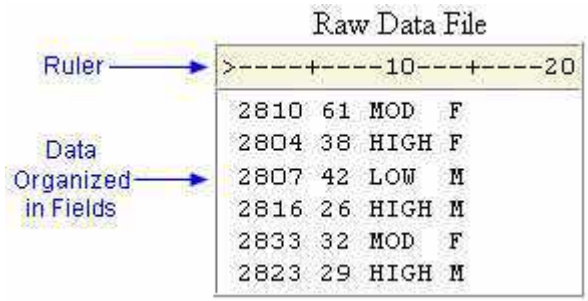
Chapter 3: Creating SAS data sets from external files

Objectives : learning about
- reference a raw data file
- name a SAS data set to be created
- specify a raw data file to be read
- read standard character and numeric values in fixed fields
- create new variables and assign values
- select observations based on conditions
- read a SAS data set and write the observations out to a raw data file.
- read instream data
- use the IMPORT procedure to read external files

Raw Data Files

A raw data file is an external text file whose records contain data values that are organized in fields. Raw data files are non-proprietary and can be read by a variety of software programs.




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

cf) noobs is used to remove the Obs column.

Steps to Create a SAS Data Set

Before reading the raw data from the file, you must first reference the SAS library in which you will store the data set. Then you can write a DATA step program to read the raw data file and create a SAS data set.

To read the raw data file, the DATA step must provide the following instructions to SAS:
- the location or name of the external text file
- a name for the new SAS data set
- a reference that identifies the external file
- a description of the data values to be read.

After using the DATA step to read the raw data, you can use a PROC PRINT step to produce a list report that displays the data values that are in the new data set.

The following FILENAME statement temporarily associates the fileref Tests with the external file that contains the data from the exercise stress tests. The complete filename is specified as C:\Users\Tmill.dat in the Windows environment.

```
filename tests 'c:₩users₩tmill.dat';
```

When reading raw data, use the INFILE statement to indicate which file the data is in. To read the raw data file to which the fileref Tests has been assigned, you write the following INFILE statement:

```
infile tests;
```

Note Instead of using a FILENAME statement, you can choose to identify the raw data file by specifying the entire filename and location in the INFILE statement. For example, the following statement points directly to the 'c:\users\tmill.dat' file:

```
iinfile 'c:₩users₩tmill.dat';
```

■ Column input

Column input specifies actual column locations for values. However, column input is appropriate only in certain situations. When you use column input, your data must be
- standard character or numeric values
- in fixed fields.

■ Standard and Nonstandard Numeric Data

Standard numeric data values can contain only
- numbers
- decimal points
- numbers in scientific or E-notation (2.3E4, for example)
- plus or minus signs.

Nonstandard numeric data includes
-values that contain special characters, such as percent signs (%), dollar signs ($), and commas (,)
- date and time values
- data in fraction, integer binary, real binary, and hexadecimal forms.

The external file that is referenced by the fileref Staff contains the personnel information for a technical writing department of a small computer manufacturer. The fields contain values for each employee's last name, first name, job title, and annual salary. Notice that the values for Salary contain commas. The values for Salary are considered to be nonstandard numeric values. You cannot use column input to read these values.

Raw Data File Staff

```
1---+----10---+----20---+---
EVANS    DONNY 112 29,996.63
HELMS    LISA  105 18,567.23
HIGGINS  JOHN  111 25,309.00
LARSON   AMY   113 32,696.78
MOORE    MARY  112 28,945.89
```

■ Fixed-Field Data

Raw data can be organized in several different ways.

This external file contains data that is in free format, meaning data that is not arranged in columns. Notice that the values for a particular field do not begin and end in the same columns. You cannot use column input to read this file.

```
1---+----10---+----20
BARNES NORTH 360.98
FARLSON WEST 243.94
LAWRENCE NORTH 195.04
NELSON EAST 169.30
STEWART SOUTH 238.45
TAYLOR WEST 318.87
```

This external file contains data that is arranged in columns or fixed fields. You can specify a beginning and ending column for each field. Let's look at how column input can be used to read this data.

```
1---+----10---+----20
2810 61 MOD  F
2804 38 HIGH F
2807 42 LOW  M
2816 26 HIGH M
2833 32 MOD  F
2823 29 HIGH M
```

The INPUT statement below assigns the character variable ID to the data in columns 1--4, the numeric variable Age to the data in columns 6--7, the character variable ActLevel to the data in columns 9--12, and the character variable Sex to the data in column 14.

```
filename exer 'c:₩users₩exer.dat';
data exercise;
        infile exer;
        input ID $ 1-4 Age 6-7 ActLevel $ 9-12 Sex $ 14;
run;
```

When you use column input, you can
- read any or all fields from the raw data file
- read the fields in any order
- specify only the starting column for values that occupy only one column.

```
input ActLevel $ 9-12 Sex $ 14 Age 6-7;
```

Submitting the DATA Step Program

The program below reads the raw data file referenced by the fileref Tests. The data is stored in a permanent SAS data set named Clinic.Stress. Don't forget a RUN statement, which tells SAS to execute the previous SAS statements.

```
data clinic.stress;
        infile tests obs=10;
        input ID 1-4 Name $ 6-25
        RestHR 27-29 MaxHR 31-33
        RecHR 35-37 TimeMin 39-40
        TimeSec 42-43 Tolerance $ 45;
run;
proc print data=clinic.stress;
run;
```

■ Invalid data

When you submit the DATA step and check the log, you could see a note indicating that invalid data appears for some variable. This note is followed by a column ruler and the actual data line that contains the invalid value for that variable.

Creating and Modifying Variables

sometimes existing data doesn't provide the information that you need. To modify existing values or to create new variables, you can use an assignment statement in any DATA step.
ex] Name='Toby Witherspoon';

■ SAS expressions

You use SAS expressions in assignment statements and many other SAS

programming statements to
- transform variables
- create new variables
- conditionally process variables
- calculate new values
- assign new values.

An expression is a sequence of operands and operators that form a set of instructions. The instructions are performed to produce a new value:

- operators in SAS expressions
: - (negative), ** (exponentiation), * (multiplication), / (division), + (addition), - (substraction)
- comparison operators
: =, ^=, >, <, >=, <=
- logical operators
: &, |

■ More examples of assigment statements

The assignment statement in the DATA step below creates a new variable, TotalTime, by multiplying the values of TimeMin by 60 and then adding the values of TimeSec.

```
data clinic.stress;
        infile tests;
        input ID 1-4 Name $ 6-25 RestHr 27-29 MaxHR 31-33
        RecHR 35-37 TimeMin 39-40 TimeSec 42-43
        Tolerance $ 45;
        TotalTime=(timemin*60)+timesec;
run;
```

The expression can also contain the variable name that is on the left side of the equal sign, as the following assignment statement shows. This statement re-defines the values of the variable RestHR as 10 percent higher.

```
data clinic.stress;
        infile tests;
        input ID 1-4 Name $ 6-25 RestHr 27-29 MaxHR 31-33
        RecHR 35-37 TimeMin 39-40 TimeSec 42-43
        Tolerance $ 45;
        resthr=resthr+(resthr*.10);
run;
```

When a variable name appears on both sides of the equal sign, the original value on the right side is used to evaluate the expression. The result is assigned to the variable on the left side of the equal sign.

■ Date constants

You can assign date values to variables in assignment statements by using date constants. To represent a constant in SAS date form, specify the date as 'ddmmmyy' or 'ddmmmyyyy', followed by a D.

In the following program, the second assignment statement assigns a date value to the variable

```
data clinic.stress;
        infile tests;
        input ID 1-4 Name $ 6-25 RestHr 27-29 MaxHR 31-33
        RecHR 35-37 TimeMin 39-40 TimeSec 42-43
        Tolerance $ 45;
        TotalTime=(timemin*60)+timesec;
        TestDate='01jan2000'd;
run;
```

Subsetting data

As you read your data, you can subset it by processing only those observations that meet a specified condition. To do this, you can use a subsetting IF statement in any DATA step.

The subsetting IF statement causes the DATA step to continue processing only those raw data records or observations that meet the condition of the expression specified in the IF statement. The resulting SAS data set or data sets contain a subset of the original external file or SAS data set.

For example, the subsetting IF statement below selects only observations whose values for Tolerance are D. The IF statement is positioned in the DATA step so that other statements do not need to process unwanted observations.

```
data clinic.stress;
        infile tests;
        input ID 1-4 Name $ 6-25 RestHr 27-29 MaxHR 31-33
        RecHR 35-37 TimeMin 39-40 TimeSec 42-43
        Tolerance $ 45;
        if tolerance='D';
        TotalTime=(timemin*60)+timesec;
run;
```

Reading Instream Data

you can also read instream data lines that you enter directly in your SAS program, rather than data that is stored in an external file. Reading instream data is extremely helpful if you want to create data and test your programming statements on a few observations that you can specify according to your needs.

To read the data for the treadmill stress tests as instream data, you can submit the following program:

```
data stress;
        input ID 1-4 Name $ 6-25 RestHr 27-29 MaxHR 31-33
        RecHR 35-37 TimeMin 39-40 TimeSec 42-43
        Tolerance $ 45
        if tolerance='D';
        TotalTime=(timemin*60)+timesec;
datalines;
2458 Murray, W              72 185 128 12 38 D
2462 Almers, C              68 171 133 10  5 I
2501 Bonaventure, T         78 177 139 11 13 I
2523 Johnson, R             69 162 114  9 42 S
2539 LaMance, K             75 168 141 11 46 D
2544 Jones, M               79 187 136 12 26 N
2552 Reberson, P            69 158 139 15 41 D
2555 King, E                70 167 122 13 13 I
2563 Pitts, D               71 159 116 10 22 S
2568 Eberhardt, S           72 182 122 16 49 N
2571 Nunnelly, A            65 181 141 15  2 I
2572 Oberon, M              74 177 138 12 11 D
2574 Peterson, V            80 164 137 14  9 D
2575 Quigley, M             74 152 113 11 26 I
2578 Cameron, L             75 158 108 14 27 I
2579 Underwood, K           72 165 127 13 19 S
2584 Takahashi, Y           76 163 135 16  7 D
2586 Derber, B              68 176 119 17 35 N
2588 Ivan, H                70 182 126 15 41 N
2589 Wilcox, E              78 189 138 14 57 I
2595 Warren, C              77 170 136 12 10 S
;
```

cf) Notice that you do not need a RUN statement here.

Create a Raw Data File

If you wanted to write a SAS data set to a raw data file, you could reverse the

process that you've been following and write out the observations from a SAS data set as records or lines to a new raw data file.

The following program creates a new raw date file named 'stress' from the SAS data set 'clinic.stress'.

```
data _null_;
        set clinic.stress;
        file 'c:₩clinic₩patients₩stress.dat';
        put id 1-4 name 6-25 resthr 27-29 maxhr 31-33
        rechr 35-37 timemin 39-40 timesec 42-43
        tolerance 45 totaltime 47-49;
run;
```

Reading Microsoft Excel Data

Use IMPORT wizard!

File => Import Data from the menu bar.

sas-code

```
filename exer 'C:\Users\ADMIN\Copy\HUFS\lecture\2016-1 SASstat\exer.dat';
data exercise;
        infile exer;
        input ID $ 1-4 Age 6-7 ActLevel $ 9-12 Sex $ 14;
run;


data exercise1;
        infile  'C:\Users\ADMIN\Copy\HUFS\lecture\2016-1 SASstat\exer.dat';
        input ID $ 1-4 Age 6-7 ActLevel $ 9-12 Sex $ 14;
run;

filename tests 'C:\Users\ADMIN\Copy\HUFS\lecture\2016-1 SASstat\stress.dat';
data clinic.stress;
infile tests;
input ID $ 1-4 Name $ 6-25
RestHR 27-29 MaxHR 31-33
RecHR 35-37 TimeMin 39-40
TimeSec 42-43 Tolerance $ 45;
run;
proc print data=clinic.stress;
run;

data clinic.stress;
        infile tests;
        input ID 1-4 Name $ 6-25 RestHr 27-29 MaxHR 31-33
        RecHR 35-37 TimeMin 39-40 TimeSec 42-43
        Tolerance $ 45;
        TotalTime=(timemin*60)+timesec;
run;

data clinic.stress;
infile tests;
input ID 1-4 Name $ 6-25 RestHr 27-29 MaxHR 31-33
RecHR 35-37 TimeMin 39-40 TimeSec 42-43
Tolerance $ 45;
resthr=resthr+(resthr*.10);
run;

data clinic.stress;
        infile tests;
        input ID 1-4 Name $ 6-25 RestHr 27-29 MaxHR 31-33
        RecHR 35-37 TimeMin 39-40 TimeSec 42-43
        Tolerance $ 45;
        TotalTime=(timemin*60)+timesec;
        TestDate='01jan2000'd;
run;

proc print data=clinic.stress;
        format testdate date9.;
run;

data clinic.stress;
        infile tests;
        input ID 1-4 Name $ 6-25 RestHr 27-29 MaxHR 31-33
        RecHR 35-37 TimeMin 39-40 TimeSec 42-43
        Tolerance $ 45;
        if tolerance='D';
        TotalTime=(timemin*60)+timesec;
run;

data stress;
```

```
input ID 1-4 Name $ 6-25 RestHr 27-29 MaxHR 31-33
RecHR 35-37 TimeMin 39-40 TimeSec 42-43
Tolerance $ 45;
if tolerance='D';
TotalTime=(timemin*60)+timesec;
datalines;
2458 Murray, W               72 185 128 12 38 D
2462 Almers, C               68 171 133 10  5 I
2501 Bonaventure, T          78 177 139 11 13 I
2523 Johnson, R              69 162 114  9 42 S
2539 LaMance, K              75 168 141 11 46 D
2544 Jones, M                79 187 136 12 26 N
2552 Reberson, P             69 158 139 15 41 D
2555 King, E                 70 167 122 13 13 I
2563 Pitts, D                71 159 116 10 22 S
2568 Eberhardt, S            72 182 122 16 49 N
2571 Nunnelly, A             65 181 141 15  2 I
2572 Oberon, M               74 177 138 12 11 D
2574 Peterson, V             80 164 137 14  9 D
2575 Quigley, M              74 152 113 11 26 I
2578 Cameron, L              75 158 108 14 27 I
2579 Underwood, K            72 165 127 13 19 S
2584 Takahashi, Y            76 163 135 16  7 D
2586 Derber, B               68 176 119 17 35 N
2588 Ivan, H                 70 182 126 15 41 N
2589 Wilcox, E               78 189 138 14 57 I
2595 Warren, C               77 170 136 12 10 S
;

data _null_;
        set clinic.stress;
        file 'c:\clinic\patients\stress.dat';
        put id 1-4 name 6-25 resthr 27-29 maxhr 31-33
        rechr 35-37 timemin 39-40 timesec 42-43
        tolerance 45 totaltime 47-49;
run;
```