Overview

The PRINT procedure prints the observations in a SAS data set, using all or some of the variables. You can create a variety of reports ranging from a simple listing to a highly customized report that groups the data and calculates totals and subtotals for numeric variables.

Output 27.1 on page 784 illustrates the simplest kind of report that you can produce. The statements that produce the output follow. A DATA step on page 799 creates the data set EXPREV.

```sas
options nodate pageno=1 linesize=64
      pagesize=60;

proc print data=exprev;
run;
```
Output 27.1  Simple Report Produced with PROC PRINT

<table>
<thead>
<tr>
<th>Obs</th>
<th>Region</th>
<th>State</th>
<th>Month</th>
<th>Expenses</th>
<th>Revenues</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Southern</td>
<td>GA</td>
<td>JAN95</td>
<td>2000</td>
<td>8000</td>
</tr>
<tr>
<td>2</td>
<td>Southern</td>
<td>GA</td>
<td>FEB95</td>
<td>1200</td>
<td>6000</td>
</tr>
<tr>
<td>3</td>
<td>Southern</td>
<td>FL</td>
<td>FEB95</td>
<td>8500</td>
<td>11000</td>
</tr>
<tr>
<td>4</td>
<td>Northern</td>
<td>NY</td>
<td>FEB95</td>
<td>3000</td>
<td>4000</td>
</tr>
<tr>
<td>5</td>
<td>Northern</td>
<td>NY</td>
<td>MAR95</td>
<td>6000</td>
<td>5000</td>
</tr>
<tr>
<td>6</td>
<td>Southern</td>
<td>FL</td>
<td>MAR95</td>
<td>9800</td>
<td>13500</td>
</tr>
<tr>
<td>7</td>
<td>Northern</td>
<td>MA</td>
<td>MAR95</td>
<td>1500</td>
<td>1000</td>
</tr>
</tbody>
</table>

Output 27.2 on page 785 is a customized report that is produced by PROC PRINT. The statements that create this report:
- customize the title and the column headings
- place dollar signs and commas in numeric output
- selectively include and control the order of variables in the report
- group the data by JobCode
- sum the values for Salary for each job code and for all job codes.

For an explanation of the program that produces this report, see Example 8 on page 816.
### Output 27.2  
Customized Report Produced with PROC PRINT

<table>
<thead>
<tr>
<th>Job Code</th>
<th>Gender</th>
<th>Annual Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>FA1</td>
<td>F</td>
<td>$23,177.00</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>$22,454.00</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>$22,268.00</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>---------------</td>
</tr>
<tr>
<td>FA1</td>
<td></td>
<td>$67,899.00</td>
</tr>
<tr>
<td>FA2</td>
<td>F</td>
<td>$28,888.00</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>$27,787.00</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>$28,572.00</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>---------------</td>
</tr>
<tr>
<td>FA2</td>
<td></td>
<td>$85,247.00</td>
</tr>
<tr>
<td>FA3</td>
<td>F</td>
<td>$32,886.00</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>$33,419.00</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>$32,217.00</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>---------------</td>
</tr>
<tr>
<td>FA3</td>
<td></td>
<td>$98,522.00</td>
</tr>
<tr>
<td>ME1</td>
<td>M</td>
<td>$29,769.00</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>$28,072.00</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>$28,619.00</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>---------------</td>
</tr>
<tr>
<td>ME1</td>
<td></td>
<td>$86,460.00</td>
</tr>
<tr>
<td>ME2</td>
<td>F</td>
<td>$35,108.00</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>$34,929.00</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>$35,345.00</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>$35,925.00</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>$35,090.00</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>$35,185.00</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>---------------</td>
</tr>
<tr>
<td>ME2</td>
<td></td>
<td>$212,582.00</td>
</tr>
<tr>
<td>ME3</td>
<td>M</td>
<td>$43,025.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$593,735.00</td>
</tr>
</tbody>
</table>

### Procedure Syntax

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

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

```sas
PROC PRINT <option(s)>;
  BY <DESCENDING> variable1 <...<DESCENDING> variable-n>
    <NOTSORTED>;
  PAGEBY BY-variable;
```

---

**Expenses Incurred for Salaries for Flight Attendants and Mechanics**

Job Code | Gender | Annual Salary
---------|--------|---------------
FA1      | F      | $23,177.00    |
          | F      | $22,454.00    |
          | M      | $22,268.00    |
---------|--------|---------------|
FA1      |        | $67,899.00    |
FA2      | F      | $28,888.00    |
          | F      | $27,787.00    |
          | M      | $28,572.00    |
---------|--------|---------------|
FA2      |        | $85,247.00    |
FA3      | F      | $32,886.00    |
          | F      | $33,419.00    |
          | M      | $32,217.00    |
---------|--------|---------------|
FA3      |        | $98,522.00    |
ME1      | M      | $29,769.00    |
          | M      | $28,072.00    |
          | M      | $28,619.00    |
---------|--------|---------------|
ME1      |        | $86,460.00    |
ME2      | F      | $35,108.00    |
          | F      | $34,929.00    |
          | M      | $35,345.00    |
          | M      | $35,925.00    |
          | M      | $35,090.00    |
          | M      | $35,185.00    |
---------|--------|---------------|
ME2      |        | $212,582.00   |
ME3      | M      | $43,025.00    |
          |        | $593,735.00   |
**SUMBY** BY-variable
**ID** variable(s);
**SUM** variable(s);
**VAR** variable(s);

<table>
<thead>
<tr>
<th>To do this</th>
<th>Use this statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Produce a separate section of the report for each BY group</td>
<td>BY</td>
</tr>
<tr>
<td>Identify observations by the formatted values of the variables that you list instead of by observation numbers</td>
<td>ID</td>
</tr>
<tr>
<td>Control page ejects that occur before a page is full</td>
<td>PAGEBY</td>
</tr>
<tr>
<td>Limit the number of sums that appear in the report</td>
<td>SUMBY</td>
</tr>
<tr>
<td>Total values of numeric variables</td>
<td>SUM</td>
</tr>
<tr>
<td>Select variables that appear in the report and determine their order</td>
<td>VAR</td>
</tr>
</tbody>
</table>

**PROC PRINT Statement**

**PROC PRINT** <option(s)>;

<table>
<thead>
<tr>
<th>To do this</th>
<th>Use this option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the input data set</td>
<td>DATA=</td>
</tr>
<tr>
<td>Control general format</td>
<td></td>
</tr>
<tr>
<td>Write a blank line between observations</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>Print the number of observations in the data set, in BY groups, or both, and specify explanatory text to print with the number</td>
<td>N=</td>
</tr>
<tr>
<td>Suppress the column in the output that identifies each observation by number</td>
<td>NOOBS</td>
</tr>
<tr>
<td>Specify a column header for the column that identifies each observation by number</td>
<td>OBS=</td>
</tr>
<tr>
<td>Round unformatted numeric values to two decimal places</td>
<td>ROUND</td>
</tr>
<tr>
<td>Control page format</td>
<td></td>
</tr>
<tr>
<td>Format the rows on a page</td>
<td>ROWS=</td>
</tr>
<tr>
<td>Use each variable’s formatted width as its column width on all pages</td>
<td>WIDTH=UNIFORM</td>
</tr>
<tr>
<td>Control column format</td>
<td></td>
</tr>
</tbody>
</table>
The PRINT Procedure

PROC PRINT Statement

To do this
Control the orientation of the column headings
Use variables' labels as column headings
Specify the split character, which controls line breaks in column headings
Determine the column width for each variable

Use this option
HEADING=
LABEL or SPLIT=
SPLIT=
WIDTH=

Options

DATA=SAS-data-set
specifies the SAS data set to print.
Main discussion: “Input Data Sets” on page 18

DOUBLE
writes a blank line between observations.
Alias: D
Restriction: This option has no effect on the HTML output.
Featured in: Example 1 on page 799

HEADING=direction
controls the orientation of the column headings, where direction is one of the following:
HORIZONTAL
prints all column headings horizontally.
Alias: H
VERTICAL
prints all column headings vertically.
Alias: V
Default: Headings are either all horizontal or all vertical. If you omit HEADING=, PROC PRINT determines the direction of the column headings as follows:
- If you do not use LABEL, spacing dictates whether column headings are vertical or horizontal.
- If you use LABEL and at least one variable has a label, all headings are horizontal.

LABEL
uses variables' labels as column headings.
Alias: L
Default: If you omit LABEL, PROC PRINT uses the variable's name as the column heading even if the PROC PRINT step contains a LABEL statement. If a variable does not have a label, PROC PRINT uses the variable's name as the column heading.
Interaction: By default, if you specify LABEL and at least one variable has a label, PROC PRINT prints all column headings horizontally. Therefore, using LABEL may increase the number of pages of output. (Use HEADING=VERTICAL in the PROC PRINT statement to print vertical column headings.)
**Interaction:** PROC PRINT sometimes conserves space by splitting labels across multiple lines. Use SPLIT= in the PROC PRINT statement to control where these splits occur. You do not need to use LABEL if you use SPLIT=.

**Tip:** To create a blank column header for a variable, use this LABEL statement in your PROC PRINT step:

```plaintext
label variable-name='00'x;
```

**See also:** For information on using the LABEL statement to create temporary labels in procedures see Chapter 3, “Statements with the Same Function in Multiple Procedures,” on page 61. For information on using the LABEL statement in a DATA step to create permanent labels, see the section on statements in SAS Language Reference Dictionary.

**Featured in:** Example 3 on page 803

**Note:** The SAS system option LABEL must be in effect in order for any procedure to use labels. For more information see the section on system options in SAS Language Reference Dictionary.

\[ \text{N<="string-1" <"string-2"\geq} \]

prints the number of observations in the data set, in BY groups, or both and specifies explanatory text to print with the number.

<table>
<thead>
<tr>
<th>If you use the N option ...</th>
<th>PROC PRINT ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>with neither a BY nor a SUM statement</td>
<td>prints the number of observations in the data set at the end of the report and labels the number with the value of string-1.</td>
</tr>
<tr>
<td>with a BY statement</td>
<td>prints the number of observations in the BY group at the end of each BY group and labels the number with the value of string-1.</td>
</tr>
<tr>
<td>with a BY statement and a SUM statement</td>
<td>prints the number of observations in the BY group at the end of each BY group and prints the number of observations in the data set at the end of the report. The numbers for BY groups are labeled with string-1; the number for the entire data set is labeled with string-2.</td>
</tr>
</tbody>
</table>

**Featured in:** Example 2 on page 801 (alone)  
Example 3 on page 803 (with a BY statement)  
Example 4 on page 805 (with a BY statement and a SUM statement)

**NOOBS**

suppresses the observation number in the output.

**Featured in:** Example 3 on page 803

**OBS="column-header”**

specifies a column header for the column that identifies each observation by number.

**Tip:** OBS= honors the split character (see the discussion of SPLIT= on page 790).

**Featured in:** Example 2 on page 801
ROUND

rounds unformatted numeric values to two decimal places. (Formatted values are already rounded by the format to the specified number of decimal places.) For both formatted and unformatted variables, PROC PRINT uses these rounded values to calculate any sums in the report.

If you omit ROUND, PROC PRINT adds the actual values of the rows to obtain the sum even though it displays the formatted (rounded) values. Any sums are also rounded by the format, but they include only one rounding error, that of rounding the sum of the actual values. The ROUND option, on the other hand, rounds values before summing them, so there may be multiple rounding errors. The results without ROUND are more accurate, but ROUND is useful for published reports where it is important for the total to be the sum of the printed (rounded) values.

Be aware that the results from PROC PRINT with the ROUND option may differ from the results of summing the same data with other methods such as PROC MEANS or the DATA step. Consider a simple case in which

- the data set contains three values for X: .003, .004, and .009.
- X has a format of 5.2.

Depending on how you calculate the sum, you can get three different answers: 0.02, 0.01, and 0.016. Figure 27.1 on page 789 shows the results of calculating the sum with PROC PRINT (without and with the ROUND option) and PROC MEANS.

<table>
<thead>
<tr>
<th>Actual Values</th>
<th>PROC PRINT without the ROUND option</th>
<th>PROC PRINT with the ROUND option</th>
<th>PROC MEANS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OBS X</td>
<td>OBS X</td>
<td>Analysis Variable : X</td>
</tr>
<tr>
<td>0.003</td>
<td>1 0.00</td>
<td>1 0.00</td>
<td>Sum</td>
</tr>
<tr>
<td>0.004</td>
<td>2 0.00</td>
<td>2 0.00</td>
<td>0.0160000</td>
</tr>
<tr>
<td>0.009</td>
<td>3 0.01</td>
<td>3 0.01</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.02</td>
<td>0.01</td>
<td></td>
</tr>
</tbody>
</table>

Notice that the sum produced without the ROUND option (.02) is closer to the actual result (0.16) than the sum produced with ROUND (0.01). However, the sum produced with ROUND reflects the numbers displayed in the report.

Alias: R

CAUTION:
Do not use ROUND with PICTURE formats. ROUND is for use with numeric values. SAS procedures treat variables that have picture formats as character variables. Using ROUND with such variables may lead to unexpected results.

ROWS= page-format

formats rows on a page. Currently, PAGE is the only value that you can use for page-format:
PAGE

prints only one row of variables for each observation per page. When you use
ROWS=PAGE, PROC PRINT does not divide the page into sections; it prints as
many observations as possible on each page. If the observations do not fill the last
page of the output, PROC PRINT divides the last page into sections and prints all
the variables for the last few observations.

Restriction: Physical page size does not mean the same thing in HTML output as it
does in traditional procedure output. Therefore, HTML output from PROC PRINT
appears the same whether or not you use ROWS=.

Tip: The PAGE value can reduce the number of pages in the output if the data set
contains large numbers of variables and observations. However, if the data set
contains a large number of variables but few observations, the PAGE value can
increase the number of pages in the output.

See also: “Page Layout” on page 796 for discussion of the default layout.

Featured in: Example 7 on page 813

SPLIT='split-character’

specifies the split character, which controls line breaks in column headers. It also
uses labels as column headers. PROC PRINT breaks a column heading when it
reaches the split character and continues the header on the next line. The split
character is not part of the column heading although each occurrence of the split
character counts toward the 256-character maximum for a label.

Alias: S=

Interaction: You do not need to use both LABEL and SPLIT= because SPLIT=
implies the use of labels.

Interaction: The OBS= option honors the split character. (See the discussion of
OBS= on page 788).

Featured in: Example 2 on page 801

Note: PROC PRINT does not split labels of BY variables in the heading preceding
each BY group even if you specify SPLIT=. Instead, PROC PRINT treats the split
character as part of the label. Therefore, you probably want to avoid using a split
character when you are using the BY statement.

UNIFORM

See WIDTH=UNIFORM on page 791.

WIDTH=column-width

determines the column width for each variable. The value of column-width must be
one of the following:

FULL

uses a variable's formatted width as the column width. If the variable does not
have a format that explicitly specifies a field width, PROC PRINT uses the default
width. For a character variable, the default width is the length of the variable.
For a numeric variable, the default width is 12. When you use WIDTH=FULL, the
column widths do not vary from page to page.

Tip: Using WIDTH=FULL can reduce execution time.

MINIMUM

uses for each variable the minimum column width that accommodates all values of
the variable.

Alias: MIN
UNIFORM

uses each variable's formatted width as its column width on all pages. If the variable does not have a format that explicitly specifies a field width, PROC PRINT uses the widest data value as the column width. When you specify WIDTH=UNIFORM, PROC PRINT normally needs to read the data set twice. However, if all the variables in the data set have formats that explicitly specify a field width (for example, BEST12. but not BEST.), PROC PRINT reads the data set only once.

Alias: U

Tip: If the data set is large and you want a uniform report, you can save computer resources by using formats that explicitly specify a field width so that PROC PRINT reads the data only once.

Tip: WIDTH=UNIFORM is the same as UNIFORM.

Restriction: When not all variables have formats that explicitly specify a width, you cannot use WIDTH=UNIFORM with an engine that supports concurrent access if another user is updating the data set at the same time.

UNIFORMBY

formats all columns uniformly within a BY group, using each variable's formatted width as its column width. If the variable does not have a format that explicitly specifies a field width, PROC PRINT uses the widest data value as the column width.

Alias: UBY

Restriction: You cannot use UNIFORMBY with a sequential data set.

Default: If you omit WIDTH= and do not specify the UNIFORM option, PROC PRINT individually constructs each page of output. The procedure analyzes the data for a page and decides how best to display them. Therefore, column widths may differ from one page to another.

Tip: Column width is affected not only by variable width but also by the length of column headings. Long column headings may lessen the usefulness of WIDTH=.

See also: For a discussion of default column widths, see “Column Width” on page 798.

BY Statement

Analyzes a separate section of the report for each BY group.

Main discussion: “BY” on page 62

Featured in: Example 3 on page 803, Example 4 on page 805, Example 5 on page 808, Example 6 on page 811, and Example 8 on page 816

BY <DESCENDING>variable1
    <...<DESCENDING>variablen>
    <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.

Using the BY Statement with an ID Statement

PROC PRINT uses a special layout if all BY variables appear in the same order at the beginning of the ID statement. (See Example 8 on page 816.)

Using the BY Statement with the NOBYLINE Option

If you use the BY statement with the SAS system option NOBYLINE, which suppresses the BY line that normally appears in output produced with BY-group processing, PROC PRINT always starts a new page for each BY group. This behavior ensures that if you create customized BY lines by putting BY-group information in the title and suppressing the default BY lines with NOBYLINE, the information in the titles matches the report on the pages. (See “Creating Titles That Contain BY-Group Information” on page 46.)

ID Statement

Identifies observations by using the formatted values of the variables that you list instead of by using observation numbers.

Featured in: Example 7 on page 813 and Example 8 on page 816

ID variable(s);
Required Arguments

variable(s)

specifies one or more variables to print instead of the observation number at the beginning of each row of the report.

Restriction: If the ID variables occupy so much space that no room remains on the line for at least one other variable, PROC PRINT writes a warning to the SAS log and does not treat all ID variables as ID variables.

Interaction: If a variable in the ID statement also appears in the VAR statement, the output contains two columns for that variable.

Using the BY Statement with an ID Statement

PROC PRINT uses a special layout if all BY variables appear in the same order at the beginning of the ID statement. (See Example 8 on page 816.)
PAGEBY Statement

Controls page ejects that occur before a page is full.

Requirements: BY statement

Featured in: Example 3 on page 803

PAGEBY BY-variable;

Required Arguments

BY-variable

identifies a variable appearing in the BY statement in the PROC PRINT step. If the value of the BY variable changes, or if the value of any BY variable that precedes it in the BY statement changes, PROC PRINT begins printing a new page.

Interaction: If you use the BY statement with the SAS system option NOBYLINE, which suppresses the BY line that normally appears in output produced with BY-group processing, PROC PRINT always starts a new page for each BY group. This behavior ensures that if you create customized BY lines by putting BY-group information in the title and suppressing the default BY lines with NOBYLINE, the information in the titles matches the report on the pages. (See “Creating Titles That Contain BY-Group Information” on page 46.)

SUM Statement

Totals values of numeric variables.

Featured in: Example 4 on page 805, Example 5 on page 808, Example 6 on page 811, and Example 8 on page 816

SUM variable(s);

Required Arguments

variable(s)

identifies the numeric variables to total in the report.

Using the SUM and BY Statements Together

When you use a SUM statement and a BY statement with one BY variable, PROC PRINT sums the SUM variables for each BY group that contains more than one observation and totals them over all BY groups (see Example 4 on page 805).
When you use a SUM statement and a BY statement with multiple BY variables, PROC PRINT sums the SUM variables for each BY group that contains more than one observation, just as it does if you use only one BY variable. However, it provides sums only for those BY variables whose values change when the BY group changes. (See Example 5 on page 808.)

Note: When the value of a BY variable changes, the SAS System considers that the values of all variables listed after it in the BY statement also change.

---

**SUMBY Statement**

Limits the number of sums that appear in the report.

Requirements: BY statement

Featured in: Example 6 on page 811

---

**SUMBY** BY-variable;

**Required Arguments**

**BY-variable**

identifies a variable that appears in the BY statement in the PROC PRINT step. If the value of the BY variable changes, or if the value of any BY variable that precedes it in the BY statement changes, PROC PRINT prints the sums of all variables listed in the SUM statement.

**What Variables Are Summed?**

If you use a SUM statement, PROC PRINT subtotals only the SUM variables. Otherwise, PROC PRINT subtotals all the numeric variables in the data set except those listed in the ID and BY statements.

---

**VAR Statement**

Selects variables that appear in the report and determines their order.

Tip: If you omit the VAR statement, PROC PRINT prints all variables in the data set.

Featured in: Example 1 on page 799 and Example 8 on page 816

**VAR** variable(s);
Required Arguments

variable(s)

identifies the variables to print. PROC PRINT prints the variables in the order that you list them.

Interaction: In the PROC PRINT output, variables that are listed in the ID statement precede variables that are listed in the VAR statement. If a variable in the ID statement also appears in the VAR statement, the output contains two columns for that variable.

Results

Procedure Output

PROC PRINT always produces a printed report. You control the appearance of the report with statements and options. See "Examples" on page 799 for a sampling of the types of reports that the procedure produces.

Page Layout

By default, PROC PRINT uses an identical layout for all observations on a page of output. First, it attempts to print observations on a single line (see Figure 27.2 on page 796).

Figure 27.2 Printing Observations on a Single Line

<table>
<thead>
<tr>
<th>Obs</th>
<th>Var_1</th>
<th>Var_2</th>
<th>Var_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>2</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>3</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>4</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>5</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>6</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
</tbody>
</table>

If PROC PRINT cannot fit all the variables on a single line, it splits the observations into two or more sections and prints the observation number or the ID variables at the beginning of each line. For example, in Figure 27.3 on page 797, PROC PRINT prints the values for the first three variables in the first section of each page and the values for the second three variables in the second section of each page.
If PROC PRINT cannot fit all the variables on one page, the procedure prints subsequent pages with the same observations until it has printed all the variables. For example, in Figure 27.4 on page 797, PROC PRINT uses the first two pages to print values for the first three observations and the second two pages to print values for the rest of the observations.

Note: You can alter the page layout with the ROWS= option in the PROC PRINT statement (see the discussion of ROWS= on page 789).
Version 7 compressed files are RADIX addressable. (The integrity of the data is not compromised; the procedure simply numbers the observations differently.)

**Column Headings**

By default, spacing dictates whether PROC PRINT prints column headings horizontally or vertically. Figure 27.2 on page 796, Figure 27.3 on page 797, and Figure 27.4 on page 797 all illustrate horizontal headings. Figure 27.5 on page 798 illustrates vertical headings.

**Figure 27.5 Using Vertical Headings**

```
1 V V V
  a a a
0 r r r
b -- -- --
s 1 2 3
1 ---- ---- ----
2 ---- ---- ----
3 ---- ---- ----
4 ---- ---- ----
5 ---- ---- ----
6 ---- ---- ----
```

Note: If you use LABEL and at least one variable has a label, PROC PRINT prints all column headings horizontally unless you specify HEADING=VERTICAL.

**Column Width**

By default, PROC PRINT uses a variable's formatted width as the column width. (The WIDTH= option overrides this default behavior.) If the variable does not have a format that explicitly specifies a field width, PROC PRINT uses the widest data value for that variable on that page as the column width.

If the formatted value of a character variable or the data width of an unformatted character variable exceeds the linesize minus the length of all the ID variables, PROC PRINT may truncate the value. Consider the following situation:

- The linesize is 80.
- IdNumber is a character variable with a length of 10. It is used as an ID variable.
- State is a character variable with a length of 2. It is used as an ID variable.
- Comment is a character variable with a length of 200.

When PROC PRINT prints these three variables on a line, it uses 14 print positions for the two ID variables and the space after each one. This leaves 80–14, or 66, print positions for COMMENT. Longer values of COMMENT are truncated.

WIDTH= controls the column width.

Note: Column width is affected not only by variable width but also by the length of column headings. Long column headings may lessen the usefulness of WIDTH=. 
Examples

Example 1: Selecting Variables to Print

Procedure features:
- PROC PRINT statement options:
  - DOUBLE
  - VAR statement

This example
- selects three variables for the report
- uses variable labels as column headings
- double spaces between rows of the report.

Program

options nodate pageno=1 linesize=70 pagesize=60;

data exprev;
  input Region $ State $ Month monyy5. Expenses Revenues;
  format month monyy5.;
  datalines;
  Southern GA JAN95 2000 8000
  Southern GA FEB95 1200 6000
  Southern FL FEB95 8500 11000
  Northern NY FEB95 3000 4000
  Northern NY MAR95 6000 5000
  Southern FL MAR95 9800 13500
  Northern MA MAR95 1500 1000
;

DOUBLE writes a blank line between observations. (This option has no effect on the HTML output.)

proc print data=exprev double;
The VAR statement creates columns for Month, State, and Expenses, in that order.

```
var month state expenses;
```

The TITLE statement specifies a title for the report.

```
title 'Monthly Expenses for Offices in Each State';
run;
```
Output

By default, PROC PRINT identifies each observation by number under the column heading Obs.

<table>
<thead>
<tr>
<th>Monthly Expenses for Offices in Each State</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obs</td>
<td>Month</td>
</tr>
<tr>
<td>-----</td>
<td>-------</td>
</tr>
<tr>
<td>1</td>
<td>JAN95</td>
</tr>
<tr>
<td>2</td>
<td>FEB95</td>
</tr>
<tr>
<td>3</td>
<td>FEB95</td>
</tr>
<tr>
<td>4</td>
<td>FEB95</td>
</tr>
<tr>
<td>5</td>
<td>MAR95</td>
</tr>
<tr>
<td>6</td>
<td>MAR95</td>
</tr>
<tr>
<td>7</td>
<td>MAR95</td>
</tr>
</tbody>
</table>

Example 2: Customizing Text in Column Headers

Procedure features:
PROC PRINT statement options:
   N
   OBS=
   SPLIT=

Other features:
   LABEL statement

Data set: EXPREV on page 799

This example
- customizes and underlines the text in column headings for variables
- customizes the column header for the column that identifies observations by number
- shows the number of observations in the report
- writes the values of Expenses with commas.

Program

options nodate pageno=1 linesize=70 pagesize=60;
SPLIT= identifies the asterisk as the character that starts a new line in column headers. The N option prints the number of observations at the end of the report. OBS= specifies the column header for the column that identifies each observation by number. The split character (*) starts a new line in the column heading. Therefore, the equal signs (=) in the value of OBS= underline the column header.

```plaintext
proc print data=exprev split='*' n obs='Observation*Number*==========';
```

The VAR statement creates columns for Month, State, and Expenses, in that order.

```plaintext
var month state expenses;
```

The LABEL statement associates a label with each variable for the duration of the PROC PRINT step. When you use SPLIT= in the PROC PRINT statement, the procedure uses labels for column headers. The split character (*) starts a new line in the column heading. Therefore, the equal signs (=) in the labels underline the column headers.

```plaintext
label month='Month**=====
state='State**=====
expenses='Expenses**========';
```

The FORMAT statement assigns a format to use for Expenses in the report. The TITLE statement specifies a title.

```plaintext
format expenses comma10.;
title 'Monthly Expenses for Offices in Each State';
run;
```

**Output**
Example 3: Creating Separate Sections of a Report for Groups of Observations

Procedure features:
- PROC PRINT statement options:
  - LABEL
  - N=
  - NOOBS
- BY statement
- PAGEBY statement

Other features:
- SORT procedure
- LABEL statement

Data set:  EXPREV on page 799

This example
- suppresses the printing of observation numbers at the beginning of each row
- presents the data for each state in a separate section of the report
- begins a new page for each region.

Program

```plaintext
options pagesize=60 pageno=1 nodate linesize=70;

PROC SORT sorts the observations by Region, State, and Month.
proc sort data=exprev;
    by region state month;
```
run;

N= prints the number of observations in a BY group at the end of that BY group. The
explanatory text that the N= option provides precedes the number. NOOBS suppresses the
printing of observation numbers at the beginning of the rows. LABEL uses variables’ labels as
column headings.

proc print data=exprev n=’Number of observations for the state: ’
   noobs label;

The VAR statement creates columns for Month, Expenses, and Revenues, in that order.

    var month expenses revenues;

The BY statement produces a separate section of the report for each BY group and prints a
heading above each one. The PAGEBY statement starts a new page each time the value of Region
changes.

    by region state;
    pageby region;

The LABEL statement associates a label with the variable Region for the duration of the PROC
PRINT step. When you use the LABEL option in the PROC PRINT statement, the procedure uses
labels for column headings.

    label region=’Sales Region’;

The FORMAT statement assigns a format to Expenses and Revenues for this report. The TITLE
statement specifies a title.

    format revenues expenses comma10.;
    title ‘Sales Figures Grouped by Region and State’;
run;
Output

Sales Figures Grouped by Region and State

------------------- Sales Region=Northern State=MA -------------------

<table>
<thead>
<tr>
<th>Month</th>
<th>Expenses</th>
<th>Revenues</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAR95</td>
<td>1,500</td>
<td>1,000</td>
</tr>
</tbody>
</table>

Number of observations for the state: 1

------------------- Sales Region=Northern State=NY -------------------

<table>
<thead>
<tr>
<th>Month</th>
<th>Expenses</th>
<th>Revenues</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEB95</td>
<td>3,000</td>
<td>4,000</td>
</tr>
<tr>
<td>MAR95</td>
<td>6,000</td>
<td>5,000</td>
</tr>
</tbody>
</table>

Number of observations for the state: 2

------------------- Sales Region=Southern State=FL -------------------

<table>
<thead>
<tr>
<th>Month</th>
<th>Expenses</th>
<th>Revenues</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEB95</td>
<td>8,500</td>
<td>11,000</td>
</tr>
<tr>
<td>MAR95</td>
<td>9,800</td>
<td>13,500</td>
</tr>
</tbody>
</table>

Number of observations for the state: 2

------------------- Sales Region=Southern State=GA -------------------

<table>
<thead>
<tr>
<th>Month</th>
<th>Expenses</th>
<th>Revenues</th>
</tr>
</thead>
<tbody>
<tr>
<td>JAN95</td>
<td>2,000</td>
<td>8,000</td>
</tr>
<tr>
<td>FEB95</td>
<td>1,200</td>
<td>6,000</td>
</tr>
</tbody>
</table>

Number of observations for the state: 2

Example 4: Summing Numeric Variables with One BY Group

Procedure features:
PROC PRINT statement options:
 N=
 BY statement
 SUM statement

Other features:
SORT procedure
TITLE statement
 #BYVAL specification
SAS system options :
 BYLINE
 NOBYLINE
Data set:  EXPREV  on page 799

This example
- sums expenses and revenues for each region and for all regions
- shows the number of observations in each BY group and in the whole report
- creates a customized title, containing the name of the region. This title replaces the default BY line for each BY group.

Program

The SAS system option NOBYLINE suppresses the printing of the default BY line. When you use PROC PRINT with NOBYLINE, each BY group starts on a new page.

```
options nodate pageno=1 linesize=70 pagesize=60 nobyline;
```

PROC SORT sorts the observations by Region.

```
proc sort data=exprev;
    by region;
run;
```

NOOBS suppresses the printing of observation numbers at the beginning of the rows. N= prints the number of observations in a BY group at the end of that BY group and (because of the SUM statement) prints the number of observations in the data set at the end of the report. The first piece of explanatory text that N= provides precedes the number for each BY group. The second piece of explanatory text that N= provides precedes the number for the entire data set.

```
proc print data=exprev noobs
    n='Number of observations for the state: '
    'Number of observations for the data set: '
```

The SUM statement alone sums the values of Expenses and Revenues for the entire data set. Because the PROC PRINT step contains a BY statement, the SUM statement also sums the values of Expenses and Revenues for each region that contains more than one observation.
sum expenses revenues;
by region;

The FORMAT statement assigns the COMMA10. format to Expenses and Revenues for this report.
format revenues expenses comma10.;

The TITLE statement specifies a title. The #BYVAL specification places the current value of the
BY variable Region in the title. Because NOBYLINE is in effect, each BY group starts on a new
page, and the title serves as a BY line.
title ‘Revenue and Expense Totals for the
#byval(region) Region’;
run;

The SAS system option BYLINE resets the printing of the default BY line.
options byline;

Output

<table>
<thead>
<tr>
<th>State</th>
<th>Month</th>
<th>Expenses</th>
<th>Revenues</th>
</tr>
</thead>
<tbody>
<tr>
<td>NY</td>
<td>FEB95</td>
<td>3,000</td>
<td>4,000</td>
</tr>
<tr>
<td>NY</td>
<td>MAR95</td>
<td>6,000</td>
<td>5,000</td>
</tr>
<tr>
<td>MA</td>
<td>MAR95</td>
<td>1,500</td>
<td>1,000</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>Region</td>
<td></td>
<td>10,500</td>
<td>10,000</td>
</tr>
</tbody>
</table>

Number of observations for the state: 3
Example 5: Summing Numeric Variables with Multiple BY Variables

Procedure features:
- BY statement
- SUM statement

Other features: SORT procedure

Data set: EXPREV on page 799

This example
- sums expenses and revenues for
  - each region
  - each state with more than one row in the report
  - all rows in the report.
- shows the number of observations in each BY group and in the whole report.

Program

options nodate pageno=1 linesize=70 pagesize=60;

PROC SORT sorts the observations by Region and State.

proc sort data=exprev;
  by region state;
run;

The N option prints the number of observations in a BY group at the end of that BY group and prints the total number of observations used in the report at the bottom of the report. NOOBS suppresses the printing of observation numbers at the beginning of the rows.
proc print data=exprev n noobs;

The BY statement produces a separate section of the report for each BY group. The SUM statement alone sums the values of Expenses and Revenues for the entire data set. Because the program contains a BY statement, the SUM statement also sums the values of Expenses and Revenues for each BY group that contains more than one observation.

   by region state;
   sum expenses revenues;

The LABEL statement associates a label with the variable Region for the duration of the PROC PRINT step. The BY line at the beginning of each BY group uses the label. The FORMAT statement assigns a format to Expenses and Revenues for this report. The TITLE statement specifies a title.

   label region=‘Sales Region’;
   format revenues expenses comma10.;
   title ‘Revenue and Expense Totals for Each State and Region’;
run;
The report uses default column headers (variable names) because neither the SPLIT= nor the LABEL option is used. Nevertheless, the BY line at the top of each section of the report shows the BY variables' labels and their values. The name of a BY variable identifies the subtotals in the report.

PROC PRINT sums Expenses and Revenues for each BY group that contains more than one observation. However, sums are shown only for the BY variables whose values change from one BY group to the next. For example, in the third BY group, where the sales region is Southern and the state is FL, expenses and revenues are summed only for the state because the next BY group is for the same region.

<table>
<thead>
<tr>
<th>Revenue and Expense Totals for Each State and Region</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sales Region=Northern State=MA</strong></td>
<td>------------</td>
</tr>
<tr>
<td><strong>Month</strong></td>
<td><strong>Expenses</strong></td>
</tr>
<tr>
<td>MAR95</td>
<td>1,500</td>
</tr>
<tr>
<td>( N = 1 )</td>
<td>( N = 1 )</td>
</tr>
<tr>
<td><strong>Sales Region=Northern State=NY</strong></td>
<td>------------</td>
</tr>
<tr>
<td><strong>Month</strong></td>
<td><strong>Expenses</strong></td>
</tr>
<tr>
<td>FEB95</td>
<td>3,000</td>
</tr>
<tr>
<td>MAR95</td>
<td>6,000</td>
</tr>
<tr>
<td>( \text{State} )</td>
<td>( 9,000 )</td>
</tr>
<tr>
<td>( \text{Region} )</td>
<td>( 10,500 )</td>
</tr>
<tr>
<td>( N = 2 )</td>
<td>( N = 2 )</td>
</tr>
<tr>
<td><strong>Sales Region=Southern State=FL</strong></td>
<td>------------</td>
</tr>
<tr>
<td><strong>Month</strong></td>
<td><strong>Expenses</strong></td>
</tr>
<tr>
<td>FEB95</td>
<td>8,500</td>
</tr>
<tr>
<td>MAR95</td>
<td>9,800</td>
</tr>
<tr>
<td>( \text{State} )</td>
<td>( 18,300 )</td>
</tr>
<tr>
<td>( N = 2 )</td>
<td>( N = 2 )</td>
</tr>
<tr>
<td><strong>Sales Region=Southern State=GA</strong></td>
<td>------------</td>
</tr>
<tr>
<td><strong>Month</strong></td>
<td><strong>Expenses</strong></td>
</tr>
<tr>
<td>JAN95</td>
<td>2,000</td>
</tr>
<tr>
<td>FEB95</td>
<td>1,200</td>
</tr>
<tr>
<td>( \text{State} )</td>
<td>( 3,200 )</td>
</tr>
<tr>
<td>( \text{Region} )</td>
<td>( 21,500 )</td>
</tr>
<tr>
<td>( \text{Total} )</td>
<td>( 32,000 )</td>
</tr>
<tr>
<td>( N = 2 )</td>
<td>( N = 2 )</td>
</tr>
<tr>
<td><strong>Total N</strong></td>
<td><strong>7</strong></td>
</tr>
</tbody>
</table>
Example 6: Limiting the Number of Sums in a Report

Features:
- BY statement
- SUM statement
- SUMBY statement

Other features:
- SORT procedure
- LABEL statement

Data set: EXPREV on page 799

This example
□ creates a separate section of the report for each combination of state and region
□ sums expenses and revenues only for each region and for all regions, not for individual states.

Program

```plaintext
options nodate pageno=1 linesize=70 pagesize=60;

PROC SORT sorts the observations by Region and State.

proc sort data=exprev;
  by region state;
run;

NOOBS suppresses the printing of observation numbers at the beginning of the rows.

proc print data=exprev noobs;

The SUM and BY statements work together to sum the values of Revenues and Expenses for each BY group as well as for the whole report. The SUMBY statement limits the subtotals to one for each region.

by region state;
sum revenues expenses;
sumby region;
```
The LABEL statement associates a label with the variable Region for the duration of the PROC PRINT step. This label is used in the BY lines.

```plaintext
label region='Sales Region';
```

The FORMAT statement assigns the COMMA10. format to Expenses and Revenues for this report. The TITLE statement specifies the title.

```plaintext
format revenues expenses comma10.;
title 'Revenue and Expense Figures for Each Region';
run;
```
Output

The report uses default column headers (variable names) because neither the SPLIT= nor the LABEL option is used. Nevertheless, the BY line at the top of each section of the report shows the BY variables' labels and their values. The name of a BY variable identifies the subtotals in the report.

<table>
<thead>
<tr>
<th>Revenue and Expense Figures for Each Region</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>------------------------------------------</td>
<td>--</td>
</tr>
<tr>
<td>------------------------------- Sales Region=Northern State=MA -------------------</td>
<td></td>
</tr>
<tr>
<td>Month</td>
<td>Expenses</td>
</tr>
<tr>
<td>MAR95</td>
<td>1,500</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>--</td>
</tr>
<tr>
<td>------------------------------- Sales Region=Northern State=NY -------------------</td>
<td></td>
</tr>
<tr>
<td>Month</td>
<td>Expenses</td>
</tr>
<tr>
<td>FEB95</td>
<td>3,000</td>
</tr>
<tr>
<td>MAR95</td>
<td>6,000</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
</tr>
<tr>
<td>Region</td>
<td>10,500</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>--</td>
</tr>
<tr>
<td>------------------------------- Sales Region=Southern State=FL -------------------</td>
<td></td>
</tr>
<tr>
<td>Month</td>
<td>Expenses</td>
</tr>
<tr>
<td>FEB95</td>
<td>8,500</td>
</tr>
<tr>
<td>MAR95</td>
<td>9,800</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
</tr>
<tr>
<td>Region</td>
<td>21,500</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
</tr>
<tr>
<td>32,000</td>
<td>48,500</td>
</tr>
</tbody>
</table>

Example 7: Controlling the Layout of a Report with Many Variables

Procedure features:
- PROC PRINT statement options:
  - ROWS=
- ID statement

Other features:
- SAS data set options:
  - OBS=

This example shows two ways of printing a data set with a large number of variables: one is the default, and the other uses ROWS=. For detailed explanations of
the layouts of these two reports, see the ROWS= option on page 789 and see “Page Layout” on page 796. These reports use a pagesize of 24 and a linesize of 64 to help illustrate the different layouts.

Note When the two reports are written as HTML output, they do not differ.

Program

```
options nodate pageno=1 linesize=64 pagesize=24 ;
```

The data set EMPDATA contains personal and job-related information about a company’s employees. A DATA step on page 1441 creates this data set.

```
data empdata;
  input IdNumber $ 1-4 LastName $ 9-19 FirstName $ 20-29
  City $ 30-42 State $ 43-44 /
  Gender $ 1 JobCode $ 9-11 Salary 20-29 @30 Birth date7.
  @43 Hired date7. HomePhone $ 54-65;
  format birth hired date7.;
  datalines;
1919 Adams Gerald Stamford CT
M TA2 34376 15SEP48 07JUN75 203/781-1255
1653 Alexander Susan Bridgeport CT
F ME2 35108 18OCT52 12AUG78 203/675-7715
... more lines of data ...
1407 Grant Daniel Mt. Vernon NY
M PT1 68096 26MAR57 21MAR78 914/468-1616
1114 Green Janice New York NY
F TA2 32928 21SEP57 30JUN75 212/588-1092;
```

The OBS= data set option uses only the first 12 observations to create the report. (This is just to conserve space here.) The ID statement identifies observations with the formatted value of IdNumber rather than with the observation number. This report is in Output 27.3 on page 815.

```
proc print data=empdata(obs=12);
  id idnumber;
  title 'Personnel Data';
run;
```

ROWS=PAGE prints only one row of variables for each observation on a page. This report is in Output 27.4 on page 816.
```sas
proc print data=empdata(obs=12) rows=page;
  id idnumber;
  title 'Personnel Data';
run;
```

## Output

### Output 27.3  Default Layout for a Report with Many Variables

In the traditional procedure output, each page of this report contains values for all variables in each observation. In the HTML output, this report is identical to the report that uses `ROWS=PAGE`.

Note that PROC PRINT automatically splits the variable names that are used as column headers at a change in capitalization if the entire name does not fit in the column. Compare, for example, the column headers for `LastName` (which fits in the column) and `FirstName` (which does not fit in the column).

### Personnel Data

<table>
<thead>
<tr>
<th>Id Number</th>
<th>LastName</th>
<th>FirstName</th>
<th>City</th>
<th>State</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>1919</td>
<td>Adams</td>
<td>Gerald</td>
<td>Stamford</td>
<td>CT</td>
<td>M</td>
</tr>
<tr>
<td>1653</td>
<td>Alexander</td>
<td>Susan</td>
<td>Bridgeport</td>
<td>CT</td>
<td>F</td>
</tr>
<tr>
<td>1400</td>
<td>Apple</td>
<td>Troy</td>
<td>New York</td>
<td>NY</td>
<td>M</td>
</tr>
<tr>
<td>1350</td>
<td>Arthur</td>
<td>Barbara</td>
<td>New York</td>
<td>NY</td>
<td>F</td>
</tr>
<tr>
<td>1401</td>
<td>Avery</td>
<td>Jerry</td>
<td>Paterson</td>
<td>NJ</td>
<td>M</td>
</tr>
<tr>
<td>1499</td>
<td>Barefoot</td>
<td>Joseph</td>
<td>Princeton</td>
<td>NJ</td>
<td>M</td>
</tr>
<tr>
<td>1101</td>
<td>Baucom</td>
<td>Walter</td>
<td>New York</td>
<td>NY</td>
<td>M</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Id Number</th>
<th>Job Code</th>
<th>Salary</th>
<th>Birth</th>
<th>Hired</th>
<th>HomePhone</th>
</tr>
</thead>
<tbody>
<tr>
<td>1919</td>
<td>TA2</td>
<td>34376</td>
<td>15SEP48</td>
<td>07JUN75</td>
<td>203/781-1255</td>
</tr>
<tr>
<td>1653</td>
<td>ME2</td>
<td>35108</td>
<td>18OCT52</td>
<td>12AUG78</td>
<td>203/675-7715</td>
</tr>
<tr>
<td>1400</td>
<td>ME1</td>
<td>29769</td>
<td>08NOV55</td>
<td>19OCT78</td>
<td>212/586-0808</td>
</tr>
<tr>
<td>1350</td>
<td>FA3</td>
<td>32886</td>
<td>03SEP53</td>
<td>01AUG78</td>
<td>718/383-1549</td>
</tr>
<tr>
<td>1401</td>
<td>TA3</td>
<td>38822</td>
<td>16DEC38</td>
<td>20NOV73</td>
<td>201/732-8787</td>
</tr>
<tr>
<td>1499</td>
<td>ME3</td>
<td>43025</td>
<td>29APR42</td>
<td>10JUN68</td>
<td>201/812-5665</td>
</tr>
<tr>
<td>1101</td>
<td>SCP</td>
<td>18723</td>
<td>09JUN50</td>
<td>04OCT78</td>
<td>212/586-8060</td>
</tr>
</tbody>
</table>

### Personnel Data 2

<table>
<thead>
<tr>
<th>Id Number</th>
<th>LastName</th>
<th>FirstName</th>
<th>City</th>
<th>State</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>1333</td>
<td>Blair</td>
<td>Justin</td>
<td>Stamford</td>
<td>CT</td>
<td>M</td>
</tr>
<tr>
<td>1402</td>
<td>Bialock</td>
<td>Ralph</td>
<td>New York</td>
<td>NY</td>
<td>M</td>
</tr>
<tr>
<td>1479</td>
<td>Bostic</td>
<td>Marie</td>
<td>New York</td>
<td>NY</td>
<td>F</td>
</tr>
<tr>
<td>1403</td>
<td>Bowden</td>
<td>Earl</td>
<td>Bridgeport</td>
<td>CT</td>
<td>M</td>
</tr>
<tr>
<td>1739</td>
<td>Boyce</td>
<td>Jonathan</td>
<td>New York</td>
<td>NY</td>
<td>M</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Id Number</th>
<th>Job Code</th>
<th>Salary</th>
<th>Birth</th>
<th>Hired</th>
<th>HomePhone</th>
</tr>
</thead>
<tbody>
<tr>
<td>1333</td>
<td>PT2</td>
<td>88606</td>
<td>02APR49</td>
<td>13FEB69</td>
<td>203/781-1777</td>
</tr>
<tr>
<td>1402</td>
<td>TA2</td>
<td>32615</td>
<td>20JAN51</td>
<td>05DEC78</td>
<td>718/384-2849</td>
</tr>
<tr>
<td>1403</td>
<td>TA3</td>
<td>38785</td>
<td>25DEC56</td>
<td>08OCT77</td>
<td>718/384-8816</td>
</tr>
<tr>
<td>1404</td>
<td>ME1</td>
<td>28072</td>
<td>31JAN57</td>
<td>24DEC79</td>
<td>203/675-3434</td>
</tr>
<tr>
<td>1739</td>
<td>PT1</td>
<td>66517</td>
<td>28DEC52</td>
<td>30JAN79</td>
<td>212/587-1247</td>
</tr>
</tbody>
</table>
Example 8: Creating a Customized Layout with BY Groups and ID Variables

Procedure features:
- BY statement
- ID statement
- SUM statement
- VAR statement

Other features:
- SORT procedure

Data set: EMPDATA on page 814

This customized report

Output 27.4  Layout Produced by the ROWS=PAGE Option

Each page of this report contains values for only some of the variables in each observation. However, each page contains values for more observations than the default report does.

<table>
<thead>
<tr>
<th>Personnel Data</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>First Name</td>
</tr>
<tr>
<td>----</td>
<td>------------</td>
</tr>
<tr>
<td>1919</td>
<td>Adams</td>
</tr>
<tr>
<td>1653</td>
<td>Alexander</td>
</tr>
<tr>
<td>1400</td>
<td>Apple</td>
</tr>
<tr>
<td>1350</td>
<td>Arthur</td>
</tr>
<tr>
<td>1401</td>
<td>Avery</td>
</tr>
<tr>
<td>1499</td>
<td>Barefoot</td>
</tr>
<tr>
<td>1101</td>
<td>Baucom</td>
</tr>
<tr>
<td>1333</td>
<td>Blair</td>
</tr>
<tr>
<td>1402</td>
<td>Blalock</td>
</tr>
<tr>
<td>1479</td>
<td>Bostic</td>
</tr>
<tr>
<td>1403</td>
<td>Bowden</td>
</tr>
<tr>
<td>1739</td>
<td>Boyce</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Personnel Data</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>Job Code</td>
</tr>
<tr>
<td>----</td>
<td>----------</td>
</tr>
<tr>
<td>1919</td>
<td>TA2</td>
</tr>
<tr>
<td>1653</td>
<td>ME2</td>
</tr>
<tr>
<td>1400</td>
<td>ME1</td>
</tr>
<tr>
<td>1350</td>
<td>FA3</td>
</tr>
<tr>
<td>1401</td>
<td>TA3</td>
</tr>
<tr>
<td>1499</td>
<td>ME3</td>
</tr>
<tr>
<td>1101</td>
<td>SCP</td>
</tr>
<tr>
<td>1333</td>
<td>PT2</td>
</tr>
<tr>
<td>1402</td>
<td>TA2</td>
</tr>
<tr>
<td>1479</td>
<td>TA3</td>
</tr>
<tr>
<td>1403</td>
<td>ME1</td>
</tr>
<tr>
<td>1739</td>
<td>PT1</td>
</tr>
</tbody>
</table>
- selects variables to include in the report and controls their order
- selects observations to include in the report
- groups the selected observations by JobCode
- sums the salaries for each job code and for all job codes
- displays numeric data with commas and dollar signs.

**Program**

PROC SORT creates a temporary data set in which the observations are sorted by JobCode and Gender.

```plaintext
options nodate pageno=1 linesize=64 pagesize=60;
proc sort data=empdata out=tempemp;
  by jobcode gender;
run;
```

SPLIT= identifies the asterisk as the character that starts a new line in column headers.

```plaintext
proc print data=tempemp split='*';
```

The VAR statement and the ID statement together select the variables to include in the report. The ID statement and the BY statement produce the special format.

```plaintext
id jobcode;
by jobcode;
var gender salary;
```

The SUM statement totals the values of Salary for each BY group and for the whole report.

```plaintext
sum salary;
```

The LABEL statement associates a label with each variable for the duration of the PROC PRINT step. When you use SPLIT= in the PROC PRINT statement, the procedure uses labels for column headings.

```plaintext
label jobcode='Job Code*========'
gender='Gender*======'
salary='Annual Salary*=============';
```
The FORMAT statement assigns a format to Salary for this report. The WHERE statement selects for the report only the observations for job codes that contain the letters 'FA' or 'ME'. The TITLE statements specify two titles.

```
format salary dollar11.2;
where jobcode contains 'FA' or jobcode contains 'ME';
title 'Expenses Incurred for';
title2 'Salaries for Flight Attendants and Mechanics';
run;
```

Output

The ID and BY statements work together to produce this layout. The ID variable is listed only once for each BY group. The BY lines are suppressed. Instead, the value of the ID variable, JobCode, identifies each BY group.

<table>
<thead>
<tr>
<th>Job Code</th>
<th>Gender</th>
<th>Annual Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>FA1</td>
<td>F</td>
<td>$23,177.00</td>
</tr>
<tr>
<td>F</td>
<td>$22,454.00</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>$22,268.00</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>---------------</td>
</tr>
<tr>
<td>FA1</td>
<td>$67,899.00</td>
<td></td>
</tr>
<tr>
<td>FA2</td>
<td>F</td>
<td>$28,888.00</td>
</tr>
<tr>
<td>F</td>
<td>$27,787.00</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>$28,572.00</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>---------------</td>
</tr>
<tr>
<td>FA2</td>
<td>$85,247.00</td>
<td></td>
</tr>
<tr>
<td>FA3</td>
<td>F</td>
<td>$32,886.00</td>
</tr>
<tr>
<td>F</td>
<td>$33,419.00</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>$32,217.00</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>---------------</td>
</tr>
<tr>
<td>FA3</td>
<td>$98,522.00</td>
<td></td>
</tr>
<tr>
<td>ME1</td>
<td>M</td>
<td>$29,769.00</td>
</tr>
<tr>
<td>M</td>
<td>$28,072.00</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>$28,619.00</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>---------------</td>
</tr>
<tr>
<td>ME1</td>
<td>$86,460.00</td>
<td></td>
</tr>
<tr>
<td>ME2</td>
<td>F</td>
<td>$35,108.00</td>
</tr>
<tr>
<td>F</td>
<td>$34,929.00</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>$35,345.00</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>$36,925.00</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>$35,090.00</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>$35,185.00</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>---------------</td>
</tr>
<tr>
<td>ME2</td>
<td>$212,582.00</td>
<td></td>
</tr>
<tr>
<td>ME3</td>
<td>M</td>
<td>$43,025.00</td>
</tr>
<tr>
<td>M</td>
<td>$593,735.00</td>
<td></td>
</tr>
</tbody>
</table>

$593,735.00
Example 9: Printing All the Data Sets in a SAS Library

Features:
- Macro facility
- DATASETS procedure
- PRINT procedure

Data set: EXPREV on page 799 and LIST on page 492

This example prints all the data sets in a SAS library. You can use the same programming logic with any procedure. Just replace the PROC PRINT step near the end of the example with whatever procedure step you want to execute. The example uses the macro language. For details about the macro language, see SAS Guide to Macro Processing, Version 6, Second Edition.

Program

libname printlib 'SAS-data-library'
options nodate pageno=1 linesize=80 pagesize=60;

PROC DATASETS copies two data sets out of the WORK library into the PRINTLIB library to limit the number of data sets available to the example.

proc datasets library=work memtype=data nolist;
   copy out=printlib;
   select list exprev;
run;

The %MACRO statement creates the macro PRINTALL. When you call the macro, you can pass it one or two parameters. The first parameter is the name of the library whose data set you want to print. The second parameter is a library used by the macro. If you do not specify this parameter, the WORK library is the default.

%macro printall(libname, worklib=work);

The %LOCAL statement creates two local macro variables, NUM and I, to use in a loop.
%local num i;

This PROC DATASETS step reads the library `LIBNAME` that you specify as a parameter when you invoke the macro. The CONTENTS statement produces an output data set called `TEMP1` in `WORKLIB`. This data set contains an observation for each variable in each data set in the library `LIBNAME`. By default, each observation includes the name of the data set that the variable is in as well as other information about the variable. However, the KEEP=data set option writes only the name of the data set to `TEMP1`.

```sas
proc datasets library=&libname memtype=data nodetails;
    contents out=&worklib..temp1(keep=memname) data=_all_ noprint;
run;
```

This DATA step increments the value of `N` each time that it reads the last occurrence of a data set name (when IF LAST.MEMNAME is true). The CALL SYMPUT statement uses the current value of `N` to create a macro variable for each unique value of MEMNAME in the data set TEMP1. The TRIM function removes extra blanks in the TITLE statement in the PROC PRINT step that follows.

```sas
data _null_;
    set &worklib..temp1 end=final;
    by memname notsorted;
    if last.memname;
    n+1;
    call symput('ds'||left(put(n,8.)),trim(memname));
```

When it reads the last observation in the data set (when FINAL is true), the DATA step assigns the value of `N` to the macro variable `NUM`. At this point in the program, the value of `N` is the number of observations in the data set.

```sas
if final then call symput('num',put(n,8.));
```
The RUN statement is crucial. It forces the DATA step to run, thus creating the macro variables used in the CALL SYMPUT statements before the %DO loop, which uses them, executes.

```
runt;
```

The %DO loop issues a PROC PRINT step for each data set. The %MEND statement ends the macro.

```
%do i=1 %to &num;
  proc print data=&libname..&&ds&i noobs;
  title "Data Set &libname..&&ds&i";
  run;
%end;
%mend printall;
```

This invocation of the PRINTALL macro prints all the data sets in the library PRINTLIB.

```
options nodate pageno=1 linesize=70 pagesize=60;
%printall(printlib)
```

## Output

**Data Set printlib.EXPREV**

<table>
<thead>
<tr>
<th>Region</th>
<th>State</th>
<th>Month</th>
<th>Expenses</th>
<th>Revenues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Northern</td>
<td>MA</td>
<td>MAR95</td>
<td>1500</td>
<td>1000</td>
</tr>
<tr>
<td>Northern</td>
<td>NY</td>
<td>FEB95</td>
<td>3000</td>
<td>4000</td>
</tr>
<tr>
<td>Southern</td>
<td>NY</td>
<td>MAR95</td>
<td>6000</td>
<td>5000</td>
</tr>
<tr>
<td>Southern</td>
<td>FL</td>
<td>FEB95</td>
<td>8500</td>
<td>11000</td>
</tr>
<tr>
<td>Southern</td>
<td>GA</td>
<td>JAN95</td>
<td>2000</td>
<td>8000</td>
</tr>
<tr>
<td>Southern</td>
<td>GA</td>
<td>FEB95</td>
<td>1200</td>
<td>6000</td>
</tr>
</tbody>
</table>

**Data Set printlib.LIST**

<table>
<thead>
<tr>
<th>Name</th>
<th>Street</th>
<th>City</th>
<th>State</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gabrielli, Theresa</td>
<td>24 Ridgetop Rd.</td>
<td>Westboro</td>
<td>MA</td>
<td>01581</td>
</tr>
<tr>
<td>Clayton, Aria</td>
<td>314 Bridge St.</td>
<td>Hanover</td>
<td>NH</td>
<td>03755</td>
</tr>
<tr>
<td>Dix, Martin L.</td>
<td>4 Shepherd St.</td>
<td>Norwich</td>
<td>VT</td>
<td>05055</td>
</tr>
<tr>
<td>Slater, Emily C.</td>
<td>2009 Cherry St.</td>
<td>York</td>
<td>PA</td>
<td>17407</td>
</tr>
<tr>
<td>Ericson, Jane</td>
<td>211 Clancy Court</td>
<td>Chapel Hill</td>
<td>NC</td>
<td>27514</td>
</tr>
<tr>
<td>An, Ing</td>
<td>95 Willow Dr.</td>
<td>Charlotte</td>
<td>NC</td>
<td>28211</td>
</tr>
<tr>
<td>Jacobson, Becky</td>
<td>7 Lincoln St.</td>
<td>Tallahassee</td>
<td>FL</td>
<td>32312</td>
</tr>
<tr>
<td>Misiewicz, Jeremy</td>
<td>43-C Lakeview Apts.</td>
<td>Madison</td>
<td>WI</td>
<td>53704</td>
</tr>
<tr>
<td>Ahmadi, Hafez</td>
<td>5203 Marston Way</td>
<td>Boulder</td>
<td>CO</td>
<td>80302</td>
</tr>
<tr>
<td>Archuleta, Ruby</td>
<td>Box 108</td>
<td>Milagro</td>
<td>NM</td>
<td>87429</td>
</tr>
</tbody>
</table>