**SAS STATEMENTS EXAMPLES**

**LIBNAME STATEMENT**

**Syntax - Libname libref “path”;**

**Ex – libname abc “c:\sample\datasets”;**

/\*\*\*\*To create Permanent library\*\*\*\*\*\*\*/

libname aa "E:\test";

libname aa2 ('E:\test' 'E:\test\zi');

**data** aa.scores\_l1;

input name $ score1 score2;

datalines;

Riley 1132 1187

Henderson 1015 1102

;

**run**;

**proc** **print** data= aa.scores\_l1;

**run**;

**data** aa.prog1;

infile "E:\test\sam.txt" ;

input name $ age;

**run**;

**proc** **print** data=aa.prog1;

**run**;

/\*\*\*\*To create Temporary library\*\*\*\*\*\*\*/

**data** work.scores\_l1;

input name $ score1 score2;

datalines;

Riley 1132 1187

Henderson 1015 1102

;

**run**;

**proc** **print** data= aa.scores\_l1;

**run**;

**data** work.prog1;

infile "E:\test\sam.txt" ;

input name $ age;

**run**;

**proc** **print** data=work.prog1;

**run**;

**INPUT TYPES**

INPUT variable start-column<end-column>;

reads the input data record using column input.

You can omit end-column if the data is only 1 byte long.

This style of input enables you to skip columns of data that you want to omit.

/\*\*\*\*\*List Input\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**data** scores\_l1;

input name $ score1 score2;

datalines;

Riley 1132 1187

Henderson 1015 1102

;

**run**;

**proc** **print** data= scores\_l1;

**run**;

**data** scores\_l2;

length name $10.;

input name $ score1 score2;

datalines;

Riley 1132 1187

Henderson 1015 1102

;

**run**;

**proc** **print** data=scores\_l2;

**run**;

/\*\*\*\*\*column Input\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**data** scores;

length name $10.;

input name $**1**-**9** score1 **11**-**14** score2 **16**-**19**;

datalines;

Riley 1132 1187

Henderson 1015 1102

;

**run**;

**proc** **print** data=scores;

**run**;

/\*\*\*\*\*\*\*\*\*\*formatted input\*\*\*\*\*\*\*\*\*\*\*/

**data** jansales\_f1;

input item $ amount;

datalines;

trucks 1,382

vans 1,235

sedans 2,391

;

**proc** **print** data=jansales\_f1;

**run**;

**data** jansales\_f2;

informat amount comma5.;

input item $ amount;

datalines;

trucks 1,382

vans 1,235

sedans 2,391

;

**proc** **print** data=jansales\_f2;

**run**;

**data** jansales\_f3;

format amount dollar8.;

input item $ amount comma5.;

datalines;

trucks 1,382

vans 1,235

sedans 2,391

;

**proc** **print** data=jansales\_f3;

**run**;

**data** jansales\_f4;

input item $ amount comma5.;

datalines;

trucks 1,382

vans 1,235

sedans 2,391

;

**proc** **print** data=jansales\_f4;

format amount dollar8.;

**run**;

**data** jansales\_f5;

input item $ st\_dte date7. amount comma5.;

datalines;

trucks 21aug16 1,382

vans 3may2000 1,235

sedans 9jan2009 2,391

;

**proc** **print** data=jansales\_f5;

format amount dollar8.;

**run**;

**data** jansales\_f5;

input item $ st\_dte date9. amount comma5.;

datalines;

trucks 21aug2016 1,382

vans 3may2017 1,235

sedans 9jan1959 2,391

;

**proc** **print** data=jansales\_f5;

format amount dollar8. st\_dte date9.;

**run**;

/\*NAMED INPUT\*/

**data** games;

input name=$ score1= score2=;

datalines;

name=riley score1=1132 score2=1187

;

**proc** **print** data=games;

**run**;

**MODIFIED LIST INPUTS**

/\*Modified List Input

List input is more versatile when you use format modifiers. The format modifiers are as follows:

“Format Modifier Purpose”

& reads character values that contain embedded blanks.

: reads data values that need the additional instructions that informats can provide but that are not aligned in columns. \*\*

~ reads delimiters within quoted character values as characters and retains the quotation marks.

\*\* Use formatted input and pointer controls to quickly read data values that are aligned in columns.

For example, use the : modifier with an informat to read character values that are longer than 8 bytes

or numeric values that contain nonstandard values.

INPUT variable : informat;

INPUT variable & informat;

reads the input data record using modified list input.

The : (colon format modifier) instructs SAS to use the informat that follows to read the data value.

The & (ampersand format modifier) instructs SAS to use the informat that follows to read the data value.

When you use the ampersand format modifier, two blanks are required to signal the end of a data value.

INPUT <pointer-control> variable informat;

reads raw data using formatted input. The informat supplies special instructions to read the data.

You can also use a pointer-control to direct SAS to start reading at a particular column.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**data** jansales;

input item : $10. amount comma5.;

datalines;

trucks 1,382

vans 1,235

sedans 2,391

;

**proc** **print** data=jansales;

**run**;

**data** scores1;

length name $ **14**;

input name $ score1 score2;

datalines;

Riley m singh 1132 1187

Henderson 1015 1102

;

**run**;

**proc** **print** data=scores1;

**run**;

**data** scores2;

input name & $ score1 score2;

datalines;

Riley m singh 1132 1187

Henderson 1015 1102

;

**run**;

**proc** **print** data=scores2;

**run**;

**data** scores3;

length name $ **14**;

input name & $ score1 score2;

datalines;

Riley m singh 1132 1187

Henderson 1015 1102

;

**run**;

**proc** **print** data=scores3;

**run**;

**data** scores3;

input name & $**1**-**14** score1 **15**-**18** score2;

datalines;

Riley m singh 1132 1187

Henderson 1015 1102

;

**run**;

**data** scores4;

length name $ **14**;

input name & $ @**15** score1 score2;

datalines;

Riley m singh 1132 1187

Henderson 1015 1102

;

**run**;

**proc** **print** data=scores4;

**run**;

**data** scores1;

infile datalines dsd;

input Name : $9. Score1-Score3 Team ~ $25. Div $;

datalines;

Smith,12,22,46,"Green Hornets, Atlanta",AAA

Mitchel,23,19,25,"High Volts, Portland",AAA

Jones,09,17,54,"Vulcans, Las Vegas",AA

;

**run**;

**proc** **print** data=scores1 noobs;

**run**;

**Writes date values as the day of the week and the date in the form *day-of-week*, *month-name dd*, *yy* (or *yyyy*)**

**data** scores2;

length Team $ **14**;

infile datalines delimiter=',';

input Name $ Score1-Score3 Team $ Final\_Date:MMDDYY10.;

format final\_date weekdate17.;

datalines;

Joe,11,32,76,Red Racers,2/3/2007

Mitchell,13,29,82,Blue Bunnies,4/5/2007

Susan,14,27,74,Green Gazelles,11/13/2007

;

**proc** **print** data=scores2;

var Name Team Score1-Score3 Final\_Date;

title 'Soccer Player Scores';

**run**;

**data** scores3;

length Team $ **14**;

infile datalines delimiter=',';

input Name $ Score1-Score3 Team $ Final\_Date:MMDDYY10.;

format final\_date weekdate17.;

datalines;

Joe,11,32,76,Red Racers,2/3/2007

Mitchell,13,29,82,Blue Bunnies,4/5/2007

Susan,14,,74,Green Gazelles,11/13/2007

;

**proc** **print** data=scores3;

var Name Team Score1-Score3 Final\_Date;

title 'Soccer Player Scores';

**run**;

**data** scores3;

length Team $ **14**;

infile datalines dsd;

input Name $ Score1-Score3 Team $ Final\_Date:MMDDYY10.;

format final\_date weekdate17.;

datalines;

Joe,11,32,76,Red Racers,2/3/2007

Mitchell,13,29,82,Blue Bunnies,4/5/2007

Susan,14,,74,Green Gazelles,11/13/2007

;

**proc** **print** data=scores3;

var Name Team Score1-Score3 Final\_Date;

title 'Soccer Player Scores';

**run**;

**LENGTH**

**data** a;

length name $9. ;

input name $ sal;

cards;

ramkumars 1000

ff 5000

;

**run**;

**proc** **print** data=a;

**run**;

**data** a;

\*length name $12. ;

input name $9. sal;

cards;

ramkumars 1000

ff 5000

;

**run**;

**proc** **print** data=a;

**run**;

**data** a;

input name $9. sal;

cards;

ramkumars 1000

ff 5000

;

**run**;

**proc** **print** data=a;

**run**;

**Column pointer controls**

**data** x2;

input region $7. @**10** jansales **5.**;

cards;

califor 10000

minnaso 50000

;

**run**;

**proc** **print** data=x2;

**run**;

**data** x;

input region $7. +**4** jansales **5.**;

cards;

califor 10000

minnaso 50000

;

**run**;

**proc** **print** data=x;

**run**;

**data** x1;

input region $7. +**4** jansales **5.**;

cards;

califor 10000

minnaso 50000

;

**run**;

**proc** **print** data=x1;

**run**;

**line Pointer Control (/,#)**

**data** y;

input Lname $ **1**-**8** Fname $ **10**-**15** /

Department $ **1**-**12** JobCode $ **15**-**19** /

Salary comma10.;

cards;

ABRAMS THOMAS\_

MARKETING     SR01\_

$25,209.03

BARCLAY ROBERT\_

EDUCATION     IN01\_

$24,435.71

;

**run**;

**proc** **print** data=y;

**run**;

/\* #n

moves the pointer to record n. \*/

**data** z;

input #**2** Department $ **1**-**12** JobCode $ **15**-**19**

#**1** Lname $ **1**-**8** Fname $ **10**-**15**

#**3** Salary comma10.;

cards;

ABRAMS THOMAS\_

MARKETING     SR01\_

$25,209.03

BARCLAY ROBERT\_

EDUCATION     IN01\_

$24,435.71

;

**run**;

**proc** **print** data=z;

**run**;

/\*

**Line-Hold Specifiers**

**single Trailing .> @**

(trailing @) prevents SAS from automatically reading a new data record into the input buffer when a new INPUT statement is executed within the same iteration of the DATA step. When used,

the trailing @ must be the last item in the INPUT statement.

@@

**(double trailing @@)** prevents SAS from automatically reading a new data record into the input buffer when the next INPUT statement is executed, even if the DATA step returns to the top for another iteration. When used, the double trailing @@ must be the last item in the INPUT statement.

\*/

**Single Trailing -> @**

**data** red\_team;

input Team $ **13**-**18** ;

if Team='red';

input IdNumber **1**-**4** StartWeight **20**-**22** EndWeight **24**-**26**;

datalines;

1023 David red 189 165

1049 Amelia yellow 145 124

1219 Alan red 210 192

1246 Ravi yellow 194 177

1078 Ashley red 127 118

1221 Jim yellow 220 .

;

**proc** **print** data=red\_team;

title 'Red Team';

**run**;

**data** red\_team;

input Team $ **13**-**18** @;

if Team='red';

input IdNumber **1**-**4** StartWeight **20**-**22** EndWeight **24**-**26**;

datalines;

1023 David red 189 165

1049 Amelia yellow 145 124

1219 Alan red 210 192

1246 Ravi yellow 194 177

1078 Ashley red 127 118

1221 Jim yellow 220 .

;

**proc** **print** data=red\_team;

title 'Red Team';

**run**;

**Double trailing -> @@**

**data** a;

input name $ age salary ;

cards;

kumar 45 1000 junh 67 2000 ion 66 3000 lam 78 4000

;

**run**;

**proc** **print** data=a;

**run**;

**data** a;

input name $ age salary @@;

cards;

kuar 45 1000 junh 67 2000 ion 66 3000 lam 78 4000

;

**run**;

**proc** **print** data=a;

**run**

**data** a;

input name $ age salary @@;

cards;

kuar 45 1000 junh 67 2000 ion 66 3000 lam 78 4000

kuar 45 2000 junh 67 3000 ion 66 3000 lam 78 4000

;

**run**;

**proc** **print** data=a;

**run**;

**INFILE**

Statements

DATALINES;

indicates that data lines immediately follow. A semicolon in the line that immediately follows the

last data line indicates the end of the data and causes the DATA step to compile and execute.

INFILE fileref< FLOWOVER | STOPOVER | MISSOVER | TRUNCOVER>;

INFILE 'external-file' <FLOWOVER | STOPOVER | MISSOVER | TRUNCOVER>;

identifies an external file to be read by an INPUT statement.

Specify a fileref that has been assigned with a FILENAME statement or with an appropriate operating environment command.

Or you can specify the actual name of the external file.

These options give you control over how SAS behaves if the end of a data record is encountered

before all of the variables are assigned values. You can use these options with list, modified list,

formatted, and column input.

FLOWOVER

is the default behavior. It causes the DATA step to look in the next

record if the end of the current record is encountered before all of the variables are assigned values

MISSOVER

causes the DATA step to assign missing values to any variables that do not

have values when the end of a data record is encountered. The DATA step continues processing.

STOPOVER

causes the DATA step to stop execution immediately and write a note to the SAS log.

TRUNCOVER

causes the DATA step to assign values to variables,

even if the values are shorter than expected by the INPUT statement,

and to assign missing values to any variables that do not have values when the end of a record is encountered.

INPUT variable <&> <$>;

reads the input data record using list input.

The & (ampersand format modifier) allows character values to contain embedded blanks.

When you use the ampersand format modifier, two blanks are required to signal the end of a data value.

The $ indicates a character variable.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**data** scores;

infile datalines ;

input test1 test2 test3;

datalines;

91 87 95

97 45 92

1 1 1

;

**proc** **print** data=scores;

**run**;

**data** scores;

infile datalines delimiter=',';

input test1 test2 test3;

datalines;

91,87,95

97,1,92

1,1,1

;

**proc** **print** data=scores;

**run**;

**data** scores1;

infile datalines dsd;

input test1 test2 test3;

datalines;

91,87,95

97,,92

,1,1

;

**proc** **print** data=scores1;

**run**;

**data** weather;

infile datalines;

input temp1-temp5;

datalines;

97.9 98.1 98.3

98.6 99.2 99.1 98.5 97.5

96.2 97.3 98.3 97.6 96.5

;

**proc** **print** data=weather;

**run**;

**data** weather;

infile datalines missover;

input temp1-temp5;

datalines;

97.9 98.1 98.3

98.6 99.2 99.1 98.5 97.5

96.2 97.3 98.3 97.6 96.5

;

**proc** **print** data=weather;

**run**;

**data** prog1;

infile "E:\test\sam.txt" ;

input name $ age;

**run**;

**proc** **print** data=prog1;

**run**;

**data** prog1;

infile "E:\test\sam.txt" firstobs=**2** ;

input name $ age;

**run**;

**proc** **print** data=prog1;

**run**;

**data** prog1;

infile "E:\test\sam1.txt" ;

input name $ age;

**run**;

**proc** **print** data=prog1;

**run**;

**data** prog1;

infile "E:\test\sam2.txt" firstobs=**2** dlm=',';

input name $ age;

**run**;

**proc** **print** data=prog1;

**run**;

**data** prog1;

infile "E:\test\sam3.txt" dlm='|';

input name $ age;

**run**;

**proc** **print** data=prog1;

**run**;

**data** a4;

infile "E:\test\sam3.txt" firstobs=**2** dlm='|';

input name $ age;

**run**;

**proc** **print** data=a4;

**run**;

**data** a4;

infile "E:\test\sam4.txt" firstobs=**2** dlm='|' dsd;

input name $ age sal;

**run**;

**proc** **print** data=a4;

**run**;

**data** a2;

infile "E:\test\sam.dat" firstobs=**2**;

input name $ age;

**run**;

**proc** **print** data=a2;

**run**;

**data** a3;

infile "E:\test\sam.csv" firstobs=**2** dlm=',';

input name $ age;

**run**;

**proc** **print** data=a3;

**run**;

**data** weather;

infile "E:\test\miss.txt";

input temp1-temp5;

**run**;

**proc** **print** data=weather;

**run**;

**data** weather;

infile "E:\test\miss.txt" missover;

input temp1-temp5;

**run**;

**proc** **print** data=weather;

**run**;

**data** aa;

infile "E:\test\trun.txt";

input num **5.**;

**run**;

**proc** **print** data=aa;

**run**;

**data** aa;

infile "E:\test\trun.txt" truncover;

input num **5.**;

**run**;

**proc** **print** data=aa;

**run**;

**data** aa;

infile "E:\test\trun.txt" truncover;

input num $5.;

**run**;

**proc** **print** data=aa;

**run**;

**FILENAME**

filename fileref "path";

Filename Myfile 'C:/Training/SAS/Data.txt';

**data** prog1;

infile myfile firstobs=**2** ;

input name $ age;

**run**;

**proc** **print** data=prog1;

**run**;

**SET**

**data** a;

length name $12. ;

input name $ sal;

cards;

gghgffjfjh 1000

ff 5000

;

**run**;

**data** new; /\* Create new dataset from existing dataset\*/

set a;

run;

**proc** **print** data=new;

**run**;

**data** b;

length name $12. ;

input name $ sal;

cards;

kumar 4000

kavitha 5000

;

**run**;

**data** d;

set a;

\*where name='ff';

if name='ff';

**run**;

**data** c;

set a b; /\* It is combine two datasets \*/

**run**;

**data** a;

\*length name $12. ;

input name : $12. sal;

cards;

gghgffjfjh 1000

ff 5000

;

**run**;

**proc** **print** data=a;

**run**;

**data** c;

length name $12. ;

input name $ sal age;

cards;

gghgffjfjh 1000 23

ff 5000 34

;

**run**;

**data** d;

length name $12. ;

input name $ sal;

cards;

kumar 4000

kavitha 5000

;

**run**;

**data** e;

set a c; /\* It is combine two datasets \*/

**run**;

**data** e;

set c d; /\* It is combine two datasets \*/

**run**;

**RENAME**

**data** n;

input name $ gender $;

cards;

rahul male

ramya female

;

**run**;

**proc** **print** data=n;

**run**;

**data** k;

set n;

rename name=emp\_name;

**run**;

**data** k;

set n;

rename name=emp\_name gender=sex;

**run**;

**data** m;

set k;

sal=**1000**; /\* Create a new variable \*/

label emp\_name=patname;

**run**;

**data** l;

set k;

where patname='rahul';

where emp\_name='rahul';

**run**;

ATTRIB

**data** a;

length name $**11**;

informat sal comma5.;

format sal comma5.;

label name='empname';

input name & $ @**13** age sal;

cards;

Ram m kumar 23 1,000

tina 45 2,000

rima 34 3,000

;

**run**;

**proc** **print** data=a label;

label name='empname';

**run**;

**data** b;

set a;

label name='empname';

**run**;

**data** c;

set b;

where empname='tina';

**run**;

**data** c;

set b;

where name='tina';

**run**;

**data** a1;

attrib name length=$11. label='emppname' sal informat=comma5. format=comma5. label='salary';

input name & $ @**13** age sal;

cards;

Ram m kumar 23 1,000

tina 45 2,000

rima 34 3,000

;

**run**;

**proc** **print** data=a1;

**run**;

**PUT Statement**

**data** st1;

a=**1**;

b=**2**;

C=a+b;

put 'the output value for' C=;

**run**;

**data** st2;

a=**1**;

b=**2**;

C=a+b;

put C=;

**run**;

**data** \_null\_;

a=**1**;

b=**2**;

C=a+b;

put 'the output value for' C=;

**run**;

/\*\*\*\*ATTRN - Returns the value of a numeric attribute for a SAS data set \*\*\*\*/

**data** \_null\_;

dsid=open("tour");

isindex=attrn(dsid,"isindex");

if isindex then put "data set is indexed";

else put "data set is not indexed";

**run**;

**data** \_null\_;

dsid=open("tour");

pw=attrn(dsid,"pw");

if pw then put "data set is protected";

**run**;

**Do While**

**/\***

This DO WHILE loop uses a WHILE condition. The SAS statements are repeatedly executed until the while condition becomes false.

\*/

**data** work.test;

x=**1**;

do while(x<**12**);

x+**1**;

output;

end;

**run**;

**proc** **print** data=test;

**run**;

**data** work.test;

x=**10**;

do while(x<=**12**);

x+**1**;

output;

end;

**run**;

**proc** **print** data=test;

**run**;

**data** work.test;

x=**13**;

do while(x<**12**);

x+**1**;

output;

end;

**run**;

**proc** **print** data=test;

**run**;

**Do Until**

The expression is not evaluated until the bottom of the loop, so a DO UNTIL loop always executes at least once.

When the expression is evaluated as true, the DO loop is not executed again.

Suppose you want to know how many years it will take to earn $50,000 if you deposit $2,000 each year

into an account that earns 10% interest. The DATA step below uses a DO UNTIL statement to perform the calculation until the value is reached. Each iteration of the DO loop represents one year of earning.

During each iteration of the DO loop,

2000 is added to the value of Capital to reflect the annual deposit of $2,000

the value of Capital with 10% interest is calculated

the value of Year is incremented by 1.

Because there is no index variable in the DO UNTIL statement, the variable Year is created in a sum statement to count

the number of iterations of the DO loop. This program produces a data set that contains the single observation shown below.

To accumulate more than $50,000 in capital requires thirteen years (and thirteen iterations of the DO loop).

\*/

**data** work.invest;

do i=**1** to **10** until(capital>**50000**);

Year+**1**;

Capital+**2000**;

capital+capital\***.10**;

output;

end;

put 'capital=' capital;

**run**;

**proc** **print** data=invest;

**run**;

**FIRST. Variable**

**data** Patients;

informat Date date7.;

format Date date7. PatientID Z4.;

input PatientID Date Weight @@;

datalines;

1021 04Jan16 302 1042 06Jan16 285

1053 07Jan16 325 1063 11Jan16 291

1053 01Feb16 299 1021 01Feb16 288

1063 09Feb16 283 1042 16Feb16 279

1021 07Mar16 280 1063 09Mar16 272

1042 28Mar16 272 1021 04Apr16 273

1063 20Apr16 270 1053 28Apr16 289

1053 13May16 295 1063 31May16 269

;

**proc** **sort** data=Patients;

by PatientID Date;

**run**;

**data** weightLoss;

set Patients;

BY PatientID;

if FIRST.PatientID then output; /\* output only the first record in each group \*/

if LAST.PatientID then output; /\* output only the last record in each group \*/

**run**;

**proc** **print** data =weightLoss noobs;

**run**;

**data** weightLoss;

set Patients;

BY PatientID;

retain startDate startWeight; /\* RETAIN the starting values \*/

if FIRST.PatientID then do;

startDate = Date; startWeight = Weight; /\* remember the initial values \*/

end;

if LAST.PatientID then do;

endDate = Date; endWeight = Weight;

elapsedDays = intck('day', startDate, endDate); /\* elapsed time (in days) \*/

weightLoss = startWeight - endWeight; /\* weight loss \*/

AvgWeightLoss = weightLoss / elapsedDays; /\* average weight loss per day \*/

output; /\* output only the last record in each group \*/

end;

**run**;

**proc** **print** noobs;

var PatientID elapsedDays startWeight endWeight weightLoss AvgWeightLoss;

**run**;

/\* The output data set summarizes each patient's activities at the clinic, including his average weight loss and the duration of his treatment. \*/

**data** tour;

input TourType $**1**-**13** Country $**14**-**30** Nights **31**-**32** landCost **39**-**46** Vendor $**47**-**54**;

cards;

architecture Spain 10 510 World

architecture Japan 8 720 Express

architecture France 8 575 World

architecture Italy 8 468 Express

scenery Switzerland 9 734 World

scenery Ireland 7 558 Express

scenery New Zealand 16 1489 Southsea

scenery Greece 12 698 Express

;

**run**;

options pagesize=**60** linesize=**80** pageno=**1** nodate;

**proc** **sort** data=tour;

by TourType;

**run**;

**data** temp;

set tour;

by TourType;

\* if last.TourType;

/\* FirstTour = first.TourType;\*/

/\* LastTour = last.TourType;\*/

if first.TourType then output;

if last.TourType then output;

**run**;

**proc** **print** data=temp;

\*var Country Tourtype FirstTour LastTour;

title 'Specifying FIRST.TOURTYPE and LAST.TOURTYPE';

**run**;

data temp1;

set tour;

by TourType;

\* if last.TourType;

/\* FirstTour = first.TourType;\*/

/\* LastTour = last.TourType;\*/

if first.TourType then N = **1**;

else N +**1**;

**run**;

**data** temp2;

set tour;

by TourType;

\* if last.TourType;

/\* FirstTour = first.TourType;\*/

/\* LastTour = last.TourType;\*/

if first.TourType then N = **1**;

else if last.TourType then N = **2**;

else N=**3**;

**run**;

proc print data=temp1;

\*var Country Tourtype FirstTour LastTour;

title 'Specifying FIRST.TOURTYPE and LAST.TOURTYPE';

run;

**data** fruit;

input x $ y $ **10**-**18** z $ **21**-**29**;

datalines;

apple banana coconut

apple banana coconut

apple blueberry citron

apricot blueberry citron

;

**data** \_null\_;

set fruit; by x y z;

if \_N\_=**1** then put 'Grouped by X Y Z';

put \_N\_= x= first.x= last.x= first.y= last.y= first.z= last.z= ;

**run**;

**data** \_null\_;

set fruit; by y x z;

if \_N\_=**1** then put 'Grouped by Y X Z';

put \_N\_= first.y= last.y= first.x= last.x= first.z= last.z= ;

**run**;

**data** have;

informat date1 date9.;

format date1 date9.;

input date1 id flag $;

cards;

21jan2018 1 n

21jan2018 1 y

21jan2018 1 y

22jan2018 1 y

22jan2018 1 y

22jan2018 1 y

22jan2018 2 n

22jan2018 2 y

22jan2018 2 y

22jan2018 2 y

22jan2018 2 y

;

**run**;

**data** test1;

set have;

by date1 id;

retain flag1;

if first.id then flag1 = flag;

**run**;

libname class '*SAS-library*';

proc sort data=class.allscores;

by id;

run;

data class.bestscores;

drop grade;

set class.allscores;

by id;

/\* Prevents HIGHEST from being reset\*/

/\* to missing for each iteration. \*/

retain highest;

/\* Sets HIGHEST to missing for each \*/

/\* different ID value. \*/

if first.id then highest=.;

/\* Compares HIGHEST to GRADE in \*/

/\* current iteration and resets \*/

/\* value if GRADE is higher. \*/

highest=max(highest,grade);

if last.id then output;

run;

**IF Statement**

**/\* IF** expression **THEN** statement;   
<**ELSE** statement;>

\*/

**data** a;

input name $ sal date date7.;

format date date7.;

cards;

gg 1000 12may08

ff 5000 23jun06

;

**run**;

**proc** **print** data=a;

format data date7.;

by sal ;

**run**;

**data** b;

set a;

if sal>**1000**;

**run**;

**proc** **print** data=b;

**run**;

**data** b;

set a;

if sal>**1000** then delete;

**run**;

**data** a;

input name $ age;

cards;

kk 100

jj 300

;

**run**;

**proc** **print** data=a;

**run**;

**data** b;

set a;

if name='kk' then delete;

**run**;

**data** a;

input item $ cost saledate mmddyy8.;

format saledate date9.;

if cost=**1000** then delete; /\*subsetting if\*/

cards;

10e 1000 12/01/09

10r 2000 11/20/08

;

**run**;

**proc** **print** data=a;

**run**;

**data** a;

input Gender $ cost saledate mmddyy8.;

format saledate date9.;

cards;

M 1000 12/01/09

F 2000 11/20/08

F 3000 11/22/12

M 4000 11/26/13

;

**run**;

**proc** **print** data=a;

**run**;

**data** x y;

set a;

if Gender='M' then output x;

else output y;

**run**;

**data** k;

format saledate date9.;

input id item $ cost saledate mmddyy8.;

cards;

101 10e 1000 12/01/09

201 10r 2000 11/20/08

301 10r 3000 12/14/12

;

**run**;

**data** b;

set k;

if id=**101** then cost=**3000**;

else if id=**201** then cost=**5000**;

else cost=**200**

;

**run**;

**proc** **print** data=b;

**run**;

**IF THEN DO Statement**

**data** a;

input name $ age sal;

cards;

kk 34 1000

jj 23 3000

;

**run**;

**proc** **print** data=a;

**run**;

**data** c;

set a;

if sal>**1000** then do;

a=sal\***100**;

end;

**run**;

**data** b;

input item $ cost saledate mmddyy8.;

format saledate date9.;

cards;

10e 1000 12/01/09

10r 2000 11/20/08

;

**run**;

**data** d;

set b;

if cost=**1000** then do;

sale=cost\***2**;

end;

else if cost=**2000** then do;

sale1=cost\***3**;

end;

**run**;

**proc** **print** data=d;

**run**;

**data** b;

input item $ cost saledate mmddyy8.;

format saledate date9.;

cards;

10e 1000 12/01/09

10r 2000 11/20/08

10a 3000 05/20/12

10b 4000 06/21/14

;

**run**;

**data** d;

set b;

if cost=**1000** then do;

sale=cost\***2**;

end;

else if cost=**2000** then do;

sale1=cost\***3**;

end;

else sale1=cost\***4**;

**run**;

**proc** **print** data=d;

**run**;

**data** b;

input item $ cost saledate mmddyy8.;

format saledate date9.;

if cost=**1000** then do;

sale=cost\***2**;

end;

else if cost=**2000** then do;

sale1=cost\***3**;

end;

cards;

10e 1000 12/01/09

10r 2000 11/20/08

;

**run**;

**proc** **print** data=b;

**run**;

**KEEP AND DROP**

The KEEP statement causes a DATA step to write only the variables that you specify to one or more SAS data sets. The KEEP statement applies to all SAS data sets that are created within the same DATA step and can appear anywhere in the step. If no KEEP or DROP statement appears, all data sets that are created in the DATA step contain all variables

* The KEEP **statement** cannot be used in SAS PROC steps. The KEEP= **data set option** can.
* The KEEP **statement** applies to all output data sets that are named in the DATA statement. To write different variables to different data sets, you must use the KEEP= **data set option**.
* The DROP statement is a parallel statement that specifies variables to omit from the output data set.
* The KEEP and DROP statements select variables to include in or exclude from output data sets. The subsetting IF statement selects observations.
* Do not confuse the KEEP statement with the RETAIN statement. The RETAIN statement causes SAS to hold the value of a variable from one iteration of the DATA step to the next iteration. The KEEP statement does not affect the value of variables but only specifies which variables to include in any output data sets.

**DATA scor;**

**INPUT nm $ score1 score2 score3 score4;**

**DATALINES;**

**kk 10 20 30 40**

**ll 20 30 40 50**

**mm 30 40 50 60**

**;**

**run;**

**data average;**

**set scor;**

**keep nm score1;**

**run;**

**data average1;**

**set scor;**

**drop score1;**

**run;**

**data** average1;

keep name avg;

INPUT name $ score1 score2 score3 score4;

avg=mean(of score1-score4);

DATALINES;

kk 10 20 30 40

ll 20 30 40 50

mm 30 40 50 60

;

**run**;

**DATA** scor;

INPUT nm $ score1 score2 score3 score4;

DATALINES;

kk 10 20 30 40

ll 20 30 40 50

mm 30 40 50 60

;

**run**;

**data** average;

set scor;

keep nm avg;

avg=mean(of score1-score4);

**run**;

**data** average;

set scor;

drop nm avg;

avg=mean(of score1-score4);

**run**;

**SUM**

variable + expression;

is called a sum statement; it adds the result of the expression on the right side of the plus sign to the variable on the left side of the plus sign and holds the new value of variable for use in subsequent observations. The expression can be a numeric variable or expression. The value of variable is retained. If the expression is a missing value, the variable maintains its previous value. Before the sum statement is executed for the first time, the default value of the variable is 0.

**data** mydata1;

input x y z;

cards;

33 3 3

24 3 4

24 3 4

;

**run**;

**data** mydata3;

set mydata1;

p=x+y+z;

**run**;

data mydata;  
input x y z;  
cards;  
33 3 3  
24 3 4  
24 3 4  
. 3 2  
23 . 3  
54 4 .  
35 4 2  
;  
run;

data mydata2;  
set mydata;  
a=sum(x,y,z);  
p=x+y+z;  
run;

options linesize=**80** pageno=**1** nodate;

**data** qtr2;

input Month qtr SalesRep $ type $ Units AmountSold;

cards;

01 1 Hollingsworth Deluxe 260 49.50

01 1 Garcia Standard 41 30.97

01 1 Hollingsworth Deluxe 330 49.50

01 1 Jensen Standard 1110 30.97

01 1 Garcia Standard 715 30.97

01 1 Jensen Deluxe 675 49.50

02 1 Jensen Standard 45 30.97

02 1 Garcia Deluxe 10 49.50

12 4 Hollingsworth Deluxe 125 49.50

12 4 Jensen Standard 1254 30.97

12 4 Hollingsworth Deluxe 175 49.50

;

**run**;

**proc** **print** data=qtr2;

var Units AmountSold;

where Units>**500** or AmountSold>**20000**;

format Units comma7. AmountSold dollar14.2;

sum Units AmountSold;

by SalesRep Month;

id SalesRep Month;

sumby SalesRep;

title1 'Sales Rep Quarterly Totals for Sales above 500 Units or $20,000';

**run**;

The EUROw.d formats and informats were introduced to support the euro currency that was established by the European Monetary Union (EMU), which was formed in 1999. EUROw.d

* uses the euro (e) currency symbol to precede Euro currency data
* uses a comma (,) as the thousands separator and a dot (.) s the decimal separator

options locale=English\_UnitedKingdom;

x=12345;

put x euro10.2;

run;

**RETAIN**

/\*The RETAIN statement simply copies retaining values by telling the SAS not to reset the variables to missing at the beginning of each iteration of the DATA step.

If you would not use retain statement then SAS would return missing at the beginning of each iteration.\*/

**data** abcd;

input x y;

cards;

1 25

1 28

1 27

2 23

2 35

2 34

3 25

3 29

;

**run**;

**data** aaa;

set abcd;

retain z **0**;

z = z + **1**;

**run**;

**data** aaa;

set abcd;

z + **1**;

**run**;

**data** aaa;

set abcd;

retain z **0**;

z = z + y;

**run**;

**data** aaa;

set abcd;

a=z + y;

**run**;

**proc** **sort** data = abcd;

by x;

**run**;

**data** aaa;

set abcd;

retain z;

if first.x then z = **1**;

else z = z + **1**;

by x;

**run**;

**data** aaa1;

set aaa;

retain z1;

if first.x then z1 = y;

else z1 = z1 + y;

by x;

**run**;

**data** aaa2;

set abcd (drop = y);

retain z;

if first.x then z = **1**;

else z = z + **1**;

by x;

if last.x then output;

**run**;

**data** temp;

input ID ID1 Score;

cards;

1 1 25

1 1 26

1 2 27

1 2 29

2 1 28

2 1 29

2 2 31

;

**run**;

**data** temp2;

set temp;

by ID ID1;

if first.ID or first.ID1 then N = **1**;

else N+**1**;

**proc** **print**;

**run**;

**RETAIN statement**

The RETAIN statement “Causes a variable that is created by an INPUT or assignment statement to retain its value from one

iteration of the DATA step to the next”1

This is in contrast to the default DATA step behavior, which is, “Without a RETAIN

statement, SAS automatically sets variables that are assigned values by an INPUT or assignment statement to missing before

each iteration of the DATA step.” 1

**DATA** intial\_cl;

INPUT account credit\_limit;

DATALINES;

1002 2000

1003 4000

1004 3000

;

**run**;

**proc** **print** data=intial\_cl;

**run**;

**data** b;

set intial\_cl;

retain total **10**;

total=account+credit\_limit;

**run**;

**proc** **print** data=b;

**run**;

**data** b;

retain CREDIT\_LIMIT account;

set intial\_cl;

retain total **10**;

total=account+credit\_limit;

**run**;

**proc** **print** data=b;

**run**;

|  |  |  |  |
| --- | --- | --- | --- |
| ***The Value of TESTNUM Using Different INFILE Statement Options*** | | | |
| **OBS** | **FLOWOVER** | **MISSOVER** | **TRUNCOVER** |
| 1 | 22 | . | 1 |
| 2 | 4444 | . | 22 |
| 3 | 55555 | . | 333 |
| 4 |  | . | 4444 |
| 5 |  | 55555 | 55555 |