



临床研究SAS高级编程 ——Report



Contents

- Data NULL
- Proc PRINT
- Proc REPORT



Data _NULL_

- Introduction
- Understanding the PUT statement
- Writing output without creating a data set
- Writing simple text
- Writing a report





Introduction

Objective

- ▶ In this section, you will learn how to do the following:
 - ┌ design output by positioning data values and character strings in an output file
 - ┌ prevent SAS from creating a data set by using the DATA _NULL_ statement
 - ┌ produce reports by using the DATA step instead of using a procedure
 - ┌ direct data to an output file by using a FILE statement





Understanding the PUT Statement

When you create output using the DATA step, you can customize that output by using the PUT statement to write text to the SAS log or to another output file. The PUT statement has the following form:



```
PUT<variable<format>><'character-string'>;
```

where

- ▶ **variable** names the variable that you want to write.
- ▶ **Format** specifies a format to use when you write variable values.
- ▶ **'character-string'** specifies a string of text to write. Be sure to enclose the string in quotation marks.



Writing Output without Creating a Data Set (1)

-  In many cases, when you use a DATA step to write a report, you do not need to create an additional data set. When you use the DATA _NULL_ statement, SAS processes the DATA step without writing observations to a data set. **Using the DATA _NULL_ statement can increase program efficiency considerably.**
-  The following is an example of a DATA _NULL_ statement:

```
data _null_;
```

Writing Output without Creating a Data Set (2)

The following program uses a PUT statement to write newspaper circulation values to the SAS log. Because the program uses a DATA _NULL_ statement, SAS does not create a data set.

```
data _null_;  
  length state $ 15;  
  input state $ morning_copies evening_copies year;  
  put state morning_copies evening_copies year;  
  datalines;  
Massachusetts 798.4 984.7 1999  
Massachusetts 834.2 793.6 1998  
Massachusetts 750.3 . 1997  
Alabama . 698.4 1999  
Alabama 463.8 522.0 1998  
Alabama 583.2 234.9 1997  
Alabama . 339.6 1996  
;
```

Writing Output without Creating a Data Set (3)

- The following output shows the results:

```
184 data _null_;  
185     length state $ 15;  
186     input state $ morning_copies evening_copies year;  
187     put state morning_copies evening_copies year;  
188     datalines;  
Massachusetts 798.4 984.7 1999  
Massachusetts 834.2 793.6 1998  
Massachusetts 750.3 . 1997  
Alabama . 698.4 1999  
Alabama 463.8 522 1998  
Alabama 583.2 234.9 1997  
Alabama . 339.6 1996  
  
196 ;
```

- ▶ SAS indicates missing numeric values with a period. Note that the log contains three missing values.





Writing Simple Text (1)

Writing a character string (1)

- ▶ In its simplest form, the PUT statement writes the character string that you specify to the SAS log, to a procedure output file, or to an external file.
- ▶ If you omit the destination (as in this example), then SAS writes the string to the log.
- ▶ In the following example, SAS executes the PUT statement once during each iteration of the DATA step. When SAS encounters missing values for MORNING_VALUES or EVENING_COPIES, the PUT statement writes a message to the log.

```
data _null_;  
  length state $ 15;  
  infile 'your-input-file';  
  input state $ morning_copies evening_copies year;  
  if morning_copies=. then put '** Morning Circulation  
                               Figures Missing';  
  else if evening_copies=. then put '** Evening Circulation  
                                    Figures Missing';  
  
run;
```



Writing Simple Text (2)

Writing a character string (2)

- ▶ The following output shows the results:

```
93  data _null_ ;
94      length state $ 15;
95      infile 'your-input-file';
96      input state $ morning_copies evening_copies year;
97      if morning_copies =. then put '** Morning Circulation Figures Missing';
98      else
99      if evening_copies =. then put '** Evening Circulation Figures Missing';
100 run;
NOTE: The infile 'your-input-file' is:
      File Name=file-name,
      Owner Name=xxxxxx,Group Name=xxxx,
      Access Permission=rw-r--r--,
      File Size (bytes)=223

** Evening Circulation Figures Missing
** Morning Circulation Figures Missing
** Morning Circulation Figures Missing
NOTE: 7 records were read from the infile 'your-input-file'.
      The minimum record length was 30.
      The maximum record length was 31.
```



Writing Simple Text (3)

Writing variable values (1)

► To identify which observations have the missing values, write the value of one or more variables along with the character string. The following program writes the value of YEAR and STATE, as well as the character string:

```
data _null_;  
  length state $ 15;  
  infile 'your-input-file';  
  input state $ morning_copies evening_copies year;  
  if morning_copies =. then put  
    '** Morning Circulation Figures Missing: ' year state;  
  else if evening_copies =. then put  
    '** Evening Circulation Figures Missing: ' year state;  
run;
```

► Notice that the last character in each of the strings is blank. This is an example of list output. In list output, SAS automatically moves one column to the right after writing a variable value, but not after writing a character string. The simplest way to include the required space is to include it in the character string.



Writing Simple Text (4)

Writing variable values (2)

- ▶ The following output shows the results:

```
164 data _null_;
165     length state $ 15;
166     infile 'your-input-file';
167     input state $ morning_copies evening_copies year;
168     if morning_copies =. then put
169         '** Morning Circulation Figures Missing: ' year state;
170     else
171         if evening_copies =. then put
172             '** Evening Circulation Figures Missing: ' year state;
173     run;
NOTE: The infile 'your-file-name' is:
      File Name=file-name,
      Owner Name=xxxxxx,Group Name=xxxx,
      Access Permission=rw-r--r--,
      File Size (bytes)=223

** Evening Circulation Figures Missing: 1997 Massachusetts
** Morning Circulation Figures Missing: 1999 Alabama
** Morning Circulation Figures Missing: 1996 Alabama
NOTE: 7 records were read from the infile 'your-input-file'.
      The minimum record length was 30.
      The maximum record length was 31.
```



Writing Simple Text (5)

Writing on the same line more than once (1)

- ▶ By default, each PUT statement begins on a new line. However, you can write on the same line if you use more than one PUT statement and at least one trailing @ (“at” sign).
- ▶ The trailing @ is a type of **pointer control** called a line-hold specifier. Pointer controls are one way to specify where SAS writes text.
- ▶ In the following example, using the trailing @ causes SAS to write the item in the second PUT statement on the same line rather than on a new line. The execution of either PUT statement holds the output line for further writing because each PUT statement has a trailing @.

```
options linesize=80 pagesize=60;
data _null_;
  length state $ 15;
  infile 'your-input-file';
  input state $ morning_copies evening_copies year;
  if morning_copies =. then put
    '** Morning Tot Missing: ' year state @;
  if evening_copies =. then put
    '** Evening Tot Missing: ' year state @;
run;
```



Writing Simple Text (6)

- Writing on the same line more than once (2)
 - ▶ The following output shows the results:

```
157 options linesize=80 pagesize=60;
158
159 data _null_;
160     length state $ 15;
161     infile 'your-input-file';
162     input state $ morning_copies evening_copies year;
163     if morning_copies =. then put
164         '** Morning Tot Missing: ' year state @;
165     if evening_copies =. then put
166         '** Evening Tot Missing: ' year state @;
167     run;
NOTE: The infile 'your-input-file' is:
      File Name=file-name,
      Owner Name=xxxxxx,Group Name=xxxx,
      Access Permission=rw-r--r--,
      File Size (bytes)=223

** Evening Tot Missing: 1997 Massachusetts ** Morning Tot Missing: 1999 Alabama
** Morning Tot Missing: 1996 Alabama
NOTE: 7 records were read from the infile 'your-input-file'.
      The minimum record length was 30.
      The maximum record length was 31.
```

- ┌ If the output line were long enough, then SAS would write all three messages about missing data on a single line.
- ┌ When it determines that an individual data value or character string does not fit on a line, SAS brings the entire item down to the next line. SAS does not split a data value or character string.



Writing Simple Text (7)

Releasing a held line (1)

- ▶ SAS determines where to write the output by the presence of the trailing @ sign in the PUT statement and the presence of a null PUT statement that releases the hold on the line.
- ▶ In the following example, the input file has five missing values. One record has missing values for both the MORNING_COPIES and EVENING_COPIES variables. Three other records have missing values for either the MORNING_COPIES or the EVENING_COPIES variable.
- ▶ To improve the appearance of your report, you can write all the missing variables for each observation on a separate line.

```
options linesize=80 pagesize=60;
data _null_;
  length state $ 15;
  infile 'your-input-file';
  input state $ morning_copies evening_copies year;
  if morning_copies=. then put
    '** Morning Tot Missing: ' year state @;
  if evening_copies=. then put
    '** Evening Tot Missing: ' year state;
  else if morning_copies=. then put;
run;
```



Writing Simple Text (8)



Releasing a held line (2)

- ▶ The following output shows the results:

```
7 data _null_;
8   length state $ 15;
9   infile 'your-input-file';
10  input state $ morning_copies evening_copies year;
11  if morning_copies=. then put
12    '** Morning Tot Missing: ' year state @;
13  if evening_copies=. then put
14    '** Evening Tot Missing: ' year state;
15  else if morning_copies=. then put;
16  run;
NOTE: The infile 'your-input-file' is:
      File Name=your-input-file,
      Owner Name=xxxxxx,Group Name=xxxx,
      Access Permission=rw-r--r--,
      File Size (bytes)=223

** Evening Tot Missing: 1997 Massachusetts
** Morning Tot Missing: 1999 Alabama
** Morning Tot Missing: 1998 Alabama ** Evening Tot Missing: 1998 Alabama
** Morning Tot Missing: 1996 Alabama
NOTE: 7 records were read from the infile 'your-input-file'.
      The minimum record length was 30.
      The maximum record length was 31.
```



Writing a Report (1)

Writing to an output file

- ▶ The simplest destination for a printed report is the SAS output file, which is the same place SAS writes output from procedures. SAS automatically defines various characteristics such as page numbers for the procedure output file, and you can take advantage of them instead of defining all the characteristics yourself.
- ▶ To route lines to the procedure output file, use the FILE statement. The FILE statement has the following form:

FILE PRINT *<options>*;

- ▶ PRINT is a reserved fileref that directs output that is produced by PUT statements to the same print file as the output that is produced by SAS procedures.

Writing a Report (2)

Designing the report

- ▶ You create the design and determine which lines and columns the text will occupy. Planning how you want your final report to look helps you write the necessary PUT statements to produce the report.
- ▶ The rest of the examples in this section show how to modify a program to produce a final report that resembles the one shown right.

```

-----1-----2-----3-----4-----5-----6-----7--
1           Morning and Evening Newspaper Circulation
2
3   State           Year           Thousands of Copies
4                                 Morning      Evening
5
6   Alabama         1984           256.3      480.5
7                                 1985           291.5      454.3
8                                 1986           303.6      454.7
9                                 1987              .      454.5
10
11                                 -----
12                                Total for each category  851.4      1844.0
13                                Combined total                2695.4
14
15   Massachusetts  1984              .              .
16                                 1985              .      68.0
17                                 1986           222.7      68.6
18                                 1987           224.1      66.7
19
20                                 -----
21                                Total for each category  446.8      203.3
22                                Combined total                650.1
23
24
25
26
27
28
29
30                                Preliminary Report
-----1-----2-----3-----4-----5-----6-----7--

```



Writing a Report (3)

Writing data values (1)

- ▶ The following program shows how to display the data values for the YEAR, MORNING_COPIES, and EVENING_COPIES variables in specific positions.
- ▶ In a PUT statement, the @*n* argument is a column-pointer control. It tells SAS to move to column *n*.
- ▶ In this example the pointer moves to the specified locations, and the PUT statement writes values at those points using list output.

```
data _null_;  
  infile 'your-input-file';  
  input state $ morning_copies evening_copies year;  
  file print notitles;  
  put @26 year @53 morning_copies @66 evening_copies;  
run;
```

Writing a Report (4)

Writing data values (2)

► The following output shows the results:

1999	798.4	984.7
1998	834.2	793.6
1997	750.3	.
1999	.	698.4
1998	463.8	522
1997	583.2	234.9
1996	.	339.6

► FILE statement *options* specify options that you can use to customize output.

! **NOTITLES**: eliminates the default title line and makes that line available for writing.



Writing a Report (5)

Improving the appearance of numeric data values

- ▶ In the design for your report, all numeric values are aligned on the decimal point. To achieve this result, you have to alter the appearance of the numeric data values by using SAS formats.

```
options pagesize=30 linesize=80 pageno=1 nodate;  
data _null_;  
  infile 'your-input-file';  
  input state $ morning_copies evening_copies year;  
  file print notitles;  
  put @26 year @53 morning_copies 5.1 @66 evening_copies 5.1;  
run;
```

- ▶ The following output shows the results:

1999	798.4	984.7
1998	834.2	793.6
1997	750.3	.
1999	.	698.4
1998	463.8	522.0
1997	583.2	234.9
1996	.	339.6



Writing a Report (6)

- Writing a value at the beginning of each BY group (1)
 - ▶ Performing a task once for a group of observations requires the use of the BY statement for BY-group processing. The BY statement has the following form:
BY *by-variable(s)*<NOTSORTED>;
 - ┌ The **by-variable** names the variable by which the data set is sorted.
 - ┌ The **optional NOTSORTED** option specifies that observations with the same BY value are grouped together but are not necessarily sorted in alphabetical or numerical order.



Writing a Report (7)

Writing a value at the beginning of each BY group (2)

- ▶ The following program creates a permanent SAS data set named NEWS.CIRCULATION, and writes the name of the state on the first line of the report for each BY group.

```
options pagesize=30 linesize=80 pageno=1 nodate;
libname news 'SAS-data-library';
data news.circulation;
  length state $ 15;
  input state $ morning_copies evening_copies year;
  datalines;
Massachusetts 798.4 984.7 1999
Massachusetts 834.2 793.6 1998
Massachusetts 750.3 . 1997
Alabama . 698.4 1999
Alabama 463.8 522.0 1998
Alabama 583.2 234.9 1997
Alabama . 339.6 1996
;
```



Writing a Report (8)

Writing a value at the beginning of each BY group (3)

```
data _null_;  
  set news.circulation;  
  by state notsorted;  
  file print notitles;  
  if first.state then put / @7 state @;  
  put @26 year @53 morning_copies 5.1 @66 evening_copies 5.1;  
run;
```

- ▶ During the first observation for a given state, a PUT statement writes the name of the state and holds the line for further writing (the year and circulation figures).
- ▶ The next PUT statement writes the year and circulation figures and releases the held line. In observations after the first, only the second PUT statement is processed. It writes the year and circulation figures and releases the line as usual.

Writing a Report (9)

Writing a value at the beginning of each BY group (4)

- ▶ The first PUT statement contains a **slash (/)**, a pointer control that moves the pointer to the beginning of the next line.
- ▶ In this example, the PUT statement prepares to write on a new line (the default action). Then the slash moves the pointer to the beginning of the next line.
- ▶ The following output shows the results:

Massachusetts	1999	798.4	984.7
	1998	834.2	793.6
	1997	750.3	.
Alabama	1999	.	698.4
	1998	463.8	522.0
	1997	583.2	234.9
	1996	.	339.6



Writing a Report (10)

Calculating totals (1)

- ▶ Sum statements accumulate the totals, and assignment statements start the accumulation at 0 for each state.
- ▶ When the last observation for a given state is being processed, an assignment statement calculates the overall total, and a PUT statement writes the totals and additional descriptive text.



Writing a Report (11)

Calculating totals (2)

```
options pagesize=30 linesize=80 pageno=1 nodate;
libname news 'SAS-data-library';

data _null_;
  set news.circulation;
  by state notsorted;
  file print notitles;
  /* Set values of accumulator variables to 0 */
  /* at beginning of each BY group. */
  if first.state then
    do;
      morning_total=0;
      evening_total=0;
      put / @7 state @;
    end;
  put @26 year @53 morning_copies 5.1 @66 evening_copies 5.1;
```



Writing a Report (12)

Calculating totals (3)

```
        /* Accumulate separate totals for morning and */  
        /* evening circulations. */  
morning_total+morning_copies;  
evening_total+evening_copies;  
  
        /* Calculate total circulation at the end of */  
        /* each BY group. */  
if last.state then  
do;  
    all_totals=morning_total+evening_total;  
    put @52 '-----' @65 '-----' /  
        @26 'Total for each category'  
        @52 morning_total 6.1 @65 evening_total 6.1 /  
        @35 'Combined total' @59 all_totals 6.1;  
end;  
  
run;
```



Writing a Report (13)

Calculating totals (4)

► The following output shows the results:

Massachusetts	1999	798.4	984.7
	1998	834.2	793.6
	1997	750.3	.
		-----	-----
	Total for each category	2382.9	1778.3
	Combined total		4161.2
Alabama	1999	.	698.4
	1998	463.8	522.0
	1997	583.2	234.9
	1996	.	339.6
		-----	-----
	Total for each category	1047.0	1794.9
	Combined total		2841.9

- Notice that Sum statements ignore missing values when they accumulate totals.
- By default, Sum statements assign the accumulator variables (in this case, MORNING_TOTAL and EVENING_TOTAL) an initial value of 0. Therefore, although the assignment statements in the DO group are executed for the first observation for both states, you need them only for the second state.



Writing a Report (14)

- Writing headings and footnotes for a one-page report (1)
 - ▶ Because this is a simple, one-page report, you can write the heading with a PUT statement that is executed only during the first iteration of the DATA step. The automatic variable `_N_` counts the number of times the DATA step has iterated or looped, and the PUT statement is executed when the value of `_N_` is 1.
 - ▶ The **FOOTNOTES** option on the FILE statement and the **FOOTNOTE** statement create the footnote.



Writing a Report (15)

Writing headings and footnotes for a one-page report (2)

```
options pagesize=30 linesize=80 pageno=1 nodate;
libname news 'SAS-data-library';

data _null_;
  set news.circulation;
  by state notsorted;
  file print notitles footnotes;
  if _n_=1 then put @16 'Morning and Evening Newspaper Circulation' //
                  @7  'State' @26 'Year' @51 'Thousands of Copies' /
                  @51 'Morning          Evening';
  if first.state then
    do;
      morning_total=0;
      evening_total=0;
      put / @7 state @;
    end;
  put @26 year @53 morning_copies 5.1 @66 evening_copies 5.1;
```



Writing a Report (16)

Writing headings and footnotes for a one-page report (3)

```
morning_total+morning_copies;
evening_total+evening_copies;

if last.state then
  do;
    all_totals=morning_total+evening_total;
    put @52 '-----' @65 '-----' /
        @26 'Total for each category'
        @52 morning_total 6.1 @65 evening_total 6.1 /
        @35 'Combined total' @59 all_totals 6.1;
  end;
  footnote 'Preliminary Report';
run;
```




Writing a Report (17)

Writing headings and footnotes for a one-page report (4)

► The following output shows the results:

Morning and Evening Newspaper Circulation			
State	Year	Thousands of Copies	
		Morning	Evening
Massachusetts	1999	798.4	984.7
	1998	834.2	793.6
	1997	750.3	.
		-----	-----
	Total for each category	2382.9	1778.3
	Combined total	4161.2	
Alabama	1999	.	698.4
	1998	463.8	522.0
	1997	583.2	234.9
	1996	.	339.6
		-----	-----
	Total for each category	1047.0	1794.9
	Combined total	2841.9	

Preliminary Report





Proc PRINT

Contents

- ▶ Overview
- ▶ Creating a basic report
- ▶ Selecting variables
- ▶ Identifying observations
- ▶ Sorting data
- ▶ Generating column totals
- ▶ Double-Spacing listing output
- ▶ Specifying titles and footnotes
- ▶ Assigning descriptive labels
- ▶ Formatting data values



Overview (1)

Types of reports

► Basic report

You can easily list the contents of a SAS data set by using a simple program like the one shown below.

```
libname clinic 'your-SAS-data-library';  
proc print data=clinic.admit;  
run;
```

Obs	ID	Name	Sex	Age	Date	Height	Weight	ActLevel	Fee
1	2458	Murray, W	M	27	1	72	168	HIGH	85.20
2	2462	Almers, C	F	34	3	66	152	HIGH	124.80
3	2501	Bonaventure, T	F	31	17	61	123	LOW	149.75
4	2523	Johnson, R	F	43	31	63	137	MOD	149.75
5	2539	LaMance, K	M	51	4	71	158	LOW	124.80
6	2544	Jones, M	M	29	6	76	193	HIGH	124.80
7	2552	Reberson, P	F	32	9	67	151	MOD	149.75
8	2555	King, E	M	35	13	70	173	MOD	149.75
9	2563	Pitts, D	M	34	22	73	154	LOW	124.80
10	2568	Eberhardt, S	F	49	27	64	172	LOW	124.80
11	2571	Nunnelly, A	F	44	19	66	140	HIGH	149.75
12	2572	Oberon, M	F	28	17	62	118	LOW	85.20
13	2574	Peterson, V	M	30	6	69	147	MOD	149.75
14	2575	Guigley, M	F	40	8	69	163	HIGH	124.80
15	2578	Cameron, L	M	47	5	72	173	MOD	124.80
16	2579	Underwood, K	M	60	22	71	191	LOW	149.75
17	2584	Takahashi, Y	F	43	29	65	123	MOD	124.80
18	2586	Derber, B	M	25	23	75	188	HIGH	85.20
19	2588	Ivan, H	F	22	20	63	139	LOW	85.20
20	2589	Wilcox, E	F	41	16	67	141	HIGH	149.75
21	2595	Warren, C	M	54	7	71	183	MOD	149.75

Overview (2)

► Column totals

You can produce column totals for numeric variables within your report.

```
libname admit 'your-SAS-data-library';  
proc print data=clinic.admit;  
    sum fee;  
run;
```

Obs	ID	Name	Sex	Age	Date	Height	Weight	ActLevel	Fee
1	2458	Murray, W	M	27	1	72	168	HIGH	85.20
2	2462	Almers, C	F	34	3	66	152	HIGH	124.80
3	2501	Bonaventure, T	F	31	17	61	123	LOW	149.75
4	2523	Johnson, R	F	43	31	63	137	MOD	149.75
5	2539	LaMance, K	M	51	4	71	158	LOW	124.80
6	2544	Jones, M	M	29	6	76	193	HIGH	124.80
7	2552	Reberson, P	F	32	9	67	151	MOD	149.75
8	2555	King, E	M	35	13	70	173	MOD	149.75
9	2563	Pitts, D	M	34	22	73	154	LOW	124.80
10	2568	Eberhardt, S	F	49	27	64	172	LOW	124.80
11	2571	Nunnelly, A	F	44	19	66	140	HIGH	149.75
12	2572	Oberon, M	F	28	17	62	118	LOW	85.20
13	2574	Peterson, V	M	30	6	69	147	MOD	149.75
14	2575	Quigley, M	F	40	8	69	163	HIGH	124.80
15	2578	Cameron, L	M	47	5	72	173	MOD	124.80
16	2579	Underwood, K	M	60	22	71	191	LOW	149.75
17	2584	Takahashi, Y	F	43	29	65	123	MOD	124.80
18	2586	Derber, B	M	25	23	75	188	HIGH	85.20
19	2588	Ivan, H	F	22	20	63	139	LOW	85.20
20	2589	Wilcox, E	F	41	16	67	141	HIGH	149.75
21	2595	Warren, C	M	54	7	71	183	MOD	149.75
									2686.95



Overview (3)

► Sorting and labels

You can sort data by the value of one or more variables and replace variable names with descriptive labels.

```
libname clinic 'your-SAS-data-library';  
proc sort data=clinic.admit out=admit;  
    by age;  
run;  
proc print data=admit label;  
    var age height weight fee;  
    label fee='Admission Fee';  
run;
```

Obs	Age	Height	Weight	Admission Fee
1	22	63	139	85.20
2	25	75	188	85.20
3	27	72	168	85.20
4	28	62	118	85.20
5	29	76	193	124.80
6	30	69	147	149.75
7	31	61	123	149.75
8	32	67	151	149.75
9	34	66	152	124.80
10	34	73	154	124.80
11	35	70	173	149.75
12	40	69	163	124.80
13	41	67	141	149.75
14	43	63	137	149.75
15	43	65	123	124.80
16	44	66	140	149.75
17	47	72	173	124.80
18	49	64	172	124.80
19	51	71	158	124.80
20	54	71	183	149.75
21	60	71	191	149.75



Overview (4)

► Selected observations and variables

You can choose the observations and variables that appear in your report and remove the default **Obs** column that display observation numbers.

```
libname clinic 'your-SAS-data-library';  
proc print data=clinic.admit noobs;  
    var age height weight fee;  
    where age>30;  
run;
```

Age	Height	Weight	Fee
34	66	152	124.80
31	61	123	149.75
43	63	137	149.75
51	71	158	124.80
32	67	151	149.75
35	70	173	149.75
34	73	154	124.80
49	64	172	124.80
44	66	140	149.75
40	69	163	124.80
47	72	173	124.80
60	71	191	149.75
43	65	123	124.80
41	67	141	149.75
54	71	183	149.75





Creating a Basic Report (1)

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 following program, 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;
```



Creating a Basic Report (2)

Notice the layout of the resulting report. By default,

- ┌ All observations and variables in the data set are printed
- ┌ A column for observation numbers appears on the far left
- ┌ Variables appear in the order in which they occur in the data set

Obs	Date	AerClass	WalkJogRun	Swim
1	JAN1999	56	78	14
2	FEB1999	32	109	19
3	MAR1999	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	78	77	54
10	OCT1999	81	62	47
11	NOV1999	84	31	52
12	DEC1999	2	44	55
13	JAN2000	37	91	83
14	FEB2000	41	102	27
15	MAR2000	52	98	19
16	APR2000	61	118	22
17	MAY2000	49	88	29
18	JUN2000	24	101	54
19	JUL2000	45	91	69
20	AUG2000	63	65	53
21	SEP2000	60	49	68
22	OCT2000	78	70	41
23	NOV2000	82	44	58
24	DEC2000	93	57	47





Selecting Variables (1)

Selecting variables

- ▶ 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.



Selecting Variables (2)

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**.

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	124.80
6	29	76	193	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
12	28	62	118	85.20
13	30	69	147	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
18	25	75	188	85.20
19	22	63	139	85.20
20	41	67	141	149.75
21	54	71	183	149.75



Selecting Variables (3)

Removing the OBS column

- ▶ 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.
- ▶ 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;
```

Age	Height	Weight	Fee
27	72	168	85.20
34	66	152	124.80
31	61	123	149.75
43	63	137	149.75
51	71	158	124.80





Identifying Observations (1)

● ID statement

- ▶ You can use one or more variables to replace the Obs column in the output.
- ▶ 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 the observation number at the beginning of each row of the report.



Identifying Observations (2)

► Example

```
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	HomePhone
1269	CASTON	FRANKLIN	STAMFORD	CT	M	NA1	41690.00	06MAY60	01DEC80	203/781-3335
1935	FERNANDEZ	KATRINA	BRIDGEPORT	CT		NA2	51081.00	31MAR42	19OCT69	203/675-2962
1417	NEWKIRK	WILLIAM	PATERSON	NJ	,	NA2	52270.00	30JUN52	10MAR77	201/732-6611
1839	NORRIS	DIANE	NEW YORK	YN	F	NA1	43433.00	02DEC58	06JUL81	718/384-1767
1111	RHODES	JEREMY	PRINCETON	NJ	M	NA1	40586.00	17JUL61	03NOV80	201/812-1837
1352	RIVERS	SIMON	NEW YORK	NY	M	NA2	5379.80	05DEC48	19OCT74	718/383-3345
1332	STEPHENSON	ADAM	BRIDGEPORT	CT	M	NA1	42178.00	20SEP58	07JUN79	203/675-1497
1443	WELLS	AGNES	STAMFORD	CT	F	NA1	422.74	20NOV56	01SEP79	203/781-5546



Identifying Observations (3)

- ▶ 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





Selecting Observations (1)

Where statement

- ▶ 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 can only be one Where statement in a step.
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.



Selecting Observations (2)

► Example

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

Here is the output:

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



Selecting Observations (3)

► 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 lower and uppercase letters exactly as it appears in the data set.



Selecting Observations (4)

- ▶ 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:

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.



Selecting Observations (5)

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=124.8 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');
```



Selecting Observations (6)

Quiz

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');

► 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.



Selecting Observations (7)

Quiz

- ▶ 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 ≤ 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;
- ▶ Correct answer: c
 - ! To ensure that the compound expression is evaluated correctly, you can use parentheses to group.





Sorting Data (1)

- Sort procedure
 - ▶ 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
 - ┆ Rearrange 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



Sorting Data (2)

- 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 than one variable in the BY statement, DESCENDING applies only to the variable that immediately follows it.



Sorting Data (3)

Example 1

```
proc sort data=clinic.admit out=work.wgtadmit;  
        by weight age;
```

```
run;
```

```
proc print data=work.wgtadmit;  
        var age height weight fee;  
        where age>30;
```

```
run;
```

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

Obs	Age	Height	Weight	Fee
2	31	61	123	149.75
3	43	65	123	124.80
4	43	63	137	149.75
6	44	66	140	149.75
7	41	67	141	149.75
9	32	67	151	149.75
10	34	66	152	124.80
11	34	73	154	124.80
12	51	71	158	124.80
13	40	69	163	124.80
15	49	64	172	124.80
16	35	70	173	149.75
17	47	72	173	124.80
18	54	71	183	149.75
20	60	71	191	149.75



Sorting Data (4)

Example 2

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 age height weight fee;  
    where age>30;  
run;
```

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

Sorting Data (5)

Quiz

► 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;
```

► 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.





Generating Column Totals (1)

SUM statement

- ▶ 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 variable names, separated by blanks.

- ▶ You do not need to name the variables in a VAR statement if you specify them in the SUM statement.

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;
```

Obs	Name	Policy	BalanceDue
2	Almers, C	95824	156.05
3	Bonaventure, 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



Generating Column Totals (2)

Quiz

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

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.



Generating Column Totals (3)

Requesting subtotals

- ▶ 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**.



Generating Column Totals (4)

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;
```



Generating Column Totals (5)

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	
ActLevel				549.10	

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	

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	



Generating Column Totals (6)

- 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.
 - ▶ 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.



Generating Column Totals (7)

Example

```
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



Generating Column Totals (8)

Quiz

In the program that produces the output shown below, which set of statements is used?

- a. `sum dest;`
`by boarded deplaned;`
- b. `sum boarded deplaned;`
`by dest;`
- c. `sum boarded deplaned;`
`by dest;`
`id dest;`
- d. `sum boarded deplaned;`
`id by dest;`

Dest	Boarded	Deplaned
LON	167	222
	150	320
LON	317	542
PAR	177	227
	177	203
PAR	354	430
	671	972

Correct answer: c

Specifying the same variable in the BY and ID statements prints the BY value only at the start of each BY group and on the line containing that group's subtotal.



Generating Column Totals (9)

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 any of the variables in the BY statement changes.

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



Generating Column Totals (10)

Example

```
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

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

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





Double Spacing Listing Output

Double option

- ▶ 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;
```

Double spacing does not apply to HTML output.

SAS Output

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



Specifying Titles and Footnotes (1)

- Title and Footnote statement
 - ▶ 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.
 - ▶ Because TITLE and FOOTNOTE statements are global statements, place them before the PRINT procedure. Titles and footnotes are assigned as soon as TITLE or FOOTNOTE statements are read; they apply to all subsequent output.



Specifying Titles and Footnotes (2)

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



Specifying Titles and Footnotes (3)

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.

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



Specifying Titles and Footnotes (4)

Example: Titles

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

```
title1 'Heart Rates for Patients with';  
title3 'Increased Stress Tolerance Levels';  
proc print data=clinic.stress;  
    var resthr maxhr rechr;  
    where tolerance='I';  
run;
```



Specifying Titles and Footnotes (5)

In **HTML output**, title lines simply appear consecutively, without extra spacing to indicate skipped title numbers.

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

In **SAS listing output**, 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



Specifying Titles and Footnotes (6)

Example: Footnotes

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

```
footnote1 '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.



Specifying Titles and Footnotes (7)

- In **HTML output**, footnote lines simply appear consecutively, without extra spacing to indicate skipped footnote numbers.
- 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**

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



Specifying Titles and Footnotes (8)

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.

```
footnote1 '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;
```



Specifying Titles and Footnotes (9)

- ▶ 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;
```



Specifying Titles and Footnotes (10)

- ▶ 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**.

```
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;
```



Specifying Titles and Footnotes (11)

Quiz

Suppose you want to specify the text Final Report on the third footnote line of a report and to cancel any other footnotes that are in effect. Which set of FOOTNOTE statements do you submit?

- a. `footnote;`
`footnote2 'Final Report';`
- b. `footnote3 'Final Report';`
`footnote;`
- c. `footnote1;`
`footnote3 'Final Report';`

Correct answer: c

You specify the FOOTNOTE1 statement to cancel any footnotes in effect, then a FOOTNOTE3 statement for your new footnote. ←



Assigning Descriptive Labels (1)

- 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.

- ▶ The LABEL statement applies only to the PROC step in which it appears.



Assigning Descriptive Labels (2)

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.

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

Obs	Date	AerClass	Walk/Jog/Run	Swim
1	JAN1999	56	78	14
2	FEB1999	32	109	19
3	MAR1999	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	78	77	54
10	OCT1999	81	62	47
11	NOV1999	84	31	52
12	DEC1999	2	44	55
13	JAN2000	37	91	83
14	FEB2000	41	102	27
15	MAR2000	52	98	19
16	APR2000	61	118	22
17	MAY2000	49	88	29
18	JUN2000	24	101	54
19	JUL2000	45	91	69
20	AUG2000	63	65	53
21	SEP2000	60	49	68
22	OCT2000	78	70	41
23	NOV2000	82	44	58
24	DEC2000	93	57	47



Assigning Descriptive Labels (3)

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;
```



Assigning Descriptive Labels (4)



Quiz

Which PROC PRINT step below creates the following output?

- 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;`

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

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.





Formatting Data Values (1)

- 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.

- ▶ The format statement applies only to the PROC step in which it appears.



Formatting Data Values (2)

- ▶ 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.

This FORMAT statement ...	Associates ...	To display values as ...
<code>format date mmddy8.;</code>	the format MMDDYY8. with the variable Date	06/05/03
<code>format net comma5.0 gross comma8.2;</code>	the format COMMA5.0 with the variable Net and the format COMMA8.2 with the variable Gross	1,234 5,678.90
<code>format net gross dollar9.2;</code>	the format DOLLAR9.2 with both variables, Net and Gross	\$1,234.00 \$5,678.90



Formatting Data Values (3)

- ▶ 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



Formatting Data Values (4)

Specifying SAS formats

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

Format	Specifies values ...	Example
COMMA <i>w.d</i>	that contain commas and decimal places	comma8.2
DOLLAR <i>w.d</i>	that contain dollar signs and commas	dollar6.2
MMDDYY <i>w.</i>	as date values of the form 09/12/97 (MMDDYY8.) or 09/12/1997 (MMDDYY10.)	mmddy10.
<i>w.</i>	rounded to the nearest integer in <i>w</i> spaces	7.
<i>w.d</i>	rounded to <i>d</i> decimal places in <i>w</i> spaces	8.2
\$ <i>w.</i>	as character values in <i>w</i> spaces	\$12.
DATE <i>w.</i>	as date values of the form 16OCT99 (DATE7.) or 16OCT1999 (DATE9.)	date9.



Formatting Data Values (5)

Field width

- ▶ 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 a 5400. To specify the COMMAw.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.

- ▶ 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
```



Formatting Data Values (6)

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. 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
```

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
```



Formatting Data Values (7)

Examples

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

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

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 'BEST' format."



Formatting Data Values (8)

- Using permanently assigned labels and formats
 - ▶ When you use a LABEL or FORMAT statement within a PROC PRINT step, the label or format applies only to the output from that step.
 - ▶ 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.



Formatting Data Values (9)

Example

```
data flights.march;  
    set flights.mar01;  
    label date='Departure Date';  
    format date date9.;  
  
run;  
  
proc print data=flights.march label;  
run;
```

(Partial Listing)

Obs	Departure Date	Dest	Boarded
1	01MAR2000	LON	198
2	01MAR2000	PAR	207
3	01MAR2000	LON	205
4	01MAR2000	COP	138
5	01MAR2000	MUN	147

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



Formatting Data Values (10)

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=options**.

```
Proc print data=reps split='*';  
  var salesrep type unitsold net commision;  
  lable 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';  
Proc print data=vcrsales;  
  var salesrep type unitsold;  
  format salesrep $repfmt.;
```

```
Run;
```





Contents

- ▶ [Overview](#)
- ▶ [Creating a default list report](#)
- ▶ [Selecting variables](#)
- ▶ [Selecting observation](#)
- ▶ [Defining variables](#)





Introduction

- ▶ To produce a variety of reports using a single report-writing tool, you can use PROC REPORT.
- ▶ In addition to creating list reports, PROC REPORT enables you to
 - ┆ Create custom reports
 - ┆ Request separate subtotals and grand totals
 - ┆ Calculate columns
 - ┆ Create and store report definitions
- ▶ You can use PROC REPORT in three ways:
 - ┆ in windowing mode with a prompting facility that guides you as you build a report
 - ┆ in windowing mode without the prompting facility
 - ┆ In nonwindowing mode. In this case, you submit a series of statements with the PROC REPORT statement, just as you do in other SAS procedures.



Creating a Default List Report (1)

The syntax

- ▶ General form, basic PROC REPORT step:

```
PROC REPORT <DATA=SAS-data-set><options>;
```

```
Run ;
```

where

SAS-data-set is the name of the SAS data set that is used for the report.

Options include:

Windows or WD, which invokes the procedure in windowing mode.

Your report appears in the interactive REPORT window. This is the default.

NOWINDOWS or NOWD, which invokes the procedure in nonwindowing mode and displays a listing of the report in the Output window.

- ▶ If you specify HTML output in your SAS preference, the HTML output appears in addition to or instead of the OUTPUT window listing, as requested. Some PROC REPORT formatting options are not applicable to HTML output.



Creating a Default List Report (2)

Example

```
Proc report data=flights.europe nowd;  
Run ;
```

This is HTML output from the PROC REPORT step.
Notice that by default

- ▶ All observations and variables in the data set are printed.
- ▶ Variables appear in the order in which they occur in the data set.

Flight	Date	Depart	Orig	Dest	Miles	Mail	Freight	Boarded	Transferred	NonRevenue	Deplaned
821	04MAR99	9:31	LGA	LON	3442	403	209	167	17	7	222
271	04MAR99	11:40	LGA	PAR	3856	492	308	146	8	3	163
271	05MAR99	12:19	LGA	PAR	3857	366	498	177	15	5	227
821	06MAR99	14:56	LGA	LON	3442	345	243	167	13	4	222
821	07MAR99	13:17	LGA	LON	3635	248	307	215	14	6	158
271	07MAR99	9:31	LGA	PAR	3442	353	205	155	18	7	172
821	08MAR99	11:40	LGA	LON	3856	391	395	186	8	1	114
271	08MAR99	12:19	LGA	PAR	3857	366	279	152	7	4	187
821	09MAR99	14:56	LGA	LON	3442	219	368	203	6	3	210
271	09MAR99	13:17	LGA	PAR	3635	357	282	159	15	4	191
821	10MAR99	9:31	LGA	LON	3442	389	479	188	8	6	211
271	10MAR99	11:40	LGA	PAR	3856	415	463	182	9	6	153





Selecting Variables (1)

Column statement

General form, COLUMN statement

`COLUMN variables`

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

Example

```
Proc report data=flights.europe nowd;  
  column flight orig dest mail freight revenue;  
Run;
```

Flight	Orig	Dest	Mail	Freight	Revenue
821	LGA	LON	403	209	150634
271	LGA	PAR	492	308	156804
271	LGA	PAR	366	498	190098
821	LGA	LON	345	243	150634
821	LGA	LON	248	307	193930
271	LGA	PAR	353	205	166470
821	LGA	LON	391	395	167772
271	LGA	PAR	366	279	163248
821	LGA	LON	219	368	183106
271	LGA	PAR	357	282	170766
821	LGA	LON	389	479	169576

Selecting Variables (2)

Quiz

Which of the following statements correctly selects and orders variables as shown in the PROC REPORT output below?

Patient	Shock	Survive	Cardiac	Urinary
203	NONSHOCK	SURV	66	110
318	OTHER	DIED	410	405
601	BACTER	DIED	260	377
98	CARDIO	SURV	260	377
4	HYPOVOL	SURV	406	200

- a. `var patient shock survive cardiac urinary;`
- b. `column patient shock survive cardiac urinary;`
- c. `column patient / cardiac urinary shock survive;`

Correct answer: b

In the COLUMN statement, you specify the variables for your report in the order you want them to appear.



Selecting Observations

Where statement

► To select observations, you can use the **Where statement**, just as you have learned to do with PROC PRINT

► Example

The following WHERE statement specifies that only observations that have the value LON or PAR for the variable Dest be printed.

```
Proc report data=flights.europe nowd;  
  column flight orig dest mail  
         freight venue;  
  where dest in ('LON', 'PAR');  
run;
```

Flight	Orig	Dest	Mail	Freight	Revenue
821	LGA	LON	403	209	150634
271	LGA	PAR	492	308	156804
271	LGA	PAR	366	498	190098
821	LGA	LON	345	243	150634
821	LGA	LON	248	307	193930
271	LGA	PAR	353	205	166470
821	LGA	LON	391	395	167772
271	LGA	PAR	366	279	163248
821	LGA	LON	219	368	183106
271	LGA	PAR	357	282	170766
821	LGA	LON	389	479	169576
271	LGA	PAR	415	463	195468
821	LGA	LON	448	282	143561
271	LGA	PAR	352	351	123456
219	LGA	LON	331	376	189065
821	LGA	LON	403	209	170766
271	LGA	PAR	492	308	125632
219	LGA	LON	485	267	197456





Defining Variables

Contents

- ▶ Overview of defining variables
- ▶ Defining column attributes
- ▶ Defining column headings
- ▶ Specifying column justification
- ▶ Enhancing the heading's appearance
- ▶ Defining variable usage





Overview of Defining Variables (1)

Introduction

- ▶ In the output that you've seen in this lesson so far, you might have noticed that PROC REPORT displays
 - ┆ Each data values the way it is stored in the data set
 - ┆ Variable names as column headings in the report
 - ┆ A default width for the report columns
 - ┆ Left-justified character values
 - ┆ Right-justified numeric values
 - ┆ Observations in the order in which they are stored in the data set
- ▶ You can enhance the report by
 - ┆ Defining how each variable is used in the report
 - ┆ Assigning formats to variables
 - ┆ Specifying column headings and widths
 - ┆ Justifying the variable values and column headings within the report columns
 - ┆ Changing the order of the rows in the report



Overview of Defining Variables (2)

Using DEFINE statement

- ▶ You can use one or more **DEFINE statement** to describe how to use and display variables in your report
- ▶ You can list Define statements in any order, or you can list options (usages, attributes, and so on) in any order in a **DEFINE statement**.



Overview of Defining Variables (3)

The syntax:

General form, basic **DEFINE statement**:

```
DEFINE variable / <usage> <attribute(s)>  
<option(s)><justification><'column-heading'>;
```

Where

- ▶ **variable** is the name of the variable that you want to define.
- ▶ **usage** specifies how to use the variable. Valid options are ACROSS, ANALYSIS, COMPUTED, DISPLAY, GROUP, and ORDER
- ▶ **attribute(s)** specifies attributes for the variable, including FORMAT=, WIDTH=, and SPACING=.
- ▶ **Option(s)** specifies formatting options, including DESCENDING, NOPRINT, NOZERO, and PAGE.
- ▶ **Justification** specifies columns justification (CENTER, LEFT, or RIGHT)
- ▶ **'column-heading'** specifies a label for the column heading.



Overview of Defining Variables (4)

Example

These DEFINE statements specify usages, attributes, options, justification, and column heading for the variable Flight and Orig.

```
Proc report data=flights.europe nowd;  
  where dest in ('LON', 'PAR');  
  column flight orig dest mail freight revenue;  
  define flight / order descending 'Flight Number'  
                center width=6 spacing=5;  
  define orig / 'Flight Origin' center width=6;  
run;
```





Defining Column Attributes (1)

Introduction

- ▶ You can easily change the appearance of your PROC REPORT output by specifying attributes for variables. For example, you can select a format for data values, specify the column width, and specify the spacing between columns.
- ▶ To enhance your PROC REPORT output, you'll use the following attributes.

Attribute	Action
FORMAT= <i>format</i>	Assigns a SAS format or a user-defined format to the item.
SPACING= <i>horizontal-positions</i>	Specifies how many blank characters to leave between the selected column and the column immediately to its left. The default is 2.
WIDTH= <i>column-width</i>	Specifies the width of the column. The default column width is just large enough to handle the specified format.



Defining Column Attributes (2)

Assigning formats

- ▶ Formats determine how data values appear in SAS output.
- ▶ If you do not specify a format for a variable within the PROC REPORT step, PROC REPORT uses the format that is stored in the data set. If no format is stored in the data set, PROC REPORT uses the default format for that variable type.
- ▶ To assign a format to a specific report column, use the **FORMAT=attribute** in the **DEFINE statement** for that column.
- ▶ You can specify any appropriate SAS format or user-defined format.



Defining Column Attributes (3)

Example

- ▶ The variable Revenue has no format assigned to it in the Flights.Europe data set. So, in your current report, revenue values appear as shown in the example below.

Partial PROC REPORT Output, HTML

Flight	Orig	Dest	Mail	Freight	Revenue
821	LGA	LON	403	209	150634
271	LGA	PAR	492	308	156804
271	LGA	PAR	366	498	190098
821	LGA	LON	345	243	150634
821	LGA	LON	248	307	193930

But suppose you want your revenue figures to be formatted with a dollar sign, commas, and two decimal places, in a total width of 15 positions. To do this, you can assign the DOLLAR15.2 format to Revenue.



Defining Column Attributes (4)

This is part of the HTML output from the program above. Notice that the format supplies the dollar sign, comma, decimal point, and decimal places.

However, because the HTML table column conforms to the width of its contents, assigning the format does not increase the column width beyond the length of the data values.

Partial PROC REPORT Output, HTML

Flight	Orig	Dest	Mail	Freight	Revenue
821	LGA	LON	403	209	\$150,634.00
271	LGA	PAR	492	308	\$156,804.00
271	LGA	PAR	366	498	\$190,098.00
821	LGA	LON	345	243	\$150,634.00
821	LGA	LON	248	307	\$193,930.00

By contrast, the monospace SAS listing of the report does display the increased column width.

Partial PROC REPORT Output, SAS Listing

Flt	Orig	Dest	Mail	Freight	Revenue
821	LGA	LON	403	209	\$150,634.00
271	LGA	PAR	492	308	\$156,804.00
271	LGA	PAR	366	498	\$190,098.00
821	LGA	LON	345	243	\$150,634.00
821	LGA	LON	248	307	\$193,930.00

You can also use FORMAT statements with PROC REPORT. However, the DEFINE statements enables you to specify more than one column attribute at a time. Also, you can use the FORMAT= attribute to define report columns (such as statistics or computed variables) that are not data set variables.



Defining Column Attributes (5)

Quiz

Which of the following statements correctly assigns the format COMMA7.3 to Cardiac?

- a. `define cardiac format=comma7.3;`
- b. `format comma7.3 cardiac;`
- c. `define cardiac / format=comma7.3;`

Correct answer: c

In the DEFINE statement, you specify the variable to be defined, followed by a slash and the FORMAT= attribute.



Defining Column Attributes (6)

Specifying column widths

- ▶ If a variable in the input data set does not have a format associated with it, the default PROC REPORT column width is
 - ┌ The variable's length for character variables
 - ┌ 9 for numeric variables
- ▶ The character variable Flight, Orig, and Dest each have a length of 3, and no format is associated with them. So 3 is their default column width.

Partial PROC REPORT Output, SAS Listing

Fli ght	Ori g	Des t	Mail	Freight	Revenue
821	LGA	LON	403	209	\$150,634.00
271	LGA	PAR	492	308	\$156,804.00
271	LGA	PAR	366	498	\$190,098.00
821	LGA	LON	345	243	\$150,634.00
821	LGA	LON	248	307	\$193,930.00

- ▶ To specify a width for columns in your report, use the WIDTH=attribute in the DEFINE statement. You can specify values from 1 to the value of the **LINESIZE=** system option.
- ▶ The WIDTH= attributes has no effect on HTML output.





Defining Column Attributes (7)

Example

To specify column widths that accommodate the column headings for Flight, Orig, and Dest, you can use the following DEFINE statements in your PROC PRINT step.

```
Proc report data =flights.europe nowd;  
  where dest in ('LON' , 'PAR');  
  column flight orig dest mail freight revenue;  
  define flight / width=6;  
  define orig / width=4;  
  define dest / width=6;
```

Run ;

Now the headings appear on one line.

Partial PROC REPORT Output, SAS Listing

Flight	Orig	Dest	Mail	Freight	Revenue
821	LGA	LON	403	209	\$150,634.00
271	LGA	PAR	492	308	\$156,804.00
271	LGA	PAR	366	498	\$190,098.00
821	LGA	LON	345	243	\$150,634.00
821	LGA	LON	248	307	\$193,930.00



Defining Column Attributes (8)

Specifying column spacing

- ▶ **Column space** is the number of blank characters between the selected column and the column immediately to its left.
- ▶ The default column spacing is 2.
- ▶ To specify a different column spacing, use the **SPACING=** attribute in the **DEFINE statement**.
- ▶ The **SPACING=** attribute has no effect on HTML output



Defining Column Attributes (9)

Example

This is PROC REPORT output without any spacing defined.

Orig and **Dest** have 2 blank characters preceding their columns.

Partial PROC REPORT Output, SAS Listing

Flight	Orig	Dest	Mail	Freight	Revenue
821	LGA	LON	403	209	\$150,634.00
271	LGA	PAR	492	308	\$156,804.00
271	LGA	PAR	366	498	\$190,098.00
821	LGA	LON	345	243	\$150,634.00
821	LGA	LON	248	307	\$193,930.00

To specify 5 blank spaces before the column headings for **Orig** and **Dest**, you can use **DEFINE statements** as shown below in your PROC REPORT step.



Defining Column Attributes (10)

```
Proc report data=flights.europe nowd;  
  where dest in ('LON' , 'PAR') ;  
  column flight orig dest mail freight revenue ;  
  define revenue / format=dollar15.2 ;  
  define flight / width=6 ;  
  define orig / width=4 spacing=5 ;  
  define dest / width=4 spacing=5 ;
```

Run ;

Now the two columns display the extra spacing

Partial PROC REPORT Output, SAS Listing

Flight	Orig	Dest	Mail	Freight	Revenue
821	LGA	LON	403	209	\$150,634.00
271	LGA	PAR	492	308	\$156,804.00
271	LGA	PAR	366	498	\$190,098.00
821	LGA	LON	345	243	\$150,634.00
821	LGA	LON	248	307	\$193,930.00



Defining Column Attributes (11)

Quiz

Suppose you produce the PROC REPORT output shown below, but you want to refine its layout. Selects the statement or statements that would define column widths so that all column headings appear on one line specify a spacing of 7 for the second through fifth columns

- a. `define all / width=8;`
`define shock-urinary / spacing=7;`
- b. `define patient / width=7;`
`define shock / spacing=7;`
`define survive / width=7 spacing=7;`
`define cardiac / spacing=7;`
`define urinary / spacing=7;`
- c. `define patient-survive / width=8;`
`define shock-urinary / spacing=7;`

Pat ien t	Shock	Surv ive	Cardiac	Urinary
203	NONSHOCK	SURV	66	110
318	OTHER	DIED	410	405
601	BACTER	DIED	260	377
98	CARDIO	SURV	260	377
4	HYPOVOL	SURV	406	200

Correct answer: b

In the DEFINE statement, you specify the variable to be defined, followed by a slash and the WIDTH= attribute. In this case, the smallest WIDTH= value would be 7.





Defining Column Headings (1)

Introduction

- ▶ To define a column heading, specify the heading text in quotation marks in the DEFINE statement.
- ▶ Example

```
proc report data=flights.europe nowd;  
  where dest in ('LON' , 'PAR');  
  column flights orig dest mail freight revenue;  
  define revenue / format=dollar15.2;  
  define flight / width=13 'Flight Number';  
  define orig / width=13 spacing=5 'Flight Origin';  
  define dest / width=18 spacing=5 'FlightDestination'  
run;
```

Partial PROC REPORT Output, SAS Listing

Flight Number	Flight Origin	Flight Destination	Mail	Freight	Revenue
821	LGA	LON	403	209	\$150,634.00
271	LGA	PAR	492	308	\$156,804.00
271	LGA	PAR	366	498	\$190,098.00
821	LGA	LON	345	243	\$150,634.00
821	LGA	LON	248	307	\$193,930.00





Defining Column Headings (2)

- Defining the split character
 - ▶ To control how words break in column headings, you can use a split character in the column label.
 - ▶ When PROC REPORT encounters the split character in a column heading, it breaks the heading and continues the heading on the next line. The split character itself does not appear in the heading.
 - ▶ To use a split character, you can do either of the following
 - ┌ Use the default slash(/) as the split character
 - ┌ Define a split character by using the **SPLIT=**option in the PROC REPORT statement.



Defining Column Headings (3)

Example

Suppose you want to break headings so that only one word appears on a line. Using the default slash as the split character, you can submit this PROC REPORT step. Notice that the column width has been reduced.

```
proc report data=flights.europe nowd;  
  where dest in ('LON','PAR');  
  column flight orig dest mail freight revenue;  
  define revenue / format=dollar15.2;  
  define flight / width=6 'Flight/Number';  
  define orig / width=6 spacing=5 'Flight/Orign';  
  define dest / width=11 spacing=5 'Flight/Destination';  
run;
```

or you can submit this program, which uses the split= option and produces exactly the same output:

```
proc report data=flights.europe nowd split='*';  
  where dest in ('LON','PAR');  
  column flight orig dest mail freight revenue;  
  define revenue / format=dollar15.2;  
  define flight / width=6 'Flight*Number';  
  define orig / width=6 spacing=5 'Flight/Orign';  
  define dest / width=11 spacing=5 'Flight/Destination';  
run;
```




Defining Column Headings (4)

Here are both types of output from both programs:

Partial PROC REPORT Output, HTML

Flight Number	Flight Origin	Flight Destination	Mail	Freight	Revenue
821	LGA	LON	403	209	\$150,634.00
271	LGA	PAR	492	308	\$156,804.00
271	LGA	PAR	366	498	\$190,098.00
821	LGA	LON	345	243	\$150,634.00
821	LGA	LON	248	307	\$193,930.00

Partial PROC REPORT Output, SAS Listing

Flight Number	Flight Origin	Flight Destination	Mail	Freight	Revenue
821	LGA	LON	403	209	\$150,634.00
271	LGA	PAR	492	308	\$156,804.00
271	LGA	PAR	366	498	\$190,098.00
821	LGA	LON	345	243	\$150,634.00
821	LGA	LON	248	307	\$193,930.00





Specifying Column Justification (1)

Introduction

- ▶ By default, PROC REPORT left-justifies character variables and right-justifies numeric variables.
- ▶ For each variable that you define, you can specify the justification option **CENTER**, **LEFT**, or **RIGHT** in the **DEFINE** statement.
- ▶ Each option justifies both the formatted values of the report item within the column width and the column headings over the values.



Specifying Column Justification (2)

Example

To center headings and values for Flight, Orig, and Dest, you can specify the **CENTER** option as shown below in your DEFINE statements.

```
proc report data=flights.europe nowd;  
  where dest in ('LON' , 'PAR' );  
  column flight orig dest mail freight revenue;  
  define revenue / format=dollar15.2;  
  define flight / width=6 'Flight/Number' center;  
  define orig / width=6 spacing=5 'Flight/Origin' center;  
  define dest / width=11 spacing=5 'Flight/Destination' center;  
run;
```

Partial PROC REPORT Output, SAS Listing

Flight Number	Flight Origin	Flight Destination	Mail	Freight	Revenue
821	LGA	LON	403	209	\$150,634.00
271	LGA	PAR	492	308	\$156,804.00
271	LGA	PAR	366	498	\$190,098.00
821	LGA	LON	345	243	\$150,634.00
821	LGA	LON	248	307	\$193,930.00

Specifying Column Justification (3)

Quiz

Which of the following programs produces the output shown below?

```
a. proc report data=sasuser.heart nowd split='+';  
   column patient shock survive;  
   where urinary>100; define patient / width=7 'Patient+ID';  
   define survive / width=7 'Patient+Outcome';  
run;
```

```
b. proc report data=sasuser.heart nowd;  
   column patient shock survive;  
   where urinary>100;  
   define patient / width=7 'Patient/ID'; define survive /  
   width=7 'Patient/Outcome';  
run;
```

c. both of the above

Patient ID	Shock	Patient Outcome
203	NONSHOCK	SURV
318	OTHER	DIED
601	BACTER	DIED
98	CARDIO	SURV
4	HYPOVOL	SURV

Correct answer: c

To split column headers over more than one line, you can use the default slash as the split character or specify another split character using the SPLIT= option.



Enhancing the Heading's Appearance (1)

Introduction

- ▶ To complete the job of enhancing headings in your list report, you can take advantage of two useful options in the PROC REPORT statement.
 - ┌ HEADLINE, which underlines all column headings and the space between them.
 - ┌ HEADSKIP, which writes a blank line beneath all column headings or after the underline if the HEADLINE options is used.
- ▶ These options have no effect on HTML output.



Enhancing the Heading's Appearance (2)

Example

```
proc report data=flights.europe nowd headline headskip;  
  where dest in ('LON' , 'PAR');  
  column flights orig dest mail freight revenue;  
  define revenue / format=dollar15.2;  
  define flight / width=6 'Flight Number' center;  
  define orig / width=6 spacing=5 'Flight Origin' center;  
  define dest / width=11 spacing=5 'Flight/Destination' center;  
run;
```

In the SAS listing output, the column headings are underlined and are followed by a blank line.

Partial PROC REPORT Output, SAS Listing

<u>Flight Number</u>	<u>Flight Origin</u>	<u>Flight Destination</u>	<u>Mail</u>	<u>Freight</u>	<u>Revenue</u>
821	LGA	LON	403	209	\$150,634.00
271	LGA	PAR	492	308	\$156,804.00
271	LGA	PAR	366	498	\$190,098.00
821	LGA	LON	345	243	\$150,634.00
821	LGA	LON	248	307	\$193,930.00



Enhancing the Heading's Appearance (3)

Quiz

Which attributes or options are reflected in this PROC REPORT output?

- a. SKIPLINE and FORMAT=
- b. CENTER, HEADLINE, HEADSKIP, and either WIDTH=, SPACING=, or FORMAT=
- c. SPACING= only
- d. CENTER, FORMAT=, and HEADLINE

Correct answer: b

The **HEADLINE** option underlines the headings, and the **HEADSKIP** option skips a line between the headings and the rows in the report. Also, Style is centered, and the column for Price is wider than the default.

Style	SqFeet	Price
RANCH	720	\$34,550
TWOSTORY	1040	\$55,850
SPLIT	1190	\$65,850
TWOSTORY	1240	\$69,250
RANCH	1250	\$64,000
SPLIT	1305	\$73,650
CONDO	1390	\$79,350
CONDO	1400	\$80,050
RANCH	1500	\$86,650
RANCH	1535	\$89,100
SPLIT	1615	\$94,450
TWOSTORY	1745	\$102,950
TWOSTORY	1810	\$107,250
CONDO	1860	\$110,700
CONDO	2105	\$127,150





Defining Variable Usage (1)

- How PROC REPORT uses variables
 - ▶ PROC REPORT uses each variable in one of six ways (**DISPLAY**, **ORDER**, **GROUP**, **ACROSS**, **ANALYSIS**, or **COMPUTED**).
 - ▶ By default, PROC REPORT uses
 - ┆ Character variables as **display variables**
 - ┆ Numeric variables as **analysis variables**, which are used to calculate the SUM statistic.
 - ▶ Because you haven't explicitly defined any variable usages, your current list report contains only display and analysis variables.
 - ┆ The character variables **Flight**, **Orig**, and **Dest** are display variables. Display variables don't affect the order of rows in the report. A report that contains one or more display variables has a detail row for each observation that is read from the data set. Each detail row contains a value for each display variable.
 - ┆ The numeric variables **Mail**, **Freight**, and **Revenue** are analysis variables. Analysis variables are used to calculate a statistic (In this case, the default **SUM**.)



Defining Variable Usage (2)

- ▶ In the illustration below, columns for display variables are shown in white. Columns for analysis variables are shown in gray.

Illustration of Partial PROC REPORT Output

	Flight Number	Flight Origin	Flight Destination	Mail	Freight	Revenue	
character variables (default=DISPLAY)	821	LGA	LON	403	209	\$150,634.00	numeric variables (default=ANALYSIS)
	271	LGA	PAR	492	308	\$156,804.00	
	271	LGA	PAR	366	498	\$190,098.00	
	821	LGA	LON	345	243	\$150,634.00	
	821	LGA	LON	248	307	\$193,930.00	



Defining Variable Usage (3)

Using order variables

- ▶ An order variable orders the detail rows in a report according to their formatted values.
- ▶ Example

```
proc report data=flights.europe nowd headline headskip;  
  where dest in ('LON' , 'PAR');  
  column flights orig dest mail freight revenue;  
  define revenue / format=dollar15.2;  
  define flight / order width=6 'Flight Number' center;  
  define orig / width=6 spacing=5 'Flight Origin' center;  
  define dest / width=11 spacing=5 'Flight/Destination' center;  
run;
```



Defining Variable Usage (4)

- ▶ This is ordered output. Notice that PROC REPORT displays only the first occurrence of each value of an order variable in a set of rows that have the same value for all ordered variable.

- ▶ By default, the order is ascending, but you can change it with the **DESCENDING** option in the DEFINE statement.

```
proc report data=flights.europe nowd headline headskip;
  where dest in ('LON' , 'PAR');
  column flights orig dest mail freight revenue;
  define revenue / format=dollar15.2;
  define flight / order descending width=6 'Flight
    Number' center;
  define orig / width=6 spacing=5 'Flight Origin'
    center;
  define dest / width=11 spacing=5 'Flight/Destination'
    center;
run;
```

Flight Number	Flight Origin	Flight Destination	Mail	Freight	Revenue
219	LGA	LON	331	376	\$189,065.00
	LGA	LON	485	267	\$197,456.00
	LGA	LON	388	298	\$162,343.00
	LGA	LON	421	356	\$134,520.00
	LGA	LON	447	299	\$106,753.00
	LGA	LON	356	547	\$122,766.00
	LGA	LON	272	370	\$198,744.00
271	LGA	PAR	492	308	\$156,804.00
	LGA	PAR	366	498	\$190,098.00
	LGA	PAR	353	205	\$166,470.00
	LGA	PAR	366	279	\$163,248.00
	LGA	PAR	357	282	\$170,766.00
	LGA	PAR	415	463	\$195,468.00
	LGA	PAR	352	351	\$123,456.00
	LGA	PAR	492	308	\$125,632.00
	LGA	PAR	366	498	\$128,972.00
	LGA	PAR	353	205	\$153,423.00
	LGA	PAR	366	279	\$133,345.00
	LGA	PAR	357	282	\$126,543.00
	LGA	PAR	415	463	\$134,976.00



Defining Variable Usage (5)

- Using group variables
 - ▶ To summarize your data using PROC REPORT, you can define one or more group variables.
 - ▶ A group variable groups the detail rows in a report according to their formatted values. If a report contains one or more group variables, PROC REPORT consolidates into one row all observations from the data set that have a unique combination of values for all group variables.
 - ▶ To define a group variable, you specify the GROUP usage option in the DEFINE statement.



Defining Variable Usage (6)

► Example

To group data in your report, you need to define in the character variables (**Flights**, **Orig**, and **Dest**) as group variables, as shown below:

```
proc report data=flights.europe nowd headline headskip;
  where dest in ('LON' , 'PAR');
  column flights orig dest mail freight revenue;
  define revenue / format=dollar15.2;
  define flight / group width=6 'Flight Number' center;
  define orig / group width=6 spacing=5 'Flight Origin' center;
  define dest / group width=11 spacing=5 'Flight/Destination' center;
run;
```

Flight Number	Flight Origin	Flight Destination	Mail	Freight	Revenue
219	LGA	LON	2700	2513	\$1,111,647.00
271	LGA	PAR	5050	4421	\$1,969,201.00
821	LGA	LON	4438	4284	\$2,077,907.00

Remember that the default statistic for the analysis variables is **SUM**.

The following table compares the effects of using order variables and group variables.

	ORDER	GROUP
Rows are ordered	yes	yes
Repetitious printing of values is suppressed	yes	yes
Rows that have the same values are collapsed	no	yes
Type of report produced	list	summary



Defining Variable Usage (7)

Specifying statistics

- ▶ The default statistic for analysis variables is **SUM**.
- ▶ To associate a statistic with an analysis variable, specify it as an attribute in the DEFINE statement.
- ▶ By specifying **MEAN** in the DEFINE statement for **Revenue**, you can display the average revenue for each flight number. The optional column heading **Average Revenue** clarifies that the MEAN statistic is displayed.

```
proc report data=flights.europe nowd headline headskip;
  where dest in ('LON' , 'PAR');
  column flights orig dest mail freight revenue;
  define revenue / mean format=dollar15.2 'Average/Revenue';
  define flight / group width=6 'Flight Number' center;
  define orig / group width=6 spacing=5 'Flight Origin' center;
  define dest / group width=11 spacing=5 'Flight/Destination' center;
run;
```

Flight Number	Flight Origin	Flight Destination	Mail	Freight	Average Revenue
219	LGA	LON	2700	2513	\$158,806.71
271	LGA	PAR	5050	4421	\$151,477.00
821	LGA	LON	4438	4284	\$159,839.00



Defining Variable Usage (8)

► You can use the following statistics in PROC REPORT

Statistic	Definition
CSS	Corrected sum of squares
USS	Uncorrected sum of squares
CV	Coefficient of variation
MAX	Maximum value
MEAN	Average
MIN	Minimum value
N	Number of observations with nonmissing values
NMISS	Number of observations with missing values
RANGE	Range
STD	Standard deviation
STDERR	Standard error of the mean
SUM	Sum
SUMWGT	Sum of the Weight variable values.
PCTN	Percentage of a cell or row frequency to a total frequency
PCTSUM	Percentage of a cell or row sum to a total sum
VAR	Variance
T	Student's t for testing the hypothesis that the population mean is 0
PRT	Probability of a greater absolute value of Student's t



Defining Variable Usage (9)

Using across variables

- ▶ You can also define variables as across variables, which are functionally similar to group variables. However, PROC REPORT displays the groups that it creates for an across variable horizontally rather than vertically.

▶ Example

```
proc report data=flights.europe nowd headline headskip;
  where dest in ('LON' , 'PAR');
  column flights orig dest mail freight revenue;
  define revenue / format=dollar15.2 'Average/Revenue' ;
  define flight / across width=6 'Flight Number' center;
  define dest / across width=11 spacing=5 'Flight/Destination' center;
run;
```

In this case, for each across variable, the table cells contain a frequency count for each unique value. For each analysis variable, the table cells represent the sum of all the variable's values.

Flight Number			Flight Destination				
219	271	821	LON	PAR	Mail	Freight	Revenue
7	13	13	20	13	12188	11218	\$5,158,755.00



Defining Variable Usage (10)

- Using computed variables
 - ▶ The last type of variable usage is reserved for computed variables, which are numeric or character variables that you define for the report.
 - ▶ They are not in the input data set, and PROC REPORT doesn't add them to the input data set.
 - ▶ You can't change the usage of a computed variable.
 - ▶ In the nonwindowing environment, you add a computed variable as follows:
 - ▶ Include the computed variable in the **COLUMN statement**.
 - ▶ Define the variable's usage as **COMPUTED** in the **DEFINE statement**.
 - ▶ Compute the value of the variable in a **compute block** that is associated with the variable.
 - ▶ The position of a computed variable is important. PROC REPORT assigns values to the columns in a row of a report from left to right.
 - ▶ Consequently, you can't base the calculation of a computed variable on any variable that appears to its right in the report.



Defining Variable Usage (11)

Example

- ▶ Suppose you want to determine the number of empty seats for each flight. To do so, you can compute the variable **Emptyseats** by subtracting the number of passengers deplaning (**Deplaned**) from the plane's total seats (**Capacity**), assuming that the plane is full.
- ▶ In the following program, you
 - ┌ specify **EmptySeats** in the **COLUMN statement** to the right of the variables that are used in its calculation.
 - ┌ define **EmptySeats** as a computed variable in a **DEFINE statement**
 - ┌ begin a compute block by specifying **EmptySeats** in a **COMPUTE statement**.
 - ┌ use DATA step statements in the compute block to calculate **EmptySeats'** value. Notice that when you refer to an analysis variable, you use a compound name that identifies both the original variable and the statistic that PROC REPORT now calculates from it. The compound name has the form **variable-name.statistic**.
 - ┌ close the compute block with an **ENDCOMP statement**.



Defining Variable Usage (12)

```
proc report data=flights.europe nowd;  
  where dest in ('LON','PAR');  
  column flight capacity deplaned emptyseats;  
  define flight / width=6;  
  define emptyseats / computed 'Empty Seats';  
  compute emptyseats;  
    emptyseats=capacity.sum-deplaned.sum;  
endcomp;  
  
run;
```

Partial PROC REPORT Output

Flight	Capacity	Deplaned	Empty Seats
821	250	222	28
271	250	163	87
271	250	227	23
821	250	222	28
821	250	158	92



Defining Variable Usage (13)

Quiz

When you define an order variable,

- a. the detail rows are ordered according to their formatted values.
- b. you can't create summary reports.
- c. PROC REPORT displays only the first occurrence of each order variable value in a set of rows that have the same value for all order variables.
- d. all of the above

Correct answer: d

Order Variables do order rows according to the formatted values of the order variable, and PROC REPORT suppresses repetitious printing of order values. However, you can't use order variables in a summary report.



Defining Variable Usage (14)



Quiz:

Which of the following programs produces this output?

Style				Average Price
CONDO	RANCH	SPLIT	TWOSTORY	
4	4	3	4	\$82,720

- a. `proc report data=sasuser.houses nowd;`
 `column style condo range split`
 `twostory price;`
 `define price / mean 'Average Price';`
 `run;`
- b. `proc report data=sasuser.houses nowd;`
 `column style price;`
 `define style / group;`
 `define price / mean 'Average Price';`
 `run;`
- c. `proc report data=sasuser.houses nowd;`
 `column style price;`
 `define style / across;`
 `define price / mean 'Average Price';`
 `run;`
- d. `proc report data=sasuser.houses nowd;`
 `column style price;`
 `define style / across 'CONDO'`
 `'RANCH' 'SPLIT' 'TWOSTORY';`
 `define price / mean 'Average Price';`
 `run;`

Correct answer: c

In this output, the table cells contain a frequency count for each unique value of an across variable, Style. You don't have to specify across variable values in your PROC REPORT step.

