Chapter 2: Creating list reports

Objectives : learning about

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

Selecting variables and observations

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. 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 noobs;
        var age height weight fee;
run;
```

cf) noobs is used to remove the Obs column.

You've learned how to remove the Obs column altogether. As another alternative, you can use one or more variables to replace the Obs column in the output. To specify which variables should replace the Obs column, use the ID statement. To replace the Obs column and identify observations based on an ID number, you can submit the following program.

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

■ Specifying where expressions

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 be only one WHERE statement in a step. For example, the following WHERE statement selects only observations for which the value of Age is greater than 30:

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

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

- enclose the value in quotation marks
- write the value with lowercase and uppercase letters exactly as it appears in the data set.

Symbol	Meaning
= or eq	equal to
^= or ne	not equal to
> or gt	greater than
< or lt	less than
>= or ge	greater than or equal to
<= or le	less than or equal to

■ Using the CONTAINS Operator

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

- where firstname CONTAINS 'Jon';
- where firstname ? 'Jon';

■ Specifying Compound WHERE Expressions

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

Operator	Meaning
And or **&**	and, both
OR or **|**	or, either

■ Examples of WHERE Statements

- Here are some examples of WHERE statements that use logical operators:

  where age<=55 and pulse>75;

  where area='A' or region='S';

  where ID>1050 and state='NC';

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

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

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

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

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

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

Sorting data

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

The SORT procedure

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

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

```
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;
```

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;
```

Generating Column Totals

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

■ Requesting Subtotals

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

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;
```

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

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

Specifying Titles and Footnotes

To make your report more meaningful and self-explanatory, you can specify 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.

cf) 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.

■ 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;
```

■ 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;
```

In 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.

■ 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;
```

Re-defining a title or footnote line cancels any higher-numbered title or footnote line, in that order.

In the example below, defining a title for line 2 in the second report automatically cancels title line 3.

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

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

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

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

Assigning Descriptive Labels

You can also enhance your PROC PRINT report by labeling columns with more descriptive text. You can assign labels in separate LABEL statements or you can assign any number of labels in a single LABEL statement.

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

Formatting Data Values

■ Modifying and Canceling Titles and Footnotes

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.

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.

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;
```

■ Specifying SAS Formats

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

Format	Specifies These Values	Example
COMMAw.d	that contain commas and decimal places	comma8.2
DOLLARw.d	that contain dollar signs, commas, and decimal places	dollar6.2
MMDDYYw.	as date values of the form 09/12/97 (MMDDYY8.) or 09/12/1997 (MMDDYY10.)	mmddyy10.
w.	rounded to the nearest integer in w spaces	7.
w.d	rounded to d decimal places in w spaces	8.2
$w.	as character values in w spaces	$12.
DATEw.	as date values of the form 16OCT99 (DATE7.) or 16OCT1999 (DATE9.)	date9.

■ Field Widths

All SAS formats specify the total field width (w) that is used for displaying the values in the output.

For example, suppose the longest value for the variable Net is a four-digit number, such as 5400.

To specify the 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, as shown in the table below.

cf) 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.

■ 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.

Formatting 15374 with a dollar sign, commas, and two decimal places requires 10 print positions.

■ Examples

This table shows you how data values are displayed when different format, field width, and decimal place specifications are used. table below describes some SAS formats that are commonly used in reports.

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

sas-code

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

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

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;

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;


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

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;

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;

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

footnote1 'Data from Treadmill Tests';
footnote3 '1st Quarter Admissions';
proc print data=clinic.stress;
```

```
        var resthr maxhr rechr;
        where tolerance='I';
run;

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;

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;

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;


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

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