be associated with a report item or with a location (at the top or bottom of a report; at the top or bottom of a page; before or after a set of observations). You create a compute block with the COMPUTE window or with the COMPUTE statement. One form of the COMPUTE statement associates the compute block with a report item. Another form associates the compute block with a location.

For a list of the SAS language elements that you can use in compute blocks, see "The Contents of Compute Blocks" on page 872.

Required Arguments

You must specify either a location or a report item in the COMPUTE statement.

location
determines where the compute block executes in relation to *target*.

AFTER
executes the compute block at a break in one of the following places:

- immediately after the last row of a set of rows that have the same value for the variable that you specify as *target* or, if there is a default summary on that variable, immediately after the creation of the preliminary summary line (see "How PROC REPORT Builds a Report" on page 946).
- near the bottom of each page, immediately before any footnotes, if you specify _PAGE_ as *target*
- at the end of the report if you omit a target.

BEFORE
executes the compute block at a break in one of the following places:

- immediately before the first row of a set of rows that have the same value for the variable that you specify as *target* or, if there is a default summary on that variable, immediately after the creation of the preliminary summary line (see "How PROC REPORT Builds a Report" on page 946).
- near the top of each page, between any titles and the column headers, if you specify _PAGE_ as *target*
- immediately before the first detail row if you omit a target.

Featured in: Example 3 on page 964 and Example 9 on page 979

report-item
specifies a data set variable, a computed variable, or a statistic to associate the compute block with. If you are working in the nonwindowing environment, you must include the report item in the COLUMN statement. If the item is a computed variable, you must include a DEFINE statement for it.

Featured in: Example 4 on page 967 and Example 5 on page 970

CAUTION:
The position of a computed variable is important. PROC REPORT assigns values to the columns in a row of a report from left to right. Consequently, you cannot base the calculation of a computed variable on any variable that appears to its right in the report. △

Options

STYLE=<*style-element-name*><[*style-attribute-specification(s)*]>
specifies the style to use for the text that is created by any LINE statements in this compute block.

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

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

Default: If you do not specify a style element, PROC REPORT uses NoteContent.

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

style-attribute-specification
describes one or more style attributes to change. Each *style-attribute-specification* has this general form:

style-attribute-name=style-attribute-value

You can set the following style attributes in the STYLE= option in the COMPUTE statement:

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

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

Featured in: Example 16 on page 999

target

controls when the compute block executes. If you specify a location (BEFORE or AFTER) for the COMPUTE statement, you can also specify *target*, which can be one of the following:

break-variable

is a group or order variable.

When you specify a break variable, PROC REPORT executes the statements in the compute block each time the value of the break variable changes.

PAGE </ justification>

causes the compute block to execute once for each page, either immediately after printing any titles or immediately before printing any footnotes. *justification* controls the placement of text and values. It can be one of the following:

CENTER	centers each line that the compute block writes.
LEFT	left-justifies each line that the compute block writes.
RIGHT	right-justifies each line that the compute block writes.

Default: CENTER

Featured in: Example 9 on page 979

type-specification

specifies the type and, optionally, the length of *report-item*. If the report item associated with a compute block is a computed variable, PROC REPORT assumes that it is a numeric variable unless you use a type specification to specify that it is a character variable. A type specification has the form

CHARACTER <LENGTH=*length*>

where

CHARACTER
specifies that the computed variable is a character variable. If you do not specify a length, the variable's length is 8.

Alias: CHAR

Featured in: Example 10 on page 983

LENGTH=*length*
specifies the length of a computed character variable.

Default: 8

Range: 1 to 200

Interaction: If you specify a length, you must use CHARACTER to indicate that the computed variable is a character variable.

Featured in: Example 10 on page 983

DEFINE Statement

Describes how to use and display a report item.

Tip: If you do not use a DEFINE statement, PROC REPORT uses default characteristics.

Featured in: Example 2 on page 961, Example 3 on page 964, Example 4 on page 967, Example 5 on page 970, Example 6 on page 973, Example 9 on page 979, and Example 10 on page 983

DEFINE *report-item* / <*usage*>
<*attribute(s)*>
<*option(s)*>
<*justification*>
<COLOR=*color*>
<'*column-header-1*' <…'*column-header-n*'>>
<*style*>;

To do this	Use this option
Specify how to use a report item (see "Usage of Variables in a Report" on page 867)	
Define the item, which must be a data set variable, as an across variable	ACROSS
Define the item, which must be a data set variable, as an analysis variable	ANALYSIS
Define the item as a computed variable	COMPUTED
Define the item, which must be a data set variable, as a display variable	DISPLAY
Define the item, which must be a data set variable, as a group variable	GROUP

To do this	Use this option
Define the item, which must be a data set variable, as an order variable	ORDER
Specify style attribute s for a report item	
Assign a SAS or user-defined format to the item	FORMAT=
Reference a HELP or CBT entry that contains Help information for the report item	ITEMHELP=
Order the values of a group, order, or across variable according to the specified order	ORDER=
Define the number of blank characters to leave between the column being defined and the column immediately to its left	SPACING=
Associate a statistic with an analysis variable	
Define the width of the column in which PROC REPORT displays the report item	WIDTH=
Specify options for a report item	
Reverse the order in which PROC REPORT displays rows or values of a group, order, or across variable	DESCENDING
Wrap the value of a character variable in its column	FLOW
Specify that the item that you are defining is an ID variable	ID
Suppress the display of the report item	NOPRINT
Suppress the display of the report item if its values are all zero or missing	NOZERO
Insert a page break just before printing the first column containing values of the report item	PAGE
Control the placement of values and column headers	
Center the formatted values of the report item within the column width and center the column header over the values	CENTER
Left-justify the formatted values of the report item within the column width and left-justify the column headers over the values	LEFT
Right-justify the formatted values of the report item within the column width and right-justify the column headers over the values	RIGHT
Specify the color in the REPORT window of the column header and of the values of the item that you define	COLOR=
Define the column header for the report item	*column-header*
Specify a style element (for the Output Delivery System) for the report item	STYLE=

Required Arguments

report-item
specifies the name or alias (established in the COLUMN statement) of the data set variable, computed variable, or statistic to define.

Note: Do not specify a usage option in the definition of a statistic. The name of the statistic tells PROC REPORT how to use it. △

Options

ACROSS
defines *item*, which must be a data set variable, as an across variable. (See "Across Variables" on page 868.)

Featured in: Example 5 on page 970

ANALYSIS
defines *item*, which must be a data set variable, as an analysis variable. (See "Analysis Variables" on page 868.)

By default, PROC REPORT calculates the Sum statistic for an analysis variable. Specify an alternate statistic with the *statistic* option in the DEFINE statement.

Note: Naming a statistic in the DEFINE statement implies the ANALYSIS option, so you never need to specify ANALYSIS. However, specifying ANALYSIS may make your code easier for novice users to understand. △

Featured in: Example 2 on page 961, Example 3 on page 964, and Example 4 on page 967

CENTER
centers the formatted values of the report item within the column width and centers the column header over the values. This option has no effect on the CENTER option in the PROC REPORT statement, which centers the report on the page.

COLOR=*color*
specifies the color in the REPORT window of the column header and of the values of the item that you are defining. You can use the following colors:

BLACK	MAGENTA
BLUE	ORANGE
BROWN	PINK
CYAN	RED
GRAY	WHITE
GREEN	YELLOW

Default: The color of **Foreground** in the SASCOLOR window. (For more information, see the online Help for the SASCOLOR window.)

Restriction: This option has no effect on the HTML or Printer output.

Note: Not all operating environments and devices support all colors, and in some operating environments and devices, one color may map to another color. For example, if the DEFINITION window displays the word BROWN in yellow characters, selecting BROWN results in a yellow item. △

column-header
defines the column header for the report item. Enclose each header in single or double quotation marks. When you specify multiple column headers, PROC REPORT

uses a separate line for each one. The split character also splits a column header over multiple lines.

In traditional (monospace) SAS output, if the first and last characters of a header are one of the following characters, PROC REPORT uses that character to expand the header to fill the space over the column:

```
:– = \_ .* +
```

Similarly, if the first character of a header is < and the last character is >, or vice-versa, PROC REPORT expands the header to fill the space over the column by repeating the first character before the text of the header and the last character after it.

Default:

Item	Header
variable without a label	variable name
variable with a label	variable label
statistic	statistic name

Tip: If you want to use names when labels exist, submit the following SAS statement before invoking PROC REPORT:

```
options nolabel;
```

Tip: HEADLINE underlines all column headers and the spaces between them. In traditional (monospace) SAS outut, you can underline column headers without underlining the spaces between them, by using the special characters `'--'` as the last line of each column header instead of using HEADLINE (see Example 4 on page 967).

See also: SPLIT= on page 889

Featured in: Example 3 on page 964, Example 4 on page 967, and Example 5 on page 970

COMPUTED

defines the specified item as a computed variable. Computed variables are variables that you define for the report. They are not in the input data set, and PROC REPORT does not add them to the input data set.

In the windowing environment, you add a computed variable to a report from the COMPUTED VAR window.

In the nonwindowing environment, you add a computed variable by

- including the computed variable in the COLUMN statement
- defining the variable's usage as COMPUTED in the DEFINE statement
- computing the value of the variable in a compute block associated with the variable.

Featured in: Example 5 on page 970 and Example 10 on page 983

DESCENDING

reverses the order in which PROC REPORT displays rows or values of a group, order, or across variable.

Tip: By default, PROC REPORT orders group, order, and across variables by their formatted values. Use the ORDER= option in the DEFINE statement to specify an alternate sort order.

DISPLAY

defines *item*, which must be a data set variable, as a display variable.

FLOW
wraps the value of a character variable in its column. The FLOW option honors the split character. If the text contains no split character, PROC REPORT tries to split text at a blank.

Restriction: This option has no effect on the HTML or Printer output.

Featured in: Example 10 on page 983

FORMAT=*format*
assigns a SAS or user-defined format to the item. This format applies to *item* as PROC REPORT displays it; the format does not alter the format associated with a variable in the data set. For data set variables, PROC REPORT honors the first of these formats that it finds:

- the format assigned with FORMAT= in the DEFINE statement
- the format assigned in a FORMAT statement when you invoke PROC REPORT
- the format associated with the variable in the data set.

If none of these is present, PROC REPORT uses BEST*w*. for numeric variables and $*w*. for character variables. The value of *w* is the default column width. For character variables in the input data set, the default column width is the variable's length. For numeric variables in the input data set and for computed variables (both numeric and character), the default column width is the value specified by COLWIDTH= in the PROC REPORT statement or in the ROPTIONS window.

In the windowing environment, if you are unsure what format to use, type a question mark (?) in the format field in the DEFINITION window to access the FORMATS window.

Featured in: Example 2 on page 961 and Example 6 on page 973

GROUP
defines *item*, which must be a data set variable, as a group variable. (See "Group Variables" on page 868.)

Featured in: Example 4 on page 967, Example 6 on page 973, and Example 14 on page 994

ID
specifies that the item that you are defining is an ID variable. An ID variable and all columns to its left appear at the left of every page of a report. ID ensures that you can identify each row of the report when the report contains more columns than will fit on one page.

Featured in: Example 6 on page 973

ITEMHELP=*entry-name*
references a HELP or CBT entry that contains help information for the report item. Use PROC BUILD in SAS/AF software to create a HELP or CBT entry for a report item. All HELP and CBT entries for a report must be in the same catalog, and you must specify that catalog with the HELP= option in the PROC REPORT statement or from the **`User Help`** fields in the ROPTIONS window.

Of course, you can access these entries only from a windowing environment. To access a Help entry from the report, select the item and issue the HELP command. PROC REPORT first searches for and displays an entry named *entry-name*.CBT. If no such entry exists, it searches for *entry-name*.HELP. If neither a CBT nor a HELP entry for the selected item exists, the opening frame of the Help for PROC REPORT is displayed.

LEFT
left-justifies the formatted values of the report item within the column width and left-justifies the column headers over the values. If the format width is the same as the width of the column, the LEFT option has no effect on the placement of values.

NOPRINT
suppresses the display of the report item. Use this option

- if you do not want to show the item in the report but you need to use its values to calculate other values that you use in the report
- to establish the order of rows in the report
- if you do not want to use the item as a column but want to have access to its values in summaries (see Example 9 on page 979).

Interaction: Even though the columns that you define with NOPRINT do not appear in the report, you must count them when you are referencing columns by number (see "Four Ways to Reference Report Items in a Compute Block" on page 872).

Interaction: SHOWALL in the PROC REPORT statement or the ROPTIONS window overrides all occurrences of NOPRINT.

Featured in: Example 3 on page 964, Example 9 on page 979, and Example 12 on page 989

NOZERO
suppresses the display of the report item if its values are all zero or missing.

Interaction: Even though the columns that you define with NOZERO do not appear in the report, you must count them when you are referencing columns by number (see "Four Ways to Reference Report Items in a Compute Block" on page 872).

Interaction: SHOWALL in the PROC REPORT statement or in the ROPTIONS window overrides all occurrences of NOZERO.

ORDER
defines *item*, which must be a data set variable, as an order variable. (See "Order Variables" on page 867.)

Featured in: Example 2 on page 961

ORDER=DATA | FORMATTED | FREQ | INTERNAL
orders the values of a group, order, or across variable according to the specified order, where

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

FORMATTED
orders values by their formatted (external) values. By default, the order is ascending.

FREQ
orders values by ascending frequency count.

INTERNAL
orders values by their unformatted values, which yields the same order that PROC SORT would yield. This order is operating environment-dependent. This sort sequence is particularly useful for displaying dates chronologically.

Default: FORMATTED

Interaction: DESCENDING in the item's definition reverses the sort sequence for an item.

Featured in: Example 2 on page 961

CAUTION:
Default for the ORDER= Option. In other SAS procedures, the default is ORDER=INTERNAL. The default for the option in PROC REPORT may change in

a future release to be consistent with other procedures. Therefore, in production jobs where it is important to order report items by their formatted values, specify ORDER=FORMATTED even though it is currently the default. Doing so ensures that PROC REPORT will continue to produce the reports you expect even if the default changes. △

PAGE
inserts a page break just before printing the first column containing values of the report item.

Interaction: PAGE is ignored if you use WRAP in the PROC REPORT statement or in the ROPTIONS window.

RIGHT
right-justifies the formatted values of the specified item within the column width and right-justifies the column headers over the values. If the format width is the same as the width of the column, RIGHT has no effect on the placement of values.

SPACING=*horizontal-positions*
defines the number of blank characters to leave between the column being defined and the column immediately to its left. For each column, the sum of its width and the blank characters between it and the column to its left cannot exceed the line size.

Default: 2

Restriction: This option has no effect on the HTML or Printer output.

Interaction: When PROC REPORT's CENTER option is in effect, PROC REPORT ignores spacing that precedes the leftmost variable in the report.

Interaction: SPACING= in an item's definition overrides the value of SPACING= in the PROC REPORT statement or in the ROPTIONS window.

statistic
associates a statistic with an analysis variable. You must associate a statistic with every analysis variable in its definition. PROC REPORT uses the statistic you specify to calculate values for the analysis variable for the observations represented by each cell of the report. You cannot use *statistic* in the definition of any other kind of variable.

Default: SUM

Note: PROC REPORT uses the name of the analysis variable as the default header for the column. You can customize the column header with the *column-header* option in the DEFINE statement. △

Use one of the following values for *statistic*:

N	CSS
NMISS	STDERR
MEAN	CV
STD	T
MIN	PRT
MAX	VAR
RANGE	SUMWGT
SUM	PCTN
USS	PCTSUM

Requirement: To compute standard error and the Student's *t*-test you must use the default value of VARDEF= which is DF.

See also: For definitions of these statistics, see "Keywords and Formulas" on page 1458.

Featured in: Example 2 on page 961, Example 3 on page 964, and Example 4 on page 967

STYLE<(*location(s)*)>=<*style-element-name*><[*style-attribute-specification(s)*]>
specifies the style element to use for column headers and for text inside cells for this report item.

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

location
identifies the areas of the column that the STYLE= option affects. *location* can be

COLUMN
affects all text that is inside the cells of the table for this report item.

HEADER
affects the column header for this report item.

Default: If you do not specify a location, STYLE= affects both the column header and the text in the cells.

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

Default: If you do not specify a style element, PROC REPORT uses Data for the COLUMN location and Header for the HEADER location.

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

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

style-attribute-name=style-attribute-value

You can set these style attributes:

ASIS=	FONT_WIDTH=
BACKGROUND=	HREFTARGET=
BACKGROUNDIMAGE=	HTMLCLASS=
BORDERCOLOR=	JUST=
BORDERCOLORDARK=	NOBREAKSPACE=
BORDERCOLORLIGHT=	POSTHTML=
BORDERWIDTH=	POSTIMAGE=
CELLHEIGHT=	POSTTEXT=
CELLWIDTH=	PREHTML=
FLYOVER=	PREIMAGE=

FONT=	PRETEXT=
FONT_FACE=	PROTECTSPECIALCHARS=
FONT_SIZE=	TAGATTR=
FONT_STYLE=	URL=
FONT_WEIGHT=	VJUST=

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

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

Featured in: Example 16 on page 999

WIDTH=*column-width*
defines the width of the column in which PROC REPORT displays *item*.

Default: A column width that is just large enough to handle the format. If there is no format, PROC REPORT uses the value of COLWIDTH=.

Range: 1 to the value of the SAS system option LINESIZE=

Restriction: This option has no effect on the HTML or Printer output.

Interaction: WIDTH= in an item definition overrides the value of COLWIDTH= in the PROC REPORT statement or the ROPTIONS window.

Tip: When you stack items in the same column in a report, the width of the item that is at the bottom of the stack determines the width of the column.

Featured in: Example 10 on page 983

ENDCOMP Statement

Marks the end of one or more programming statements that PROC REPORT executes as it builds the report.

Restriction: A COMPUTE statement must precede the ENDCOMP statement.

ENDCOMP;

See also: COMPUTE statement

Featured in: Example 2 on page 961

FREQ Statement

Treats observations as if they appear multiple times in the input data set.

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

See also: For an example that uses the FREQ statement, see "Example" on page 71

FREQ *variable*;

Required Arguments

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

Frequency Information Is Not Saved

When you store a report definition, PROC REPORT does not store the FREQ statement.

LINE Statement

Provides a subset of the features of the PUT statement for writing customized summaries.

Restriction: This statement is valid only in a compute block that is associated with a location in the report.

Restriction: You cannot use the LINE statement in conditional statements (IF-THEN, IF-THEN/ELSE, and SELECT) because it does not take effect until PROC REPORT has executed all other statements in the compute block.

Featured in: Example 2 on page 961, Example 3 on page 964, and Example 9 on page 979

LINE *specification(s)*;

Required Arguments

specification(s)
can have one of the following forms. You can mix different forms of specifications in one LINE statement.

item item-format
specifies the item to display and the format to use to display it, where

item
is the name of a data set variable, a computed variable, or a statistic in the report. For information about referencing report items see "Four Ways to Reference Report Items in a Compute Block" on page 872.

item-format
is a SAS or user-defined format. You must specify a format for each item.

Featured in: Example 2 on page 961

'character-string '
specifies a string of text to display. When the string is a blank and nothing else is in *specification(s)*, PROC REPORT prints a blank line.

Featured in: Example 2 on page 961

number-of-repetitions'character-string '*
specifies a character string and the number of times to repeat it.

Featured in: Example 3 on page 964

pointer-control
specifies the column in which PROC REPORT displays the next specification. You can use either of the following forms for pointer controls:

@*column-number*
specifies the number of the column in which to begin displaying the next item in the specification list.

+*column-increment*
specifies the number of columns to skip before beginning to display the next item in the specification list.

Both *column-number* and *column-increment* can be either a variable or a literal value.

Restriction: The pointer controls are designed for monospace output. They have no effect on the HTML or Printer output. Do not use pointer controls if you are writing to the HTML or Printer destination.

Featured in: Example 3 on page 964 and Example 5 on page 970

Differences between the LINE and PUT Statements

The LINE statement does not support the following features of the PUT statement:

- automatic labeling signaled by an equals sign (=), also known as named output
- the _ALL_, _INFILE_, and _PAGE_ arguments and the OVERPRINT option
- grouping items and formats to apply one format to a list of items
- pointer control using expressions
- line pointer controls (# and /)
- trailing at signs (@ and @@)
- format modifiers
- array elements.

RBREAK Statement

Produces a default summary at the beginning or end of a report or at the beginning and end of each BY group.

Featured in: Example 1 on page 958 and Example 10 on page 983

RBREAK *location* </ *option(s)*>;

To do this	Use this option
Specify the color of the break lines in the REPORT window	COLOR=
Double overline each value	DOL
Double underline each value	DUL
Overline each value	OL
Start a new page after the last break line of a break located at the beginning of the report	PAGE
Write a blank line for the last break line of a break located at the beginning of the report	SKIP
Specify a style element (for the Output Delivery System) for default summary lines, customized summary lines, or both	STYLE=
Include a summary line as one of the break lines	SUMMARIZE
Underline each value	UL

Required Arguments

location
controls the placement of the break lines and is either of the following:

AFTER
places the break lines at the end of the report.

BEFORE
places the break lines at the beginning of the report.

Options

COLOR=*color*
specifies the color of the break lines in the REPORT window. You can use the following colors:

BLACK	MAGENTA
BLUE	ORANGE
BROWN	PINK
CYAN	RED
GRAY	WHITE
GREEN	YELLOW

Default: The color of **`Foreground`** in the SASCOLOR window. (For more information, see the online Help for the SASCOLOR window.)

Restriction: This option has no effect on the HTML or Printer output.

Note: Not all operating environments and devices support all colors, and in some operating environments and devices, one color may map to another color. For example, if the DEFINITION window displays the word BROWN in yellow characters, selecting BROWN results in a yellow item. △

DOL
(for double overlining) uses the thirteenth formatting character to overline each value

- that appears in the summary line
- that would appear in the summary line if you specified the SUMMARIZE option.

Default: equals sign (=)

Restriction: This option has no effect on the HTML or Printer output.

Interaction: If you specify both the OL and DOL options, PROC REPORT honors only OL.

See also: the discussion of FORMCHAR= on page 882.

Featured in: Example 1 on page 958

DUL
(for double underlining) uses the thirteenth formatting character to underline each value

- that appears in the summary line
- that would appear in the summary line if you specified the SUMMARIZE option.

Default: equals sign (=)

Restriction: This option has no effect on the HTML or Printer output.

Interaction: If you specify both the UL and DUL options, PROC REPORT honors only UL.

See also: the discussion of FORMCHAR= on page 882.

OL
(for overlining) uses the second formatting character to overline each value

- that appears in the summary line
- that would appear in the summary line if you specified the SUMMARIZE option.

Default: hyphen (–)

Restriction: This option has no effect on the HTML or Printer output.

Interaction: If you specify both the OL and DOL options, PROC REPORT honors only OL.

See also: the discussion of FORMCHAR= on page 882.

Featured in: Example 10 on page 983

PAGE
starts a new page after the last break line of a break located at the beginning of the report.

SKIP
writes a blank line after the last break line of a break located at the beginning of the report.

Restriction: This option has no effect on the HTML or Printer output.

STYLE=<*style-element-name*><[*style-attribute-specification(s)*]>
specifies the style element to use for default summary lines that are created with the RBREAK statement. You can alter the default style element of the summary lines or specify another style element entirely.

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

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

Default: If you do not specify a style element, PROC REPORT uses DataEmphasis.

See also: For information about Institute-supplied style definitions, see "What Style Definitions Are Shipped with the Software?" on page 43.

For information about PROC TEMPLATE and the Output Delivery System, see *The Complete Guide to the SAS Output Delivery System*.

style-attribute-specification
describes one or more style attributes to change. Each *style-attribute-specification* has this general form:

style-attribute-name=style-attribute-value

You can set these style attributes:

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

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

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

SUMMARIZE
includes a summary line as one of the break lines. A summary line at the beginning or end of a report contains values for

- statistics
- analysis variables
- computed variables.

The following table shows how PROC REPORT calculates the value for each kind of report item in a summary line created by the RBREAK statement:

If the report item is ...	Then its value is ...
a statistic	the value of the statistic over all observations in the set
an analysis variable	the value of the statistic specified as the usage option in the DEFINE statement. PROC REPORT calculates the value of the statistic over all observations in the set. The default usage is SUM.
a computed variable	the results of the calculations based on the code in the corresponding compute block (see "COMPUTE Statement" on page 905).

Featured in: Example 1 on page 958 and Example 10 on page 983

UL

(for underlining) uses the second formatting character to underline each value

- that appears in the summary line
- that would appear in the summary line if you specified the SUMMARIZE option.

Default: hyphen (–)

Restriction: This option has no effect on the HTML or Printer output.

Interaction: If you specify both the UL and DUL options, PROC REPORT honors only UL.

See also: the discussion of FORMCHAR= on page 882.

Order of Break Lines

When a default summary contains more than one break line, the order in which the break lines appear is

1. overlining or double overlining (OL or DOL)
2. summary line (SUMMARIZE)
3. underlining or double underlining (UL or DUL)
4. skipped line (SKIP)
5. page break (PAGE).

Note: If you define a customized summary for the break, customized break lines appear after underlining or double underlining. For more information about customized break lines, see "COMPUTE Statement" on page 905 and "LINE Statement" on page 917. △

WEIGHT Statement

Specifies weights for analysis variables in the statistical calculations.

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

WEIGHT *variable*;

Required Arguments

variable
specifies a numeric variable whose values weight the values of the analysis variables. The value of the variable does not have to be an integer. If the value of *variable* is

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

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

Tip: When you use the WEIGHT statement, consider which value of the VARDEF= option is appropriate. See VARDEF= on page 892 and the calculation of weighted statistics in "Keywords and Formulas" on page 1458 for more information.

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

Weight Information Is Not Saved

When you store a report definition, PROC REPORT does not store the WEIGHT statement.

PROC REPORT Windows

The windowing environment in PROC REPORT provides essentially the same functionality as the statements, with one major exception. You cannot use the Output Delivery System from the windowing environment.

BREAK

Controls PROC REPORT's actions at a change in the value of a group or order variable or at the top or bottom of a report.

Path

Edit ▶ Summarize information

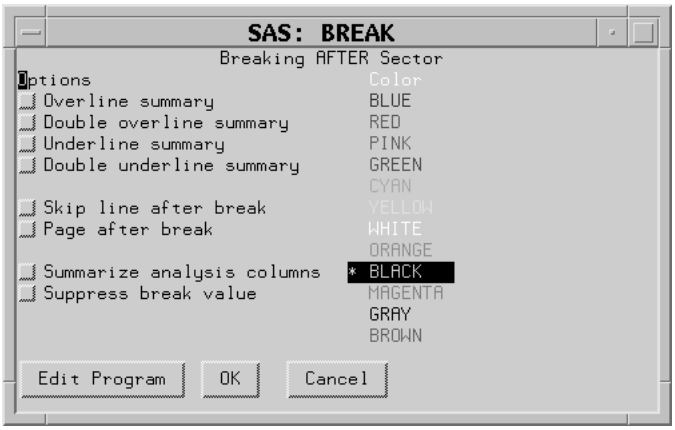
After you select **`Summarize Information`**, PROC REPORT offers you four choices for the location of the break:

- **Before Item**
- **After Item**
- **At the top**
- **At the bottom.**

After you select a location, the BREAK window opens.

Note: To create a break before or after detail lines (when the value of a group or order variable changes), you must select a variable before you open the BREAK window. △

Description



Note: For information about changing the formatting characters that are used by the line drawing options in this window, see the discussion of FORMCHAR= on page 882. △

Options

Overline summary
uses the second formatting character to overline each value

- that appears in the summary line
- that would appear in the summary line if you specified the SUMMARIZE option.

Default: hyphen (-)

Interaction: If you specify options to overline and to double overline, PROC REPORT overlines.

Double overline summary
uses the thirteenth formatting character to overline each value

- that appears in the summary line
- that would appear in the summary line if you specified the SUMMARIZE option.

Default: equals sign (=)

Interaction: If you specify options to overline and to double overline, PROC REPORT overlines.

Underline summary
uses the second formatting character to underline each value

- that appears in the summary line
- that would appear in the summary line if you specified the SUMMARIZE option.

Default: hyphen (-)

Interaction: If you specify options to underline and to double underline, PROC REPORT underlines.

Double underline summary
uses the thirteenth formatting character to underline each value

- that appears in the summary line
- that would appear in the summary line if you specified the SUMMARIZE option.

Default: equals sign (=)

Interaction: If you specify options to overline and to double overline, PROC REPORT overlines.

Skip line after break
writes a blank line for the last break line.
This option has no effect if you use it in a break at the end of a report.

Page after break
starts a new page after the last break line. This option has no effect in a break at the end of a report.

Interaction: If you use this option in a break on a variable and you create a break at the end of the report, the summary for the whole report is on a separate page.

Summarize analysis columns
writes a summary line in each group of break lines. A summary line contains values for

- statistics
- analysis variables
- computed variables.

A summary line between sets of observations also contains

- the break variable (which you can suppress with **`Suppress break value`**)
- other group or order variables to the left of the break variable.

The following table shows how PROC REPORT calculates the value for each kind of report item in a summary line created by the BREAK window:

If the report item is ...	Then its value is ...
the break variable	the current value of the variable (or a missing value if you select **`suppress break value`**)
a group or order variable to the left of the break variable	the current value of the variable
a group or order variable to the right of the break variable, or a display variable anywhere in the report	missing*
a statistic	the value of the statistic over all observations in the set
an analysis variable	the value of the statistic specified as the usage option in the item's definition. PROC REPORT calculates the value of the statistic over all observations in the set. The default usage is SUM.

If the report item is ...	Then its value is ...
a computed variable	the results of the calculations based on the code in the corresponding compute block (see "COMPUTE Statement" on page 905).

*If you reference a variable with a missing value in a customized summary line, PROC REPORT displays that variable as a blank (for character variables) or a period (for numeric variables).

Suppress break value
suppresses printing of

- the value of the break variable in the summary line
- any underlining and overlining in the break lines in the column containing the break variable.

If you select **`Suppress break value`**, the value of the break variable is unavailable for use in customized break lines unless you assign it a value in the compute block associated with the break.

Color

From the list of colors, select the one to use in the REPORT window for the column header and the values of the item that you are defining.

Default: The color of **`Foreground`** in the SASCOLOR window. (For more information, see the online Help for the SASCOLOR window.)

Note: Not all operating environments and devices support all colors, and in some operating environments and devices, one color may map to another color. For example, if the DEFINITION window displays the word BROWN in yellow characters, selecting BROWN results in a yellow item.

Pushbuttons

Edit Program
opens the COMPUTE window and enables you to associate a compute block with a location in the report.

OK
applies the information in the BREAK window to the report and closes the window.

Cancel
closes the BREAK window without applying information to the report.

COMPUTE

Attaches a compute block to a report item or to a location in the report. Use the SAS Text Editor commands to manipulate text in this window.

Path

From Edit Program in the COMPUTED VAR, DEFINITION, or BREAK window.

Description

For information about the SAS language features that you can use in the COMPUTE window, see "The Contents of Compute Blocks" on page 872.

COMPUTED VAR

Adds a variable that is not in the input data set to the report.

Path

Select a column. Then select

Edit ▶ Add Item ▶ Computed Column

After you select **`Computed Column`**, PROC REPORT prompts you for the location of the computed column relative to the column that you have selected. After you select a location, the COMPUTED VAR window opens.

Description

Enter the name of the variable at the prompt. If it is a character variable, select the **`Character data`** check box and, if you want, enter a value in the **`Length`** field. The length can be any integer between 1 and 200. If you leave the field blank, PROC REPORT assigns a length of 8 to the variable.

After you enter the name of the variable, select Edit Program to open the COMPUTE window. Use programming statements in the COMPUTE window to define the computed variable. After closing the COMPUTE and COMPUTED VAR windows, open the DEFINITION window to describe how to display the computed variable.

CAUTION:

The Position of a Computed Variable Is Important. PROC REPORT assigns values to the columns in a row of a report from left to right. Consequently, you cannot base the calculation of a computed variable on any variable that appears to its right in the report. △

DATA COLUMNS

Lists all variables in the input data set so that you can add one or more data set variables to the report.

Path

Select a report item. Then select

Edit ▶ Add Item ▶ Data Column

After you select **Data column**, PROC REPORT prompts you for the location of the computed column relative to the column that you have selected. After you select a location, the DATA COLUMNS window opens.

Description

Select one or more variables to add to the report. When you select the first variable, it moves to the top of the list in the window. If you select multiple variables, subsequent selections move to the bottom of the list of selected variables. An asterisk (*) identifies each selected variable. The order of selected variables from top to bottom determines their order in the report from left to right.

DATA SELECTION

Loads a data set into the current report definition.

Path

File ▶ Open Data Set

Description

The first list box in the DATA SELECTION window lists all the librefs defined for your SAS session. The second one lists all the SAS data sets in the selected library.

CAUTION:

Use Data Compatible with the Current Report Definition. The data set that you load must contain variables whose names are the same as the variable names in the current report definition. △

Pushbuttons

OK
loads the selected data set into the current report definition.

Cancel
closes the DATA SELECTION window without loading new data.

DEFINITION

Displays the characteristics associated with an item in the report and lets you change them.

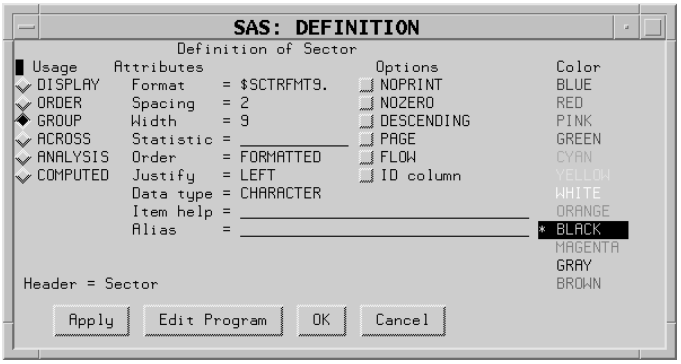
Path

Select a report item. Then select

Edit ▶ Define

Note: Alternatively, double-click on the selected item. (Not all operating environments support this method of opening the DEFINITION window.) △

Description



Usage

For an explanation of each type of usage see “Laying Out a Report” on page 866.

DISPLAY
defines the selected item as a display variable. DISPLAY is the default for character variables.

ORDER
defines the selected item as an order variable.

GROUP
defines the selected item as a group variable.

ACROSS
defines the selected item as an across variable.

ANALYSIS
defines the selected item as an analysis variable. You must specify a statistic (see the discussion of statistic= on page 930) for an analysis variable. ANALYSIS is the default for numeric variables.

COMPUTED
defines the selected item as a computed variable. Computed variables are variables that you define for the report. They are not in the input data set, and PROC REPORT does not add them to the input data set. However, computed variables are included in an output data set if you create one.

In the windowing environment, you add a computed variable to a report from the COMPUTED VAR window.

Attributes

Format=
assigns a SAS or user-defined format to the item. This format applies to the selected item as PROC REPORT displays it; the format does not alter the format associated with a variable in the data set. For data set variables, PROC REPORT honors the first of these formats that it finds:

- the format assigned with FORMAT= in the DEFINITION window
- the format assigned in a FORMAT statement when you start PROC REPORT
- the format associated with the variable in the data set.

If none of these is present, PROC REPORT uses BEST*w*. for numeric variables and $*w*. for character variables. The value of *w* is the default column width. For character variables in the input data set, the default column width is the variable's length. For numeric variables in the input data set and for computed variables (both numeric and character), the default column width is the value of the COLWIDTH= attribute in the ROPTIONS window.

If you are unsure what format to use, type a question mark (?) in the format field in the DEFINITION window to access the FORMATS window.

Spacing=
defines the number of blank characters to leave between the column being defined and the column immediately to its left. For each column, the sum of its width and the blank characters between it and the column to its left cannot exceed the line size.

Default: 2

Interaction: When PROC REPORT's CENTER option is in effect, PROC REPORT ignores spacing that precedes the leftmost variable in the report.

Interaction: SPACING= in an item definition overrides the value of SPACING= in the PROC REPORT statement or the ROPTIONS window.

Width=
defines the width of the column in which PROC REPORT displays the selected item.

Range: 1 to the value of the SAS system option LINESIZE=

Default: A column width that is just large enough to handle the format. If there is no format, PROC REPORT uses the value of COLWIDTH=.

Note: When you stack items in the same column in a report, the width of the item that is at the bottom of the stack determines the width of the column. △

Statistic=
associates a statistic with an analysis variable. You must associate a statistic with every analysis variable in its definition. PROC REPORT uses the statistic that you specify to calculate values for the analysis variable for the observations represented by each cell of the report. You cannot use *statistic* in the definition of any other kind of variable.

Default: SUM

Note: PROC REPORT uses the name of the analysis variable as the default header for the column. You can customize the column header with the **`Header`** field of the DEFINITION window. △

You can use the following values for *statistic*:

N	CSS
NMISS	STDERR
MEAN	CV
STD	T
MIN	PRT

MAX	VAR
RANGE	SUMWGT
SUM	PCTN
USS	PCTSUM

Requirement: To compute standard error and the Student's *t*-test you must use the default value of VARDEF= which is DF.

See also: For definitions of these statistics, see "Keywords and Formulas" on page 1458.

Order=
orders the values of a GROUP, ORDER, or ACROSS variable according to the specified order, where

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

FORMATTED
orders values by their formatted (external) values. By default, the order is ascending.

FREQ
orders values by ascending frequency count.

INTERNAL
orders values by their unformatted values, which yields the same order that PROC SORT would yield. This order is operating environment-dependent. This sort sequence is particularly useful for displaying dates chronologically.

Default: FORMATTED

Interaction: DESCENDING in the item's definition reverses the sort sequence for an item.

CAUTION:
Default for the ORDER= Option. In other SAS procedures, the default is ORDER=INTERNAL. The default for the option in PROC REPORT may change in a future release to be consistent with other procedures. Therefore, in production jobs where it is important to order report items by their formatted values, specify ORDER=FORMATTED even though it is currently the default. Doing so ensures that PROC REPORT will continue to produce the reports you expect even if the default changes. △

Justify=
You can justify the placement of the column header and of the values of the item that you are defining within a column in one of three ways:

LEFT
left-justifies the formatted values of the item that you are defining within the column width and left-justifies the column header over the values. If the format width is the same as the width of the column, LEFT has no effect on the placement of values.

RIGHT
right-justifies the formatted values of the item that you are defining within the column width and right-justifies the column header over the values. If the format width is the same as the width of the column, RIGHT has no effect on the placement of values.

CENTER
centers the formatted values of the item that you are defining within the column width and centers the column header over the values. This option has no effect on the setting of the SAS system option CENTER.

When justifying values, PROC REPORT justifies the field width defined by the format of the item within the column. Thus, numbers are always aligned.

Data type=
shows you if the report item is numeric or character. You cannot change this field.

Item Help=
references a HELP or CBT entry that contains help information for the selected item. Use PROC BUILD in SAS/AF software to create a HELP or CBT entry for a report item. All HELP and CBT entries for a report must be in the same catalog, and you must specify that catalog with the HELP= option in the PROC REPORT statement or from the **User Help** fields in the ROPTIONS window.

To access a help entry from the report, select the item and issue the HELP command. PROC REPORT first searches for and displays an entry named *entry-name*.CBT. If no such entry exists, PROC REPORT searches for *entry-name*.HELP. If neither a CBT nor a HELP entry for the selected item exists, the opening frame of the help for PROC REPORT is displayed.

Alias=
By entering a name in the **Alias** field, you create an alias for the report item that you are defining. Aliases let you distinguish between different uses of the same report item. When you refer in a compute block to a report item that has an alias, you must use the alias (see Example 3 on page 964).

Options

NOPRINT
suppresses the display of the item that you are defining. Use this option

- if you do not want to show the item in the report but you need to use the values in it to calculate other values you use in the report
- to establish the order of rows in the report
- if you do not want to use the item as a column but want to have access to its values in summaries (see Example 9 on page 979).

Interaction: Even though the columns that you define with NOPRINT do not appear in the report, you must count them when you are referencing columns by number (see "Four Ways to Reference Report Items in a Compute Block" on page 872).

Interaction: SHOWALL in the PROC REPORT statement or the ROPTIONS window overrides all occurrences of NOPRINT.

NOZERO
suppresses the display of the item that you are defining if its values are all zero or missing.

Interaction: Even though the columns that you define with NOZERO do not appear in the report, you must count them when you are referencing columns by number (see "Four Ways to Reference Report Items in a Compute Block" on page 872).

Interaction: SHOWALL in the PROC REPORT statement or the ROPTIONS window overrides all occurrences of NOZERO.

DESCENDING
reverses the order in which PROC REPORT displays rows or values of a group, order, or across variable.

PAGE
inserts a page break just before printing the first column containing values of the selected item.

Interaction: PAGE is ignored if you use WRAP in the PROC REPORT statement or in the ROPTIONS window.

FLOW
wraps the value of a character variable in its column. The FLOW option honors the split character. If the text contains no split character, PROC REPORT tries to split text at a blank.

ID column
specifies that the item that you are defining is an ID variable. An ID variable and all columns to its left appear at the left of every page of a report. ID ensures that you can identify each row of the report when the report contains more columns than will fit on one page.

Color

From the list of colors, select the one to use in the REPORT window for the column header and the values of the item that you are defining.

Default: The color of **`Foreground`** in the SASCOLOR window. (For more information, see the online Help for the SASCOLOR window.)

Note: Not all operating environments and devices support all colors, and in some operating environments and devices, one color may map to another color. For example, if the DEFINITION window displays the word BROWN in yellow characters, selecting BROWN results in a yellow item.

Pushbuttons

Apply
applies the information in the open window to the report and keeps the window open.

Edit Program
opens the COMPUTE window and enables you to associate a compute block with the variable that you are defining.

OK
applies the information in the DEFINITION window to the report and closes the window.

Cancel
closes the DEFINITION window without applying changes made with APPLY.

DISPLAY PAGE

Displays a particular page of the report.

Path

View ▶ Display Page

Description

You can get to the last page of the report by entering a large number for the page number. When you are on the last page of the report, PROC REPORT sends a note to the message line of the REPORT window.

EXPLORE

Lets you experiment with your data.

Restriction: You cannot open the EXPLORE window unless your report contains at least one group or order variable.

Path

Edit ▶ Explore Data

Description

In the EXPLORE window you can

- subset the data with list boxes
- suppress the display of a column with the **`Remove Column`** checkbox
- change the order of the columns with Rotate columns.

Note: The results of your manipulations in the EXPLORE window appear in the REPORT window but are not saved in report definitions. △

Window Features

list boxes

The EXPLORE window contains three list boxes. These boxes contain the value "All levels" as well as actual values for the first three group or order variables in your report. The values reflect any WHERE clause processing that is in effect. For example, if you use a WHERE clause to subset the data so that it includes only the northeast and northwest sectors, the only values that appear in the list box for Sector are **`All levels`**, **`Northeast`**, and **`Northwest`**. Selecting **`All levels`** in this case displays rows of the report for only the northeast and northwest sectors. To see data for all the sectors, you must clear the WHERE clause before you open the EXPLORE window.

Selecting values in the list boxes restricts the display in the REPORT window to the values that you select. If you select incompatible values, PROC REPORT returns an error.

Remove Column
Above each list box in the EXPLORE window is a check box labeled **Remove Column**. Selecting this check box and applying the change removes the column from the REPORT window. You can easily restore the column by clearing the check box and applying that change.

Pushbuttons

OK
applies the information in the EXPLORE window to the report and closes the window.

Apply
applies the information in the EXPLORE window to the report and keeps the window open.

Rotate columns
changes the order of the variables displayed in the list boxes. Each variable that can move one column to the left does; the leftmost variable moves to the third column.

Cancel
closes the EXPLORE window without applying changes made with APPLY.

FORMATS

Displays a list of formats and provides a sample of each one.

Path

From the DEFINE window, type a question mark (?) in the **Format** field and select any of the pushbuttons except **Cancel**, or press RETURN.

Description

When you select a format in the FORMATS window, a sample of that format appears in the **Sample:** field. Select the format that you want to use for the variable that you are defining.

Pushbuttons

OK
writes the format that you have selected into the **Format** field in the DEFINITION window and closes the FORMATS window. To see the format in the report, select Apply in the DEFINITION window.

Cancel
closes the FORMATS window without writing a format into the **Format** field.

LOAD REPORT

Loads a stored report definition.

Path

File ▶ Open Report

Description

The first list box in the LOAD REPORT window lists all the librefs defined for your SAS session. The second one lists all the catalogs in the selected library. The third one lists descriptions of all the stored report definitions (entry types of REPT) in the selected catalog. If there is no description for an entry, the list box contains the entry's name.

Pushbuttons

OK
loads the current data into the selected report definition.

Cancel
closes the LOAD REPORT window without loading a new report definition.

Note: Issuing the END command in the REPORT window returns you to the previous report definition (with the current data). △

MESSAGES

Automatically opens to display notes, warnings, and errors returned by PROC REPORT.

You must close the MESSAGES window by selecting OK before you can continue to use PROC REPORT.

PROFILE

Customizes some features of the PROC REPORT environment by creating a report profile.

Path

Tools ▶ Report Profile

Description

The PROFILE window creates a report profile that

- specifies the SAS library, catalog, and entry that define alternative menus to use in the REPORT and COMPUTE windows. Use PROC PMENU to create catalog

entries of type PMENU that define these menus. PMENU entries for both windows must be in the same catalog.

- sets defaults for WINDOWS, PROMPT, and COMMAND. PROC REPORT uses the default option whenever you start the procedure unless you specifically override the option in the PROC REPORT statement.

Specify the catalog that contains the profile to use with the PROFILE= option in the PROC REPORT statement (see the discussion of PROFILE= on page 888).

Pushbuttons

OK
stores your profile in a file that is called SASUSER.PROFILE.REPORT.PROFILE.

Note: Use PROC CATALOG or the EXPLORER window to copy the profile to another location. △

Cancel
closes the window without storing the profile.

PROMPTER

Prompts you for information as you add items to a report.

Path

Specify the PROMPT option when you start PROC REPORT or select PROMPT from the ROPTIONS window. The PROMPTER window opens the next time that you add an item to the report.

Description

The prompter guides you through parts of the windows most commonly used to build a report. As the content of the PROMPTER window changes, the title of the window changes to the name of the window that you would use to perform a task if you were not using the prompter. The title change is to help you begin to associate the windows with their functions and to learn what window to use if you later decide to change something.

If you start PROC REPORT with prompting, the first window gives you a chance to limit the number of observations used during prompting. When you exit the prompter, PROC REPORT removes the limit.

Pushbuttons

OK
applies the information in the open window to the report and continues the prompting process.

Note: When you select OK from the last prompt window, PROC REPORT removes any limit on the number of observations that it is working with. △

Apply
applies the information in the open window to the report and keeps the window open.

Backup
returns you to the previous PROMPTER window.

Exit Prompter
closes the PROMPTER window without applying any more changes to the report. If you have limited the number of observations to use during prompting, PROC REPORT removes the limit.

REPORT

Is the surface on which the report appears.

Path

Use WINDOWS or PROMPT in the PROC REPORT statement.

Description

You cannot write directly in any part of the REPORT window except column headers. To change other aspects of the report, you select a report item as the target of the next command and issue the command. To select an item, use a mouse or cursor keys to position the cursor over it. Then click the mouse button or press RETURN. To execute a command, make a selection from the menu bar at the top of the REPORT window. PROC REPORT displays the effect of a command immediately unless the DEFER option is on.

Note: Issuing the END command in the REPORT window returns you to the previous report definition with the current data. If there is no previous report definition, END closes the REPORT window. △

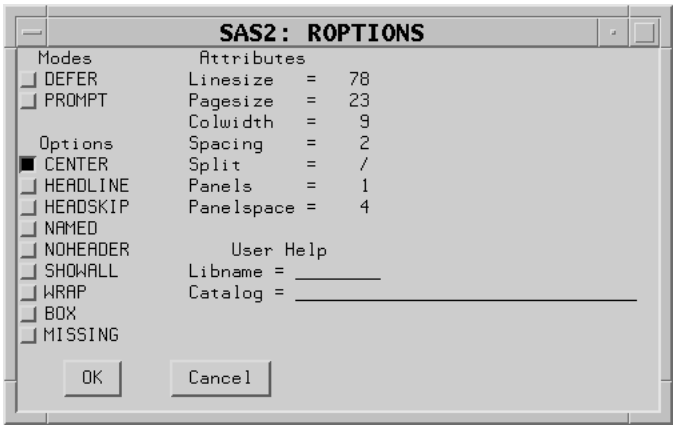
ROPTIONS

Displays choices that control the layout and display of the entire report and identifies the SAS data library and catalog containing CBT or HELP entries for items in the report.

Path

Tools ▶ Options ▶ Report

Description



Modes

DEFER
stores the information for changes and makes the changes all at once when you turn DEFER mode off or select

View ▶ Refresh

DEFER is particularly useful when you know that you need to make several changes to the report but do not want to see the intermediate reports.

By default, PROC REPORT redisplays the report in the REPORT window each time you redefine the report by adding or deleting an item, by changing information in the DEFINITION window, or by changing information in the BREAK window.

PROMPT
opens the PROMPTER window the next time that you add an item to the report.

Options

CENTER
centers the report and summary text (customized break lines). If CENTER is not selected, the report is left-justified.

PROC REPORT honors the first of these centering specifications that it finds:

- the CENTER or NOCENTER option in the PROC REPORT statement or the CENTER toggle in the ROPTIONS window
- the CENTER or NOCENTER option stored in the report definition loaded with REPORT= in the PROC REPORT statement
- the SAS system option CENTER or NOCENTER.

When PROC REPORT's CENTER option is in effect, PROC REPORT ignores spacing that precedes the leftmost variable in the report.

HEADLINE
underlines all column headers and the spaces between them at the top of each page of the report.

HEADLINE underlines with the second formatting character. (See the discussion of FORMCHAR= on page 882.)

Default: hyphen (-)

Tip: In traditional (monospace) SAS output, you can underline column headers without underlining the spaces between them, by using `'--'` as the last line of each column header instead of using HEADLINE.

HEADSKIP
writes a blank line beneath all column headers (or beneath the underlining that the HEADLINE option writes) at the top of each page of the report.

NAMED
writes *name*= in front of each value in the report, where *name* is the column header for the value.

Tip: Use NAMED in conjunction with WRAP to produce a report that wraps all columns for a single row of the report onto consecutive lines rather than placing columns of a wide report on separate pages.

Interaction: When you use NAMED, PROC REPORT automatically uses NOHEADER.

NOHEADER
suppresses column headers, including those that span multiple columns.

Once you suppress the display of column headers in the windowing environment, you cannot select any report items.

SHOWALL
overrides the parts of a definition that suppress the display of a column (NOPRINT and NOZERO). You define a report item with a DEFINE statement or in the DEFINITION window.

WRAP
displays one value from each column of the report, on consecutive lines if necessary, before displaying another value from the first column. By default, PROC REPORT displays values for only as many columns as it can fit on one page. It fills a page with values for these columns before starting to display values for the remaining columns on the next page.

Interaction: When WRAP is in effect, PROC REPORT ignores PAGE in any item definitions.

Tip: Typically, you use WRAP in conjunction with NAMED to avoid wrapping column headers.

BOX
uses formatting characters to add line-drawing characters to the report. These characters

- surround each page of the report
- separate column headers from the body of the report
- separate rows and columns from each other.

Interaction: You cannot use BOX if you use WRAP in the PROC REPORT statement or ROPTIONS window or if you use FLOW in any item's definition.

See also: For information about formatting characters, see the discussion of FORMCHAR= on page 882.

MISSING
considers missing values as valid values for group, order, or across variables. Special missing values that are used to represent numeric values (the letters A through Z and the underscore (_) character) are each considered as a different value. A group for each missing value appears in the report. If you omit the MISSING option, PROC REPORT does not include observations with a missing value for one or more group, order, or across variables in the report.

Attributes

Linesize
specifies the line size for a report. PROC REPORT honors the first of these line-size specifications that it finds:

- LS= in the PROC REPORT statement or Linesize= in the ROPTIONS window
- the LS= setting stored in the report definition loaded with REPORT= in the PROC REPORT statement
- the SAS system option LINESIZE=.

Range: 64-256 (integer)

Tip: If the line size is greater than the width of the REPORT window, use SAS windowing environment commands RIGHT and LEFT to display portions of the report that are not currently in the display.

Pagesize
specifies the page size for a report. PROC REPORT honors the first of these pagesize specifications that it finds

- PS= in the PROC REPORT statement or **`Pagesize=`** in the ROPTIONS window
- the PS= setting stored in the report definition loaded with REPORT= in the PROC REPORT statement
- the SAS system option PAGESIZE=.

Range: 15-32,767 (integer)

Colwidth
specifies the default number of characters for columns containing computed variables or numeric data set variables.

Range: 1 to the linesize

Default: 9

Interaction: When setting the width for a column, PROC REPORT first looks at WIDTH= in the definition for that column. If WIDTH= is not present, PROC REPORT uses a column width large enough to accommodate the format for the item. (For information about formats, see the discussion of Format= on page 930.) If no format is associated with the item, the column width depends on variable type:

If the variable is a ...	Then the column width is the ...
character variable in the input data set	length of the variable
numeric variable in the input data set	value of the COLWIDTH= option
computed variable (numeric or character)	value of the COLWIDTH= option

SPACING=*space-between-columns*
specifies the number of blank characters between columns. For each column, the sum of its width and the blank characters between it and the column to its left cannot exceed the line size.

Default: 2

Interaction: PROC REPORT separates all columns in the report by the number of blank characters specified by SPACING= in the PROC REPORT statement or the

ROPTIONS window unless you use SPACING= in the definition of a particular item to change the spacing to the left of that item.

Interaction: When CENTER is in effect, PROC REPORT ignores spacing that precedes the leftmost variable in the report.

SPLIT='*character*'
specifies the split character. PROC REPORT breaks a column header when it reaches that character and continues the header on the next line. The split character itself is not part of the column header although each occurrence of the split character counts toward the 40-character maximum for a label.

Default: slash (/)

Interaction: The FLOW option in the DEFINE statement honors the split character.

Note: If you are typing over a header (rather than entering one from the PROMPTER or DEFINITION window), you do not see the effect of the split character until you refresh the screen by adding or deleting an item, by changing the contents of a DEFINITION or a BREAK window, or by selecting

View ► Refresh

PANELS=*number-of-panels*
specifies the number of panels on each page of the report. If the width of a report is less than half of the line size, you can display the data in multiple sets of columns so that rows that would otherwise appear on multiple pages appear on the same page. Each set of columns is a *panel*. A familiar example of this kind of report is a telephone book, which contains multiple panels of names and telephone numbers on a single page.

When PROC REPORT writes a multipanel report, it fills one panel before beginning the next.

The number of panels that fits on a page depends on the

- width of the panel
- space between panels
- line size.

Default: 1

Tip: If *number-of-panels* is larger than the number of panels that can fit on the page, PROC REPORT creates as many panels as it can. Let PROC REPORT put your data in the maximum number of panels that can fit on the page by specifying a large number of panels (for example, 99).

See also: For information about specifying the space between panels see the discussion of PSPACE= on page 942. For information about setting the linesize, see the discussion of Linesize on page 941).

PSPACE=*space-between-panels*
specifies the number of blank characters between panels. PROC REPORT separates all panels in the report by the same number of blank characters. For each panel, the sum of its width and the number of blank characters separating it from the panel to its left cannot exceed the line size.

Default: 4

User Help
identifies the library and catalog containing user-defined help for the report. This help can be in CBT or HELP catalog entries. You can write a CBT or HELP entry for

each item in the report with the BUILD procedure in SAS/AF software. You must store all such entries for a report in the same catalog.

Specify the entry name for help for a particular report item in the DEFINITION window for that report item or in a DEFINE statement.

SAVE DATA SET

Lets you specify an output data set in which to store the data from the current report.

Path

File ▶ Save Data Set

Description

To specify an output data set, enter the name of the SAS data library and the name of the data set (called **member** in the window) that you want to create in the Save Data Set window.

Pushbuttons

OK
Creates the output data set and closes the Save Data Set window.

Cancel
Closes the Save Data Set window without creating an output data set.

SAVE DEFINITION

Saves a report definition for subsequent use with the same data set or with a similar data set.

Path

File ▶ Save Report

Description

The SAVE DEFINITION window prompts you for the complete name of the catalog entry in which to store the definition of the current report and for an optional description of the report. This description shows up in the LOAD REPORT window and helps you to select the appropriate report.

SAS stores the report definition as a catalog entry of type REPT. You can use a report definition to create an identically structured report for any SAS data set that contains variables with the same names as those used in the report definition.

Pushbuttons

OK
Creates the report definition and closes the SAVE DEFINITION window.

Cancel
Closes the SAVE DEFINITION window without creating a report definition.

SOURCE

Lists the PROC REPORT statements that build the current report.

Path

Tools ▶ Report Statements

STATISTICS

Displays statistics that are available in PROC REPORT.

Path

Edit ▶ Add item ▶ Statistic

After you select **`Statistic`**, PROC REPORT prompts you for the location of the statistic relative to the column that you have selected. After you select a location, the STATISTICS window opens.

Description

Select the statistics that you want to include in your report and close the window. When you select the first statistic, it moves to the top of the list in the window. If you select multiple statistics, subsequent selections move to the bottom of the list of selected statistics. The order of selected statistics from top to bottom determines their order in the report from left to right.

Note: If you double-click on a statistic, PROC REPORT immediately adds it to the report. The STATISTICS window remains open. △

To compute standard error and the Student's *t* test you must use the default value of VARDEF= which is DF.

WHERE

Selects observations from the data set that meet the conditions that you specify.

Path

Subset ▶ Where

Description

Enter a *where-expression* in the **Enter where clause** field. A *where-expression* is an arithmetic or logical expression that generally consists of a sequence of operands and operators. For information about constructing a *where-expression*, see the documentation of the WHERE statement in the section on statements in *SAS Language Reference: Dictionary*.

Note: You can clear all *where-expressions* by leaving the **Enter where clause** field empty and by selecting OK. △

Pushbuttons

OK
Applies the *where-expression* to the report and closes the WHERE window.

Cancel
Closes the WHERE window without altering the report.

WHERE ALSO

Selects observations from the data set that meet the conditions that you specify and any other conditions that are already in effect.

Path

Subset ▶ Where Also

Description

Enter a *where-expression* in the **Enter where clause** field. A *where-expression* is an arithmetic or logical expression that generally consists of a sequence of operands and operators. For information about constructing a *where-expression*, see the documentation of the WHERE statement in the chapter on statements in *SAS Language Reference: Dictionary*.

Pushbuttons

OK
Adds the *where-expression* to any other *where-expressions* that are already in effect and applies them all to the report. It also closes the WHERE ALSO window.

Cancel
Closes the WHERE ALSO window without altering the report.

How PROC REPORT Builds a Report

This section first explains the process of building a report. Following this explanation are illustrations of how PROC REPORT creates two sample reports. The examples use programming statements; you can construct the same reports in the windowing environment.

To understand the process of building a report, you must understand the difference between report variables and DATA step variables. Variables that appear only in one or more compute blocks are *DATA step variables*. Variables that appear in one or more columns of the report are *report variables*. A report variable may or may not appear in a compute block.

Sequence of Events

PROC REPORT constructs a report as follows:

1. It consolidates the data by group, order, and across variables. It calculates all statistics for the report, those for detail rows as well as those for summary lines in breaks. Statistics include those computed for analysis variables. PROC REPORT calculates statistics for summary lines whether or not they appear in the report. It stores all this information in a temporary file.
2. It initializes all DATA step variables to missing.
3. It begins constructing the rows of the report.
   a. At the beginning of each row, it initializes all report variables to missing.
   b. It fills in values for report variables from left to right.
      - Values for computed variables come from executing the statements in the corresponding compute blocks.
      - Values for all other variables come from the temporary file created at the beginning of the report-building process.
   c. Whenever it comes to a break, PROC REPORT first constructs the break lines created with the BREAK or RBREAK statement or with options in the BREAK window. It then executes the statements in the compute block attached to the break (if there is one).

   Note: Because of the way PROC REPORT builds a report, you can
   - use group statistics in compute blocks for a break before the group variable.
   - use statistics for the whole report in a compute block at the beginning of the report.

   This document references these statistics with the appropriate compound name. For information about referencing report items in a compute block, see “Four Ways to Reference Report Items in a Compute Block” on page 872. △

Construction of Summary Lines

PROC REPORT constructs a summary line for a break if either of the following conditions is true:

- You summarize numeric variables in the break.
- You use a compute block at the break. (You can attach a compute block to a break without using a BREAK or RBREAK statement or without selecting any options in the BREAK window.)

  For more information about using compute blocks, see "Using Compute Blocks" on page 871 and the discussion of the COMPUTE statement on page 905.

The summary line that PROC REPORT constructs at this point is preliminary. If no compute block is attached to the break, the preliminary summary line becomes the final summary line. However, if a compute block is attached to the break, the statements in the compute block can alter the values in the preliminary summary line.

PROC REPORT prints the summary line only if you summarize numeric variables in the break.

Using Compound Names

When you use a statistic in a report, you generally refer to it in compute blocks by a compound name like Sales.sum. However, in different parts of the report, that same name has different meanings. Consider the report in Output 32.1 on page 947. The statements that create the output follow. The user-defined formats that are used are created by a PROC FORMAT step on page 960.

```
libname proclib 'SAS-data-library';


options nodate pageno=1 linesize=64
        pagesize=60 fmtsearch=(proclib);
proc report data=grocery nowindows;
   column sector manager sales;
   define sector / group format=$sctrfmt.;
   define sales  / analysis sum
                   format=dollar9.2;
   define manager / group format=$mgrfmt.;
   break after sector / summarize skip ol;
   rbreak after / summarize dol dul;
   compute after;
      sector='Total:';
   endcomp;
run;
```

Output 32.1 Three Different Meanings of Sales.sum

```
                         The SAS System                          1

Sector     Manager      Sales
Northeast  Alomar       $786.00      ❶
           Andrews    $1,045.00
---------              ---------
Northeast              $1,831.00      ❷

Northwest  Brown        $598.00
           Pelfrey     $746.00
           Reveiz    $1,110.00
---------              ---------
Northwest              $2,454.00

Southeast  Jones        $630.00
           Smith        $350.00
---------              ---------
Southeast                $980.00

Southwest  Adams        $695.00
           Taylor       $353.00
---------              ---------
Southwest              $1,048.00

=========              =========
Total:                 $6,313.00      ❸
=========              =========
```

Here Sales.sum has three different meanings:

1 In detail rows, the value is the sales for one manager's store in a sector of the city. For example, the first detail row of the report shows that the sales for the store that Alomar manages were $786.00.

2 In the group summary lines, the value is the sales for all the stores in one sector. For example, the first group summary line shows that sales for the Northeast sector were $1,831.00.

3 In the report summary line, the value ($6,313.00) is the sales for all stores in the city.

CAUTION:

When to Use an Alias Unless you use the NOALIAS option in the PROC REPORT statement, when you refer in a compute block to a statistic that has an alias, you do not use a compound name. Generally, you must use the alias. However, if the statistic shares a column with an across variable, you must reference it by column number (see "Four Ways to Reference Report Items in a Compute Block" on page 872). △

Building a Report That Uses Groups and a Report Summary

The report in Output 32.2 on page 949 contains five columns:

- Sector and Department are group variables.
- Sales is an analysis variable that is used to calculate the Sum statistic.
- Profit is a computed variable whose value is based on the value of Department.
- The N statistic indicates how many observations each row represents.

At the end of the report a break summarizes the statistics and computed variables in the report and assigns to Sector the value of **TOTALS:**.

The following statements produce Output 32.2 on page 949. The user-defined formats that are used are created by a PROC FORMAT step on page 960.

```
libname proclib 'SAS-data-library';


options nodate pageno=1 linesize=64
        pagesize=60 fmtsearch=(proclib);
proc report data=grocery headline headskip;
   column sector department sales Profit N;
   define sector / group format=$sctrfmt.;
   define department   / group format=$deptfmt.;
   define sales  / analysis sum
                   format=dollar9.2;
   define profit / computed format=dollar9.2;

   compute profit;
      if department='np1' or department='np2'
         then profit=0.4*sales.sum;
      else profit=0.25*sales.sum;
   endcomp;

   rbreak after / dol dul summarize;
   compute after;
      sector='TOTALS:';
   endcomp;

   where sector contains 'n';
   title 'Report for Northeast and Northwest Sectors';
run;
```

Output 32.2 Report with Groups and a Report Summary

Report for Northeast and Northwest Sectors 1

Sector	Department	Sales	Profit	N
Northeast	Canned	$840.00	$336.00	2
	Meat/Dairy	$490.00	$122.50	2
	Paper	$290.00	$116.00	2
	Produce	$211.00	$52.75	2
Northwest	Canned	$1,070.00	$428.00	3
	Meat/Dairy	$1,055.00	$263.75	3
	Paper	$150.00	$60.00	3
	Produce	$179.00	$44.75	3
TOTALS:		$4,285.00	$1,071.25	20

A description of how PROC REPORT builds this report follows:

1 PROC REPORT starts building the report by consolidating the data (Sector and Department are group variables) and by calculating the statistics (Sales.sum and N) for each detail row and for the break at the end of the report. It stores these values in a temporary file.

2 Now, PROC REPORT is ready to start building the first row of the report. This report does not contain a break at the beginning of the report or a break before any groups, so the first row of the report is a detail row. The procedure initializes all report variables to missing, as Figure 32.9 on page 950 illustrates. Missing values for a character variable are represented by a blank, and missing values for a numeric variable are represented by a period.

Figure 32.9 First Detail Row with Values Initialized

Sector	Department	Sales	Profit	N
		.	.	.

3 Figure 32.10 on page 950 illustrates the construction of the first three columns of the row. PROC REPORT fills in values for the row from left to right. Values come from the temporary file created at the beginning of the report-building process.

Figure 32.10 First Detail Row with Values Filled in from Left to Right

Sector	Department	Sales	Profit	N
Northeast		.	.	.

Sector	Department	Sales	Profit	N
Northeast	Canned	.	.	.

Sector	Department	Sales	Profit	N
Northeast	Canned	$840.00	.	.

4 The next column in the report contains the computed variable Profit. When it gets to this column, PROC REPORT executes the statements in the compute block that is attached to Profit. Nonperishable items (which have a value of **np1** or **np2**) return a profit of 40%; perishable items (which have a value of **p1** or **p2**) return a profit of 25%.

```
if department='np1' or department='np2'
   then profit=0.4*sales.sum;
else profit=0.25*sales.sum;
```

The row now looks like Figure 32.11 on page 951.

CAUTION:

The position of a computed variable is important. PROC REPORT assigns values to the columns in a row of a report from left to right. Consequently, you cannot

base the calculation of a computed variable on any variable that appears to its right in the report. △

Figure 32.11 A Computed Variable Added to the First Detail Row

Sector	Department	Sales	Profit	N
Northeast	Canned	$840.00	$336.00	.

5 Next, PROC REPORT fills in the value for the N statistic. The value comes from the temporary file created at the beginning of the report-building process. Figure 32.12 on page 951 illustrates the completed row.

Figure 32.12 First Complete Detail Row

Sector	Department	Sales	Profit	N
Northeast	Canned	$840.00	$336.00	2

6 The procedure writes the completed row to the report.

7 PROC REPORT repeats steps 2, 3, 4, 5, and 6 for each detail row in the report.

8 At the break at the end of the report, PROC REPORT constructs the break lines described by the RBREAK statement. These lines include double underlining, double overlining, and a preliminary version of the summary line. The statistics for the summary line were calculated earlier (see step 1). The value for the computed variables is calculated when PROC REPORT reaches the appropriate column, just as it is in detail rows. PROC REPORT uses these values to create the preliminary version of the summary line (see Figure 32.13 on page 951).

Figure 32.13 Preliminary Summary Line

Sector	Department	Sales	Profit	N
		$4,285.00	$1,071.25	20

9 If no compute block is attached to the break, the preliminary version of the summary line is the same as the final version. However, in this example, a compute block is attached to the break. Therefore, PROC REPORT now executes the statements in that compute block. In this case, the compute block contains one statement:

```
sector='TOTALS:';
```

This statement replaces the value of Sector, which in the summary line is missing by default, with the word **TOTALS:**. After PROC REPORT executes the statement, it modifies the summary line to reflect this change to the value of Sector. The final version of the summary line appears in Figure 32.14 on page 952.

Figure 32.14 Final Summary Line

Sector	Department	Sales	Profit	N
TOTALS:		$4,285.00	$1,071.25	20

10 Finally, PROC REPORT writes all the break lines–underlining, overlining, and the final summary line–to the report.

Building a Report That Uses DATA Step Variables

PROC REPORT initializes report variables to missing at the beginning of each row of the report. The value for a DATA step variable is initialized to missing before PROC REPORT begins to construct the rows of the report, and it remains missing until you specifically assign a value to it. PROC REPORT retains the value of a DATA step variable from the execution of one compute block to another.

Because all compute blocks share the current values of all variables, you can initialize DATA step variables at a break at the beginning of the report or at a break before a break variable. This report initializes the DATA step variable Sctrtot at a break before Sector.

CAUTION:

Timing at Breaks. PROC REPORT creates a preliminary summary line for a break before it executes the corresponding compute block. If the summary line contains computed variables, the computations are based on the values of the contributing variables in the preliminary summary line. If you want to recalculate computed variables based on values you set in the compute block, you must do so explicitly in the compute block. This report illustrates this technique.

If no compute block is attached to a break, the preliminary summary line becomes the final summary line. △

The report in Output 32.3 on page 953 contains five columns:

- Sector and Department are group variables.
- Sales is an analysis variable that is used twice in this report: once to calculate the Sum statistic, and once to calculate the Pctsum statistic.
- Sctrpct is a computed variable whose values are based on the values of Sales and a DATA step variable, Sctrtot, which is the total sales for a sector.

At the beginning of the report, a customized report summary tells what the sales for all stores are. At a break before each group of observations for a department, a default summary summarizes the data for that sector. At the end of each group a break inserts a blank line.

The following statements produce Output 32.3 on page 953. The user-defined formats that are used are created by a PROC FORMAT step on page 960.

Note: Calculations of the percentages do not multiply their results by 100 because PROC REPORT prints them with the PERCENT. format. △

```
libname proclib 'SAS-data-library';


options nodate pageno=1 linesize=64
        pagesize=60 fmtsearch=(proclib);
proc report data=grocery noheader nowindows;
  column sector department sales
         Sctrpct sales=Salespct;

  define sector     / 'Sector' group
                      format=$sctrfmt.;
  define department / group format=$deptfmt.;
  define sales      / analysis sum
                      format=dollar9.2 ;
  define sctrpct    / computed
                      format=percent9.2 ;
  define salespct   / pctsum format=percent9.2;

  compute before;
     line ' ';
     line @16 'Total for all stores is '
          sales.sum dollar9.2;
     line ' ';
     line @29 'Sum of' @40 'Percent'
          @51 'Percent of';
     line @6 'Sector' @17 'Department'
          @29 'Sales'
          @40 'of Sector' @51 'All Stores';
     line @6 55*'=';
     line ' ';
  endcomp;

  break before sector / summarize ul;
  compute before sector;
     sctrtot=sales.sum;
     sctrpct=sales.sum/sctrtot;
  endcomp;

  compute sctrpct;
     sctrpct=sales.sum/sctrtot;
  endcomp;

  break after sector/skip;
  where sector contains 'n';
  title 'Report for Northeast and Northwest Sectors';
run;
```

Output 32.3 Report with DATA Step Variables

```
          Report for Northeast and Northwest Sectors          1


              Total for all stores is $4,285.00

                          Sum of      Percent     Percent of
Sector      Department    Sales       of Sector   All Stores
=======================================================

Northeast                 $1,831.00    100.00%      42.73%
---------                 ---------  ---------   ---------
Northeast   Canned          $840.00     45.88%      19.60%
            Meat/Dairy      $490.00     26.76%      11.44%
            Paper           $290.00     15.84%       6.77%
            Produce         $211.00     11.52%       4.92%

Northwest                 $2,454.00    100.00%      57.27%
---------                 ---------  ---------   ---------
Northwest   Canned        $1,070.00     43.60%      24.97%
            Meat/Dairy    $1,055.00     42.99%      24.62%
            Paper           $150.00      6.11%       3.50%
            Produce         $179.00      7.29%       4.18%
```

A description of how PROC REPORT builds this report follows:

1 PROC REPORT starts building the report by consolidating the data (Sector and Department are group variables) and by calculating the statistics (Sales.sum and Sales.pctsum) for each detail row, for the break at the beginning of the report, for the breaks before each group, and for the breaks after each group. It stores these values in a temporary file.

2 PROC REPORT initializes the DATA step variable, Sctrtot, to missing (see Figure 32.15 on page 954).

Figure 32.15 Initialized DATA Step Variables

Report Variables					DATA Step Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
		.	.	.	.

3 Because this PROC REPORT step contains a COMPUTE BEFORE statement, the procedure constructs a preliminary summary line for the break at the beginning of the report. This preliminary summary line contains values for the statistics (Sales.sum and Sales.pctsum) and the computed variable (Sctrpct).

At this break, Sales.sum is the sales for all stores, and Sales.pctsum is the percentage those sales represent for all stores (100%). PROC REPORT takes the values for these statistics from the temporary file that it created at the beginning of the report-building process.

The value for Sctrpct comes from executing the statements in the corresponding compute block. Because the value of Sctrtot is missing, PROC REPORT cannot calculate a value for Sctrpct. Therefore, in the preliminary summary line (which is not printed in this case), this variable also has a missing value (see Figure 32.16 on page 955).

The statements in the COMPUTE BEFORE block do not alter any variables. Therefore, the final summary line is the same as the preliminary summary line.

Note: The COMPUTE BEFORE statement creates a break at the beginning of the report. You do not need to use an RBREAK statement. △

Figure 32.16 Preliminary and Final Summary Line for the Break at the Beginning of the Report

Report Variables					DATA Step Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
		$4,285.00	.	100.00%	.

4 Because the program does not include an RBREAK statement with the SUMMARIZE option, PROC REPORT does not write the final summary line to the report. Instead, it uses LINE statements to write a customized summary that embeds the value of Sales.sum into a sentence and to write customized column headers. (The NOHEADER option in the PROC REPORT statement suppresses the default column headers, which would have appeared before the customized summary.)

5 Next, PROC REPORT constructs a preliminary summary line for the break before the first group of observations. (This break both uses the SUMMARIZE option in the BREAK statement and has a compute block attached to it. Either of these conditions generates a summary line.) The preliminary summary line contains values for the break variable (Sector), the statistics (Sales.sum and Sales.pctsum), and the computed variable (Sctrpct). At this break, Sales.sum is the sales for one sector (the northeast sector). PROC REPORT takes the values for Sector, Sales.sum, and Sales.pctsum from the temporary file that it created at the beginning of the report-building process.

The value for Sctrpct comes from executing the statements in the corresponding compute blocks. Because the value of Sctrtot is still missing, PROC REPORT cannot calculate a value for Sctrpct. Therefore, in the preliminary summary line, Sctrpct has a missing value (see Figure 32.17 on page 955).

Figure 32.17 Preliminary Summary Line for the Break before the First Group of Observations

Report Variables					DATA Step Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
Northeast		$1,831.00	.	42.73%	.

6 PROC REPORT creates the final version of the summary line by executing the statements in the COMPUTE BEFORE SECTOR compute block. These statements execute once each time the value of Sector changes.

- The first statement assigns the value of Sales.sum, which in that part of the report represents total sales for one Sector, to the variable Sctrtot.
- The second statement completes the summary line by recalculating Sctrpct from the new value of Sctrtot. Figure 32.18 on page 956 shows the final summary line.

CAUTION:

Recalculating Values in the Final Summary Line. If you do not recalculate the value for Sctrpct, it will be missing because the value of Sctrtot is missing at the time that the COMPUTE Sctrpct block executes. △

Figure 32.18 Final Summary Line for the Break before the First Group of Observations

Report Variables					DATA Step Variable
Sector	**Department**	**Sales.sum**	**Sctrpct**	**Sales.pctsum**	**Sctrtot**
Northeast		$1,831.00	100.00%	42.73%	$1,831.00

7 Because the program contains a BREAK BEFORE statement with the SUMMARIZE option, PROC REPORT writes the final summary line to the report. The UL option in the BREAK statement underlines the summary line.

8 Now, PROC REPORT is ready to start building the first detail row of the report. It initializes all report variables to missing. Values for DATA step variables do not change. Figure 32.19 on page 956 illustrates the first detail row at this point.

Figure 32.19 First Detail Row with Initialized Values

Report Variables					DATA Step Variable
Sector	**Department**	**Sales.sum**	**Sctrpct**	**Sales.pctsum**	**Sctrtot**
		.	.	.	$1,831.00

9 Figure 32.20 on page 956 illustrates the construction of the first three columns of the row. PROC REPORT fills in values for the row from left to right. The values come from the temporary file it created at the beginning of the report-building process.

Figure 32.20 Filling in Values from Left to Right

Report Variables					DATA Step Variable
Sector	**Department**	**Sales.sum**	**Sctrpct**	**Sales.pctsum**	**Sctrtot**
Northeast		.	.	.	$1,831.00

Report Variables					DATA Step Variable
Sector	**Department**	**Sales.sum**	**Sctrpct**	**Sales.pctsum**	**Sctrtot**
Northeast	Canned	.	.	.	$1,831.00

Report Variables					DATA Step Variable
Sector	**Department**	**Sales.sum**	**Sctrpct**	**Sales.pctsum**	**Sctrtot**
Northeast	Canned	$840.00	.	.	$1,831.00

10 The next column in the report contains the computed variable Sctrpct. When it gets to this column, PROC REPORT executes the statement in the compute block

attached to Sctrpct. This statement calculates the percentage of the sector's total sales that this department accounts for:

```
sctrpct=sales.sum/sctrtot;
```

The row now looks like Figure 32.21 on page 957.

Figure 32.21 First Detail Row with the First Computed Variable Added

Report Variables					DATA Step Variable
Sector	**Department**	**Sales.sum**	**Sctrpct**	**Sales.pctsum**	**Sctrtot**
Northeast	Canned	$840.00	45.88%	.	$1,831.00

11 The next column in the report contains the statistic Sales.pctsum. PROC REPORT gets this value from the temporary file. The first detail row is now complete (see Figure 32.22 on page 957).

Figure 32.22 First Complete Detail Row

Report Variables					DATA Step Variable
Sector	**Department**	**Sales.sum**	**Sctrpct**	**Sales.pctsum**	**Sctrtot**
Northeast	Canned	$840.00	45.88%	19.60%	$1,831.00

12 PROC REPORT writes the detail row to the report. It repeats steps 8, 9, 10, 11, and 12 for each detail row in the group.

13 After writing the last detail row in the group to the report, PROC REPORT constructs the default group summary. Because no compute block is attached to this break and because the BREAK AFTER statement does not include the SUMMARIZE option, PROC REPORT does not construct a summary line. The only action at this break is that the SKIP option in the BREAK AFTER statement writes a blank line after the last detail row of the group.

14 Now the value of the break variable changes from **`Northeast`** to **`Northwest`**. PROC REPORT constructs a preliminary summary line for the break before this group of observations. As at the beginning of any row, PROC REPORT initializes all report variables to missing but retains the value of the DATA step variable. Next, it completes the preliminary summary line with the appropriate values for the break variable (Sector), the statistics (Sales.sum and Sales.pctsum), and the computed variable (Sctrpct). At this break, Sales.sum is the sales for the Northwest sector. Because the COMPUTE BEFORE Sector block has not yet executed, the value of Sctrtot is still $1,831.00, the value for the Northeast sector. Thus, the value that PROC REPORT calculates for Sctrpct in this preliminary summary line is incorrect (see Figure 32.23 on page 958). The statements in the compute block for this break calculate the correct value (see the following step).

Figure 32.23 Preliminary Summary Line for the Break before the Second Group of Observations

Report Variables					DATA Step Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
Northwest		$2,454.00	134.00%	57.27%	$1,831.00

CAUTION:

Synchronizing Values for Computed Variables in Break Lines. If the PROC REPORT step does not recalculate Sctrpct in the compute block attached to the break, the value in the final summary line will not be synchronized with the other values in the summary line, and the report will be incorrect. △

15 PROC REPORT creates the final version of the summary line by executing the statements in the COMPUTE BEFORE Sector compute block. These statements execute once each time the value of Sector changes.

- The first statement assigns the value of Sales.sum, which in that part of the report represents sales for the Northwest sector, to the variable Sctrtot.
- The second statement completes the summary line by recalculating Sctrpct from the new, appropriate value of Sctrtot. Figure 32.24 on page 958 shows the final summary line.

Figure 32.24 Final Summary Line for the Break before the Second Group of Observations

Report Variables					DATA Step Variable
Sector	Department	Sales.sum	Sctrpct	Sales.pctsum	Sctrtot
Northwest		$2,454.00	100.00%	57.27%	$2,454.00

Because the program contains a BREAK BEFORE statement with the SUMMARIZE option, PROC REPORT writes the final summary line to the report. The UL option in the BREAK statement underlines the summary line.

16 Now, PROC REPORT is ready to start building the first row for this group of observations. It repeats steps 8 through 16 until it has processed all observations in the input data set (stopping with step 14 for the last group of observations).

Examples

Example 1: Selecting Variables for a Report

Procedure features:

PROC REPORT statement options:

NOWD

COLUMN statement
default variable usage
RBREAK statement options:
DOL
SUMMARIZE

Other features:
FORMAT statement
FORMAT procedure:
LIBRARY=
SAS system options:
FMTSEARCH=
Automatic macro variables:
SYSDATE

This example uses a permanent data set and permanent formats to create a report that contains

- one row for every observation
- a default summary for the whole report.

Program

```
libname proclib 'SAS-data-library';


options nodate pageno=1 linesize=64 pagesize=60;
```

The data set GROCERY contains one day's sales figures for eight stores in the Grocery Mart chain. Each observation contains one day's sales data for one department in one store.

```
data grocery;
   input Sector $ Manager $ Department $ Sales @@;
   datalines;
se 1 np1 50    se 1 p1 100   se 1 np2 120   se 1 p2 80
se 2 np1 40    se 2 p1 300   se 2 np2 220   se 2 p2 70
nw 3 np1 60    nw 3 p1 600   nw 3 np2 420   nw 3 p2 30
nw 4 np1 45    nw 4 p1 250   nw 4 np2 230   nw 4 p2 73
nw 9 np1 45    nw 9 p1 205   nw 9 np2 420   nw 9 p2 76
sw 5 np1 53    sw 5 p1 130   sw 5 np2 120   sw 5 p2 50
sw 6 np1 40    sw 6 p1 350   sw 6 np2 225   sw 6 p2 80
ne 7 np1 90    ne 7 p1 190   ne 7 np2 420   ne 7 p2 86
ne 8 np1 200   ne 8 p1 300   ne 8 np2 420   ne 8 p2 125
;
```

PROC FORMAT creates permanent formats for Sector, Manager, and Department. The LIBRARY= option specifies a permanent storage location so that the formats are available in subsequent SAS sessions. These formats are used for examples throughout this section.

```
proc format library=proclib;
   value $sctrfmt 'se' = 'Southeast'
                  'ne' = 'Northeast'
                  'nw' = 'Northwest'
                  'sw' = 'Southwest';

   value $mgrfmt '1' = 'Smith'   '2' = 'Jones'
                 '3' = 'Reveiz'  '4' = 'Brown'
                 '5' = 'Taylor'  '6' = 'Adams'
                 '7' = 'Alomar'  '8' = 'Andrews'
                 '9' = 'Pelfrey';

   value $deptfmt 'np1' = 'Paper'
                  'np2' = 'Canned'
                  'p1'  = 'Meat/Dairy'
                  'p2'  = 'Produce';
run;
```

The SAS system option FMTSEARCH= adds the SAS data library PROCLIB to the search path that is used to locate formats.

```
options fmtsearch=(proclib);
```

The NOWD option runs the REPORT procedure without the REPORT window and sends its output to the SAS procedure output.

```
proc report data=grocery nowd;
```

The report contains a column for Manager, Department, and Sales. Because there is no DEFINE statement for any of these variables, PROC REPORT uses the character variables (Manager and Department) as display variables and the numeric variable (Sales) as an analysis variable that is used to calculate the sum statistic.

```
    column manager department sales;
```

The RBREAK statement produces a default summary at the end of the report. DOL writes a line of equal signs (=) above the summary information. SUMMARIZE sums the value of Sales for all observations in the report.

```
    rbreak after / dol summarize;
```

The WHERE statement selects for the report only the observations for stores in the southeast sector.

```
    where sector='se';
```

The FORMAT statement assigns formats to use in the report. You can use the FORMAT statement only with data set variables.

```
   format manager $mgrfmt.;
   format department $deptfmt.;
   format sales dollar11.2;
```

SYSDATE is an automatic macro variable that returns the date when the SAS job or SAS session began. The TITLE2 statement uses double rather than single quotes so that the macro variable resolves.

```
   title 'Sales for the Southeast Sector';
   title2 "for &sysdate";
run;
```

Output

```
                Sales for the Southeast Sector                1
                          for 14MAY98

               Manager  Department        Sales
               Smith    Paper            $50.00
               Smith    Meat/Dairy      $100.00
               Smith    Canned          $120.00
               Smith    Produce          $80.00
               Jones    Paper            $40.00
               Jones    Meat/Dairy      $300.00
               Jones    Canned          $220.00
               Jones    Produce          $70.00
                                       ==========
                                        $980.00
```

Example 2: Ordering the Rows in a Report

Procedure features:

PROC REPORT statement options:
- COLWIDTH=
- HEADLINE
- HEADSKIP
- SPACING=

BREAK statement options:
- OL
- SKIP
- SUMMARIZE

COMPUTE statement arguments:
- AFTER

DEFINE statement options:
- ANALYSIS
- FORMAT=
- ORDER
- ORDER=
- SUM

ENDCOMP statement

LINE statement:

with quoted text
with variable values

Data set: GROCERY on page 959

Formats: $MGRFMT. and $DEPTFMT. on page 960

This example

- arranges the rows alphabetically by the formatted values of Manager and the internal values of Department (so that sales for the two departments that sell nonperishable goods precede sales for the two departments that sell perishable goods)
- controls the default column width and the spacing between columns
- underlines the column headers and writes a blank line beneath the underlining
- creates a default summary of Sales for each manager
- creates a customized summary of Sales for the whole report.

Program

```
libname proclib 'SAS-data-library';


options nodate pageno=1 linesize=64 pagesize=60
        fmtsearch=(proclib);
```

The NOWD option runs PROC REPORT without the REPORT window and sends its output to the SAS procedure output. COLWIDTH=10 sets the default column width to 10 characters. SPACING= puts five blank characters between columns. HEADLINE underlines all column headers and the spaces between them at the top of each page of the report. HEADSKIP writes a blank line beneath the underlining that HEADLINE writes.

```
proc report data=grocery nowd
            colwidth=10
            spacing=5
            headline headskip;
```

The report contains a column for Manager, Department, and Sales.

```
    column manager department sales;
```

The values of all variables with the ORDER option in the DEFINE statement determine the order of the rows in the report. In this report, PROC REPORT arranges the rows first by the value of Manager (because it is the first variable in the COLUMN statement) and then by the values of Department.

ORDER= specifies the sort order for a variable. This report arranges the rows according to the formatted values of Manager and the internal values of Department (np1, np2, p1, and p2). FORMAT= specifies the formats to use in the report.

```
define manager / order order=formatted format=$mgrfmt.;
define department / order order=internal format=$deptfmt.;
```

Sum calculates the sum statistic for all observations represented by the current row of the row. In this report each row represents only one observation. Therefore, the Sum statistic is the same as the value of Sales for that observation in the input data set. Using Sales as an analysis variable in this report enables you to summarize the values for each group and at the end of the report.

```
define sales / analysis sum format=dollar7.2;
```

This BREAK statement produces a default summary after the last row for each manager. OL writes a row of hyphens above the summary line. SUMMARIZE writes the value of Sales (the only analysis or computed variable) in the summary line. PROC REPORT sums the values of Sales for each manager because Sales is an analysis variable used to calculate the Sum statistic. SKIP writes a blank line after the summary line.

```
break after manager / ol
                      summarize
                      skip;
```

This COMPUTE statement begins a compute block that produces a customized summary at the end of the report. The LINE statement places the quoted text and the value of Sales.sum (with the DOLLAR9.2 format) in the summary. An ENDCOMP statement must end the compute block.

```
compute after;
   line 'Total sales for these stores were: '
        sales.sum dollar9.2;
endcomp;
```

The WHERE statement selects for the report only the observations for stores in the southeast sector.

```
where sector='se';
```

The TITLE statement specifies the title.

```
   title 'Sales for the Southeast Sector';
run;
```

Output

```
                Sales for the Southeast Sector                1

              Manager     Department       Sales
              ----------------------------------

              Jones       Paper            $40.00
                          Canned          $220.00
                          Meat/Dairy      $300.00
                          Produce          $70.00
              -------                     -------
              Jones                       $630.00

              Smith       Paper            $50.00
                          Canned          $120.00
                          Meat/Dairy      $100.00
                          Produce          $80.00
              -------                     -------
              Smith                       $350.00

          Total sales for these stores were:   $980.00
```

Example 3: Using Aliases to Obtain Multiple Statistics for the Same Variable

Procedure features:

COLUMN statement:

with aliases

COMPUTE statement arguments:

AFTER

DEFINE statement options:

ANALYSIS
MAX
MIN
NOPRINT
customizing column headers

LINE statement:

pointer controls
quoted text
repeating a character string
variable values and formats
writing a blank line

Other features:

automatic macro variables:

SYSDATE

Data set: GROCERY on page 959

Formats: $MGRFMT. and $DEPTFMT. on page 960

The customized summary at the end of this report displays the minimum and maximum values of Sales over all departments for stores in the southeast sector. To determine these values, PROC REPORT needs the MIN and MAX statistic for Sales in

every row of the report. However, to keep the report simple, the display of these statistics is suppressed.

Program

```
libname proclib 'SAS-data-library';


options nodate pageno=1 linesize=64 pagesize=60
        fmtsearch=(proclib);
```

The NOWD option runs PROC REPORT without the REPORT window and sends its output to the SAS procedure output. HEADLINE underlines all column headers and the spaces between them at the top of each page of the report. HEADSKIP writes a blank line beneath the underlining that HEADLINE writes.

```
proc report data=grocery nowd headline headskip;
```

The report contains columns for Manager and Department. It also contains three columns for Sales. The column specifications SALES=SALESMIN and SALES=SALESMAX create aliases for Sales. These aliases enable you to use a separate definition of Sales for each of the three columns.

```
    column manager department sales
           sales=salesmin
           sales=salesmax;
```

The values of all variables with the ORDER option in the DEFINE statement determine the order of the rows in the report. In this report, PROC REPORT arranges the rows first by the value of Manager (because it is the first variable in the COLUMN statement) and then by the values of Department. The ORDER= option specifies the sort order for a variable. This report arranges the values of Manager by their formatted values and arranges the values of Department by their internal values (np1, np2, p1, and p2). FORMAT= specifies the formats to use in the report. Text in quotation marks specifies column headers.

```
    define manager / order
                     order=formatted
                     format=$mgrfmt.
                     'Manager';
    define department    / order
                     order=internal
                     format=$deptfmt.
                     'Department';
```

The value of an analysis variable in any row of a report is the value of the statistic associated with it (in this case Sum) calculated for all observations represented by that row. In a detail report each row represents only one observation. Therefore, the Sum statistic is the same as the value of Sales for that observation in the input data set.

```
    define sales / analysis sum format=dollar7.2 'Sales';
```

These DEFINE statements use aliases from the COLUMN statement to create separate columns for the MIN and MAX statistics for the analysis variable Sales. NOPRINT suppresses the printing of these statistics. Although PROC REPORT does not print these values in columns, it has access to them so that it can print them in the summary.

```
    define salesmin / analysis min noprint;
    define salesmax / analysis max noprint;
```

This COMPUTE statement begins a compute block that executes at the end of the report. The first LINE statement writes a blank line. The second LINE statement writes 53 hyphens (-), beginning in column 7.

```
    compute after;
       line ' ';
       line @7 53*'-';
```

The first line of this LINE statement writes the text in quotation marks, beginning in column 7. The second line writes the value of Salesmin with the DOLLAR7.2 format, beginning in the next column. The cursor then moves one column to the right (+1), where PROC REPORT writes the text in quotation marks. Again, the cursor moves one column to the right, and PROC REPORT writes the value of Salesmax with the DOLLAR7.2 format. (Note that the program must reference the variables by their aliases.) The third line writes the text in quotation marks, beginning in the next column.

```
       line @7 '| Departmental sales ranged from'
            salesmin dollar7.2  +1 'to' +1 salesmax dollar7.2
            '. |';
       line @7 53*'-';
```

An ENDCOMP statement must end the compute block.

```
    endcomp;
```

The WHERE statement selects for the report only the observations for stores in the southeast sector. The TITLE statements specify two titles. SYSDATE is an automatic macro variable that returns the date when the SAS job or SAS session began. The TITLE2 statement uses double rather than single quotes so that the macro variable resolves.

```
    where sector='se';
    title 'Sales for the Southeast Sector';
    title2 ''for &sysdate'';
run;
```

Output

```
                Sales for the Southeast Sector                1
                         for 27MAY98

                 Manager  Department    Sales
                 ---------------------------

                 Jones    Paper        $40.00
                          Canned      $220.00
                          Meat/Dairy  $300.00
                          Produce      $70.00
                 Smith    Paper        $50.00
                          Canned      $120.00
                          Meat/Dairy  $100.00
                          Produce      $80.00

      ------------------------------------------------------
      | Departmental sales ranged from $40.00 to $300.00. |
      ------------------------------------------------------
```

Example 4: Consolidating Multiple Observations into One Row of a Report

Procedure features:

BREAK statement options:
- OL
- SKIP
- SUMMARIZE
- SUPPRESS

CALL DEFINE statement

Compute block
- associated with a data set variable

COMPUTE statement arguments:
- AFTER
- a data set variable as *report-item*

DEFINE statement options:
- ANALYSIS
- GROUP
- SUM
- customizing column headers

LINE statement:
- quoted text
- variable values

Data set: GROCERY on page 959

Formats: $MGRFMT. and $DEPTFMT. on page 960

This example creates a summary report that

- consolidates information for each combination of Sector and Manager into one row of the report
- contains default summaries of sales for each sector

- contains a customized summary of sales for all sectors
- uses one format for sales in detail rows and a different format in summary rows
- uses customized column headers.

Program

```
libname proclib 'SAS-data-library';
```

The NOWD option runs PROC REPORT without the REPORT window and sends its output to the SAS procedure output. HEADLINE underlines all column headers and the spaces between them at the top of each page of the report. HEADSKIP writes a blank line beneath the underlining that HEADLINE writes..

```
options nodate pageno=1 linesize=64 pagesize=60
        fmtsearch=(proclib);

proc report data=grocery nowd headline headskip;
```

The report contains columns for Sector, Manager, and Sales.

```
    column sector manager sales;
```

In this report, Sector and Manager are group variables. Sales is an analysis variable used to calculate the Sum statistic. Each detail row represents a set of observations that have a unique combination of formatted values for all group variables. The value of Sales in each detail row is the sum of Sales for all observations in the group. FORMAT= specifies the format to use in the report. Text in quotation marks in a DEFINE statement specifies the column header.

```
    define sector / group
                    format=$sctrfmt.
                    'Sector';
    define manager / group
                     format=$mgrfmt.
                     'Manager';
    define sales / analysis sum
                   format=comma10.2
                   'Sales';
```

This BREAK statement produces a default summary after the last row for each sector. OL writes a row of hyphens above the summary line. SUMMARIZE writes the value of Sales in the summary line. PROC REPORT sums the values of Sales for each manager because Sales is an analysis variable used to calculate the Sum statistic. SUPPRESS prevents PROC REPORT from displaying the value of Sector in the summary line. SKIP writes a blank line after the summary line.

```
    break after sector / ol
                         summarize
                         suppress
```

```
                        skip;
```

This compute block creates a customized summary at the end of the report. The LINE statement writes the quoted text and the value of Sales.sum (with a format of DOLLAR9.2) in the summary. An ENDCOMP statement must end the compute block.

```
    compute after;
       line 'Combined sales for the northern sectors were '
             sales.sum dollar9.2 '.';
    endcomp;
```

In detail rows, PROC REPORT displays the value of Sales with the format specified in its definition (COMMA10.2). The compute block specifies an alternate format to use in the current column on summary rows. Summary rows are identified as a value other than a blank for _BREAK_.

```
    compute sales;
       if _break_ ne ' ' then
       call define(_col_,"format","dollar11.2");
    endcomp;
```

The WHERE statement selects for the report only the observations for stores in the northeast and northwest sectors. The TITLE statement specifies the title.

```
    where sector contains 'n';
    title 'Sales Figures for Northern Sectors';
run;
```

Output

```
              Sales Figures for Northern Sectors              1

                Sector     Manager       Sales
                ------------------------------

                Northeast  Alomar         786.00
                           Andrews      1,045.00
                                       ----------
                                       $1,831.00

                Northwest  Brown          598.00
                           Pelfrey        746.00
                           Reveiz       1,110.00
                                       ----------
                                       $2,454.00

    Combined sales for the northern sectors were $4,285.00.
```

Example 5: Creating a Column for Each Value of a Variable

Procedure features:

PROC REPORT statement options:

SPLIT=

BREAK statement options:

SKIP

COLUMN statement:

stacking variables

COMPUTE statement arguments:

with a computed variable as *report-item*
AFTER

DEFINE statement options:

ACROSS
ANALYSIS
COMPUTED
SUM

LINE statement:

pointer controls

Data set: GROCERY on page 959

Formats: $SCTRFMT., $MGRFMT., and $DEPTFMT. on page 960

The report in this example

- consolidates multiple observations into one row
- contains a column for each value of Department that is selected for the report (the departments that sell perishable items)
- contains a variable that is not in the input data set
- uses customized column headers, some of which contain blank lines
- double-spaces between detail rows
- uses pointer controls to control the placement of text and variable values in a customized summary.

Program

```
libname proclib 'SAS-data-library';


options nodate pageno=1 linesize=64 pagesize=60
        fmtsearch=(proclib);
```

The NOWD option runs PROC REPORT without the REPORT window and sends its output to the SAS procedure output. HEADLINE underlines the column headers. HEADSKIP writes a blank line beneath the underlining that HEADLINE writes. SPLIT= defines the split character as an asterisk (*) because the default split character (/) is part of the name of a department.

```
proc report data=grocery nowd
            headline
            headskip
            split='*';
```

Department and Sales are separated by a comma in the COLUMN statement, so they collectively determine the contents of the column that they define. Each item generates a header, but the header for Sales is set to blank in its definition. Because Sales is an analysis variable, its values fill the cells created by these two variables.

```
    column sector manager department,sales perish;
```

In this report, Sector and Manager are group variables. Each detail row of the report consolidates the information for all observations with the same values of the group variables. FORMAT= specifies the formats to use in the report. Text in quotation marks in the DEFINE statements specifies column headers. These statements illustrate two ways to write a blank line in a column header. **`'Sector' ''`** writes a blank line because each quoted string is a line of the column header. The two adjacent quotation marks write a blank line for the second line of the header. **`'Manager* '`** writes a blank line because the split character (*) starts a new line of the header. That line contains only a blank.

```
    define sector / group format=$sctrfmt. 'Sector' '';
    define manager / group format=$mgrfmt. 'Manager* ';
```

PROC REPORT creates a column and a column header for each formatted value of the across variable Department. PROC REPORT orders the columns by these values. PROC REPORT also generates a column header that spans all these columns. Quoted text in the DEFINE statement for Department customizes this header. In traditional (monospace) SAS output, PROC REPORT expands the header with underscores to fill all columns created by the across variable. Sales is an analysis variable that is used to calculate the sum statistic. In each case, the value of Sales is the sum of Sales for all observations in one department in one group. (In this case, the value represents a single observation.)

```
    define department / across format=$deptfmt. '_Department_';
    define sales / analysis sum format=dollar11.2 ' ';
```

The COMPUTED option indicates that PROC REPORT must compute values for Perish. You compute the variable's values in a compute block associated with Perish.

```
    define perish / computed format=dollar11.2
                 'Perishable*Total';
```

This BREAK statement creates a default summary after the last row for each value of Manager. The only option in use is SKIP, which writes a blank line. You can use this technique to double-space in many reports that contains a group or order variable.

```
    break after manager / skip;
```

This compute block computes the value of Perish from the values for the Meat/Dairy department and the Produce department. Because the variables Sales and Department collectively define these columns, there is no way to identify the values to PROC REPORT by name. Therefore, the assignment statement uses column numbers to unambiguously specify the values to use. Each time PROC REPORT needs a value for Perish, it sums the values in the third and fourth columns of that row of the report.

```
compute perish;
   perish=_c3_+_c4_;
endcomp;
```

This compute block creates a customized summary at the end of the report. The first LINE statement writes 57 hyphens (-) starting in column 4. Subsequent LINE statements write the quoted text in the specified columns and the values of the variables _C3_, _C4_, and _C5_ with the DOLLAR11.2 format.

```
compute after;
   line @4 57*'-';
   line @4 '|   Combined sales for meat and dairy : '
        @46 _c3_ dollar11.2 '   |';
   line @4 '|   Combined sales for produce : '
        @46 _c4_ dollar11.2 '   |';
   line @4 '|' @60 '|';
   line @4 '|   Combined sales for all perishables: '
        @46 _c5_ dollar11.2 '   |';
   line @4 57*'-';
endcomp;
```

The WHERE statement selects for the report only the observations for departments **p1** and **p2** in stores in the northeast or northwest sector. The TITLE statement specifies the title.

```
   where sector contains 'n'
         and (department='p1' or department='p2');
   title "Sales Figures for Perishables in Northern Sectors";
run;
```

Output

```
      Sales Figures for Perishables in Northern Sectors       1

                        _______Department_______
Sector     Manager    Meat/Dairy      Produce    Perishable
                                                      Total
-------------------------------------------------------------

Northeast  Alomar        $190.00       $86.00       $276.00

           Andrews       $300.00      $125.00       $425.00

Northwest  Brown         $250.00       $73.00       $323.00

           Pelfrey       $205.00       $76.00       $281.00

           Reveiz        $600.00       $30.00       $630.00

-------------------------------------------------------------
|   Combined sales for meat and dairy :     $1,545.00   |
|   Combined sales for produce :              $390.00   |
|                                                       |
|   Combined sales for all perishables:     $1,935.00   |
-------------------------------------------------------------
```

Example 6: Displaying Multiple Statistics for One Variable

Procedure features:

PROC REPORT statement options:

LS=
PS=

COLUMN statement:

specifying statistics for stacked variables

DEFINE statement options:

FORMAT=
GROUP
ID

Data set: GROCERY on page 959

Formats: $MGRFMT. on page 960

The report in this example displays six statistics for the sales for each manager's store. The output is too wide to fit all the columns on one page, so three of the statistics appear on the second page of the report. In order to make it easy to associate the statistics on the second page with their group, the report repeats the values of Manager and Sector on every page of the report.

Program

```
libname proclib 'SAS-data-library';
```

```
options nodate pageno=1 linesize=80 pagesize=60
        fmtsearch=(proclib);
```

The NOWD option runs PROC REPORT without the REPORT window and sends its output to the SAS procedure output. HEADLINE underlines all column headers and the spaces between them at the top of each page of the report. HEADSKIP writes a blank line beneath the underlining that HEADLINE writes. LS= sets the line size for the report to 66, and PS= sets the page size to 18.

```
proc report data=grocery nowd headline headskip
            ls=66 ps=18;
```

This COLUMN statement creates a column for Sector, Manager, and each of the six statistics associated with Sales.

```
column sector manager (Sum Min Max Range Mean Std),sales;
```

ID specifies that Manager is an ID variable. An ID variable and all columns to its left appear at the left of every page of a report. In this report, Sector and Manager are group variables. Each detail row of the report consolidates the information for all observations with the same values of the group variables. FORMAT= specifies the formats to use in the report.

```
   define manager / group format=$mgrfmt. id;
   define sector / group format=$sctrfmt.;
   define sales / format=dollar11.2 ;
```

The TITLE statement specifies a title for the report.

```
   title 'Sales Statistics for All Sectors';
run;
```

Output

```
              Sales Statistics for All Sectors                1

                                Sum          Min          Max
Sector     Manager            Sales        Sales        Sales
-----------------------------------------------------------

Northeast  Alomar           $786.00       $86.00      $420.00
           Andrews        $1,045.00      $125.00      $420.00
Northwest  Brown            $598.00       $45.00      $250.00
           Pelfrey          $746.00       $45.00      $420.00
           Reveiz         $1,110.00       $30.00      $600.00
Southeast  Jones            $630.00       $40.00      $300.00
           Smith            $350.00       $50.00      $120.00
Southwest  Adams            $695.00       $40.00      $350.00
           Taylor           $353.00       $50.00      $130.00
```

```
              Sales Statistics for All Sectors                2

                              Range          Mean           Std
Sector      Manager           Sales         Sales         Sales
-----------------------------------------------------------------

Northeast   Alomar        $334.00       $196.50       $156.57
            Andrews       $295.00       $261.25       $127.83
Northwest   Brown         $205.00       $149.50       $105.44
            Pelfrey       $375.00       $186.50       $170.39
            Reveiz        $570.00       $277.50       $278.61
Southeast   Jones         $260.00       $157.50       $123.39
            Smith          $70.00        $87.50        $29.86
Southwest   Adams         $310.00       $173.75       $141.86
            Taylor         $80.00        $88.25        $42.65
```

Example 7: Storing and Reusing a Report Definition

Procedure features:

PROC REPORT statement options:

NAMED
OUTREPT=
REPORT=
WRAP

Other features:

TITLE statement

WHERE statement

Data set: GROCERY on page 959

Formats: $SCTRFMT., $MGRFMT. and $DEPTFMT. on page 960

The first PROC REPORT step in this example creates a report that displays one value from each column of the report, using two rows to do so, before displaying another value from the first column. (By default, PROC REPORT displays values for only as many columns as it can fit on one page. It fills a page with values for these columns before starting to display values for the remaining columns on the next page.)

Each item in the report is identified in the body of the report rather than in a column header.

The report definition created by the first PROC REPORT step is stored in a catalog entry. The second PROC REPORT step uses it to create a similar report for a different sector of the city.

Program to Store a Report Definition

```
libname proclib 'SAS-data-library';


options nodate pageno=1 linesize=80 pagesize=60
        fmtsearch=(proclib);
```

The NOWD option runs PROC REPORT without the REPORT window and sends its output to the SAS procedure output. NAMED writes *name*= in front of each value in the report, where *name*= is the column header for the value. When you use NAMED, PROC REPORT suppresses the display of column headers at the top of each page.

```
proc report data=grocery nowd
            named
            wrap
            ls=64 ps=36
            outrept=proclib.reports.namewrap;
```

The report contains a column for Sector, Manager, Department, and Sales.

```
    column sector manager department sales;
```

Because no usage is specified in the DEFINE statements, PROC REPORT uses the defaults. The character variables (Sector, Manager, and Department) are display variables. Sales is an analysis variable that is used to calculate the sum statistic. FORMAT= specifies the formats to use in the report.

```
    define sector / format=$sctrfmt.;
    define manager / format=$mgrfmt.;
    define department / format=$deptfmt.;
    define sales / format=dollar11.2;
```

A report definition may differ from the SAS program that creates the report. In particular, PROC REPORT stores neither WHERE statements nor TITLE statements.

```
    where manager='1';
    title "Sales Figures for Smith on &sysdate";
run;
```

Output

This is the output from the first PROC REPORT step, which creates the report definition.

```
              Sales Figures for Smith on 01OCT98              1

    Sector=Southeast  Manager=Smith    Department=Paper
    Sales=     $50.00
    Sector=Southeast  Manager=Smith    Department=Meat/Dairy
    Sales=    $100.00
    Sector=Southeast  Manager=Smith    Department=Canned
    Sales=    $120.00
    Sector=Southeast  Manager=Smith    Department=Produce
    Sales=     $80.00
```

Program to Use a Report Definition

REPORT= uses the report definition stored in PROCLIB.REPORTS.NAMEWRAP to produce the report. The second report differs from the first one because it uses different WHERE and TITLE statements.

```
options nodate pageno=1 fmtsearch=(proclib);
proc report data=grocery report=proclib.reports.namewrap
            nowd;
   where sector='sw';
   title "Sales Figures for the Southwest Sector on &sysdate";
run;
```

Output

```
       Sales Figures for the Southwest Sector on 01OCT98       1

    Sector=Southwest  Manager=Taylor   Department=Paper
    Sales=     $53.00
    Sector=Southwest  Manager=Taylor   Department=Meat/Dairy
    Sales=    $130.00
    Sector=Southwest  Manager=Taylor   Department=Canned
    Sales=    $120.00
    Sector=Southwest  Manager=Taylor   Department=Produce
    Sales=     $50.00
    Sector=Southwest  Manager=Adams    Department=Paper
    Sales=     $40.00
    Sector=Southwest  Manager=Adams    Department=Meat/Dairy
    Sales=    $350.00
    Sector=Southwest  Manager=Adams    Department=Canned
    Sales=    $225.00
    Sector=Southwest  Manager=Adams    Department=Produce
    Sales=     $80.00
```

Example 8: Condensing a Report into Multiple Panels

Procedure features:

PROC REPORT statement options:

- FORMCHAR=
- HEADLINE
- LS=
- PANELS=
- PS=
- PSPACE=

BREAK statement options:

- SKIP

Other features:

SAS system option FORMCHAR=

Data set: GROCERY on page 959

Formats: $MGRFMT. and $DEPTFMT. on page 960

The report in this example

- uses panels to condense a two-page report to one page. Panels compactly present information for long, narrow reports by placing multiple rows of information side by side.
- uses a default summary to place a blank line after the last row for each manager.
- changes the default underlining character for the duration of this PROC REPORT step.

Program

```
libname proclib 'SAS-data-library';


options nodate pageno=1 linesize=80 pagesize=60
        fmtsearch=(proclib);
```

The NOWD option runs PROC REPORT without the REPORT window and sends its output to the SAS procedure output. HEADLINE underlines all column headers and the spaces between them at the top of each panel of the report. FORMCHAR= sets the value of the second formatting character (the one that HEADLINE uses) to the tilde (~). Therefore, the tilde underlines the column headers in the output. HEADSKIP writes a blank line beneath the underlining that HEADLINE writes. LS= sets the line size for the report to 64, and PS= sets the page size to 18. PANELS= creates a multipanel report. Specifying PANELS=99 ensures that PROC REPORT fits as many panels as possible on one page. PSPACE=8 places 8 spaces between panels.

```
proc report data=grocery nowd headline
            formchar(2)='~'
            panels=99 pspace=6
            ls=64 ps=18;
```

The report contains a column for Manager, Department, and Sales.

```
    column manager department sales;
```

The values of all variables with the ORDER option in the DEFINE statement determine the order of the rows in the report. In this report, PROC REPORT arranges the rows first by the value of Manager (because it is the first variable in the COLUMN statement) and then, within each value of Manager, by the values of Department. The ORDER= option specifies the sort order for a variable. This report arranges the values of Manager by their formatted values and arranges the values of Department by their internal values (np1, np2, p1, and p2). FORMAT= specifies the formats to use in the report.

```
    define manager / order
                     order=formatted
```

```
                  format=$mgrfmt.;
   define department / order
                order=internal
                format=$deptfmt.;
   define sales / format=dollar7.2;
```

This BREAK statement produces a default summary after the last row for each manager. Because SKIP is the only option in the BREAK statement, each break consists of only a blank line.

```
   break after manager / skip;
```

The WHERE statement selects for the report only the observations for stores in the northwest or southwest sector. The TITLE statement specifies a title for the report.

```
   where sector='nw' or sector='sw';
   title "Sales for the Western Sectors";
run;
```

Output

```
                Sales for the Western Sectors                 1

Manager  Department    Sales      Manager  Department    Sales
~~~~~~~~~~~~~~~~~~~~~~~~~~~~      ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Adams    Paper         $40.00
         Canned       $225.00     Reveiz   Paper         $60.00
         Meat/Dairy   $350.00              Canned       $420.00
         Produce       $80.00              Meat/Dairy   $600.00
                                           Produce       $30.00
Brown    Paper         $45.00
         Canned       $230.00     Taylor   Paper         $53.00
         Meat/Dairy   $250.00              Canned       $120.00
         Produce       $73.00              Meat/Dairy   $130.00
                                           Produce       $50.00
Pelfrey  Paper         $45.00
         Canned       $420.00
         Meat/Dairy   $205.00
         Produce       $76.00
```

Example 9: Writing a Customized Summary on Each Page

Procedure features:

BREAK statement options:

OL
PAGE
SUMMARIZE

COMPUTE statement arguments:

with a computed variable as *report-item*
BEFORE *break-variable*
AFTER *break-variable* with conditional logic
BEFORE _PAGE_

DEFINE statement options:
NOPRINT

LINE statement:
pointer controls
quoted text
repeating a character string
variable values and formats

Data set: GROCERY on page 959

Formats: $SCTRFMT., $MGRFMT., and $DEPTFMT. on page 960

The report in this example displays a record of one day's sales for each store. The rows are arranged so that all the information about one store is together, and the information for each store begins on a new page. Some variables appear in columns. Others appear only in the page header that identifies the sector and the store's manager.

The header that appears at the top of each page is created with the _PAGE_ argument in the COMPUTE statement.

Profit is a computed variable based on the value of Sales and Department.

The text that appears at the bottom of the page depends on the total of Sales for the store. Only the first two pages of the report appear here.

Program

```
libname proclib 'SAS-data-library';


options nodate pageno=1 linesize=64 pagesize=30
        fmtsearch=(proclib);
```

The NOWD option runs PROC REPORT without the REPORT window and sends its output to the SAS procedure output. NOHEADER in the PROC REPORT statement suppresses the default column headers.

```
proc report data=grocery nowd
            headline headskip;
```

The TITLE statement specifies a title for the report.

```
    title 'Sales for Individual Stores';
```

The report contains a column for Sector, Manager, Department, Sales, and Profit, but the NOPRINT option suppresses the printing of the columns for Sector and Manager. The page header (created later in the program) includes their values. To get these variable values into the page header, Sector and Manager must be in the COLUMN statement.

```
column sector manager department sales Profit;
```

In this report, Sector, Manager, and Department are group variables. Each detail row of the report consolidates the information for all observations with the same values of the group variables. Profit is a computed variable whose values are calculated in the next section of the program. FORMAT= specifies the formats to use in the report.

```
define sector / group noprint;
define manager / group noprint;
define profit / computed format=dollar11.2;
define sales / analysis sum format=dollar11.2;
define department / group format=$deptfmt.;
```

Profit is computed as a percentage of Sales. For nonperishable items, the profit is 40% of the sale price. For perishable items the profit is 25%. Notice that in the compute block you must reference the variable Sales with a compound name (Sales.sum) that identifies both the variable and the statistic that you calculate with it.

```
compute profit;
   if department='np1' or department='np2'
      then profit=0.4*sales.sum;
   else profit=0.25*sales.sum;
endcomp;
```

This compute block executes at the top of each page, after PROC REPORT writes the title. It writes the page header for the current manager's store. The LEFT option left-justifies the text in the LINE statements. Each LINE statement writes the text in quotation marks just as it appears in the statement. The LINE statement writes a variable value with the format specified immediately after the variable's name. The at sign (@) specifies the column to write in.

```
 compute before _page_ / left;
   line sector $sctrfmt. ' Sector';
   line 'Store managed by ' manager $mgrfmt.;
   line ' ';
   line ' ';
   line ' ';
endcomp;
```

This BREAK statement creates a default summary after the last row for each manager. OL writes a row of hyphens above the summary line. SUMMARIZE writes the value of Sales (the only analysis or computed variable) in the summary line. The PAGE option starts a new page after each default summary so that the page header created in the preceding compute block always pertains to the correct manager.

```
break after manager / ol summarize page;
```

This compute block places conditional text in a customized summary that appears after the last detail row for each manager.

```
compute after manager;
```

The LENGTH statement assigns a length of 35 to the DATA step variable TEXT. In this particular case, the LENGTH statement is unnecessary because the longest version appears in the first IF/THEN statement. However, using the LENGTH statement ensures that even if the order of the conditional statements changes, TEXT will be long enough to hold the longest version.

```
   length text $ 35;
```

You cannot use the LINE statement in conditional statements (IF-THEN, IF-THEN/ELSE, and SELECT) because it does not take effect until PROC REPORT has executed all other statements in the compute block. These IF-THEN/ELSE statements assign a value to TEXT based on the value of Sales.sum in the summary row. A LINE statement writes that variable, whatever its value happens to be.

```
   if sales.sum lt 500 then
      text='Sales are below the target region.';
   else if sales.sum ge 500 and sales.sum lt 1000 then
      text='Sales are in the target region.';
   else if sales.sum ge 1000 then
      text='Sales exceeded goal!';
   line ' ';
   line text $35.;
endcomp;
run;
```

Output

```
                  Sales for Individual Stores                  1

Northeast Sector
Store managed by Alomar


              Department        Sales       Profit
              -------------------------------------

              Canned          $420.00      $168.00
              Meat/Dairy      $190.00       $47.50
              Paper            $90.00       $36.00
              Produce          $86.00       $21.50
                             ----------  ----------
                              $786.00      $196.50

                Sales are in the target region.
```

```
                  Sales for Individual Stores                  2

Northeast Sector
Store managed by Andrews


              Department        Sales       Profit
              -----------------------------------

              Canned          $420.00      $168.00
              Meat/Dairy      $300.00       $75.00
              Paper           $200.00       $80.00
              Produce         $125.00       $31.25
                            ----------  ----------
                            $1,045.00      $261.25

                      Sales exceeded goal!
```

Example 10: Calculating Percentages

Procedure features:

COLUMN statement arguments:

- PCTSUM
- SUM
- spanning heads

COMPUTE statement options:

- CHAR
- LENGTH=

DEFINE statement options:

- COMPUTED
- FLOW
- WIDTH=

RBREAK statement options:

- OL
- SUMMARIZE

Other features:

TITLE statement

Data set: GROCERY on page 959

Formats: $MGRFMT. and $DEPTFMT. on page 960

The summary report in this example shows the total sales for each store and the percentage that these sales represent of sales for all stores. Each of these columns has its own header. A single header also spans all the columns. This header looks like a title, but it differs from a title because it would be stored in a report definition. You must submit a null TITLE statement whenever you use the report definition, or the report will contain both a title and the spanning header.

The report includes a computed character variable, COMMENT, that flags stores with an unusually high percentage of sales. The text of COMMENT wraps across multiple rows. It makes sense to compute COMMENT only for individual stores.

Therefore, the compute block that does the calculation includes conditional code that prevents PROC REPORT from calculating COMMENT on the summary line.

Program

```
libname proclib 'SAS-data-library';


options nodate pageno=1 linesize=64 pagesize=60
        fmtsearch=(proclib);
```

The NOWD option runs PROC REPORT without the REPORT window and sends its output to the SAS procedure output. HEADLINE underlines all column headers and the spaces between them at the top of each page of the report. The null TITLE statement suppresses the title for the report.

```
proc report data=grocery nowd headline;
   title;
```

The COLUMN statement uses the text in quotation marks as a spanning header. The header spans all the columns in the report because they are all included in the pair of parentheses that contains the header. The COLUMN statement associates two statistics with Sales: Sum and Pctsum. The Sum statistic sums the values of Sales for all observations that are included in a row of the report. The Pctsum statistic shows what percentage that sum is of Sales for all observations in the report.

```
    column ('Individual Store Sales as a Percent of All Sales'
             sector manager sales,(sum pctsum) comment);
```

In this report, Sector and Manager are group variables. Each detail row represents a set of observations that have a unique combination of formatted values for all group variables. Sales is, by default, an analysis variable used to calculate the Sum statistic. However, because statistics are associated with Sales in the column statement, those statistics override the default. FORMAT= specifies the formats to use in the report. Text between quotation marks specifies the column header.

```
    define manager / group
                     format=$mgrfmt.;
    define sector / group
                    format=$sctrfmt.;
    define sales / format=dollar11.2
                   '';
    define sum / format=dollar9.2
                 'Total Sales';
```

The DEFINE statement for Pctsum specifies a column header, a format, and a column width of 8. The PERCENT. format presents the value of Pctsum as a percentage rather than a decimal. The DEFINE statement for COMMENT defines it as a computed variable and assigns it a column width of 20 and a blank column header. The FLOW option wraps the text for COMMENT onto multiple lines if it exceeds the column width.

```
    define pctsum / 'Percent of Sales' format=percent6. width=8;
    define comment / computed width=20 '' flow;
```

Options in the COMPUTE statement define COMMENT as a character variable with a length of 40.

```
    compute comment / char length=40;
```

This compute block creates a comment that says "Sales substantially above expectations." for every store where sales exceeded 15% of the sales for all stores. Of course, on the summary row for the report, the value of Pctsum is 100. However, it is inappropriate to flag this row as having exceptional sales. The automatic variable _BREAK_ distinguishes detail rows from summary rows. In a detail row, the value of _BREAK_ is blank. The THEN statement executes only on detail rows where the value of Pctsum exceeds 0.15.

```
       if sales.pctsum gt .15 and _break_ = ' '
       then comment='Sales substantially above expectations.';
       else comment=' ';
    endcomp;
```

This RBREAK statement creates a default summary at the end of the report. OL writes a row of hyphens above the summary line. SUMMARIZE writes the values of Sales.sum and Sales.pctsum in the summary line.

```
    rbreak after / ol summarize;
run;
```

Output

```
                                                                        1

      Individual Store Sales as a Percent of All Sales

                          Total   Percent
Sector     Manager        Sales  of Sales
------------------------------------------------------------
Northeast  Alomar       $786.00       12%
           Andrews    $1,045.00       17%   Sales substantially
                                            above expectations.
Northwest  Brown        $598.00        9%
           Pelfrey      $746.00       12%
           Reveiz     $1,110.00       18%   Sales substantially
                                            above expectations.
Southeast  Jones        $630.00       10%
           Smith        $350.00        6%
Southwest  Adams        $695.00       11%
           Taylor       $353.00        6%
                      ---------   --------
                      $6,313.00      100%
```

Example 11: How PROC REPORT Handles Missing Values

Procedure features:

PROC REPORT statement options:

MISSING

COLUMN statement

with the N statistic

Other features:

TITLE statement

Formats: $MGRFMT. on page 960

This example illustrates the difference between the way PROC REPORT handles missing values for group (or order or across) variables with and without the MISSING option. The differences in the reports are apparent if you compare the values of N for each row and compare the totals in the default summary at the end of the report.

Program with Data Set with No Missing Values

```
libname proclib 'SAS-data-library';


options nodate pageno=1 linesize=64 pagesize=60
        fmtsearch=(proclib);
```

GROCMISS is identical to GROCERY except that it contains some observations with missing values for Sector, Manager, or both.

```
data grocmiss;
   input Sector $ Manager $ Department $ Sales @@;
datalines;
se 1 np1 50    .  1 p1 100   se . np2 120   se 1 p2 80
se 2 np1 40    se 2 p1 300   se 2 np2 220   se 2 p2 70
nw 3 np1 60    nw 3 p1 600   .  3 np2 420   nw 3 p2 30
nw 4 np1 45    nw 4 p1 250   nw 4 np2 230   nw 4 p2 73
nw 9 np1 45    nw 9 p1 205   nw 9 np2 420   nw 9 p2 76
sw 5 np1 53    sw 5 p1 130   sw 5 np2 120   sw 5 p2 50
.  . np1 40    sw 6 p1 350   sw 6 np2 225   sw 6 p2 80
ne 7 np1 90    ne . p1 190   ne 7 np2 420   ne 7 p2 86
ne 8 np1 200   ne 8 p1 300   ne 8 np2 420   ne 8 p2 125
;
```

The NOWD option runs PROC REPORT without the REPORT window and sends its output to the SAS procedure output. HEADLINE underlines all column headers and the spaces between them.

```
proc report data=grocmiss nowd headline;
```

The report contains a column for Sector, Manager, the N statistic, and Sales.

```
   column sector manager N sales;
```

In this report, Sector and Manager are group variables. Sales is, by default, an analysis variable used to calculate the Sum statistic. Each detail row represents a set of observations that have a unique combination of formatted values for all group variables. The value of Sales in each detail row is the sum of Sales for all observations in the group. In this PROC REPORT step, the procedure does not include observations with a missing value for the group variable. FORMAT= specifies formats to use in the report.

```
   define sector / group format=$sctrfmt.;
   define manager / group format=$mgrfmt.;
   define sales / format=dollar9.2;
```

This RBREAK statement creates a default summary at the end of the report. DOL writes a row of equals signs above the summary line. SUMMARIZE writes the values of N and Sales.sum in the summary line.

```
   rbreak after / dol summarize;
```

The TITLE statement specifies a title for the report.

```
   title 'Summary Report for All Sectors and Managers';
run;
```

Output with No Missing Values

```
          Summary Report for All Sectors and Managers          1

            Sector     Manager          N      Sales
            ---------------------------------------
            Northeast  Alomar           3    $596.00
                       Andrews          4  $1,045.00
            Northwest  Brown            4    $598.00
                       Pelfrey          4    $746.00
                       Reveiz           3    $690.00
            Southeast  Jones            4    $630.00
                       Smith            2    $130.00
            Southwest  Adams            3    $655.00
                       Taylor           4    $353.00
                                 =========  =========
                                        31  $5,443.00
```

Program with Data Set with Missing Values

The MISSING option in the second PROC REPORT step includes the observations with missing values for the group variable.

```
proc report data=grocmiss nowd headline missing;
   column sector manager N sales;
   define sector / group format=$sctrfmt.;
   define manager / group format=$mgrfmt.;
   define sales / format=dollar9.2;
   rbreak after / dol summarize;
run;
```

Output with Missing Values

```
                                                               3

            Sector     Manager          N      Sales
            ---------------------------------------
                                        1     $40.00
                       Reveiz           1    $420.00
                       Smith            1    $100.00
            Northeast                   1    $190.00
                       Alomar           3    $596.00
                       Andrews          4  $1,045.00
            Northwest  Brown            4    $598.00
                       Pelfrey          4    $746.00
                       Reveiz           3    $690.00
            Southeast                   1    $120.00
                       Jones            4    $630.00
                       Smith            2    $130.00
            Southwest  Adams            3    $655.00
                       Taylor           4    $353.00
                                 =========  =========
                                        36  $6,313.00
```

Example 12: Creating and Processing an Output Data Set

Procedure features:

PROC REPORT statement options:

BOX
OUT=

DEFINE statement options:

ANALYSIS
GROUP
NOPRINT
SUM

Other features:

Data set options:

WHERE=

Data set: GROCERY on page 959

Formats: $MGRFMT. on page 960

This example uses WHERE processing as it builds an output data set. This technique enables you to do WHERE processing after you have consolidated multiple observations into a single row.

The first PROC REPORT step creates a report (which it does not display) in which each row represents all the observations from the input data set for a single manager. The second PROC REPORT step builds a report from the output data set. This report uses line-drawing characters to separate the rows and columns.

Program to Create Output Data Set

```
libname proclib 'SAS-data-library';


options nodate pageno=1 linesize=64 pagesize=60
        fmtsearch=(proclib);
```

The NOWD option runs PROC REPORT without the REPORT window and sends its output to the SAS procedure output. OUT= creates the output data set TEMP. The output data set contains a variable for each column in the report (Manager and Sales) as well as for the variable _BREAK_, which is not used in this example. Each observation in the data set represents a row of the report. Because Manager is a group variable and Sales is an analysis variable used to calculate the Sum statistic, each row in the report (and therefore each observation in the output data set) represents multiple observations from the input data set. In particular, each value of Sales in the output data set is the total of all values of Sales for that manager. The WHERE= data set option in the OUT= option filters those rows as PROC REPORT creates the output data set. Only those observations with sales that exceed $1,000 become observations in the output data set.

```
proc report data=grocery nowd
            out=temp( where=(sales gt 1000) );
```

```
   column manager sales;
```

Because the definitions of all report items in this report include the NOPRINT option, PROC REPORT does not print a report. However, the PROC REPORT step does execute and create an output data set.

```
   define manager / group noprint;
   define sales / analysis sum noprint;
run;
```

Output Showing the Output Data Set

This is the output data set that PROC REPORT creates. It is used as the input DATA step in the next PROC REPORT step.

```
                    The Data Set TEMP                          1

 Manager       Sales  _____________BREAK_____________
 3              1110
 8              1045
```

Program That Uses the Output Data Set

DATA= specifies the output data set from the previous PROC REPORT step as the input data set for this report. The BOX option draws an outline around the output, separates the column headers from the body of the report, and separates rows and columns of data. The TITLE statements specify a title for the report.

```
proc report data=temp box nowd;
   column manager sales;
   define manager / group format=$mgrfmt.;
   define sales / analysis sum format=dollar11.2;
   title 'Managers with Daily Sales';
   title2 'of over';
   title3 'One Thousand Dollars';
run;
```

Report Based on the Output Data Set

```
                  Managers with Daily Sales                    1
                           of over
                      One Thousand Dollars

                     ----------------------
                     |Manager        Sales|
                     |--------------------|
                     |Andrews|   $1,045.00|
                     |-------+------------|
                     |Reveiz |   $1,110.00|
                     ----------------------
```

Example 13: Storing Computed Variables as Part of a Data Set

Procedure features:
PROC REPORT statement options:
OUT=
COMPUTE statement:
with a computed variable as *report-item*
DEFINE statement options:
COMPUTED
Other features: CHART procedure
Data set: GROCERY on page 959
Formats: $SCTRFMT. on page 960

The report in this example
- creates a computed variable
- stores it in an output data set
- uses that data set to create a chart based on the computed variable.

Program that Creates the Output Data Set

```
libname proclib 'SAS-data-library';


options nodate pageno=1 linesize=64 pagesize=60
        fmtsearch=(proclib);
```

The NOWD option runs PROC REPORT without the REPORT window and sends its output to the SAS procedure output. OUT= creates the output data set PROFIT.

```
title;
proc report data=grocery nowd out=profit;
```

The report contains a column for Manager, Department, Sales, and Profit, which is not in the input data set. Because the purpose of this report is to generate an output data set to use in another procedure, the report layout simply uses the default usage for all the data set variables to list all the observations. DEFINE statements for the data set variables are unnecessary.

```
    column sector manager department sales Profit;
```

The COMPUTED option tells PROC REPORT that Profit is defined in a compute block somewhere in the PROC REPORT step.

```
    define profit / computed;
```

Profit is computed as a percentage of Sales. For nonperishable items, the profit is 40% of the sale price. For perishable items the profit is 25%. Notice that in the compute block, you must reference the variable Sales with a compound name (Sales.sum) that identifies both the variable and the statistic that you calculate with it.

```
       /* Compute values for Profit. */
    compute profit;
       if department='np1' or department='np2' then profit=0.4*sales.sum;
       else profit=0.25*sales.sum;
    endcomp;
run;
```

The Output Data Set

This is the output data set created by PROC REPORT. It is used as input for PROC CHART.

The Data Set PROFIT 1

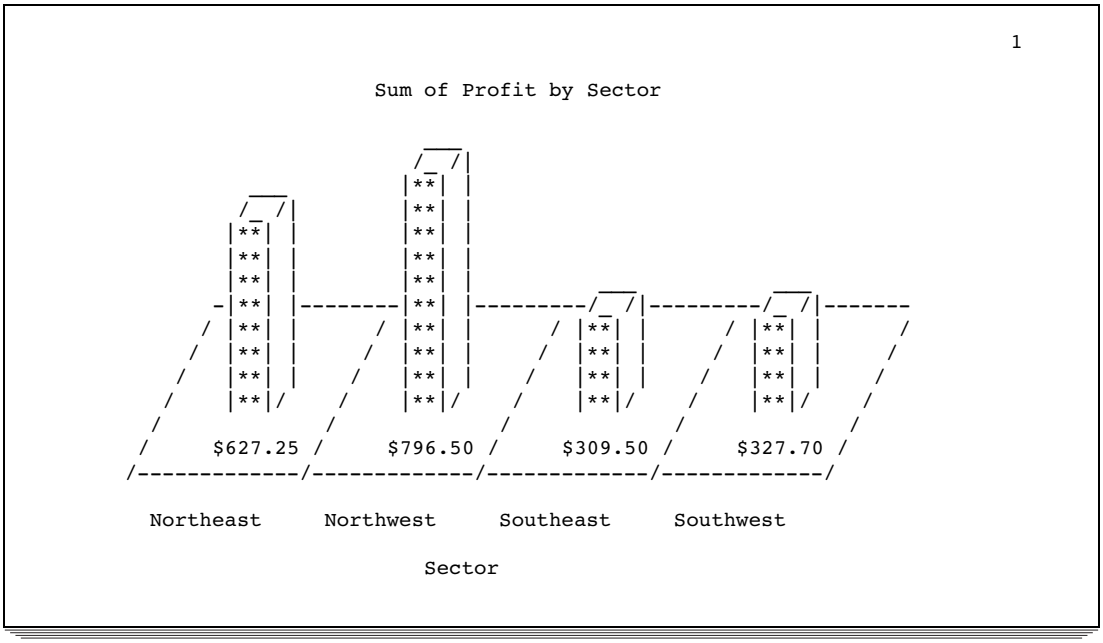
Sector	Manager	Department	Sales	Profit	_BREAK__
se	1	np1	50	20	
se	1	p1	100	25	
se	1	np2	120	48	
se	1	p2	80	20	
se	2	np1	40	16	
se	2	p1	300	75	
se	2	np2	220	88	
se	2	p2	70	17.5	
nw	3	np1	60	24	
nw	3	p1	600	150	
nw	3	np2	420	168	
nw	3	p2	30	7.5	
nw	4	np1	45	18	
nw	4	p1	250	62.5	
nw	4	np2	230	92	
nw	4	p2	73	18.25	
nw	9	np1	45	18	
nw	9	p1	205	51.25	
nw	9	np2	420	168	
nw	9	p2	76	19	
sw	5	np1	53	21.2	
sw	5	p1	130	32.5	
sw	5	np2	120	48	
sw	5	p2	50	12.5	
sw	6	np1	40	16	
sw	6	p1	350	87.5	
sw	6	np2	225	90	
sw	6	p2	80	20	
ne	7	np1	90	36	
ne	7	p1	190	47.5	
ne	7	np2	420	168	
ne	7	p2	86	21.5	
ne	8	np1	200	80	
ne	8	p1	300	75	
ne	8	np2	420	168	
ne	8	p2	125	31.25	

Program That Uses the Output Data Set

PROC CHART uses the output data set from the previous PROC REPORT step to chart the sum of Profit for each sector.

```
title;
options nodate pageno=1 linesize=80 pagesize=60
        fmtsearch=(proclib);
proc chart data=profit;
   block sector / sumvar=profit;
   format sector $sctrfmt.;
   format profit dollar7.2;
run;
```

Output from Processing the Output Data Set



Example 14: Using a Format to Create Groups

Procedure features:
DEFINE statement options:
GROUP

Other features: FORMAT procedure

Data set: GROCERY on page 959

Formats: $MGRFMT. on page 960

This example shows how to use formats to control the number of groups that PROC REPORT creates. The program creates a format for Department that classifies the four departments as one of two types: perishable or nonperishable. Consequently, when Department is an across variable, PROC REPORT creates only two columns instead of four. The column header is the formatted value of the variable.

Program

```
libname proclib 'SAS-data-library';


options nodate pageno=1 linesize=64 pagesize=60
        fmtsearch=(proclib);
```

PROC FORMAT creates a format for Department. This variable has four different values in the data set, but the format has only two values.

```
proc format;
   value $perish 'p1','p2'='Perishable'
                 'np1','np2'='Nonperishable';
run;
```

The NOWD option runs the REPORT procedure without the REPORT window and sends its output to the SAS procedure output. HEADLINE underlines all column headers and the spaces between them at the top of each page of the report. HEADSKIP writes a blank line beneath the underlining that HEADLINE writes.

```
proc report data=grocery nowd
     headline
     headskip;
```

Department and Sales are separated by a comma in the COLUMN statement, so they collectively determine the contents of the column that they define. Because Sales is an analysis variable, its values fill the cells created by these two variables. The report also contains a column for Manager and a column for Sales by itself (which is the sales for all departments).

```
    column manager department,sales sales;
```

Manager is a group variable. Each detail row of the report consolidates the information for all observations with the same value of Manager. Department is an across variable. PROC REPORT creates a column and a column header for each formatted value of Department. ORDER=FORMATTED arranges the values of Manager and Department alphabetically according to their formatted values. FORMAT= specifies the formats to use. The empty quotation marks in the definition of Department specify a blank column header, so no header spans all the departments. However, PROC REPORT uses the formatted values of Department to create a column header for each individual department.

```
    define manager / group order=formatted
                     format=$mgrfmt.;
    define department / across order=formatted
                  format=$perish. '';
```

Sales is an analysis variable used to calculate the Sum statistic. Sales appears twice in the COLUMN statement, and the same definition applies to both occurrences. FORMAT= specifies the format to use in the report. WIDTH= specifies the width of the column. Notice that the column headers for the columns that both Department and Sales create are a combination of the header for Department and the (default) header for Sales.

```
    define sales / analysis sum
                   format=dollar9.2 width=13;
```

This COMPUTE statement begins a compute block that produces a customized summary at the end of the report. The LINE statement places the quoted text and the value of Sales.sum (with the DOLLAR9.2 format) in the summary. An ENDCOMP statement must end the compute block.

```
compute after;
   line ' ';
   line 'Total sales for these stores were: '
         sales.sum dollar9.2;
endcomp;
```

The TITLE statement specifies a title for the report.

```
title 'Sales Summary for All Stores';
run;
```

Output

```
                Sales Summary for All Stores                1

            Nonperishable     Perishable
Manager              Sales          Sales          Sales
-------------------------------------------------------

Adams             $265.00        $430.00        $695.00
Alomar            $510.00        $276.00        $786.00
Andrews           $620.00        $425.00      $1,045.00
Brown             $275.00        $323.00        $598.00
Jones             $260.00        $370.00        $630.00
Pelfrey           $465.00        $281.00        $746.00
Reveiz            $480.00        $630.00      $1,110.00
Smith             $170.00        $180.00        $350.00
Taylor            $173.00        $180.00        $353.00

    Total sales for these stores were: $6,313.00
```

Example 15: Specifying Style Elements for HTML Output in the PROC REPORT Statement

Procedure features: STYLE= option in the PROC REPORT statement
Other features: ODS HTML statement
Data set: GROCERY on page 959
Formats: $MGRFMT. and $DEPTFMT. on page 960

This example creates HTML files and sets the style elements for each location in the report in the PROC REPORT statement.

Program

```
libname proclib 'SAS-data-library';
```

```
options nodate fmtsearch=(proclib);
```

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

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

The NOWD option runs PROC REPORT without the REPORT window. In this case, SAS writes the output to both the traditional procedure output and to the HTML body file.

```
proc report data=grocery nowd headline headskip
```

This STYLE= option sets the style element for the structural part of the report. Because no style element is specified, PROC REPORT uses all the style attributes of the default style element for this location except for CELLSPACING=, BORDERWIDTH=, and BORDERCOLOR=.

```
style(report)=[cellspacing=5 borderwidth=10 bordercolor=blue]
```

This STYLE= option sets the style element for all column headers. Because no style element is specified, PROC REPORT uses all the style attributes of the default style element for this location except for those that are specified here.

```
style(header)=[foreground=yellow font_face=lucida
               font_style=italic font_size=6]
```

This STYLE= option sets the style element for all the cells in all the columns. Because no style element is specified, PROC REPORT uses all the style attributes of the default style element for this location except for those that are specified here.

```
style(column)=[foreground=moderate brown
               font_face=helvetica font_size=4]
```

This STYLE= option sets the style element for all the LINE statement in all compute blocks. Because no style element is specified, PROC REPORT uses all the style attributes of the default style element for this location except for those that are specified here.

```
style(lines)=[foreground=white background=black font_face=lucida
              font_style=italic font_weight=bold font_size=5]
```

This STYLE= option sets the style element for all the default summary lines. Because no style element is specified, PROC REPORT uses all the style attributes of the default style element for this location except for those that are specified here.

```
style(summary)=[foreground=cx3e3d73 background=cxaeadd9
                font_face=helvetica font_size=3 just=r];
```

The report contains a column for Manager, Department, and Sales.

```
    column manager department sales;
```

In this report Manager is an order variable. PROC REPORT arranges the rows first by the value of Manager (because it is the first variable in the COLUMN statement). ORDER= specifies that values of Manager are arranged according to their formatted values. FORMAT= specifies the format to use for this variable. Text in quotation marks specifies the column headers.

```
    define manager / order
                     order=formatted
```

```
                    format=$mgrfmt.
                    'Manager';
```

In this report Department is an order variable. PROC REPORT arranges the rows first by the value of Manager (because it is the first variable in the COLUMN statement), then by the value of Department. ORDER= specifies that values of Department are arranged according to their internal values. FORMAT= specifies the format to use for this variable. Text in quotation marks specifies the column header.

```
define department / order
                    order=internal
                    format=$deptfmt.
                    'Department';
```

The BREAK statement produces a default summary after the last row for each manager. SUMMARIZE writes the values of Sales (the only analysis or computed variable in the report) in the summary line. PROC REPORT sums the values of Sales for each manager because Sales is an analysis variable that is used to calculate the Sum statistic.

```
break after manager / summarize;
```

The COMPUTE statement begins a compute block that produces a customized summary after each value of Manager. The LINE statement places the quoted text and the values of Manager and Sales.sum (with the formats $MGRFMT. and DOLLAR7.2) in the summary. An ENDCOMP statement must end the compute block.

```
compute after manager;
   line 'Subtotal for ' manager $mgrfmt. 'is '
         sales.sum dollar7.2 '.';
endcomp;
```

This COMPUTE statement begins a compute block that executes at the end of the report. The LINE statement writes the quoted text and the value of Sales.sum (with the DOLLAR7.2 format). An ENDCOMP statement must end the compute block.

```
compute after;
    line 'Total for all departments is: '
         sales.sum dollar7.2 '.';
endcomp;
```

The WHERE statement selects for the report only the observations for stores in the southeast sector.

```
where sector='se';
```

The TITLE statement specifies the title.

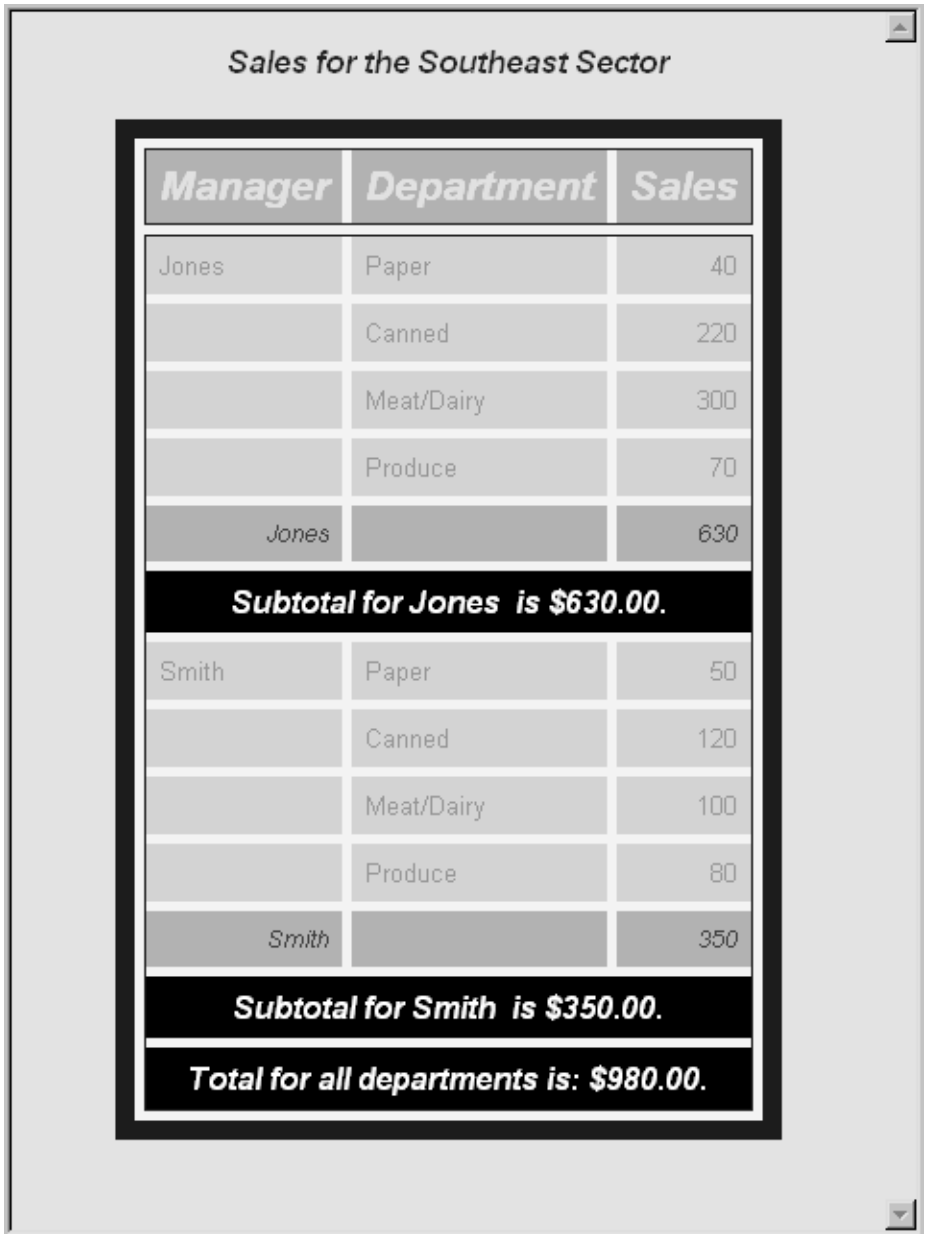
```
    title 'Sales for the Southeast Sector';
run;
```

The ODS HTML statement closes the HTML destination.

```
ods html close;
```

HTML Body File

This report uses customized style elements to control things like font faces, font sizes, and justification, as well as the width of the border of the table and the width of the spacing between the cells. All these customizations are done in the PROC REPORT statement.



Sales for the Southeast Sector

Manager	Department	Sales
Jones	Paper	40
	Canned	220
	Meat/Dairy	300
	Produce	70
Jones		*630*
Subtotal for Jones is $630.00.		
Smith	Paper	50
	Canned	120
	Meat/Dairy	100
	Produce	80
Smith		*350*
Subtotal for Smith is $350.00.		
Total for all departments is: $980.00.		

Example 16: Specifying Style Elements for HTML Output in Multiple Statements

Procedure features:

STYLE= option in

PROC REPORT statement
CALL DEFINE statement

COMPUTE statement
DEFINE statement

Other features: ODS HTML statement

Data set: GROCERY on page 959

Formats: $MGRFMT. on page 960 and $DEPTFMT. on page 960

This example creates HTML files and sets the style elements for each location in the report in the PROC REPORT statement. It then overrides some of these settings by specifying style elements in other statements.

Program

```
libname proclib 'SAS-data-library';

options nodate fmtsearch=(proclib);
```

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

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

The NOWD option runs PROC REPORT without the REPORT window. In this case, SAS writes the output to both the traditional procedure output and to the HTML body file.

```
proc report data=grocery nowd headline headskip
```

This STYLE= option sets the style element for the structural part of the report. Because no style element is specified, PROC REPORT uses all the style attributes of the default style element for this location except for CELLSPACING= and BORDERWIDTH=.

```
style(report)=[cellspacing=5 borderwidth=10 bordercolor=blue]
```

This STYLE= option sets the style element for all column headers. Because no style element is specified, PROC REPORT uses all the style attributes of the default style element for this location except for those that are specified here.

```
style(header)=[foreground=yellow font_face=lucida
               font_style=italic font_size=6]
```

This STYLE= option sets the style element for all the cells in all the columns. Because no style element is specified, PROC REPORT uses all the style attributes of the default style element for this location except for those that are specified here.

```
style(column)=[foreground=moderate brown
               font_face=helvetica font_size=4]
```

This STYLE= option sets the style element for all the LINE statement in all compute blocks. Because no style element is specified, PROC REPORT uses all the style attributes of the default style element for this location except for those that are specified here.

```
style(lines)=[foreground=white background=black font_face=lucida
              font_style=italic font_weight=bold font_size=5]
```

This STYLE= option sets the style element for all the default summary lines. Because no style element is specified, PROC REPORT uses all the style attributes of the default style element for this location except for those that are specified here.

```
style(summary)=[foreground=cx3e3d73 background=cxaeadd9
                font_face=helvetica font_size=3 just=r];
```

The report contains a column for Manager, Department, and Sales.

```
column manager department sales;
```

In this report Manager is an order variable. PROC REPORT arranges the rows first by the value of Manager (because it is the first variable in the COLUMN statement). ORDER= specifies that values of Manager are arranged according to their formatted values. FORMAT= specifies the format to use for this variable. Text in quotation marks specifies the column headers.

```
define manager / order
                 order=formatted
                 format=$mgrfmt.
                 'Manager'
```

The STYLE= option sets the foreground and background colors of the column header for Manager. The other style attributes for the column header will match those that were established for the HEADER location in the PROC REPORT statement.

```
                 style(header)=[foreground=white
                                background=black];
```

In this report Department is an order variable. PROC REPORT arranges the rows first by the value of Manager (because it is the first variable in the COLUMN statement), then by the value of Department. ORDER= specifies that values of Department are arranged according to their internal values. FORMAT= specifies the format to use for this variable. Text in quotation marks specifies the column header.

```
define department / order
                    order=internal
                    format=$deptfmt.
                    'Department'
```

The STYLE= option sets the font of the cells in the column Department to **italic**. The other style attributes for the cells will match those that were established for the COLUMN location in the PROC REPORT statement.

```
                    style(column)=[font_style=italic];
```

The BREAK statement produces a default summary after the last row for each manager. SUMMARIZE writes the values of Sales (the only analysis or computed variable in the report) in the summary line. PROC REPORT sums the values of Sales for each manager because Sales is an analysis variable that is used to calculate the Sum statistic.

```
break after manager / summarize;
```

The COMPUTE statement begins a compute block that produces a customized summary at the end of the report. This STYLE= option specifies the style element to use for the text that is created by the LINE statement in this compute block. This style element switches the foreground and background colors that were specified for the LINES location in the PROC REPORT statement. It also changes the font style, the font weight, and the font size.

```
    compute after manager
            / style=[font_style=roman font_size=3 font_weight=bold
              background=white foreground=black];
```

The LINE statement places the quoted text and the values of Manager and Sales.sum (with the formats $MGRFMT. and DOLLAR7.2) in the summary. An ENDCOMP statement must end the compute block.

```
       line 'Subtotal for ' manager $mgrfmt. 'is '
             sales.sum dollar7.2 '.';
    endcomp;
```

This compute block specifies a background color and a bold font for all cells in the Sales column that contain values of 100 or greater and that are not summary lines.

```
    compute sales;
       if sales.sum>100 and _break_=' ' then
       call define(_col_, "style",
                   "style=[background=yellow
                           font_face=helvetica
                           font_weight=bold]");
    endcomp;
```

This COMPUTE statement begins a compute block that executes at the end of the report. The LINE statement writes the quoted text and the value of Sales.sum (with the DOLLAR7.2 format). An ENDCOMP statement must end the compute block.

```
    compute after;
        line 'Total for all departments is: '
             sales.sum dollar7.2 '.';
    endcomp;
```

The WHERE statement selects for the report only the observations for stores in the southeast sector.

```
    where sector='se';
```

The TITLE statement specifies the title.

```
    title 'Sales for the Southeast Sector';
run;
```

The ODS HTML statement closes the HTML destination.

```
ods html close;
```

HTML Body File

Display 32.1 HTML Body File

This report uses customized style elements to control things like font faces, font sizes, and justification, as well as the width of the border of the table and the width of the spacing between cells. Some of these customizations are done in the PROC REPORT statement. Others are set in subordinate statements.

Sales for the Southeast Sector

Manager	Department	Sales
Jones	Paper	40
	Canned	220
	Meat/Dairy	300
	Produce	70
Jones		*630*
Subtotal for Jones is $630.00.		
Smith	Paper	50
	Canned	120
	Meat/Dairy	100
	Produce	80
Smith		*350*
Subtotal for Smith is $350.00.		
Total for all departments is: $980.00.		

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

SAS® Procedures Guide, Version 8

ISBN 1–58025–482–9

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

1st printing, October 1999