Chapters 4, 10 CREATING LIST REPORTS

By default, SAS produces a traditional listing output. A more presentable style of SAS output is **HyperText Markup Language** (**HTML) output**.

Traditional Listing Output HTML Output

| **ID** | **Age** | **ActLevel** | **Gender** |
| --- | --- | --- | --- |
| 2810 | 61 | MOD | F |
| 2804 | 38 | HIGH | F |
| 2807 | 42 | LOW | M |
| 2816 | 26 | HIGH | M |
| 2833 | 32 | MOD | F |
| 2823 | 29 | HIGH | M |

Act

ID Age Level Gender

2810 61 MOD F

2804 38 HIGH F

2807 42 LOW M

2816 26 HIGH M

2833 32 MOD F

2823 29 HIGH M

SAS uses **Output Delivery System (ODS)** statement to generate an HTML output. The syntax is

ODS HTML BODY='html-file-pathname';

SAS procedures

ODS HTML CLOSE;

Example. The following program creates the HTML output above.

data exercise;

input id $ age actlevel $ gender $;

cards;

2810 61 MOD F

2804 38 HIGH F

2807 42 LOW M

2816 26 HIGH M

2833 32 MOD F

2823 29 HIGH M

;

**ods html body='C:\Users\mstudent\Desktop\outdata.html';**

title;

options nonumber nodate;

proc print noobs;

run;

**ods html close;**

In fact, this program produces

* the traditional SAS listing output in the SAS “Output” window,
* the HTML output in file outdata.html on C-drive,
* the HTML output in the SAS “Results Viewer-SAS Output” window.

To **suppress the traditional SAS listing output** and produce only the HTML output, type

**ODS LISTING CLOSE;**

ODS HTML BODY='html-file-pathname';

SAS procedures

ODS HTML CLOSE;

**ODS LISTING;**

Example. In the above example, type

ods listing close;

ods html body='G:\outdata.html';

title;

options nonumber nodate;

proc print noobs;

run;

ods html close;

ods listing;

No traditional SAS listing output is created in the “Output” window (Verify!).

MODIFYING LIST REPORTS

Several basic statements are often useful for modification of list reports. As example, we will consider only HTML outputs, even thought the same manipulations are valid for traditional SAS listing outputs.

Recall that we studied how to add variable labels, titles, footnotes, remove Obs column, time, date, and page number.

Example. In the exercise example, after the DATA statement, type

ods listing close;

ods html body='C:\Users\mstudent\Desktop\outdata.html';

title 'Exercise Data Set';

footnote 'HTML output';

options nonumber nodate;

proc print noobs label;

label actlevel='Activity Level';

run;

ods html close;

ods listing;

The output has the form:

|  |
| --- |
| ***Exercise Data Set*** |

| **id** | **age** | **Activity Level** | **gender** |
| --- | --- | --- | --- |
| 2810 | 61 | MOD | F |
| 2804 | 38 | HIGH | F |
| 2807 | 42 | LOW | M |
| 2816 | 26 | HIGH | M |
| 2833 | 32 | MOD | F |
| 2823 | 29 | HIGH | M |

|  |
| --- |
| ***HTML output*** |

Additional typically used statements include:

* sorting
* selecting variables
* identifying observations
* selecting observations
* generating column totals
* outputting subsets of data by variable name

PROC SORT

A dataset may be sorted in ascending or descending order with respect to a specified variable(s). By default, sorting is carried out in ascending order.

The syntax is

proc sort data=*dataname* out=*dataout*;

by <descending> *variable1* *variable2*;

run;

* If out=*dataout* is not specified, the original dataset is permanently sorted.
* If the option descending is specified, it refers only to the variable that follows it, not subsequent variables.

Example. In the exercise example, type

proc sort data=exercise out=exerciseout;

by descending age;

run;

proc print data=exerciseout noobs;

run;

The output is sorted with respect to variable age in descending order. Note that the original dataset is **not** sorted.

| **id** | **age** | **actlevel** | **gender** |
| --- | --- | --- | --- |
| 2810 | 61 | MOD | F |
| 2807 | 42 | LOW | M |
| 2804 | 38 | HIGH | F |
| 2833 | 32 | MOD | F |
| 2823 | 29 | HIGH | M |
| 2816 | 26 | HIGH | M |

IMPORTANT KEYNOTE:

If data=*dataname* is not specified, SAS uses the **previously created dataset**.

For example, in the above SAS program, data= statements may be omitted.

proc sort~~data=exercise~~ out=exerciseout;

by descending age;

run;

proc print ~~data=exerciseout~~ noobs;

run;

SAS understands that data=exercise should be used in proc sort, and newly created dataset exerciseout should be used in proc print. If dataset exercise is used in subsequent procedures, data=exercise must be specified, since it is **not** the latest dataset created.

SELECTING VARIABLES

If a subset of variables is to be displayed, use the **VAR statement** in PROC PRINT step. The syntax is

proc print;

var *variable1 variable2 …*;

run;

Example. In the exercise example, type

proc print noobs;

var id actlevel;

run;

The output is

| **id** | **actlevel** |
| --- | --- |
| 2810 | MOD |
| 2804 | HIGH |
| 2807 | LOW |
| 2816 | HIGH |
| 2833 | MOD |
| 2823 | HIGH |

IDENTIFYING OBSERVATIONS

When the Obs column is removed, it might be necessary to specify a variable that identifies observations. This technique is particularly useful when observations are too long to print on one line. To specify which variables should replace the Obs column, use the **ID statement** in PROC PRINT step. The syntax is

proc print;

ID variable(s);

run;

Example. In the exercise example, type

proc print noobs;

ID id;

run;

The output is

| **id** | **age** | **actlevel** | **gender** |
| --- | --- | --- | --- |
| **2810** | 61 | MOD | F |
| **2804** | 38 | HIGH | F |
| **2807** | 42 | LOW | M |
| **2816** | 26 | HIGH | M |
| **2833** | 32 | MOD | F |
| **2823** | 29 | HIGH | M |

SELECTING OBSERVATIONS

Recall that options firstobs=*n* and obs=*n* may be used to display observations in a certain range. A more sophisticated way to select observations is to use a **WHERE statement** in PROC PRINT step. The syntax is

WHERE *where-expression*;

where *where-expression* specifies a condition for selecting observations. For example, where age>30 or where gender='F'.

The **comparison operators** that may be used in where-expression are

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Meaning** | **Example** |
| = or eq | Equal to | where Age=30;  where ID='1533'; |
| ^= or ne | Not equal to | where Age^=30;  where ID ne '1533'; |
| > or gt | Greater than | where Age>30;  where ID gt '1533'; |
| < or lt | Less than | where Age lt 30;  where ID<'1533'; |
| >= or ge | Greater than or equal to | where Age>=30;  where ID ge'1533'; |
| <= or le | Less than or equal to | where Age le 30;  where ID<='1533'; |

**Compound** where-expression may be specified that selects observations satisfying **multiple conditions**. The **logical operators** that may be used are

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Meaning** | **Example** |
| AND or & | **Both** conditions must hold | where Age=30 and ID<= '1533' |
| OR or | | **Either** condition must hold | where (Age ne 30 or ID ge'1533') |

* Note that multiple conditions may be enclosed in parentheses for reading ease, even though it is not necessary.

Example. In the exercise example, type

proc print noobs;

where (age>30 and id<'2820');

run;

The output is

| **id** | **age** | **actlevel** | **gender** |
| --- | --- | --- | --- |
| 2810 | 61 | MOD | F |
| 2804 | 38 | HIGH | F |
| 2807 | 42 | LOW | M |

* An **IN operator** may be used in the WHERE statement.

Example. In the exercise example, type

proc print noobs;

where age in (26, 29, 38);

run;

The output is

| **id** | **age** | **actlevel** | **gender** |
| --- | --- | --- | --- |
| 2804 | 38 | HIGH | F |
| 2816 | 26 | HIGH | M |
| 2823 | 29 | HIGH | M |

* The **CONTAINS operator** may be used in the WHERE statement. It selects all observations for a **character variable** that include the specified substring. Alternatively, a question mark (?) may be used. The syntax is

proc print;

where *variablename* contains 'substring';

run;

Example. In the exercise example, type

proc print noobs;

where actlevel contains 'O';

run;

or

proc print noobs;

where actlevel ? 'O';

run;

The output is

| **id** | **age** | **actlevel** | **gender** |
| --- | --- | --- | --- |
| 2810 | 61 | MOD | F |
| 2807 | 42 | LOW | M |
| 2833 | 32 | MOD | F |

* The **LIKE operator** may be used in the WHERE statement. It selects all observations for a **character variable** that match a specified pattern. In the pattern, an underscore (\_) may be used in place of any single character, and a percent sign (%) may replace any number of characters (0 or more). The syntax is

proc print;

where *variablename* like 'pattern';

run;

Example. In the exercise example, type

proc print noobs;

where id like '%1\_';

run;

The output is

| **id** | **age** | **actlevel** | **gender** |
| --- | --- | --- | --- |
| 2810 | 61 | MOD | F |
| 2816 | 26 | HIGH | M |

GENERATING COLUMN TOTAL

To calculate column totals for numeric variables, use a **SUM statement** in PROC PRINT step. The syntax is

sum *variable1 variable2* …;

Example. Suppose in the exercise example, payment due is recorded for each member. To compute the total payment amount due, type

data exercise;

input id $ age actlevel $ gender $ payment;

cards;

2810 61 MOD F 34.00

2804 38 HIGH F 29.00

2807 42 LOW M 27.50

2816 26 HIGH M 20.25

2833 32 MOD F 15.50

2823 29 HIGH M 23.00

;

ods listing close;

ods html body='C:\Users\mstudent\Desktop\outdata.html';

title;

options nonumber nodate;

proc print noobs label;

label payment='Payment Due';

**sum payment;**

run;

ods html close;

ods listing;

The output is

| **id** | **age** | **actlevel** | **gender** | **Payment Due** |
| --- | --- | --- | --- | --- |
| 2810 | 61 | MOD | F | 34.00 |
| 2804 | 38 | HIGH | F | 29.00 |
| 2807 | 42 | LOW | M | 27.50 |
| 2816 | 26 | HIGH | M | 20.25 |
| 2833 | 32 | MOD | F | 15.50 |
| 2823 | 29 | HIGH | M | 23.00 |
|  |  |  |  | **149.25** |

Total Payment Due

OUTPUTTING SUBSETS OF DATA BY VARIABLE NAME

Suppose subtotals for a numeric variable1 must be computed for different values of variable2. Firstly, the data must be **sorted** by variable2. Then a **BY statement** may be used. The syntax is

sum *variable1*;

by *variable2*;

Example. Suppose the total payments due for different activity levels are of interest. The code is

proc sort;

by actlevel;

run;

proc print noobs label;

label payment='Payment Due';

sum payment;

by actlevel;

run;

The output is

actlevel=HIGH

| **id** | **age** | **gender** | **Payment Due** |
| --- | --- | --- | --- |
| 2804 | 38 | F | 29.00 |
| 2816 | 26 | M | 20.25 |
| 2823 | 29 | M | 23.00 |
| **actlevel** |  |  | **72.25** |

actlevel=LOW

| **id** | **age** | **gender** | **Payment Due** |
| --- | --- | --- | --- |
| 2807 | 42 | M | 27.5 |

actlevel=MOD

| **id** | **age** | **gender** | **Payment Due** |
| --- | --- | --- | --- |
| 2810 | 61 | F | 34.00 |
| 2833 | 32 | F | 15.50 |
| **actlevel** |  |  | **49.50** |
|  |  |  | **149.25** |