SAS Certification Prep Guide: Base Programming for SAS 9, Third Edition

by SAS Institute
SAS Institute. (c) 2011. Copying Prohibited.

Chapter 4: Creating List Reports

Overview

Introduction

To list the information in a data set, you can create a report with a PROC PRINT step. Then you can enhance the report with additional statements and options to create reports like those shown below.

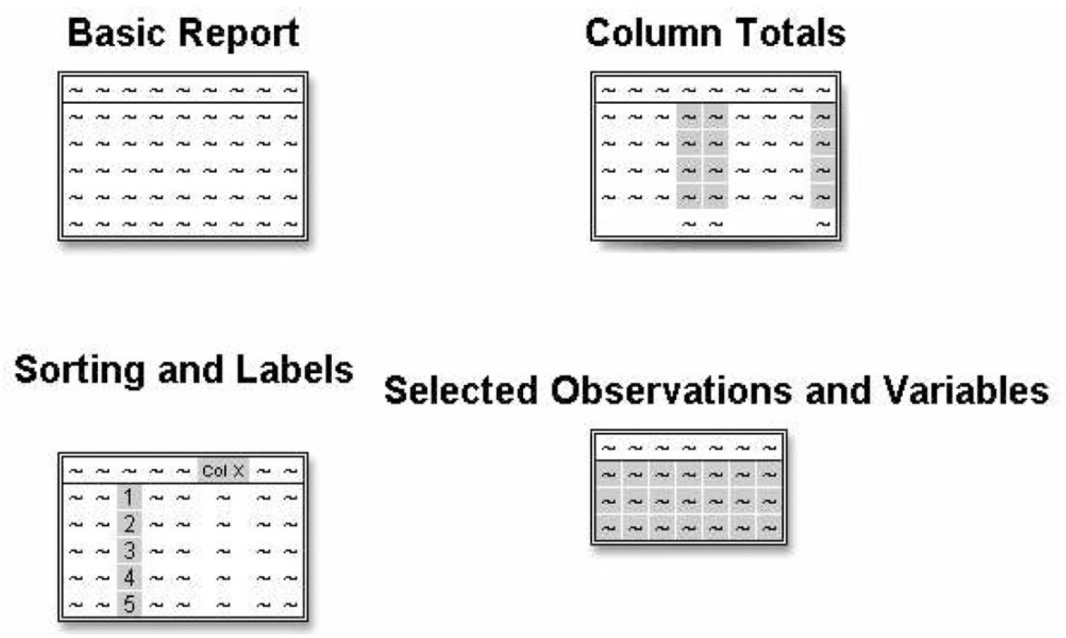


Figure 4.1: PROC PRINT

Objectives

In this chapter, you learn to

- specify SAS data sets to print
- select variables and observations to print
- sort data by the values of one or more variables
- specify column totals for numeric variables
- double space LISTING output
- add titles and footnotes to procedure output
- assign descriptive labels to variables
- apply formats to the values of variables.

Creating a Basic Report

To produce a simple list report, you first reference the library in which your SAS data set is stored. If you want, you can also set system options to control the appearance of your reports. Then you submit a basic PROC PRINT step.

General form, basic PROC PRINT step:

PROC PRINT DATA=*SAS-data-set;*

RUN;

where *SAS-data-set* is the name of the SAS data set to be printed.

In the program below, the PROC PRINT statement invokes the PRINT procedure and specifies the data set Therapy in the SAS data library to which the libref Patients has been assigned.

```
libname patients 'c:\records\patients';
proc print data=patients.therapy;
run;
```

Notice the layout of the resulting report below. By default,

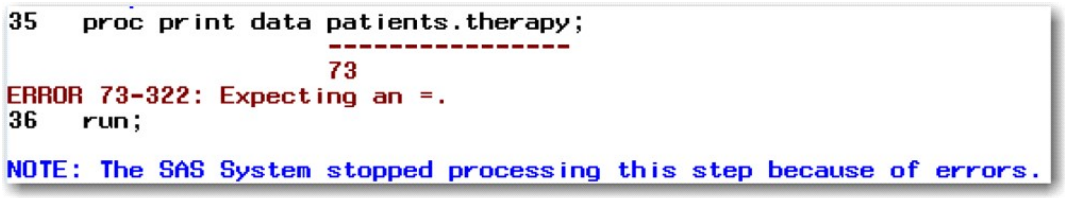
- all observations and variables in the data set are printed
- a column for observation numbers appears on the far left
- variables and observations appear in the order in which they occur in the data set.

The SAS System

Obs	Date	AerClass	WalkJogRun	Swim
1	JAN1999	56	73	14
2	FEB1999	32	109	19
3	MARI 999	35	106	22
4	APR1999	47	115	24
5	MAY1999	55	121	31
6	JUN1999	61	114	67
7	JUL1999	67	102	72
8	AUG1999	64	76	77
9	SEP1999	73	77	54
10	OCT1999	31	62	47
11	NOVI 999	84	31	52
12	DEC1999	2	44	55
13	JAN2000	37	91	33
14	FEB2000	41	102	27
15	MAR2000	52	93	19
16	APR2000	61	113	22
17	MAV2000	49	33	29
18	JUN2000	24	101	54
19	JUL2000	45	91	69
20	AUG2000	63	65	53
21	SEP2000	60	49	63
22	OCT2000	73	70	41
23	NOV2000	32	44	53
24	DEC2000	93	57	47

Figure 4.2: Patients.Therapy Data Set

Additional Note Be sure to specify the equal sign in the DATA= option in SAS procedures. If you omit the equal sign, your program produces an error similar to the following in the SAS log:



```
35    proc print data patients.therapy;
                      -----------------
                      73
ERROR 73-322: Expecting an =.
36    run;

NOTE: The SAS System stopped processing this step because of errors.
```

Figure 4.3: Error Message

Selecting Variables

Overview

By default, a PROC PRINT step lists all the variables in a data set. You can select variables and control the order in which they appear by using a VAR statement in your PROC PRINT step.

General form, VAR statement:

VAR *variable(s);*

where *variable(s)* is one or more variable names, separated by blanks.

For example, the following VAR statement specifies that only the variables Age, Height, Weight, and Fee be printed, in that order:

```
proc print data=clinic.admit;
   var age height weight fee;
run;
```

The procedure output from the PROC PRINT step with the VAR statement lists only the values for the variables Age, Height, Weight, and Fee.

The SAS System

Obs	Age	Height	Weight	Fee
1	27	72	168	35.20
2	34	66	152	1.24.80
3	31	61	123	149.75
4	43	63	137	149.75
5	51	71	158	1.24.80
6	29	76	193	1.24.80
7	32	67	151	149.75
8	35	70	173	149.75
9	34	73	154	1.24.80
10	49	64	172	1.24.80
11	44	66	140	149.75
12	20	62	118	35.20
13	30	69	147	149.75
14	40	69	163	1.24.80
15	47	72	173	1.24.80
16	60	71	191	149.75
17	43	65	123	1.24.80
18	25	75	188	35.20
19	22	63	139	35.20

20	41	67	141	149.75
21	54	71	183	149.75

Figure 4.4: Procedure Output

In addition to selecting variables, you can control the default Obs column that PROC PRINT displays to list observation numbers. If you prefer, you can choose not to display observation numbers.

Obs	Age	Height	Weight	Fee
1	27	72	168	85 20
2	34	66	152	124.80
3	31	61	123	149.75
4	43	63	137	149.75
5	51	71	158	124.00

Figure 4.5: Printing Observations

Removing the OBS Column

To remove the Obs column, specify the NOOBS option in the PROC PRINT statement.

```
proc print data=work.example noobs;
   var age height weight fee;
run;
```

Identifying Observations

You've learned how to remove the Obs column altogether. As another alternative, you can use one or more variables to replace the Obs column in the output.

Using the ID Statement

To specify which variables should replace the Obs column, use the ID statement. This technique is particularly useful when observations are too long to print on one line.

General form, ID statement:

ID *variable(s);*

where *variable(s)* specifies one or more variables to print instead of the observation number at the beginning of each row of the report.

Example

To replace the Obs column and identify observations based on an employee's ID number and last name, you can submit the following program.

```
proc print data=sales.reps;
    id idnum lastname;
run;
```

This is HTML output from the program:

IDnum	LastName	FirstName	City	State	Sex	JobCode	Salary	Birth	Hired	Ho
1269	CASTON	FRANKLIN	STAMFORD	CT	M	NA1	41690.00	06MAY60	01DEC80	20 33
1935	FERNANDEZ	KATRINA	BRIDGEPORT	CT		NA2	51081.00	31MAR42	190CT69	20

										29
1417	NEWKIRK	WILLIAM	PATERSON	NJ	,	NA2	52270.00	30JUN52	10MAR77	20 66
1839	NORRIS	DIANE	NEW YORK	YN	F	NA1	43433.00	02DEC58	06JUL81	71 17
1111	RHODES	JEREMY	PRINCETON	NJ	M	NA1	40586.00	17JUL61	03NOV80	20 18
1352	RIVERS	SIMON	NEW YORK	NY	M	NA2	5379.80	05DEC48	190CT74	71 33
1332	STEPHENSON	ADAM	BRIDGEPORT	CT	M	NAI	42178.00	20SEP58	07JUN79	20 14
1443	WELLS	AGNES	STAMFORD	CT	F	NA1	422.74	20NOV56	01SEP79	20 55

Figure 4.6: HTML Output

Additional Note In LISTING output, the IDnum and LastName columns are repeated for each observation that is printed on more than one line.

```
IDnum   LastName    FirstName   City        State     Sex   JobCode
1269    GASTON      FRANKLIN    STAMFORD    CT        M     NA1
1935    FERNANDEZ   KATRINA     BRIDGEPO    CT              NA2
1417    NEWKIRK     WILLIAM     PATERSON    NJ        '     NA2
1839    NORRIS      DIANE       NEW YORK    NY        F     NA1
1111    RHODES      JEREMY      PRINCETO    NJ        M     NA1
1352    RIVERS      SIMON       NEW YORK    NY        M     NA2
1332    STEPHENS    ADAM        BRIDGEPO    CT        M     NA1
1443    WELLS       AGNES       STAMFORD    CT        F     NA1

IDnum   LastName    Salary      Birth       Hired       HomePhone

1269    GASTON      41690.0 0   06MAY60     01DEC80    203/781-3335
1935    FERNANDEZ   51081.00    31MAR42     190CT69    203/675-2962
1417    NEWKIRK     52270.00    30JUN52     10MAR77    201/732-6611
1839    NORRIS      43433.00    02DEC58     06JUL81    718/384-1767
1111    RHODES      40586.00    17JUL61     O3NOV80    201/812-1837
1352    RIVERS      5379.80     05DEC48     190CT74    718/383-3345
1332    STEPHENS    42178.00    20SEP58     07JUN79    203/675-1497
1443    WELLS       422.74      20NOV56     01SEP79    203/781-5546
```

Figure 4.7: LISTING Output

If a variable in the ID statement also appears in the VAR statement, the output contains two columns for that variable. In the example below, the variable IDnum appears twice.

```
proc print data=sales.reps;
    id idnum lastname;
    var idnum sex jobcode salary;
run;
```

IDnum	LastName	IDnum	Sex	JobCode	Salary
1269	CASTON	1269	M	NA1	41690.00
1935	FERNANDEZ	1935		NA2	51081.00

1417	NEWKIRK	1417	,	NA2	52270.00
1839	NORRIS	1839	F	NA1	43433.00
1111	RHODES	1111	M	NA1	40586.00
1352	RIVERS	1352	M	NA2	5379.80
1332	STEPHENSON	1332	M	NA1	42178.00
1443	WELLS	1443	F	NA1	422.74

Figure 4.8: IDNUM Output

Selecting Observations

By default, a PROC PRINT step lists all the observations in a data set. You can control which observations are printed by adding a WHERE statement to your PROC PRINT step. There should be only one WHERE statement in a step. If multiple WHERE statements are issued, only the last statement is processed.

General form, WHERE statement:

WHERE *where-expression;*

where *where-expression* specifies a condition for selecting observations. The *where-expression* can be any valid SAS expression.

For example, the following WHERE statement selects only observations for which the value of Age is greater than 30:

```
proc print data=clinic.admit;
   var age height weight fee;
   where age>30;
run;
```

Here is the procedure output from the PROC PRINT step with the WHERE statement:

Obs	Age	Height	Weight	Fee
2	34	66	152	124.80
3	31	61	123	149.75
4	43	63	137	149.75
5	51	71	158	124.80
7	32	67	151	149.75
8	35	70	173	149.75
9	34	73	154	124.80
10	49	64	172	124.80
11	44	66	140	149.75
14	40	69	163	124.80
15	47	72	173	124.80
16	60	71	191	149.75
17	43	65	123	124.80
20	41	67	141	149.75
21	54	71	183	149.75

Figure 4.9: PROC PRINT Output with WHERE Statement

Specifying WHERE Expressions

In the WHERE statement you can specify any variable in the SAS data set, not just the variables that are specified in the VAR statement. The WHERE statement works for both character and numeric variables. To specify a condition based on the value of a character variable, you must

- enclose the value in quotation marks
- write the value with lowercase, uppercase, or mixed case letters exactly as it appears in the data set.

You use the following comparison operators to express a condition in the WHERE statement:

Table 4.1: Comparison Operators in a WHERE Statement

Symbol	Meaning	Example
= or eq	equal to	**`where name='Jones, C.';`**
^= or ne	not equal to	**`where temp ne 212;`**
> or gt	greater than	**`where income>20000;`**
< or lt	less than	**`where partno lt "BG05";`**
>= or ge	greater than or equal to	**`where id>='1543';`**
<= or le	less than or equal to	**`where pulse le 85;`**

Additional Note You can learn more about valid SAS expressions in Chapter 5, "Creating SAS Data Sets from External Files," on page 151.

Using the CONTAINS Operator

The CONTAINS operator selects observations that include the specified substring. The symbol for the CONTAINS operator is ?. You can use either the CONTAINS keyword or the symbol in your code, as shown below.

```
where firstname CONTAINS 'Jon';
where firstname ? 'Jon';
```

Specifying Compound WHERE Expressions

You can also use WHERE statements to select observations that meet multiple conditions. To link a sequence of expressions into compound expressions, you use logical operators, including the following:

Table 4.2: Compound WHERE Expression Operators

Operator		Meaning
AND	&	and, both. If both expressions are true, then the compound expression is true.
OR	|	or, either. If either expression is true, then the compound expression is true.

Examples of WHERE Statements

•You can use compound expressions like these in your WHERE statements:

```
where age<=55 and pulse>75;
where area='A' or region='S';
where ID>'1050' and state='NC';
```

•When you test for multiple values of the same variable, you specify the variable name in each expression:

```
where actlevel='LOW' or actlevel='MOD';
where fee=124.80 or fee=178.20;
```

•You can use the IN operator as a convenient alternative:

```
where actlevel in ('LOW','MOD');
where fee in (124.80,178.20);
```

•To control the way compound expressions are evaluated, you can use parentheses (expressions in parentheses are evaluated first):

```
where (age<=55 and pulse>75) or area='A';
```

```
   where age<=55 and (pulse>75 or area='A');
```

Sorting Data

Overview

By default, PROC PRINT lists observations in the order in which they appear in your data set. To sort your report based on values of a variable, you must use PROC SORT to sort your data before using the PRINT procedure to create reports from the data.

The SORT procedure

- rearranges the observations in a SAS data set
- creates a new SAS data set that contains the rearranged observations
- replaces the original SAS data set by default
- can sort on multiple variables
- can sort in ascending or descending order
- does not generate printed output
- treats missing values as the smallest possible values.

General form, simple PROC SORT step:

```
PROC SORT DATA=SAS-data-set<OUT=SAS-data-set>;
              BY <DESCENDING> BY-variable(s);
RUN;
```

where

- the **DATA=** option specifies the data set to be read
- the **OUT=** option creates an output data set that contains the data in sorted order
- *BY-variable(s)* in the required **BY** statement specifies one or more variables whose values are used to sort the data
- the **DESCENDING** option in the BY statement sorts observations in descending order. If you have more that one variable in the BY statement, DESCENDING applies only to the variable that immediately follows it.

Caution If you don't use the OUT= option, PROC SORT overwrites the data set specified in the DATA= option.

Example

In the following program, the PROC SORT step sorts the permanent SAS data set Clinic.Admit by the values of the variable Age within the values of the variable Weight and creates the temporary SAS data set Wgtadmit. Then the PROC PRINT step prints a subset of the Wgtadmit data set.

```
proc sort data=clinic.admit out=work.wgtadmit;
   by weight age;
run;
proc print data=work.wgtadmit;
   var weight age height fee;
   where age>3 0;
run;
```

The report displays observations in ascending order of age within weight.

Obs	Weight	Age	Height	Fee
2	123	31	61	149.75

3	123	43	65	124.80
4	137	43	63	149.75
6	140	44	66	149.75
7	141	41	67	149.75
9	151	32	67	149.75
10	152	34	66	124.80
11	154	34	73	124.80
12	153	51	71	124.80
13	1S3	40	69	124.80
15	172	49	64	124.80
16	173	35	70	149.75
17	173	47	72	124.80
18	183	54	71	149.75
20	191	60	71	149.75

Figure 4.10: Observations Displayed in Ascending Order of Age Within Weight

Adding the DESCENDING option to the BY statement sorts observations in ascending order of age within descending order of weight. Notice that DESCENDING applies only to the variable Weight.

```
proc sort data=clinic.admit out=work.wgtadmit;
   by descending weight age;
run;
proc print data=work.wgtadmit;
   var weight age height fee;
   where age>30;
run;
```

Obs	Weight	Age	Height	Fee
2	191	60	71	149.75
4	183	54	71	149.75
5	173	35	70	149.75
6	173	47	72	124.80
7	172	49	64	124.80
9	163	40	69	124.80
10	158	51	71	124.80
11	154	34	73	124.80
12	152	34	66	124.80
13	151	32	67	149.75
15	141	41	67	149.75
16	140	44	66	149.75
18	137	43	63	149.75
19	123	31	61	149.75
20	123	43	65	124.80

Figure 4.11: Observations Displayed in Descending Order

Generating Column Totals

Overview

To produce column totals for numeric variables, you can list the variables to be summed in a SUM statement in your PROC PRINT step.

General form, SUM statement:

SUM *variable(s);*

where *variable(s)* is one or more numeric variable names, separated by blanks.

The SUM statement in the following PROC PRINT step requests column totals for the variable BalanceDue:

```
proc print data=clinic.insure;
   var name policy balancedue;
   where pctinsured < 100;
   sum balancedue;
run;
```

Column totals appear at the end of the report in the same format as the values of the variables.

Obs	Name	Policy	BalanceDue
2	Almers, C	95824	156.05
3	Bonaventura, T	87795	9.48
4	Johnson. R	39022	61.04
5	LaMance, K	63265	43.68
6	Jones, M	92478	52.42
7	Reberson. P	25530	207.41
8	King. E	18744	27 19
9	Pitts. D	60976	310.82
10	Eberhardt, S	81589	173.17
13	Peterson. V	75986	228.00
14	Quigley, M	97048	99.01
15	Cameron. L	42351	111.41
17	Takahashi. Y	54219	186.58
18	Derber. B	74653	236 11
20	Wilcox. E	94034	212.20
21	Warren. C	20347	164.44
			2279.0

Figure 4.12: Column Totals

Requesting Subtotals

You might also want to subtotal numeric variables. To produce subtotals, add both a SUM statement and a BY statement to your PROC PRINT step.

General form, BY statement in the PRINT procedure:

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

where

- *BY-variable* specifies a variable that the procedure uses to form BY groups. You can specify more than one variable, separated by blanks.
- the DESCENDING option specifies that the data set is sorted in descending order by the variable that immediately follows.
- the NOTSORTED option specifies that observations are not necessarily sorted in alphabetic or numeric order. If observations that have the same values for the BY variables are not contiguous, the procedure treats each contiguous set as a separate BY group.

Caution 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.

Example

The SUM statement in the following PROC PRINT step requests column totals for the variable Fee, and the BY statement produces a subtotal for each value of ActLevel.

```
proc sort data=clinic.admit out=work.activity;
   by actlevel;
run;
proc print data=work.activity;
   var age height weight fee;
   where age>30;
   sum fee;
   by actlevel;
run;
```

In the output, the BY variable name and value appear before each BY group. The BY variable name and the subtotal appear at the end of each BY group.

ActLevel=HIGH				
Obs	**Age**	**Height**	**Weight**	**Fee**
2	34	66	152	124.80
4	44	66	140	149.75
5	40	69	163	124.80
7	41	67	141	149.75
Act Level				**549.10**

Figure 4.13: BY Group Output: High

ActLevel=LOW				
Obs	**Age**	**Height**	**Weight**	**Fee**
8	31	61	123	149.75
9	51	71	158	124.80
10	34	73	154	124.80
11	49	64	172	124.80
13	60	71	191	149.75
ActLevel				**673.90**

Figure 4.14: BY Group Output: Low

ActLevel=MOD				
Obs	**Age**	**Height**	**Weight**	**Fee**

15	43	63	137	149.75
16	32	67	151	149.75
17	35	70	173	149.75
19	47	72	173	124.80
20	43	65	123	124.80
21	54	71	183	149.75
ActLevel				**848.60**
				2071.60

Figure 4.15: BY Group Output: Mod

Creating a Customized Layout with BY Groups and ID Variables

In the previous example, you may have noticed the redundant information for the BY variable. For example, in the partial PROC PRINT output below, the BY variable ActLevel is identified both before the BY group and for the subtotal.

ActLevel=HIGH				
Obs	**Aget**	**Heigh**	**Weight**	**Fee**
2	34	66	152	124.80
4	44	66	140	149.75
5	40	69	163	124.80
7	41	67	141	149.75
ActLevel				**549.10**

Figure 4.16: Creating a Customized Layout with BY Groups and ID Variables

To show the BY variable heading only once, you can use an ID statement and a BY statement together with the SUM statement. When an ID statement specifies the same variable as the BY statement,

- the Obs column is suppressed
- the ID/BY variable is printed in the left-most column
- each ID/BY value is printed only at the start of each BY group and on the line that contains that group's subtotal.

Example

The ID, BY, and SUM statements work together to produce the output shown below. The ID variable is listed only once for each BY group and once for each sum. The BY lines are suppressed. Instead, the value of the ID variable, ActLevel, identifies each BY group.

```
proc sort data=clinic.admit out=work.activity;
   by actlevel;
run;
proc print data=work.activity;
   var age height weight fee;
   where age>30;
   sum fee;
   by actlevel;
   id actlevel;
run;
```

ActLevel	Age	Height	Weight	Fee
HIGH	34	66	152	124.80
	44	66	140	149.75
	40	69	163	124.80

	41	67	141	149.75
HIGH				**549.10**
LOW	31	61	123	149.75
	51	71	158	124.80
	34	73	154	124.80
	49	64	172	124.80
	60	71	191	149.75
LOW				**673.90**
MOD	43	63	137	149.75
	32	67	151	149.75
	35	70	173	149.75
	47	72	173	124.80
	43	65	123	124.80
	54	71	183	149.75
MOD				**848.60**
				2071.60

Figure 4.17: Creating Custom Output Example Output

Requesting Subtotals on Separate Pages

As another enhancement to your PROC PRINT report, you can request that each BY group be printed on a separate page by using the PAGEBY statement.

General form, PAGEBY statement:

PAGEBY *BY-variable:*

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

Caution The variable specified in the PAGEBY statement must also be specified in the BY statement in the PROC PRINT step.

Example

The PAGEBY statement in the program below prints BY groups for the variable ActLevel separately. The BY groups appear separated by horizontal lines in the HTML output.

```
proc sort data=clinic.admit out=work.activity;
   by actlevel;
run;
proc print data=work.activity;
   var age height weight fee;
   where age>30;
   sum fee;
   by actlevel;
   id actlevel;
   pageby actlevel;
run;
```

ActLevel	Age	Height	Weight	Fee
HIGH	34	66	152	124.80

	44	66	140	149.75
	40	69	163	124.80
	41	67	141	149.75
HIGH				**549.10**

Figure 4.18: PAGEBY Example: High

ActLevel	Age	Height	Weight	Fee
LOW	31	61	123	149.75
	51	71	158	124.80
	34	73	154	124.80
	49	64	172	124.80
	60	71	191	149.75
LOW				**673.90**

Figure 4.19: PAGEBY Example: Low

ActLevel	Age	Height	Weight	Fee
MOD	43	63	137	149.75
	32	67	151	149.75
	35	70	173	149.75
	47	72	173	124.80
	43	65	123	124.80
	54	71	183	149.75
MOD				**848.60**
				2071.60

Figure 4.20: PAGEBY Example: Mod

Double Spacing LISTING Output

If you are generating SAS LISTING output, one way to control the layout is to double space it. To do so, specify the DOUBLE option in the PROC PRINT statement. For example,

```
proc print data=clinic.stress double;
   var resthr maxhr rechr;
   where tolerance='I';
run;
```

Additional Note Double spacing does not apply to HTML output.

Additional Note To generate SAS LISTING output, you must select **Tools** ⇨ **Options** ⇨ **Preferences**. Select the **Results** tab. Select the **Create listing** option.

SAS Output

```
OBS   ResttHR   MaxHR   RecHR
  2        68     171     133
  3        78     177     139
  8        70     167     122
 11        65     181     141
 14        74     152     113
 15        75     158     108
```

```
   20        78       189       138
```

Figure 4.21: Double-Spaced LISTING Output

Specifying Titles and Footnotes

Overview

Now you've learned how to structure your PRINT procedure output. However, you might also want to make your reports easy to interpret by

- adding titles and footnotes
- replacing variable names with descriptive labels
- formatting variable values.

Although this chapter focuses on PROC PRINT, you can apply these enhancements to most SAS procedure output.

TITLE and FOOTNOTE Statements

To make your report more meaningful and self-explanatory, you can associate up to 10 titles with procedure output by using TITLE statements before the PROC step. Likewise, you can specify up to 10 footnotes by using FOOTNOTE statements before the PROC step.

Additional Note Because TITLE and FOOTNOTE statements are global statements, place them anywhere within or before the PRINT procedure. Titles and footnotes are assigned as soon as TITLE or FOOTNOTE statements are read; they apply to all subsequent output.

General form, TITLE and FOOTNOTE statements:

TITLE<*n*> *'text';*

FOOTNOTE<*n*> *'text';*

where *n* is a number from 1 to 10 that specifies the title or footnote line, and *'text'* is the actual title or footnote to be displayed. The maximum title or footnote length depends on your operating environment and on the value of the LINESIZE= option.

The keyword **`title`** is equivalent to title1. Likewise, **`footnote`** is equivalent to footnote1. If you don't specify a title, the default title is The SAS System. No footnote is printed unless you specify one.

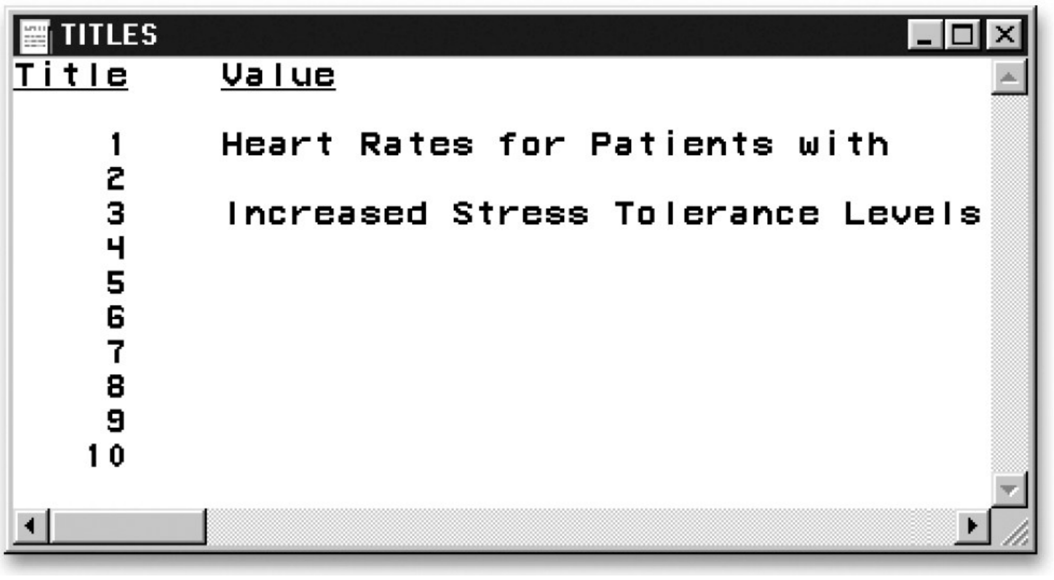
Caution Be sure to match quotation marks that enclose the title or footnote text.

Using the TITLES and FOOTNOTES Windows

You can also specify titles in the TITLES window and footnotes in the FOOTNOTES window. Titles and footnotes that you specify in these windows are not stored with your program, and they remain in effect only during your SAS session.

To open the TITLES window, issue the TITLES command. To open the FOOTNOTES window, issue the FOOTNOTES command.

To specify a title or footnote, type in the text you want next to the number of the line where the text should appear. To cancel a title or footnote, erase the existing text. Notice that you do not enclose text in quotation marks in these windows.



```
TITLES
Title     Value

    1     Heart Rates for Patients with
    2
    3     Increased Stress Tolerance Levels
    4
    5
    6
    7
    8
    9
   10
```

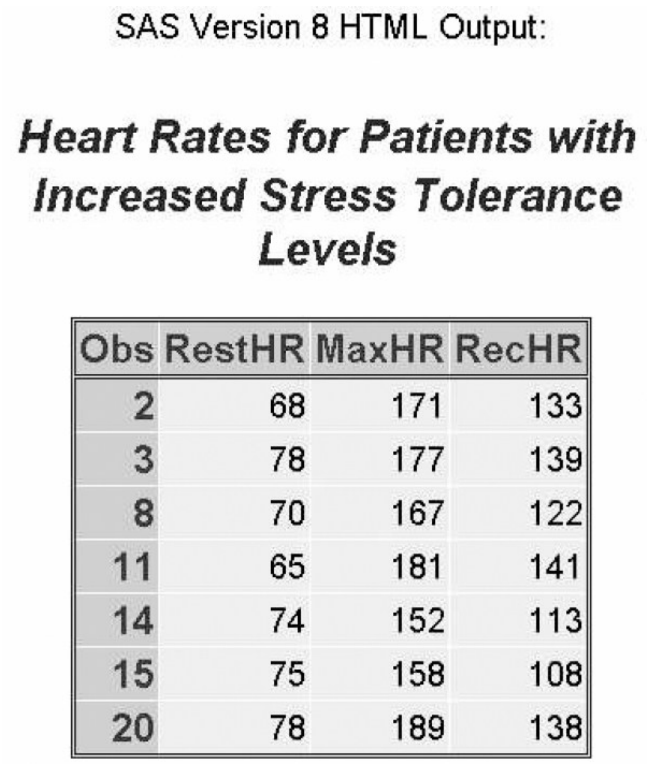
Figure 4.22: Titles Window

Example: Titles

The two TITLE statements below, specified for lines 1 and 3, define titles for the PROC PRINT output.

```
titlel 'Heart Rates for Patients with'
title3 'Increased Stress Tolerance Levels';
proc print data=clinic.stress;
   var resthr maxhr rechr;
   where tolerance='I';
run;
```

Title lines for HTML output appear differently depending on the version of SAS that you use. In SAS Version 8, title lines simply appear consecutively, without extra spacing to indicate skipped title numbers. In SAS®9 HTML output, title line 2 is blank.

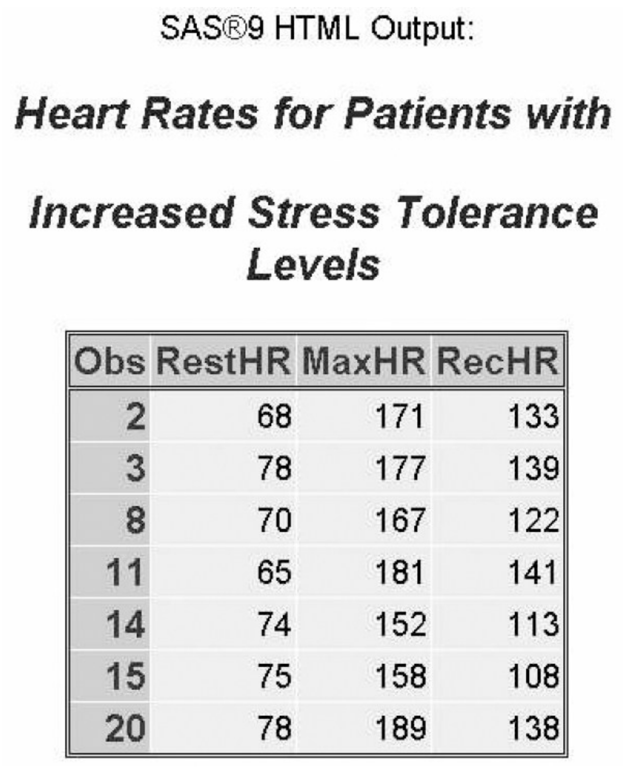


SAS Version 8 HTML Output:

Heart Rates for Patients with
Increased Stress Tolerance Levels

Obs	RestHR	MaxHR	RecHR
2	68	171	133
3	78	177	139
8	70	167	122
11	65	181	141
14	74	152	113
15	75	158	108
20	78	189	138

Figure 4.23: HTML Output with Titles: SAS®8



SAS®9 HTML Output:

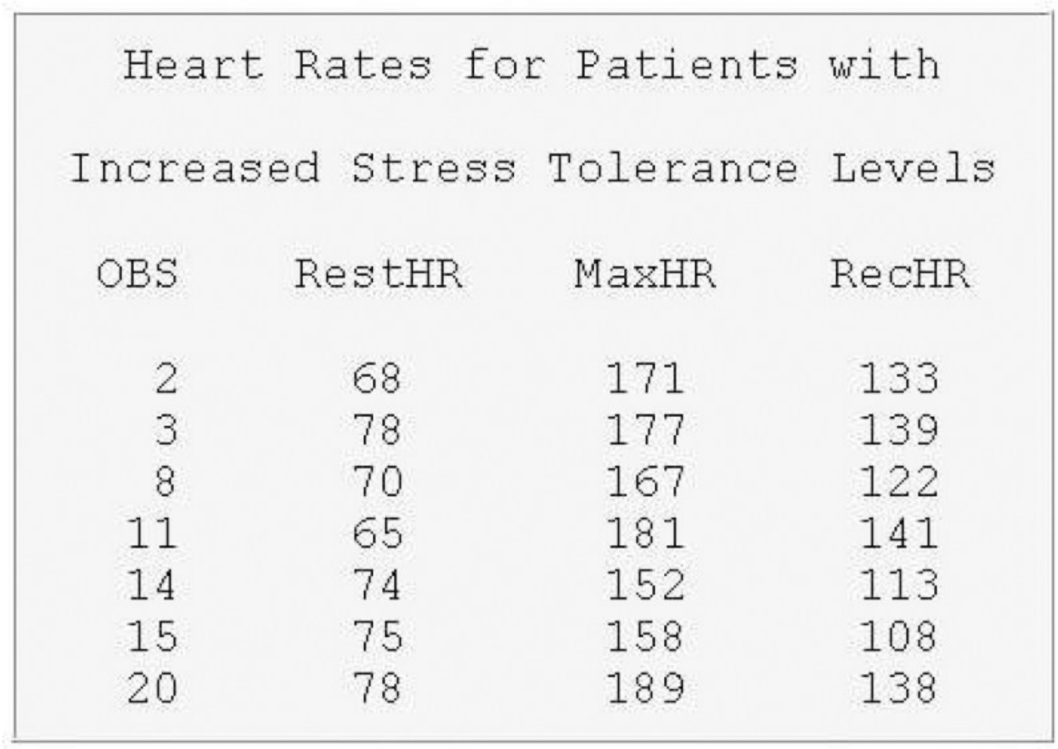
Heart Rates for Patients with

Increased Stress Tolerance Levels

Obs	RestHR	MaxHR	RecHR
2	68	171	133
3	78	177	139
8	70	167	122
11	65	181	141
14	74	152	113
15	75	158	108
20	78	189	138

Figure 4.24: HTML Output with Titles: SAS®9

In SAS LISTING output for all versions of SAS, title line 2 is blank, as shown below. Titles are centered by default.



```
     Heart Rates for Patients with

  Increased Stress Tolerance Levels

 OBS     RestHR     MaxHR     RecHR

   2       68        171       133
   3       78        177       139
   8       70        167       122
  11       65        181       141
  14       74        152       113
  15       75        158       108
  20       78        189       138
```

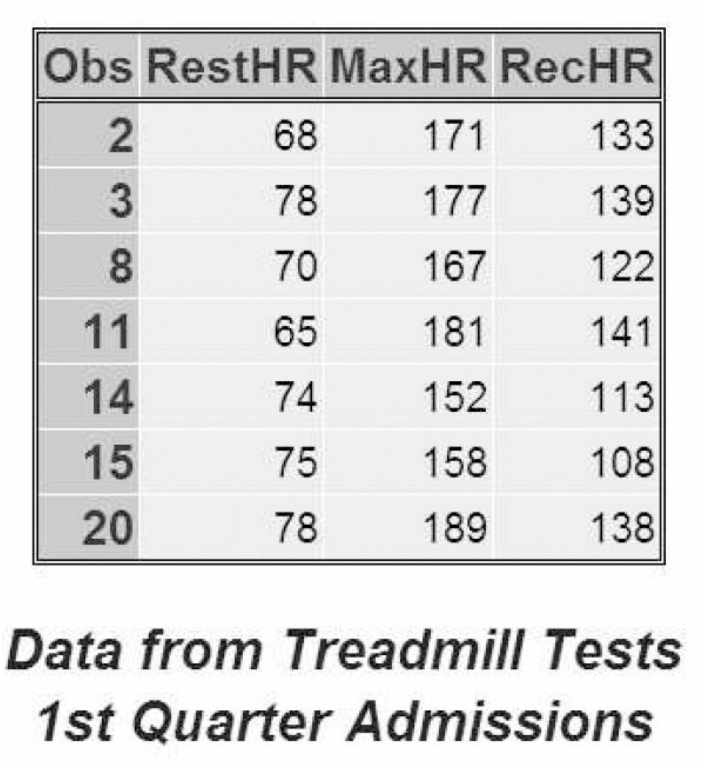
Figure 4.25: LISTING Output with Titles: All Versions

Example: Footnotes

The two FOOTNOTE statements below, specified for lines 1 and 3, define footnotes for the PROC PRINT output. Since there is no footnote2, a blank line is inserted between footnotes 1 and 2 in the output.

```
footnotel 'Data from Treadmill Tests';
footnote3 '1st Quarter Admissions';
proc print data=clinic.stress;
   var resthr maxhr rechr;
   where tolerance='I';
run;
```

Footnotes appear at the bottom of each page of procedure output. Notice that footnote lines are *pushed up* from the bottom. The FOOTNOTE statement that has the largest number appears on the bottom line.



Obs	RestHR	MaxHR	RecHR
2	68	171	133
3	78	177	139
8	70	167	122
11	65	181	141
14	74	152	113
15	75	158	108
20	78	189	138

Data from Treadmill Tests

1st Quarter Admissions

Figure 4.26: HTML Output with Footnotes

In SAS LISTING output, footnote line 2 is blank, as shown below. Footnotes are centered by default.

```
OBS   RestHR   MaxHR   RecHR
  2       68     171     133
  3       78     177     139
  8       70     167     122
 11       65     181     141
 14       74     152     113
 15       75     158     108
 20       78     189     138
     Data from Treadmill Tests
      1st Quarter Admissions
```

Figure 4.27: LISTING Output with Footnotes

Modifying and Canceling Titles and Footnotes

TITLE and FOOTNOTE statements are global statements. That is, after you define a title or footnote, it remains in effect until you modify it, cancel it, or end your SAS session.

For example, the footnotes that are assigned in the PROC PRINT step below also appear in the output from the PROC TABULATE step.

```
footnotel 'Data from Treadmill Tests';
footnote3 '1st Quarter Admissions';
proc print data=clinic.stress;
   var resthr maxhr rechr;
   where tolerance='I';
run;
proc tabulate data=clinic.stress;
   where tolerance='I';
   var resthr maxhr;
   table mean*(resthr maxhr);
run;
```

Redefining a title or footnote line cancels any higher-numbered title or footnote lines, respectively. In the example below, defining a title for line 2 in the second report automatically cancels title line 3.

```
title3 'Participation in Exercise Therapy';
proc print data=clinic.therapy;
   var swim walkjogrun aerclass;
run;
title2 'Report for March';
proc print data=clinic.therapy;
run;
```

To cancel all previous titles or footnotes, specify a null TITLE or FOOTNOTE statement (a TITLE or FOOTNOTE statement with no number or text) or a TITLE1 or FOOTNOTE1 statement with no text. This will also cancel the default title The SAS System.

For example, in the program below, the null TITLE1 statement cancels all titles that are in effect before either PROC step executes. The null FOOTNOTE statement cancels all footnotes that are in effect after the PROC PRINT step executes. The PROC TABULATE output appears without a title or a footnote.

```
title1;
footnote1 'Data from Treadmill Tests';
footnote3 '1st Quarter Admissions';
proc print data=clinic.stress;
   var resthr maxhr rechr;
   where tolerance='I';
run;
footnote;
proc tabulate data=clinic.stress;
   var timemin timesec;
   table max*(timemin timesec);
run;
```

Assigning Descriptive Labels

Temporarily Assigning Labels to Variables

You can also enhance your PROC PRINT report by labeling columns with more descriptive text. To label columns, you use

- the LABEL statement to assign a descriptive label to a variable
- the LABEL option in the PROC PRINT statement to specify that the labels be displayed.

General form, LABEL statement:

```
LABEL variable1 = 'label1'
        variable2='label2'
        …;
```

Labels can be up to 256 characters long. Enclose the label in quotation marks.

Additional Note The LABEL statement applies only to the PROC step in which it appears.

Example

In the PROC PRINT step below, the variable name WalkJogRun is displayed with the label Walk/Jog/Run. Note the LABEL option in the PROC PRINT statement. Without the LABEL option in the PROC PRINT statement, PROC PRINT would use the name of the column heading **`walkjogrun`** even though you specified a value for the variable.

```
proc print data=clinic.therapy label;
   label walkjogrun=Walk/Jog/Run;
run;
```

Obs	Date	AerClIass	Walk/Jog/Run	Swim
6	JUN1999	61	114	67
7	JUL1999	67	102	72
8	AUG1999	64	76	77
9	SEP1999	78	77	54
10	OCT1999	31	62	47
11	NOVI 999	84	31	52
16	APR2000	61	118	22
20	AUG20C0	63	65	53
22	OCT2000	78	70	41
23	NOV2000	82	44	58
24	OEC2000	93	57	47

Figure 4.28: Output Created Without the LABEL Option

Using Single or Multiple LABEL Statements

You can assign labels in separate LABEL statements …

```
proc print data=clinic.admit label;
   var age height;
   label age='Age of Patient';
   label height='Height in Inches';
run;
```

… or you can assign any number of labels in a single LABEL statement.

```
proc print data=clinic.admit label;
   var actlevel height weight;
   label actlevel='Activity Level'
          height='Height in Inches'
          weight='Weight in Pounds';
run;
```

Formatting Data Values

Temporarily Assigning Formats to Variables

In your SAS reports, formats control how the data values are displayed. To make data values more understandable when they are displayed in your procedure output, you can use the FORMAT statement, which associates formats with variables.

Formats affect only how the data values appear in output, *not* the actual data values as they are stored in the SAS data set.

General form, FORMAT statement:

FORMAT *variable(s) format-name*;

where

- *variable(s)* is the name of one or more variables whose values are to be written according to a particular pattern
- *format-name* specifies a SAS format or a user-defined format that is used to write out the values.

Additional Note The FORMAT statement applies only to the PROC step in which it appears.

You can use a separate FORMAT statement for each variable, or you can format several variables (using either the same format or different formats) in a single FORMAT statement.

Table 4.3: Formats That Are Used to Format Data

This FORMAT statement …	Associates …	To display values as …
`format date mmddyy8.;`	the format MMDDYY8. with the variable Date	**06/05/03**
`format net comma5.0 gross comma8.2;`	the format COMMA5.0 with the variable Net and the format COMMA8.2 with the variable Gross	**1,234** **5,678.90**
`format net gross dollar9.2;`	the format DOLLAR9.2 with both variables, Net and Gross	**$1,234.00** **$5,678.90**

For example, the FORMAT statement below writes values of the variable Fee using dollar signs, commas, and no decimal places.

```
proc print data=clinic.admit;
   var actlevel fee;
   where actlevel='HIGH';
   format fee dollar4.;
run;
```

Obs	ActLevel	Fee
1	HIGH	$85
2	HIGH	$125
6	HIGH	$125
11	HIGH	$150
14	HIGH	$125
18	HIGH	$85
20	HIGH	$150

Figure 4.29: FORMAT Statement Example

Specifying SAS Formats

The table below describes some SAS formats that are commonly used in reports.

Table 4.4: Commonly Used SAS Formats

Format	Specifies values …	Example
COMMA**w.d**	that contain commas and decimal places	**comma8 .2**
DOLLAR **w.d**	that contain dollar signs, commas, and decimal places	**dollar 6.2**
MMDDYY**w.**	as date values of the form 09/12/97 (MMDDYY8.) or 09/12/1997 (MMDDYY10.)	**mmddyy 10.**
w.	rounded to the nearest integer in **w** spaces	**7.**
w.d	rounded to d decimal places in **w** spaces	**8.2**
$w.	as character values in **w** spaces	**$12.**

DATEw.	as date values of the form 16OCT99 (DATE7.) or 16OCT1999 (DATE9.)	**date9.**

Field Widths

All SAS formats specify the total field width **(w)** that is used for displaying the values in the output. For example, suppose the longest value for the variable Net is a four-digit number, such as **5400.** To specify the COMMA**w.d** format for Net, you specify a field width of 5 or more. You must count the comma, because it occupies a position in the output.

Caution When you use a SAS format, be sure to specify a field width (w) that is wide enough for the largest possible value. Otherwise, values might not be displayed properly.

```
format net comma5.0;
5  ,  4  0  0
1  2  3  4  5
```

Figure 4.30: Specifying a Field Width (w) with the FORMAT Statement

Decimal Places

For numeric variables you can also specify the number of decimal places **(d)**, if any, to be displayed in the output. Numbers are rounded to the specified number of decimal places. In the example above, no decimal places are displayed.

Writing the whole number 2030 as 2,030.00 requires eight print positions, including two decimal places and the decimal point.

```
   format qtr3tax comma8.2;
2  ,  0  3  0  .  0  0
1  2  3  4  5  6  7  8
```

Figure 4.31: Whole Number Decimal Places

Formatting 15374 with a dollar sign, commas, and two decimal places requires ten print positions.

```
         format totsales dollar10.2;
$  1  5  ,  3  7  4  .  0  0
1  2  3  4  5  6  7  8  9  10
```

Figure 4.32: Specifying 10 Decimal Places

Examples

This table shows you how data values are displayed when different format, field width, and decimal place specifications are used.

Table 4.5: Displaying Data Values with Formats

Stored Value	Format	Displayed Value
38245.3975	COMMA12.2	38,245.40
38245.3975	12.2	38245.40
38245.3975	DOLLAR12.2	$38,245.40
38245.3975	DOLLAR9.2	$38245.40

38245.3975	DOLLAR8.2	38245.40
0	MMDDYY8.	01/01/60
0	MMDDYY10.	01/01/1960
0	DATE7.	01JAN60
0	DATE9.	01JAN1960

Additional Note If a format is too small, the following message is written to the SAS log: "NOTE: At least one W.D format was too small for the number to be printed. The decimal may be shifted by the 'BEST' format."

Using Permanently Assigned Labels and Formats

You have seen how to *temporarily* assign labels and formats to variables. When you use a LABEL or FORMAT statement within a PROC step, the label or format applies only to the output from that step.

However, in your PROC steps, you can also take advantage of *permanently* assigned labels or formats. Permanent labels and formats can be assigned in the DATA step. These labels and formats are saved with the data set, and they can later be used by procedures that reference the data set.

For example, the DATA step below creates Flights.March and defines a format and label for the variable Date. Because the LABEL and FORMAT statements are inside the DATA step, they are written to the Flights.March data set and are available to the subsequent PRINT procedure.

```
data sasuser.paris;
set sasuser.laguardia;
   where dest="PAR" and (boarded=155 or boarded=146);
   label date='Departure Date';
   format date date9.;
run;

proc print data=sasuser.paris;
   var date dest boarded;
run;
```

Obs	Departure Date	Dest	Boarded
1	04MAR1999	PAR	146
2	07MAR1999	PAR	155
3	04MAR1999	PAR	146
4	07MAR1999	PAR	155

Figure 4.33: Using Permanent Labels and Formats

Notice that the PROC PRINT statement still requires the LABEL option in order to display the permanent labels. Other SAS procedures display permanently assigned labels and formats without additional statements or options.

Additional Note You can learn about permanently assigning labels and formats in "Creating and Managing Variables" on page 304.

Additional Features

When you create list reports, you can use several other features to enhance your procedure output. For example, you can

- control where text strings split in labels by using the SPLIT= option.

  ```
  proc print data=reps split='*';
     var salesrep type unitsold net commission;
     label salesrep='Sales*Representative';
  run;
  ```

- create your own formats, which are particularly useful for formatting character values.

  ```
  proc format;
     value $repfmt
            'TFB'='Bynum'
  ```

```
             'MDC'='Crowley'
             'WKK'='King';
   run;
   proc print data=vcrsales;
       var salesrep type unitsold;
       format salesrep $repfmt.;
   run;
```

Additional Note You can learn more about user-defined formats in "Creating and Applying User-Defined Formats" on page 233 .

Chapter Summary

Text Summary

Creating a Basic Report

To list the information in a SAS data set, you can use PROC PRINT. You use the PROC PRINT statement to invoke the PRINT procedure and to specify the data set that you are listing. Include the DATA= option to specify the data set that you are using. By default, PROC PRINT displays all observations and variables in the data set, includes a column for observation numbers on the far left, and displays observations and variables in the order in which they occur in the data set. If you use a LABEL statement with PROC PRINT, you must specify the LABEL option in the PROC PRINT statement.

To refine a basic report, you can

- select which variables and observations are processed
- sort the data
- generate column totals for numeric variables.

Selecting Variables

You can select variables and control the order in which they appear by using a VAR statement in your PROC PRINT step. To remove the Obs column, you can specify the NOOBS option in the PROC PRINT statement. As an alternative, you can replace the Obs column with one or more variables by using the ID statement.

Selecting Observations

The WHERE statement enables you to select observations that meet a particular condition in the SAS data set. You use comparison operators to express a condition in the WHERE statement. You can also use the CONTAINS operator to express a condition in the WHERE statement. To specify a condition based on the value of a character variable, you must enclose the value in quotation marks, and you must write the value with lower and uppercase letters exactly as it appears in the data set. You can also use the WHERE statement to select a subset of observations based on multiple conditions. To link a sequence of expressions into compound expressions, you use logical operators. When you test for multiple values of the same variable, you specify the variable name in each expression. You can use the IN operator as a convenient alternative. To control how compound expressions are evaluated, you can use parentheses.

Sorting Data

To display your data in sorted order, you use PROC SORT to sort your data before using PROC PRINT to create reports. By default, PROC SORT sorts the data set specified in the DATA= option permanently. If you do not want your data to be sorted permanently, you must create an output data set that contains the data in sorted order. The OUT= option in the PROC SORT statement specifies an output data set. If you need sorted data to produce output for only one SAS session, you should specify a temporary SAS data set as the output data set. The BY statement, which is required with PROC SORT, specifies the variable(s) whose values are used to sort the data.

Generating Column Totals

To total the values of numeric variables, use the SUM statement in the PROC PRINT step. You do not need to specify the variables in a VAR statement if you specify them in the SUM statement. Column totals appear at the end of the report in the same format as the values of the variables. To produce subtotals, add both the SUM statement and the BY statement to your PROC PRINT step. To show BY variable headings only once, use an ID and BY statement together with the SUM statement. As another enhancement to your report, you can request that each BY group be printed on a separate page by using the PAGEBY statement.

Double Spacing Output

To double space your SAS LISTING output, you can specify the DOUBLE option in the PROC PRINT statement.

Specifying Titles

To make your report more meaningful and self-explanatory, you can associate up to 10 titles with procedure output by using TITLE statements anywhere within or preceding the PROC step. After you define a title, it remains in effect until you modify it, cancel it, or end your SAS session. Redefining a title line cancels any higher-numbered title lines. To cancel all previous titles, specify a null TITLE statement (a TITLE statement with no number or text).

Specifying Footnotes

To add footnotes to your output, you can use the FOOTNOTE statement. Like TITLE statements, FOOTNOTE statements are global. Footnotes appear at the bottom of each page of procedure output, and footnote lines are *pushed up* from the bottom. The FOOTNOTE statement that has the largest number appears on the bottom line. After you define a footnote, it remains in effect until you modify it, cancel it, or end your SAS session. Redefining a footnote line cancels any higher-numbered footnote lines. To cancel all previous footnotes, specify a null FOOTNOTE statement (a FOOTNOTE statement with no number or text).

Assigning Descriptive Labels

To label the columns in your report with more descriptive text, you use the LABEL statement, which assigns a descriptive label to a variable. To display the labels that were assigned in a LABEL statement, you must specify the LABEL option in the PROC PRINT statement.

Formatting Data Values

To make data values more understandable when they are displayed in your procedure output, you can use the FORMAT statement, which associates formats with variables. The FORMAT statement remains in effect only for the PROC step in which it appears. Formats affect only how the data values appear in output, not the actual data values as they are stored in the SAS data set. All SAS formats can specify the total field width (w) that is used for displaying the values in the output. For numeric variables you can also specify the number of decimal places (d), if any, to be displayed in the output.

Using Permanently Assigned Labels and Formats

You can take advantage of permanently assigned labels or formats without adding LABEL or FORMAT statements to your PROC step.

Syntax

```
LIBNAME libref'SAS-data-library';
OPTIONS options;
PROC SORT DATA=SAS-data-set OUT=SAS-data-set;
            BY variable(s);
RUN;
TITLE<n> 'text';
FOOTNOTE<n> 'text';
PROC PRINT DATA=SAS-data-set
            BY<DESCENDING>BY-variable-1<…<DESCENDING><BY-variable-n>>
            <NOTSORTED>;
            PAGEBYBY-variable;
            NOOBS LABEL DOUBLE;
            ID variable(s);
            VAR variable(s);
            WHERE where-expression;
            SUM variable(s);
      LABEL variable1='labeU' variable2='label2' …
     FORMAT variable(s) format-name;
RUN;
```

Sample Program

```
libname clinic 'c:\stress\labdata';
options nodate number pageno=15;
proc sort data=clinic.stress out=work.maxrates;
```

```
   by maxhr;
   where tolerance='I' and resthr>60;
run;
title 'August Admission Fees';
footnote 'For High Activity Patients';
proc print data=work.maxrates label double noobs;
   id name;
   var resthr maxhr rechr;
   label rechr='Recovery HR';
run;

proc print data=clinic.admit label;
   var actlevel fee;
   where actlevel='HIGH';
   label fee='Admission Fee';
   sum fee;
   format fee dollar4.;
run;
```

Points to Remember

- VAR, WHERE, SUM, FORMAT and LABEL statements remain in effect only for the PROC step in which they appear.
- If you don't use the OUT= option, PROC SORT permanently sorts the data set specified in the DATA= option.
- TITLE and FOOTNOTE statements remain in effect until you modify them, cancel them, or end your SAS session.
- Be sure to match the quotation marks that enclose the text in TITLE, FOOTNOTE, and LABEL statements.
- To display labels in PRINT procedure output, remember to add the LABEL option to the PROC PRINT statement.
- To permanently assign labels or formats to data set variables, place the LABEL or FORMAT statement inside the DATA step.

Chapter Quiz

Select the best answer for each question. After completing the quiz, you can check your answers using the answer key in the appendix.

1. Which PROC PRINT step below creates the following output? ?

Date	On	Changed	Flight
04MAR99	232	18	219
05MAR99	160	4	219
06MAR99	163	14	219
07MAR99	241	9	219
08MAR99	183	11	219
09MAR99	211	18	219
10MAR99	167	7	219

a.
```
proc print data=flights.laguardia noobs;
   var on changed flight;
   where on<=160;
run;
```

b.
```
proc print data=flights.laguardia;
   var date on changed flight;
   where changed>3;
run;
```

c.
```
proc print data=flights.laguardia label;
   id date;
   var boarded transferred flight;
   label boarded='On' transferred='Changed';
```

```
      where flight='219';
   run;
```

d.
```
proc print flights.laguardia noobs;
   id date;
   var date on changed flight;
   where flight='219';
run;
```

2. Which of the following PROC PRINT steps is correct if labels are not stored with the data set? ?

a.
```
proc print data=allsales.totals label;
   label region8='Region 8 Yearly Totals';
run;
```

b.
```
proc print data=allsales.totals;
   label region8='Region 8 Yearly Totals';
run;
```

c.
```
proc print data allsales.totals label noobs;
run;
```

d.
```
proc print allsales.totals label;
run;
```

3. Which of the following statements selects from a data set only those observations for which the value of the variable Style is **RANCH, SPLIT**, or **TWOSTORY?** ?

a. `where style='RANCH' or 'SPLIT' or 'TWOSTORY';`

b. `where style in 'RANCH' or 'SPLIT' or 'TWOSTORY';`

c. `where style in (RANCH, SPLIT, TWOSTORY);`

d. `where style in ('RANCH','SPLIT','TWOSTORY');`

4. If you want to sort your data and create a temporary data set named Calc to store the sorted data, which of the following steps should you submit? ?

a.
```
proc sort data=work.calc out=finance.dividend;
run;
```

b.
```
proc sort dividend out=calc;
   by account;
run;
```

c.
```
proc sort data=finance.dividend out=work.calc;
    by account;
run;
```

d.
```
proc sort from finance.dividend to calc;
   by account;
run;
```

5. Which options are used to create the following PROC PRINT output? ?

```
                 13:27 Monday, March22, 1999
patient   Arterial   Heart   Cardiac   Urinary
   203       88        95       66        110
    54       83       183       95          0
```

664	72	111	332	12
210	74	97	369	0
101	80	130	291	0

a. `the DATE system option and the LABEL option in PROC PRINT`

b. `the DATE and NONUMBER system options and the DOUBLE and NOOBS options in PROC PRINT`

c. `the DATE and NONUMBER system options and the DOUBLE option in PROC PRINT`

d. `the DATE and NONUMBER system options and the NOOBS option in PROC PRINT`

6. Which of the following statements can you use in a PROC PRINT step to create this output?

Month	Instructors	AerClass	WalkJogRun	Swim
01	1	37	91	83
02	2	41	102	27
03	1	52	98	19
04	1	61	118	22
05	3	49	88	29
	8	**240**	**497**	**180**

a.
```
var month instructors;
sum instructors aerclass walkjogrun swim;
```

b.
```
var month;
sum instructors aerclass walkjogrun swim;
```

c.
```
var month instructors aerclass;
sum instructors aerclass walkjogrun swim;
```

d. `all of the above`

7. What happens if you submit the following program?

```
proc sort data=clinic.diabetes;
run;
proc print data=clinic.diabetes;
   var age height weight pulse;
   where sex='F';
run;
```

a. The PROC PRINT step runs successfully, printing observations in their sorted order.

b. The PROC SORT step permanently sorts the input data set.

c. The PROC SORT step generates errors and stops processing, but the PROC PRINT step runs successfully, printing observations in their original (unsorted) order.

d. The PROC SORT step runs successfully, but the PROC PRINT step generate errors and stops processing.

8. If you submit the following program, which output does it create?

```
proc sort data=finance.loans out=work.loans;
   by months amount;
run;
proc print data=work.loans noobs;
   var months;
   sum amount payment;
   where months<360;
run;
```

Months	Amount	Payment
12	$3,500	$308.52

a.

24	$8,700	$403.47
36	$10,000	$325.02
48	$5,000	$128.02
	$27,200	$1,165.03

b.

Months	Amount	Payment
12	$3,500	$308.52
24	$8,700	$403.47
36	$10,000	$325.02
48	$5,000	$128.02
	27,200	1,165.03

c.

Months	Amount	Payment
12	$3,500	$308.52
48	$5,000	$128.02
24	$8,700	$403.47
36	$10,000	$325.02
	$27,200	$1,165.03

d.

Months	Amount	Payment
12	$3,500	$308.52
24	$8,700	$403.47
36	$10,000	$325.02
48	$5,000	$128.02
		$1,165.03

9. Choose the statement below that selects rows in which ?

- the amount is less than or equal to $5000
- the account is 101-1092 or the rate equals 0.095.

a.
```
where amount <= 5000 and
      account='101-1092' or rate = 0.095;
```

b.
```
where (amount le 5000 and account='101-1092')
      or rate = 0.095;
```

c.
```
where amount <= 5000 and
      (account='101-1092' or rate eq 0.095);
```

d.
```
where amount <= 5000 or account='101-1092'
      and rate = 0.095;
```

10. What does PROC PRINT display by default? ?

a. PROC PRINT does not create a default report; you must specify the rows and columns to be displayed.

b. PROC PRINT displays all observations and variables in the data set. If you want an additional column for observation numbers, you can request it.

c. PROC PRINT displays columns in the following order: a column for observation numbers, all character variables, and all numeric variables.

d. PROC PRINT displays all observations and variables in the data set, a column for observation numbers on the far left, and variables in the order in which they occur in the data set.

Answers

1. Correct answer: c

   The DATA= option specifies the data set that you are listing, and the ID statement replaces the Obs column with the specified variable. The VAR statement specifies variables and controls the order in which they appear, and the WHERE statement selects rows based on a condition. The LABEL option in the PROC PRINT statement causes the labels specified in the LABEL statement to be displayed.

2. Correct answer: a

   You use the DATA= option to specify the data set to be printed. The LABEL option specifies that variable labels appear in output instead of variable names.

3. Correct answer: d

   In the WHERE statement, the IN operator enables you to select observations based on several values. You specify values in parentheses and separated by spaces or commas. Character values must be enclosed in quotation marks and must be in the same case as in the data set.

4. Correct answer: c

   In a PROC SORT step, you specify the DATA= option to specify the data set to sort. The OUT= option specifies an output data set. The required BY statement specifies the variable(s) to use in sorting the data.

5. Correct answer: b

   The DATE and NONUMBER system options cause the output to appear with the date but without page numbers. In the PROC PRINT step, the DOUBLE option specifies double spacing, and the NOOBS option removes the default Obs column.

6. Correct answer: d

   You do not need to name the variables in a VAR statement if you specify them in the SUM statement, but you can. If you choose not to name the variables in the VAR statement as well, then the SUM statement determines their order in the output.

7. Correct answer: c

   The BY statement is required in PROC SORT. Without it, the PROC SORT step fails. However, the PROC PRINT step prints the original data set as requested.

8. Correct answer: a

   Column totals appear at the end of the report in the same format as the values of the variables, so b is incorrect. Work.Loans is sorted by Month and Amount, so c is incorrect. The program sums both Amount and Payment, so d is incorrect.

9. Correct answer: c

   To ensure that the compound expression is evaluated correctly, you can use parentheses to group

   ```
   account='101-1092' or rate eq 0.095
   ```

OBS	Account.	Amount.	Rate	Months	Payment.

1	101-1092	$22,000	10.00%	60	$467.43
2	101-1731	$114,000	9.50%	360	$958.57
3	101-1289	$10,000	10.50%	36	$325.02
4	101-3144	$3,500	10.50%	12	$308.52
5	103-1135	$8,700	10.50%	24	$403.47
6	103-1994	$18,500	10.00%	60	$393.07
7	103-2335	$5,000	10.50%	48	$128.02
8	103-3864	$ 87,500	9.50%	360	$735.75
9	103-3891	$ 3 0,000	9.7 5%	360	$257.75

For example, from the data set above, a and b above select observations 2 and 8 (those that have a rate of 0.095); c selects no observations; and d selects observations 4 and 7 (those that have an amount less than or equal to 5000).

10. Correct answer: d

You can remove the column for observation numbers. You can also specify the variables you want, and you can select observations according to conditions.