MARK CARPENTER, Ph.D.

Module 3 : DATA STEP OPTIONS/STATEMENTS

Description. In this module, we cover in greater detail, using special options, all SAS commands and statements covered in Module 1. For example, we provide tools to deal with missing data in the input file, use the DELIMITER option in the INFILE statement to import a comma delimited file and show how to import different/non-standard data structures.

SAS Keywords : KEEP, DROP, RENAME, IF, WHERE, FIRSTOBS, OBS, MISSOVER, @@, DELIMITER, ‘:’

New Topics : Missing Data, delimited files, inputs of variable length, colon modifier, DSD, more data informats/formats, creating variables in SAS data set or during a data step (assignment statement)

New PROCs : SORT, TRANSPOSE, DATASTEP, GCHART
Shared Language Statements
The following Base SAS language statements are also part of SAS/GRAPH:

- **FORMAT statement**
  assigns a format to a variable. SAS/GRAPH procedures use formatted values to determine such aspects of the graph as midpoints, axis labels, tick mark values, and legend entries.

- **FILENAME**
  associates a SAS fileref with an external text file or output device. See FILENAME Statement for a more detailed description of this statement.

- **RUN statement**
  see line 7 in Figure: Typical SAS Program
  executes the statements in the PROC step.

- **LABEL statement**
  assigns a descriptive text string (a “label”) to a variable. The label appears in place of the variable name on the axis and legend.

- **LIBNAME**
  associates a libref with a SAS library. See LIBNAME Statement for a more detailed description of this statement.

- **ODS statements**
  control the output of SAS/GRAPH procedures, where the output is sent (destination), the appearance of the output (STYLE=), and the output file type (DEVICE=). See Getting Started With SAS/GRAPH for information about using ODS with SAS/GRAPH procedures.

- **OPTIONS statement**
  changes the value of one or more SAS system options.

- **QUIT statement**
  see line 8 in Figure: Typical SAS Program
  executes any statements that have not executed and ends the procedure. It also ends a procedure that is using RUN-GROUP processing.

- **WHERE statement**
  specifies observations from SAS data sets that meet a particular condition. You can use a WHERE statement to easily subset your data.
DATA Options, Subset Observations and Assignment Statements

In this section, we explore various options and statements that can be used in a DATA Step. These statements can produce datasets that

• are subsets of the original set of observations (IF, WHERE),

• contains fewer variables than original set (KEEP/DROP), and

• contains new variables produced by assignment statements, as well as other options.
TYPES OF DATA STEP STATEMENTS

• Declarative versus Executable Statements/options
• Overview of Data Step Statements/Options
• Data Step Statements versus Options
  ➢ Statements/Options that act on Variables (DROP, KEEP, RENAME)
  ➢ Statements/Options that subset Observations (IF/WHERE/FIRSTOBS/OBS)
• Other Important Data Step Statements
  ➢ Assignment Statements and SAS Expressions
  ➢ SAS Expression Operators
  ➢ SAS Functions
DATA step statements are executable or declarative statements that can appear in the DATA step.

- **Executable statements** result in some action during individual iterations of the DATA step;

- **Declarative statements** supply information to SAS and take effect when the system compiles program statements.

The following tables show the SAS executable and declarative statements that you can use in the DATA step.
**Executable Statements** result in some action during individual iterations of the DATA step;

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABORT</td>
<td>IF, Subsetting</td>
<td>PUT, Column</td>
</tr>
<tr>
<td>Array Reference</td>
<td>IF-THEN/ELSE</td>
<td>PUT, Formatted</td>
</tr>
<tr>
<td>Assignment</td>
<td>INFILE</td>
<td>PUT, List</td>
</tr>
<tr>
<td>CALL</td>
<td>INPUT</td>
<td>PUT, Named</td>
</tr>
<tr>
<td>CONTINUE</td>
<td>GO TO</td>
<td>PUT</td>
</tr>
<tr>
<td>DECLARE</td>
<td>INPUT, Column</td>
<td>PUT, ODS</td>
</tr>
<tr>
<td>DELETE</td>
<td>INPUT, Formatted</td>
<td>PUTLOG</td>
</tr>
<tr>
<td>DESCRIBE</td>
<td>INPUT, List</td>
<td>REDIRECT</td>
</tr>
<tr>
<td>DISPLAY</td>
<td>INPUT, Named</td>
<td>REMOVE</td>
</tr>
<tr>
<td>DO</td>
<td>LEAVE</td>
<td>REPLACE</td>
</tr>
<tr>
<td>DO, Iterative</td>
<td>LINK</td>
<td>RESETLINE</td>
</tr>
<tr>
<td>DO UNTIL</td>
<td>LIST</td>
<td>RETURN</td>
</tr>
<tr>
<td>DO WHILE</td>
<td>LOSTCARD</td>
<td>SELECT</td>
</tr>
<tr>
<td>ERROR</td>
<td>MERGE</td>
<td>SET</td>
</tr>
<tr>
<td>EXECUTE</td>
<td>MODIFY</td>
<td>STOP</td>
</tr>
<tr>
<td>FILE</td>
<td>Null</td>
<td>Sum</td>
</tr>
<tr>
<td>FILE, ODS</td>
<td>OUTPUT</td>
<td>UPDATE</td>
</tr>
</tbody>
</table>
Declarative Statements supply information to SAS and take effect when the system compiles program statements

<table>
<thead>
<tr>
<th>Declarative Statement</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARRAY</td>
<td>DATALINES4</td>
<td>Labels, Statement</td>
</tr>
<tr>
<td>ATTRIB</td>
<td>DROP</td>
<td>LENGTH</td>
</tr>
<tr>
<td>BY</td>
<td>END</td>
<td>RENAME</td>
</tr>
<tr>
<td>CARDS</td>
<td>FORMAT</td>
<td>RETAIN</td>
</tr>
<tr>
<td>CARDS4</td>
<td>INFORMAT</td>
<td>WHERE</td>
</tr>
<tr>
<td>DATA</td>
<td>KEEP</td>
<td>WINDOW</td>
</tr>
<tr>
<td>DATALINES</td>
<td>LABEL</td>
<td></td>
</tr>
</tbody>
</table>
DATA STEP OPTIONS AND STATEMENTS

DATA STEP LAYOUT

Revised Form 1:
DATA <data-set-name>(data-set-options)>;
statement-1;
.
.
.
statement-n;
RUN;

-(data-set-options) specifies optional arguments that the DATA step applies when it writes observations to the output.

SAS Statements can include, INFILE, INPUT, SET, MERGE, FORMAT statements, for example, as well as, assignment statements and statements that can be used to invoke the options described above (WHERE, DROP, KEEP, etc).

A DATA STEP is a group of SAS language statements that begin with a DATA statement. The group of language statements contains other programming statements that manipulate existing SAS data sets or create SAS data sets from raw data files.
OPTIONS/STATEMENTS TO BE COVERED HERE:

- **DROP**= Data Set Option: For an input data set, excludes the specified variables from processing; for an output data set, excludes the specified variables from being written to the data set.

- **KEEP**= Data Set Option: For an input data set, specifies the variables to process; for an output data set, specifies the variables to write to the data set.

- **RENAME**= Data Set Option: Changes the name of a variable

- **FIRSTOBS**= Data Set Option: Specifies the first observation that SAS processes in a SAS data set.

- **OBS**= Data Set Option: Specifies the last observation that SAS processes in a data set.

- **WHERE**= Data Set Option: Specifies specific conditions to use to select observations from a SAS data set.
Statements Versus Options
Although, the statements and options can produce the same result in the completed data step, the methods differ in how and/or when the variables are processed. The main differences are described below:

- **Data Step Options:** Work on both input and output variables. Controls how variables and observations get processed during input and output stages of the DATA Step.

- **Data Step Statements:** Work only on output variables. Controls which variables get written to the output data set. For example, if you used a RENAME statement to rename a variable, the new name doesn’t take effect until the DATA Step is completed, i.e., new name appears in the output dataset only.
VARIBALES, OBSERVATIONS, ASSIGNMENTS

We will highlight three topics:

• drop, keep, or rename variables in the output data set resulting from a DATA step.

• creating data sets whose observations represent subsets of the set of original observations, using IF and WHERE clauses.

• assignment statements, operations, expressions and functions (built-in SAS functions).
VARIABLES, OBSERVATIONS, ASSIGNMENTS

We will highlight three topics:

- drop, keep, or rename variables in the output data set resulting from a DATA step.

- creating data sets whose observations represent subsets of the set of original observations, using IF and WHERE clauses.

- assignment statements, operations, expressions and functions (built-in SAS functions).
Module 3.1. Dropping, Keeping or Renaming Variables in a DATA Step

Some of the options that we will examine today are the relate to dropping, keeping or renaming variables in the output data set. This can be done using the DROP, KEEP and RENAME statements in a DATA Step. Alternatively, the DROP, KEEP and RENAME statements can be used as data-set-options, as indicated above.

- **Data Step Statements**: Works only on output variables. Controls with variables get written to the output data set.

  DROP variable-list;
  KEEP variable-list;
  RENAME = (old-name-1=new-name-1 <old-name-n=new-name-n>);

- **Data Step Options**: Works on both input and output variables

  DROP = variable-list;
  KEEP = variable-list;
  RENAME = (old-name-1=new-name-1 <old-name-n=new-name-n>)
THE DROP STATEMENT

**DROP Statement**
Excludes variables from output SAS data sets.

<table>
<thead>
<tr>
<th>Valid in:</th>
<th>DATA step</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category:</td>
<td>Information</td>
</tr>
<tr>
<td>Type:</td>
<td>Declarative</td>
</tr>
</tbody>
</table>

**Syntax**
DROP `variable-list`;

**Details**
The DROP statement applies to all the SAS data sets that are created within the same DATA step and can appear anywhere in the step. The variables in the DROP statement are available for processing in the DATA step. If no DROP or KEEP statement appears, all data sets that are created in the DATA step contain all variables. Do not use both DROP and KEEP statements within the same DATA step.
DROP STATEMENTS

EXAMPLE 1: The Following data set Example1 will be used for Examples 1 a, b, c

```sas
DATA example1;
   INFORMAT purchase dollar7. repair dollar7.;
   INPUT shift batchnum count1-count5 purchase repair;
DATALINES;
  1 1 15 20 12 4 12 $10,000 $20,000
  2 1 12 12 15 3 22 $5,000  $50,000
  1 2 10  9 11 7 22 $9,000  $8,000
  2 2 11 15 11 2 22 $6,000  $5,000
;```

DROP STATEMENTS

EXAMPLE 1 (a): Create a data set Data1a from example1 and drop the following variables: shift, batch, purchase and repair. Only count1-count5 will be written to data1a.

```
DATA data1a;
  SET example1;
  DROP shift batchnum purchase repair;
RUN;

PROC PRINT DATA=data1a NOOBS;
  VAR count1-count5;
RUN;
```
DROP STATEMENTS

EXAMPLE 1 (b): Create a data set Data1b from example1 and drop the following variables: count1-count5, purchase and repair. Only shift, batchnum and totcost will be written to the output data set Data1b.

```sas
DATA data1b;
  SET example1;
  DROP count1-count5 purchase repair;
RUN;

PROC PRINT DATA=data1b NOOBS;
  VAR shift batchnum totcost;
RUN;
```
DROP STATEMENTS

EXAMPLE 1 (c): Create a data set Data1c from example1 and drop the following variables: count1-count5, purchase and repair. In this example, the variables PURCHASE and REPAIR are used in processing but are not written to the output data set INVENTORY:

```sas
DATA data1c;
    SET example1;
    DROP purchase repair count1-count5;
    totcost=sum(purchase,repair);
RUN;

PROC PRINT DATA=data1c NOOBS;
    VAR shift batchnum totcost;
    FORMAT totcost dollar8.;
RUN;
```
THE DROP= DATA-Step-Option

DROP= Data Set Option
For an input data set, excludes the specified variables from processing; for an output data set, excludes the specified variables from being written to the data set.

Valid in: DATA step and PROC steps
Category: Variable Control

DROP= data set option differs from the DROP statement in these ways:

• In DATA steps, the DROP= data set option can apply to both input and output data sets. The DROP statement applies only to output data sets.

• In DATA steps, when you create multiple output data sets, use the DROP= data set option to write different variables to different data sets. The DROP statement applies to all output data sets.

• In PROC steps, you can use only the DROP= data set option, not the DROP statement.
Example 1: Excluding Variables from Input. In this example, the variables SALARY and GENDER are not included in processing and they are not written to either output data set:

```
DATA plan1 plan2;
  SET payroll(drop=salary gender);
  IF hired<>'01jan98'd THEN OUTPUT plan1;
  ELSE OUTPUT plan2;
RUN;
```

You cannot use SALARY or GENDER in any logic in the DATA step because DROP= prevents the SET statement from reading them from PAYROLL.
Example 2: Processing Variables without Writing Them to a Data Set. In this example, SALARY and GENDER are not written to PLAN2, but they are written to PLAN1.

```
DATA plan1 plan2(drop=salary gender);
  SET payroll;
  IF hired < '01jan98'd THEN OUTPUT plan1;
  ELSE OUTPUT plan2;
RUN;
```
**KEEP STATEMENTS**

**KEEP statement**
Specifies the variables to include in output SAS data sets.

<table>
<thead>
<tr>
<th>Valid in:</th>
<th>DATA step</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category:</td>
<td>Information</td>
</tr>
<tr>
<td>Type:</td>
<td>Declarative</td>
</tr>
</tbody>
</table>

**Syntax**

KEEP `variable-list`;

**Arguments**

`variable-list`

specifies the names of the variables to write to the output data set.

**Details**

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.

**Note:** Do not use both the KEEP and DROP statements within the same DATA step.
KEEP STATEMENT COMPARISONS

• 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.
Example 1: KEEP Statement Basic Usage
These examples show the correct syntax for listing variables in the KEEP statement:

```sas
KEEP name address city state zip phone;
KEEP repl1-repl5;
```
Example 2: Keeping Variables in the Data Set
This example uses the KEEP statement to include only the variables NAME and AVG in the output data set. The variables SCORE1 through SCORE20, from which AVG is calculated, are not written to the data set AVERAGE.

```
DATA average;
   KEEP name avg;
   INFILE file-specification;
   INPUT name $ score1-score20;
   avg=mean(of score1-score20);
RUN;
```
**KEEP = Data-Step-Options**

**KEEP Data Step Option**
For an input data set, specifies the variables to process; for an output data set, specifies the variables to write to the data set.

<table>
<thead>
<tr>
<th>Valid in:</th>
<th>DATA step and PROC steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category:</td>
<td>Variable Control</td>
</tr>
</tbody>
</table>

**Syntax**

**KEEP=variable-1 <...variable-n>**

*Syntax Description*

`variable-1 <...variable-n>`
list one or more variable names. You can list the variables in any form that SAS allows.

**Details**

If the KEEP= data set option is associated with an input data set, only those variables that are listed after the KEEP= data set option are available for processing. If the KEEP= data set option is associated with an output data set, only the variables listed after the option are written to the output data set. All variables are available for processing.
**KEEP = Data-Step-Options Comparisons**

The KEEP= data set option differs from the KEEP statement in the following ways:

- In DATA steps, the KEEP= data set option can apply to both input and output data sets. The KEEP statement applies only to output data sets.
- In DATA steps, when you create multiple output data sets, use the KEEP= data set option to write different variables to different data sets. The KEEP statement applies to all output data sets.
- In PROC steps, you can use only the KEEP= data set option, not the KEEP statement.
- The DROP= data set option specifies variables to omit during processing or to omit from the output data set.
Example 3: In this example, only IDNUM and SALARY are read from PAYROLL, and they are the only variables in PAYROLL that are available for processing.

```
DATA bonus;
  SET payroll(keep=idnum salary);
  bonus=salary*1.1;
RUN;
```
RENAME STATEMENTS

RENAME Statement
Specifies new names for variables in output SAS data sets.

<table>
<thead>
<tr>
<th>Valid in:</th>
<th>DATA step</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category:</td>
<td>Information</td>
</tr>
<tr>
<td>Type:</td>
<td>Declarative</td>
</tr>
</tbody>
</table>

Syntax
RENAME *old-name-1* = *new-name-1* ... <
*old-name-n* = *new-name-n*

Details
The RENAME statement allows you to change the names of one or more variables, variables in a list, or a combination of variables and variable lists. The new variable names are written to the output data set only. Use the old variable names in programming statements for the current DATA step. RENAME applies to all output data sets.

Note: The RENAME statement has an effect on data sets opened in output mode only.
RENAME STATEMENTS

Example: These examples show the correct syntax for renaming variables using the RENAME statement

```sas
rename street=address;
rename timel=temp1 time2=temp2 time3=temp3;
rename name=Firstname score1-score3=Newscore1-Newscore3;
```

Example uses the old name of the variable in program statements. The variable Olddept is named Newdept in the output data set, and the variable Oldaccount is named Newaccount.

```sas
rename Olddept=Newdept Oldaccount=Newaccount;
if Oldaccount>5000;
keep Olddept Oldaccount items volume;
```
Comparisons

• RENAME cannot be used in PROC steps, but the RENAME= data set option can.
• The RENAME= data set option allows you to specify the variables you want to rename for each input or output data set. Use it in input data sets to rename variables before processing.
• If you use the RENAME= data set option in an output data set, you must continue to use the old variable names in programming statements for the current DATA step. After your output data is created, you can use the new variable names.
• The RENAME= data set option in the SET statement renames variables in the input data set. You can use the new names in programming statements for the current DATA step.
• To rename variables as a file management task, use the DATASETS procedure or access the variables through the SAS windowing interface. These methods are simpler and do not require DATA step processing.
**RENAME= Data Set Option**

Changes the name of a variable.

Valid in: \(\text{DATA step and PROC steps}\)

Category: \(\text{Variable Control}\)

**Note:** The RENAME= data set option has an effect only on data sets that are opened in output mode.

**Syntax**

\[
\text{RENAME=}( \text{old-name-1=new-name-1 < ...old-name-n=new-name-n>} )
\]

**Details**

If you use the RENAME= data set option when you create a data set, the new variable name is included in the output data set. If you use RENAME= on an input data set, the new name is used in DATA step programming statements.

If you use RENAME= on an input data set that is used in a SAS procedure, SAS changes the name of the variable in that procedure. If you use RENAME= with WHERE processing such as a WHERE statement or a WHERE= data set option, the new name is applied before the data is processed. You must use the new name in the WHERE expression.

If you use RENAME= in the same DATA step with either the DROP= or the KEEP= data set option, the DROP= and the KEEP= data set options are applied before RENAME=. You must use the old name in the DROP= and KEEP= data set options. You cannot drop and rename the same variable in the same statement.
The RENAME= data set option differs from the RENAME statement in the following ways:

• The RENAME= data set option can be used in PROC steps and the RENAME statement cannot.

• The RENAME statement applies to all output data sets. If you want to rename different variables in different data sets, you must use the RENAME= data set option.

• To rename variables before processing begins, you must use a RENAME= data set option on the input data set or data sets.

• Use the RENAME statement or the RENAME= data set option when program logic requires that you rename variables such as two input data sets that have variables with the same name. To rename variables as a file management task, use the DATASETS procedure.
RENAME= Data Set Option

Example: Renaming a Variable at Time of Output
This example uses RENAME= in the DATA statement to show that the variable is renamed at the time it is written to the output data set. The variable keeps its original name, X, during the DATA step processing:

```sas
data one;
  input x y z;
  datalines;
  24 595 439
  243 343 034
;
DATA two
  (RENAME=(x=keys));
  SET one;
  z=x+y;
RUN;
```
RENAME= Data Set Option

Example: Renaming a Variable at Time of Input
This example renames variable X to a variable named KEYS in the SET statement, which is a rename before DATA step processing. KEYS, not X, is the name to use for the variable for DATA step processing.

```sas
DATA three;
  SET one(rename=(x=keys));
  z=keys+y;
RUN;
```
RENAME= Data Set Option

Example: Renaming a Variable for a SAS Procedure with WHERE Processing
This example renames variable Score1 to a variable named Score2 for the PRINT procedure. Because the new name is applied before the data is processed, the new name must be specified in the WHERE statement.

```sas
PROC PRINT DATA=test(rename=(score1=score2));
   WHERE score2 gt 75;
RUN;
```
ASSIGNMENT STATEMENTS

<table>
<thead>
<tr>
<th>Valid in:</th>
<th>DATA step</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category:</td>
<td>Action</td>
</tr>
<tr>
<td>Type:</td>
<td>Executable</td>
</tr>
</tbody>
</table>

Syntax

```
variable = expression;
```

Arguments

`variable`

names a new or existing variable.

Range: `variable` can be a variable name, array reference, or SUBSTR function.

Tip: Variables that are created by the Assignment statement are not automatically retained.
ASSIGNMENT STATEMENTS

*expression* is any SAS expression.

**Tip:** *expression* can contain the variable that is used on the left side of the equal sign. When a variable appears on both sides of a statement, the original value on the right side is used to evaluate the expression, and the result is stored in the variable on the left side of the equal sign. For more information, see [Expressions](#) in *SAS Language Reference: Concepts*.

**Details**
Assignment statements evaluate the expression on the right side of the equal sign and store the result in the variable that is specified on the left side of the equal sign.

```sas
name='Amanda Jones';
WholeName='Ms. '||name;
a=a+b;
```
**OBS= Data Set Option**

Specifies the last observation that SAS processes in a data set.

<table>
<thead>
<tr>
<th>Valid in:</th>
<th>DATA step and PROC steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category:</td>
<td>Observation Control</td>
</tr>
<tr>
<td>Default:</td>
<td>MAX</td>
</tr>
<tr>
<td>Restrictions:</td>
<td>Use with input data sets only</td>
</tr>
<tr>
<td></td>
<td>Cannot use with PROC SQL views</td>
</tr>
</tbody>
</table>

**Syntax**

OBS= n | nK | nM | nG | nT | hexX | MIN | MAX

$n | nK | nM | nG | nT$

specifies a number to indicate when to stop processing observations, with $n$ being an integer. Using one of the letter notations results in multiplying the integer by a specific value. That is, specifying K (kilo) multiplies the integer by 1,024, M (mega) multiplies by 1,048,576, G (giga) multiplies by 1,073,741,824, or T (tera) multiplies by 1,099,511,627,776. For example, a value of 20 specifies 20 observations, while a value of 3m specifies 3,145,728 observations.
OBS= tells SAS when to stop processing observations. To determine when to stop processing, SAS uses the value for OBS= in a formula that includes the value for OBS= and the value for FIRSTOBS=. The formula is
\[(\text{obs} - \text{firstobs}) + 1 = \text{results}\]

For example, if OBS=10 and FIRSTOBS=1 (which is the default for FIRSTOBS=), the result is ten observations, that is \((10 - 1) + 1 = 10\).

If OBS=10 and FIRSTOBS=2, the result is nine observations, that is \((10 - 2) + 1 = 9\). OBS= is valid only when an existing SAS data set is read.
Comparisons

- The OBS= data set option overrides the OBS= system option for the individual data set.
- While the OBS= data set option specifies an ending point for processing, the FIRSTOBS= data set option specifies a starting point. The two options are often used together to define a range of observations to be processed.
- The OBS= data set option enables you to select observations from SAS data sets. You can select observations to be read from external data files by using the OBS= option in the INFILE statement.
data Ages;
   input Name $ Age;
datalines;
Miguel 53
Brad 27
Willie 69
Marc 50
Sylvia 40
Arun 25
Gary 40
Becky 51
Alma 39
Tom 62
Kris 66
Paul 60
Randy 43
Barbara 52
Virginia 72;

proc print data=Ages (firstobs=2 obs=12);
run;
proc print data=Ages;
   where Age LT 65;
run;
proc print data=Ages (firstobs=2 obs=10);
   where Age LT 65;
run;