# STA 311 Statistical Computing and Data Management

Instructor: Cheng Peng, Ph.D.
Department of Mathematics
West Chester University
West Chester, PA 19383

Office: 25 University Avenue, RM 111
Phone: 610-436-2369
Email: cpeng@wcupa.edu

## 9. Data Management: Modifying SAS Data Sets

# List of Topics

- More on Date and Time
- Automatic Variables
- Calculating New Variables
- Logical Expressions
- Operators with WHERE statement

# More on Date Formats

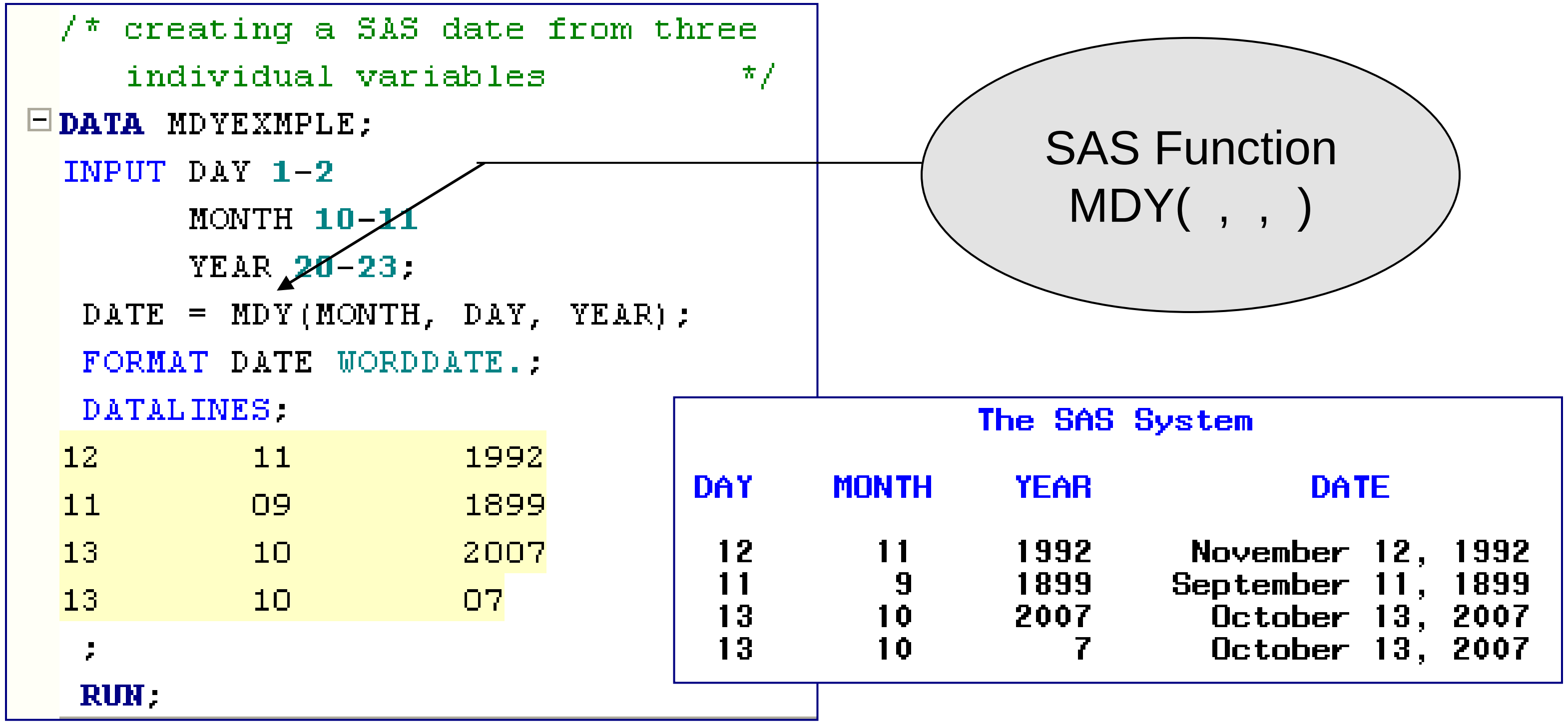
Commonly used SAS informats for date.

For example, September 19, 2007

| Date | informats |
|---|---|
| 09/19/07 | MMDDYY8. |
| 09-19-07 | MMDDYY8. |
| 09+19,07 | MMDDYY8. |
| 19SEP07 | DATE7. |
| 091907 | MMDDYY6. |
| 09/19/2007 | MMDDYY10. |
| 19/09/07 | DDMMYY8. |
| September 19, 2007 | WORDDATE. |
| Wed, Sept, 19, 2007 | WEEKDATE. |

# Define New Variables Using Date Functions

It is not uncommon that the sources data file contains three separate variables representing day, month, and year respectively. Example: Creating a SAS date from month, day, and year.

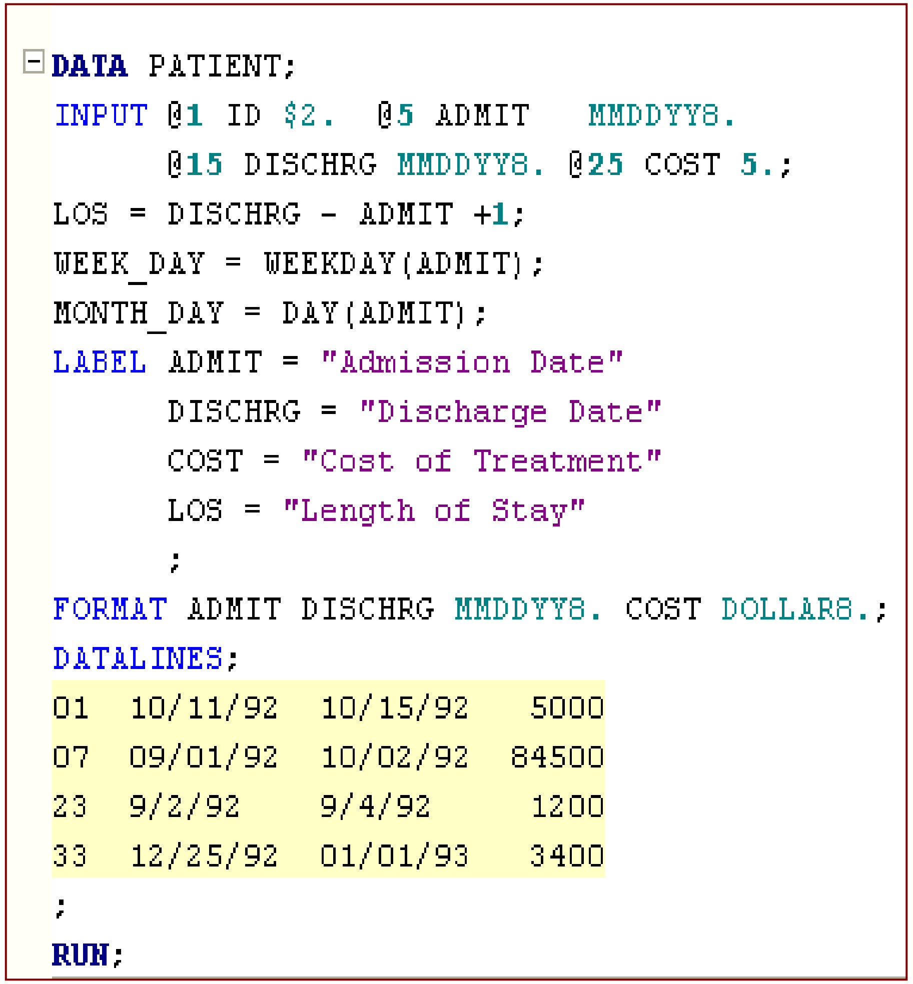


```
/* creating a SAS date from three
   individual variables          */
DATA MDYEXMPLE;
INPUT DAY 1-2
      MONTH 10-11
      YEAR 20-23;
 DATE = MDY(MONTH, DAY, YEAR);
 FORMAT DATE WORDDATE.;
 DATALINES;
12        11        1992
11        09        1899
13        10        2007
13        10        07
 ;
 RUN;
```

SAS Function
MDY( , , )

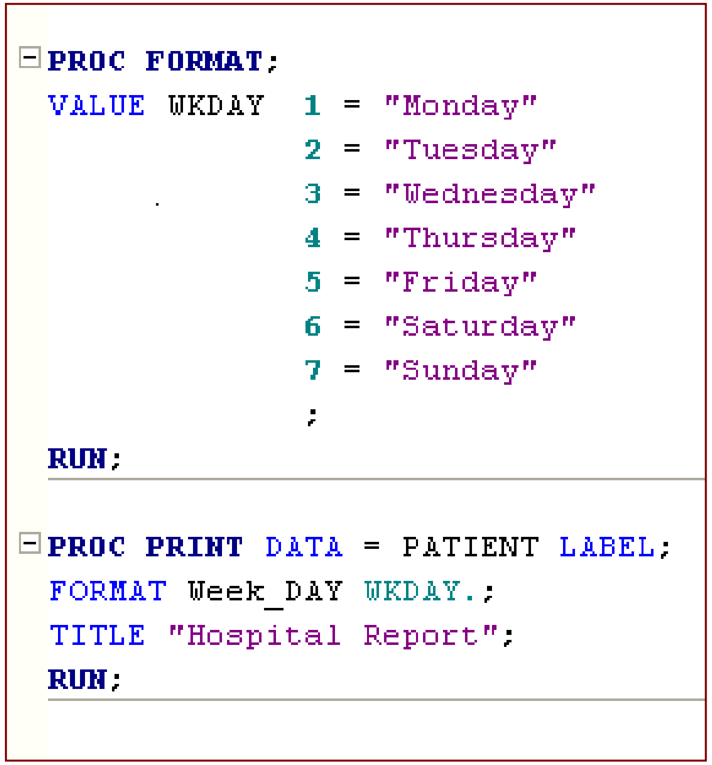
```
                     The SAS System

DAY      MONTH      YEAR                DATE

 12        11       1992      November 12, 1992
 11         9       1899     September 11, 1899
 13        10       2007       October 13, 2007
 13        10          7       October 13, 2007
```

# Define New Variables Using Date Functions



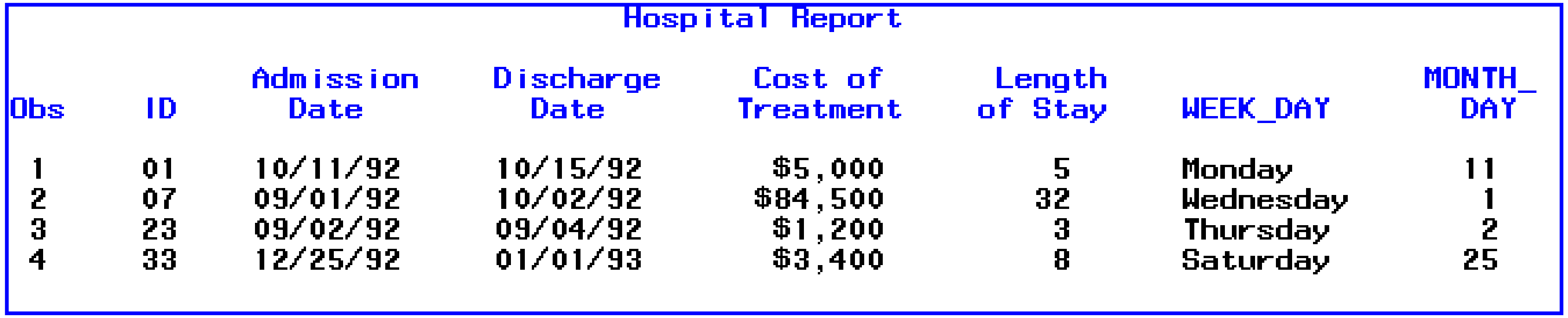
```
DATA PATIENT;
INPUT @1 ID $2.  @5 ADMIT   MMDDYY8.
      @15 DISCHRG MMDDYY8. @25 COST 5.;
LOS = DISCHRG - ADMIT +1;
WEEK_DAY = WEEKDAY(ADMIT);
MONTH_DAY = DAY(ADMIT);
LABEL ADMIT = "Admission Date"
      DISCHRG = "Discharge Date"
      COST = "Cost of Treatment"
      LOS = "Length of Stay"
      ;
FORMAT ADMIT DISCHRG MMDDYY8. COST DOLLAR8.;
DATALINES;
01  10/11/92  10/15/92   5000
07  09/01/92  10/02/92  84500
23  9/2/92    9/4/92     1200
33  12/25/92  01/01/93   3400
;
RUN;
```

**Extracting Day of the week (month) from a SAS Date: SAS functions DAY and WEEKDAY can be used to achieve this goal.**



```
PROC FORMAT;
VALUE WKDAY  1 = "Monday"
             2 = "Tuesday"
             3 = "Wednesday"
             4 = "Thursday"
             5 = "Friday"
             6 = "Saturday"
             7 = "Sunday"
             ;
RUN;

PROC PRINT DATA = PATIENT LABEL;
FORMAT Week_DAY WKDAY.;
TITLE "Hospital Report";
RUN;
```

# Define New Variables Using Date Functions



```
                              Hospital Report

                 Admission    Discharge     Cost of     Length               MONTH_
Obs      ID        Date         Date       Treatment    of Stay    WEEK_DAY      DAY

 1       01      10/11/92     10/15/92       $5,000         5      Monday        11
 2       07      09/01/92     10/02/92      $84,500        32      Wednesday      1
 3       23      09/02/92     09/04/92       $1,200         3      Thursday       2
 4       33      12/25/92     01/01/93       $3,400         8      Saturday      25
```

# More on Date: Relevant Functions

## Useful SAS functions commonly used with SAS Dates

### 1. INT() and ROUND() - Examples

| | | | | | |
|---|---|---|---|---|---|
| a=int(5); | 5 | g=round(5); | 5 | l=round(2222,10); | 2220 |
| b=int(7.3); | 7 | h=round(7.3); | 7 | m=round(2222,100); | 2200 |
| c=int(7.6); | 7 | i=round(7.6); | 8 | n=round(2222,1000); | 2000 |
| d=int(-3); | -3 | j=round(-9.2); | -9 | o=round(15.125,.1); | 15.1 |
| e=int(-9.2); | -9 | k=round(-9.8); | -10 | p=round(15.125,.01); | 15.13 |
| f=int(-9.8); | -9 | | | q=round(15.125,.001); | 15.125 |

### 2. Computing Date Intervals: INTCK() and INTNX()

```
Work_Yrs = INTCK('YEAR', DATEHIRE, TODAY())
Followup = INTNX('MONTH', VISIT, 10)
```

**Caution:** both functions works with internal boundaries (the 1st of each month or year depending on the number of months or the number of years to be computed.)

# Automatic Variables

**_N_** denotes the observation number.

**_error_** equals to 0 if no error occurs when reading an observation and equal to 1 if an error occurs.

**FIRST.var** associated with the BY var; statement, equals 1 if the observation is the first observation with a particular value of the BY var, zero otherwise.

**LAST.var** associated with the BY var; statement, equals 1 if the observation is the last observation with a particular value of the BY var, zero otherwise.

# Automatic Variables

```
DATA pets1;
 INPUT @1 name $9. @10 time time5. @20 date mmddyy8. @30 species $;
 mistakes=_error_; /* New variable mistakes=1 if error in reading obs*/
DATALINES;
Fluffy   9:00      02/13/98   cat
Tom      10:00     02/13/98   cat
Rex      13:00     02/31/98   dog
Fido     14:00     02/13/98   dog
Felix    9:30      02/13/98   cat
Spot     15:00     02/13/98   dog
;
PROC SORT data=pets1;
  BY species time;
RUN;;
DATA pets1;      /* Make changes to dataset PETS1. */
  SET pets1;
  BY species;
  pet_num=_n_;   /* Pet number equal to observation number */
  firstgrp=first.species; /* firstgrp=1 if first obs of each species*/
  lastgrp=last.species;   /* lastgrp=1 if last obs of each species */
RUN;
PROC PRINT data=pets1;
  VAR pet_num name species time date mistakes firstgrp lastgrp;
  FORMAT time time5. date mmddyy8.;
RUN;
```

# Automatic Variables

```
                         The SAS System

OBS PET_NUM NAME    SPECIES  TIME     DATE MISTAKES FIRSTGRP LASTGRP

 1      1   Fluffy    cat     9:00 02/13/98     0         1        0
 2      2   Felix     cat     9:30 02/13/98     0         0        0
 3      3   Tom       cat    10:00 02/13/98     0         0        1
 4      4   Rex       dog    13:00        .     1         1        0
 5      5   Fido      dog    14:00 02/13/98     0         0        0
 6      6   Spot      dog    15:00 02/13/98     0         0        1
```

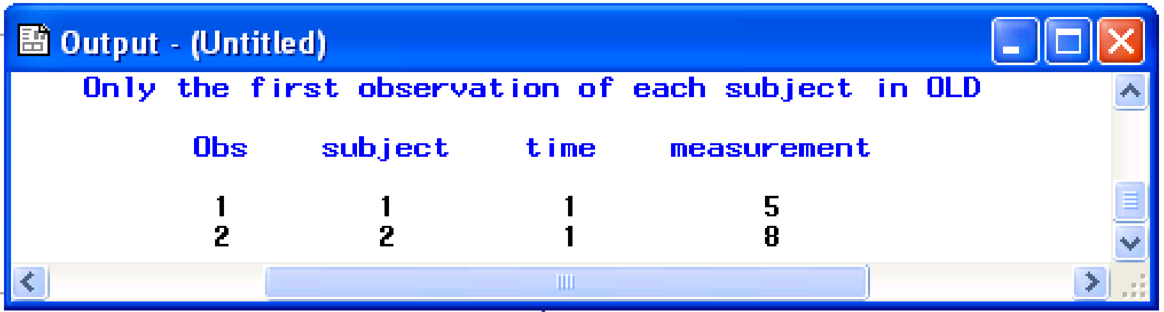
```
1      options ls=70 ps=200 nodate nocenter;
2      data pets1;
3      input @1 name $9. @10 time time5. @20 date mmddyy8. @30 species
$;
4      mistakes=_error_; /* New variable mistakes=1 if error in
reading obs*/
5      datalines;

NOTE: Invalid data for DATE in line 8 20-27.
RULE:----+----1----+----2----+----3----+----4----+----5----+----6----+
8     Rex      13:00     02/31/98   dog
NAME=Rex TIME=46800 DATE=. SPECIES=dog MISTAKES=1 _ERROR_=1 _N_=3
NOTE: The data set WORK.PETS1 has 6 observations and 5 variables.
NOTE: The DATA statement used 0.93 seconds.
```

# Automatic Variables

```
OPTIONS NONUMBER NODATE;
/* automatic variable */

DATA OLD;
INPUT subject time measurement @@;
DATALINES;
1 1 5 1 2 6 1 3 7 2 1 8 2 2 9 2 3 4
;
RUN;

PROC SORT DATA = OLD;
BY subject time;
RUN;

DATA NEW;
SET OLD;
BY subject;
IF FIRST.subject;
RUN;

PROC PRINT DATA = NEW;
TITLE "Only the first observation of each subject in OLD";
RUN;
```

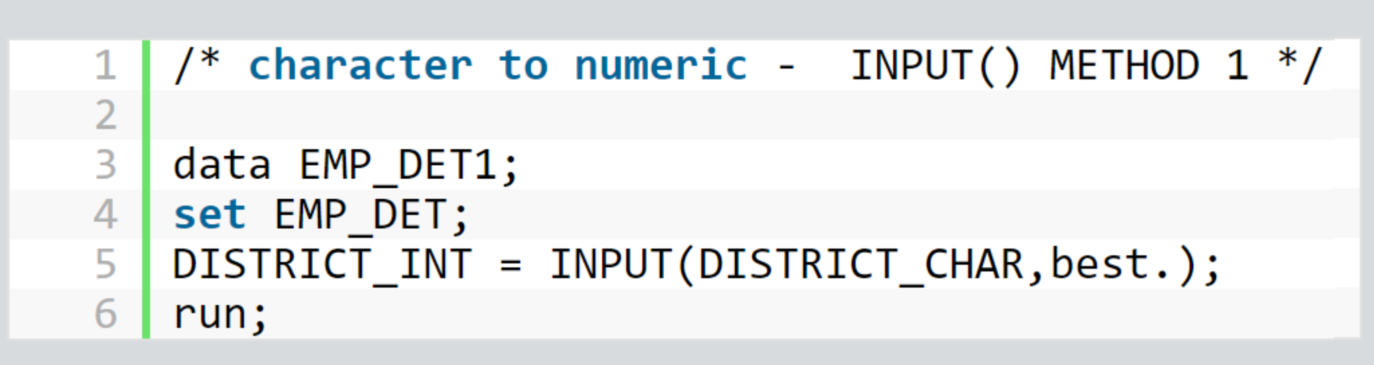
**Creating a single observation from multiple observations**

**A cautionary note: don’t automatically accept _N_ as a true observation counter in all DATA steps. Data steps that contain LOOPS and OUTPUT statements may cause the relationship of the observation number and the internal variable _N_ to fall apart!**

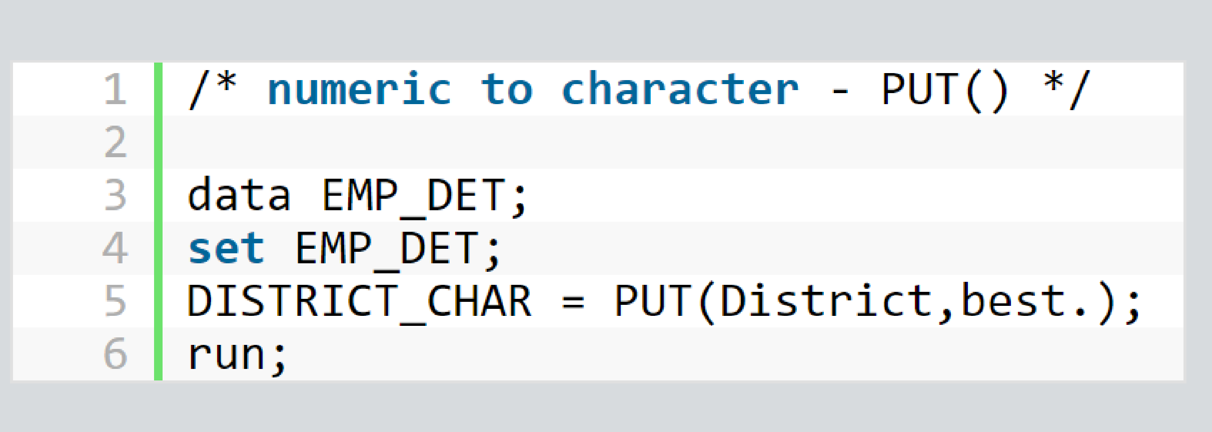


```
   Only the first observation of each subject in OLD

          Obs     subject     time     measurement

            1        1          1           5
            2        2          1           8
```

# Converting Variable Types



```
/* character to numeric -  INPUT() METHOD 1 */

data EMP_DET1;
set EMP_DET;
DISTRICT_INT = INPUT(DISTRICT_CHAR,best.);
run;
```



```
/* numeric to character - PUT() */

data EMP_DET;
set EMP_DET;
DISTRICT_CHAR = PUT(District,best.);
run;
```

# Creating New Variables with Basic Mathematics Operations

**New variables are created from input data values using standard algebraic expressions and mathematical functions.**

| | |
|---|---|
| Newvar = var1 + var2 ; | addition |
| Newvar = var1 - var2; | subtraction |
| Newvar = var1 * var2; | multiplication |
| Newvar = var1 / var2; | division |
| Newvar = var1 ** var2; | exponentiation ($var1^{var2}$) |

*Order of operation can be controlled by parenthesis*.

Newvar = (var1 + var2) ** (var3 / (var4*var5)) - var6 ;

```
Gigabyte = harddriv/1e9 ;
kilobaud = modem / 1000;
perimeter = (2*(3/5)*monitor + 2*(4/5)*monitor)*2.54;
area = ((((3/5)*monitor)*((4/5)*monitor))/2)*(2.54**2);
```

# Creating New Variables with Basic Mathematics Functions

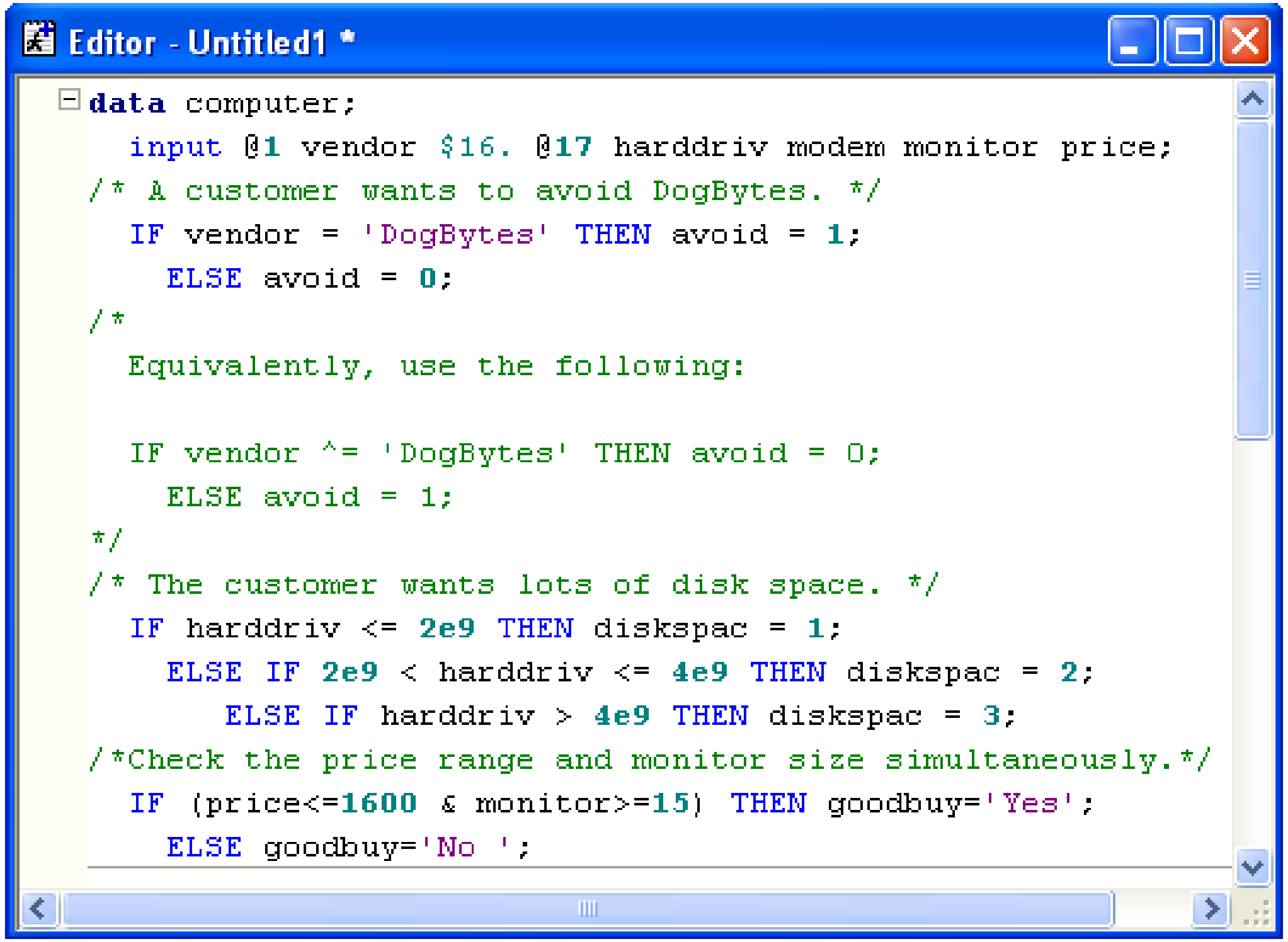
| | |
|---|---|
| Newvar = log(var1); | value is natural logarithm of var1 |
| Newvar = log10(var1); | value is common logarithm of var1 |
| Newvar = log2(var1); | value is base 2 logarithm of var1 |
| Newvar = sqrt(var1); | value is square root of var1 |
| Newvar = mdy(month, day, year); | create SAS date variable from individual month, day and year values. |
| Newvar = abs(var1); | value is the absolute value of var1 |

# Logical Operators

```
IF logical_condition_true THEN action_1;
                ELSE action_2;
```

| | |
|---|---|
| Logical operators | EQ equals  (=) |
| | NE not equal (~=, ^=) |
| | GT greater than (>) |
| | LT  less than (<) |
| | GE greater than or equal to (>=) |
| | LE  less than or equal to (<=) |
| | AND all comparisons must be true (&) |
| | OR only one comparison must be true (!, \|) |
| Actions | ANY SAS data or macro statement |
| | (e.g. another assignment, IF, DO …) |

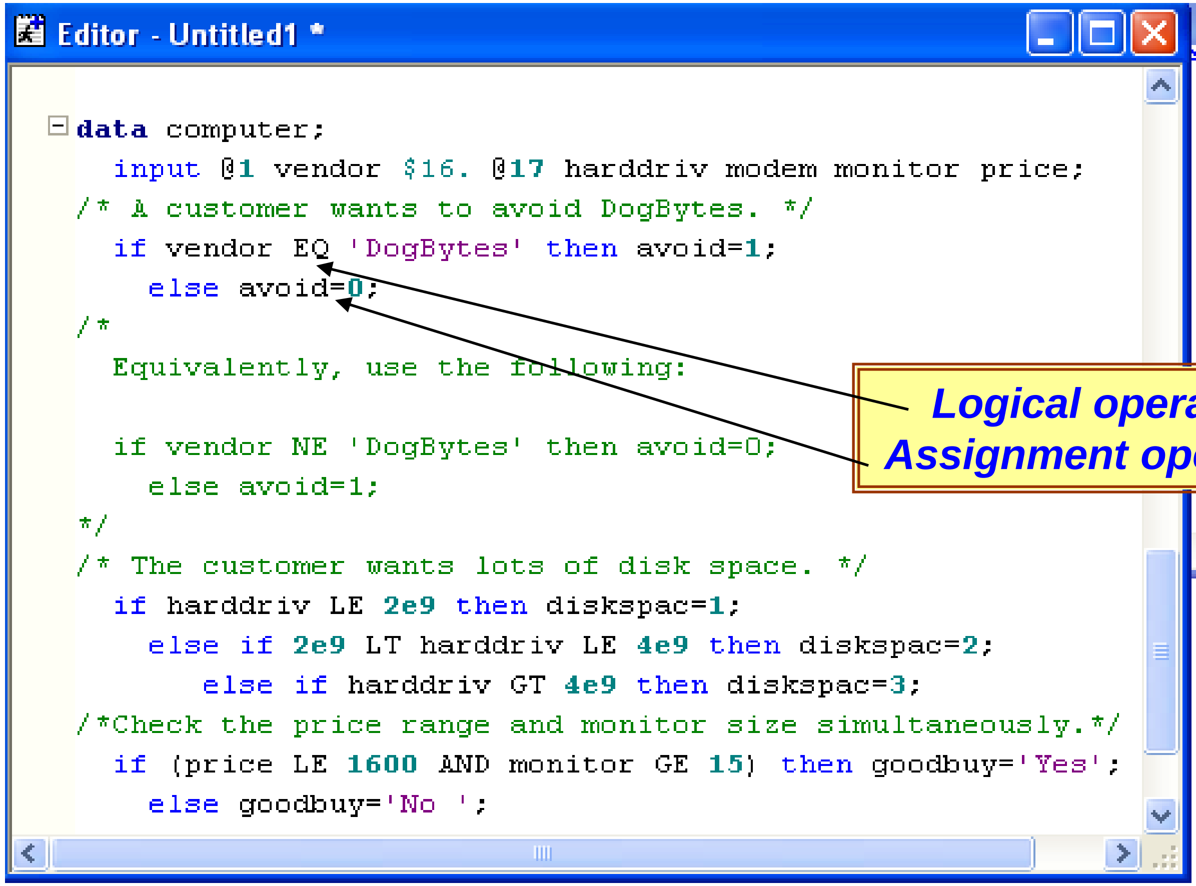
# Logical Operators



```
data computer;
  input @1 vendor $16. @17 harddriv modem monitor price;
/* A customer wants to avoid DogBytes. */
  IF vendor = 'DogBytes' THEN avoid = 1;
    ELSE avoid = 0;
/*
  Equivalently, use the following:

  IF vendor ^= 'DogBytes' THEN avoid = 0;
    ELSE avoid = 1;
*/
/* The customer wants lots of disk space. */
  IF harddriv <= 2e9 THEN diskspac = 1;
    ELSE IF 2e9 < harddriv <= 4e9 THEN diskspac = 2;
       ELSE IF harddriv > 4e9 THEN diskspac = 3;
/*Check the price range and monitor size simultaneously.*/
  IF (price<=1600 & monitor>=15) THEN goodbuy='Yes';
    ELSE goodbuy='No ';
```

# Logical Operators

**RECOMMENDATION: Use logical operators EQ, LT, LE, GT, GE, AND, OR**



```
data computer;
 input @1 vendor $16. @17 harddriv modem monitor price;
/* A customer wants to avoid DogBytes. */
 if vendor EQ 'DogBytes' then avoid=1;
  else avoid=0;
/*
 Equivalently, use the following:

 if vendor NE 'DogBytes' then avoid=0;
  else avoid=1;
*/
/* The customer wants lots of disk space. */
 if harddriv LE 2e9 then diskspac=1;
  else if 2e9 LT harddriv LE 4e9 then diskspac=2;
   else if harddriv GT 4e9 then diskspac=3;
/*Check the price range and monitor size simultaneously.*/
 if (price LE 1600 AND monitor GE 15) then goodbuy='Yes';
  else goodbuy='No ';
```

*Logical operator.*
*Assignment operator.*

# Why Use Logical Operators

1) To remove observations or otherwise subset the data set.

```
DATA cheap;
  SET computer;
  IF price LE 1600;
```

2) To differentially assign values.

```
IF age LE 21 THEN age_grp=1;
  ELSE IF 21 LT age LE 55 THEN age_grp=2;
      ELSE IF age GT 55 THEN age_grp=3;
```

# WHERE Statement Operators

**Operator:**

BETWEEN – AND

**Action:**

Selects observations which fall (inclusively) within a specified range

**Example(S):**

```
WHERE AGE BETWEEN 20 AND 40;
```

(Selects age between 20 and 40 inclusively)

# WHERE Statement Operators

**Operator:**

CONTAINS or ?

**Action:**

Used for character variable only, selects records that include or contain the specified string.

NOTE: the string in the quote is CASE-SENSITIVE!!

**Example(S):**

```
WHERE NAME CONTAINS ‘eng’ ;
 or
WHERE NAME ?  ‘eng’ ;
```

(Selects all names that contain the string *eng*. This would match my name Cheng Peng!)

# WHERE Statement Operators

**Operator:**

IS MISSING or IS NULL

**Action:**

Selects observations for which the value of the variable is missing. This is particularly useful since it works with both numeric and character variable

**Example(S):**

```
WHERE AGE IS MISSING;
```

(Selects all observations where AGE is missing)

```
WHERE NAME IS NULL ;
```

(Selects all observations where NAME is missing)

# WHERE Statement Operators

**Operator:**

LIKE

**Action:**

Allows you to select observations based on patterns using the percent sign(%) and underscore(_) wildcard operators.

The percent sign (%) is a variable length wildcard (like * in DOS or UNIX). It matches on any string (including a null string).

The underscore (_) wildcard operator is a patter matching for one character only.

NOTE: The LIKE operator is only used with character variables and is case sensitive!

# WHERE Statement Operators

**Operator:**

LIKE

**Example(S):**

WHERE NAME LIKE ‘BOY%’ ;

(Examples of matches are: BOY BOYCE BOYXYZ, etc.)

WHERE NAME LIKE ‘A___’ ;   *  3 underscores

(Selects all names of length 4, beginning with A)

WHERE NAME LIKE ‘A_%’ ;

(Selects all names that begin with A and are at least two characters in length)

# WHERE Statement Operators

**Operator:**

=*

**Action:**

A phonetic match (called a SOUNDEX operator) used for matches that “sound like” the given expression. The =* operator attempts a phonetic match based on a Soundex algorithm. It is a very powerful operator and should be used with care. It is useful for “fuzzy matches” where you suspect a name might be misspelled or you are not sure of the correct spelling of a name.

NOTE: The SOUNDEX operator is NOT case-sensitive!

# WHERE Statement Operators

**Operator:**

=*

**Example(S):**

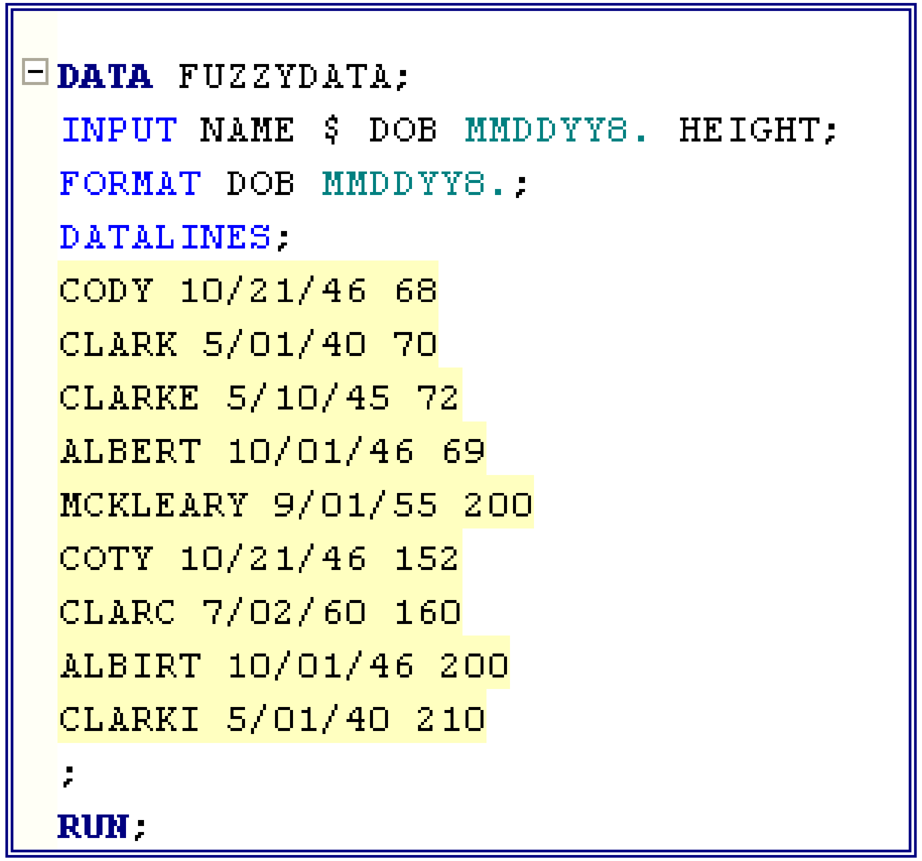
```
WHERE NAME =* 'CODY' ;
```

(Given the names: CODY Coedy, Kody, COTY, and COOky, the above code selects: CODY, Coedy, Kody and Koty, but not COOky.)

```
WHERE NAME =* 'MCHENRY' ;
```

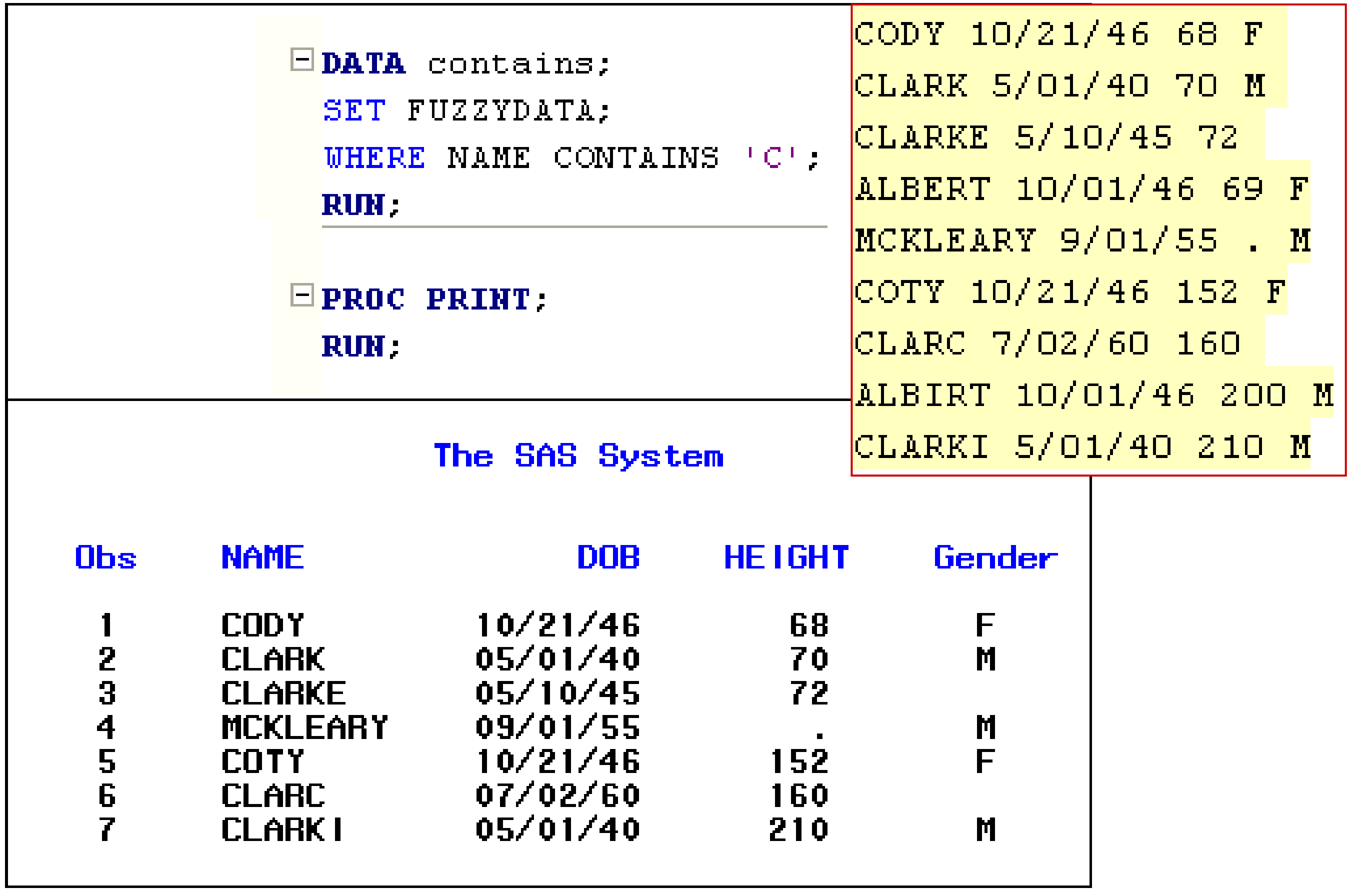
(Given the names: MCHENRY, MACHENRY, MCHENRI, and MKHENRY, the above code selects all names. This is obviously an operator to be used with considerable caution.)

# WHERE Statement Operators



```
DATA FUZZYDATA;
INPUT NAME $ DOB MMDDYY8. HEIGHT;
FORMAT DOB MMDDYY8.;
DATALINES;
CODY 10/21/46 68
CLARK 5/01/40 70
CLARKE 5/10/45 72
ALBERT 10/01/46 69
MCKLEARY 9/01/55 200
COTY 10/21/46 152
CLARC 7/02/60 160
ALBIRT 10/01/46 200
CLARKI 5/01/40 210
;
RUN;
```

# WHERE Statement Operators

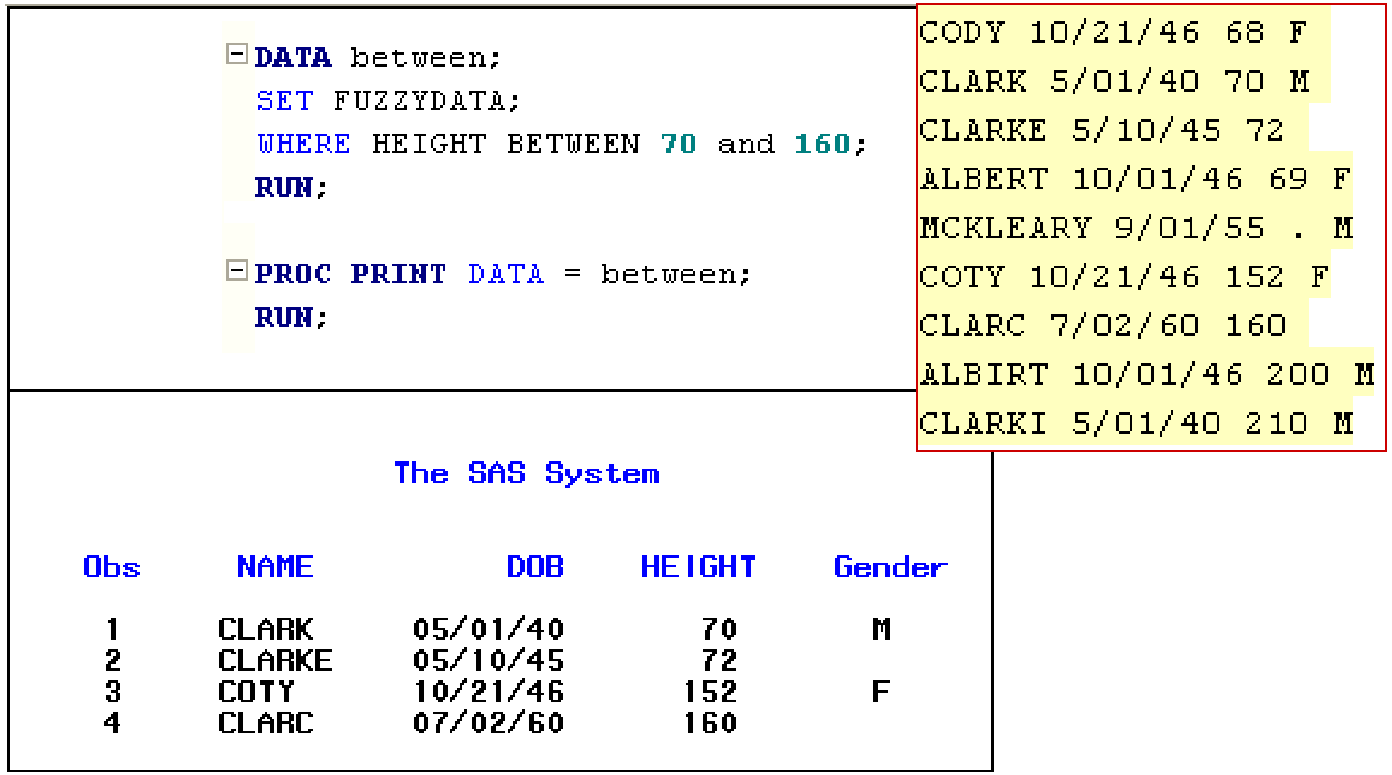


```
DATA contains;
SET FUZZYDATA;
WHERE NAME CONTAINS 'C';
RUN;

PROC PRINT;
RUN;
```

```
CODY 10/21/46 68 F
CLARK 5/01/40 70 M
CLARKE 5/10/45 72
ALBERT 10/01/46 69 F
MCKLEARY 9/01/55 . M
COTY 10/21/46 152 F
CLARC 7/02/60 160
ALBIRT 10/01/46 200 M
CLARKI 5/01/40 210 M
```

```
                        The SAS System

Obs    NAME                DOB    HEIGHT    Gender

 1     CODY           10/21/46       68       F
 2     CLARK          05/01/40       70       M
 3     CLARKE         05/10/45       72
 4     MCKLEARY       09/01/55        .       M
 5     COTY           10/21/46      152       F
 6     CLARC          07/02/60      160
 7     CLARKI         05/01/40      210       M
```

# WHERE Statement Operators

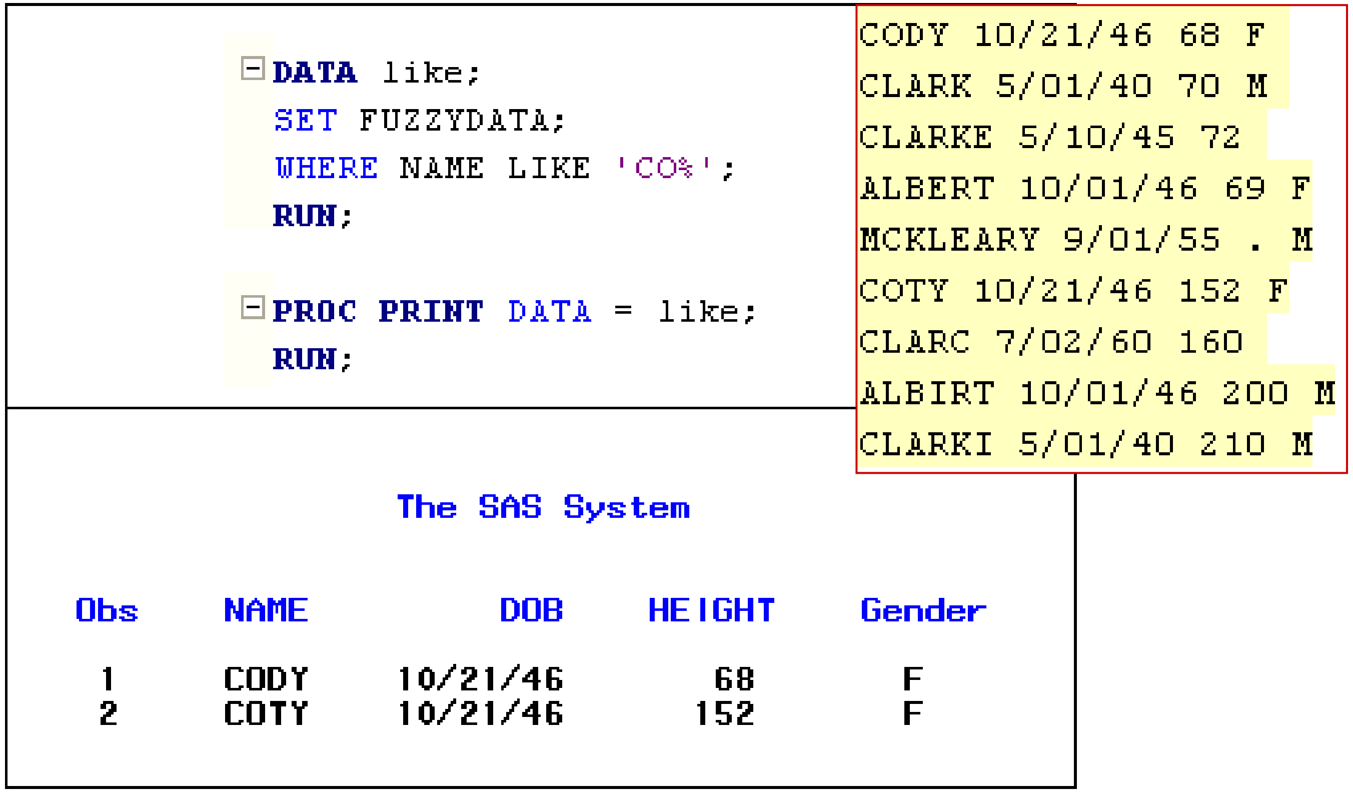


```
DATA between;
SET FUZZYDATA;
WHERE HEIGHT BETWEEN 70 and 160;
RUN;

PROC PRINT DATA = between;
RUN;
```

```
CODY 10/21/46  68 F
CLARK 5/01/40 70 M
CLARKE 5/10/45 72
ALBERT 10/01/46 69 F
MCKLEARY 9/01/55 . M
COTY 10/21/46 152 F
CLARC 7/02/60 160
ALBIRT 10/01/46 200 M
CLARKI 5/01/40 210 M
```

```
                    The SAS System

 Obs     NAME            DOB     HEIGHT    Gender

  1      CLARK      05/01/40        70       M
  2      CLARKE     05/10/45        72
  3      COTY       10/21/46       152       F
  4      CLARC      07/02/60       160
```

# WHERE Statement Operators

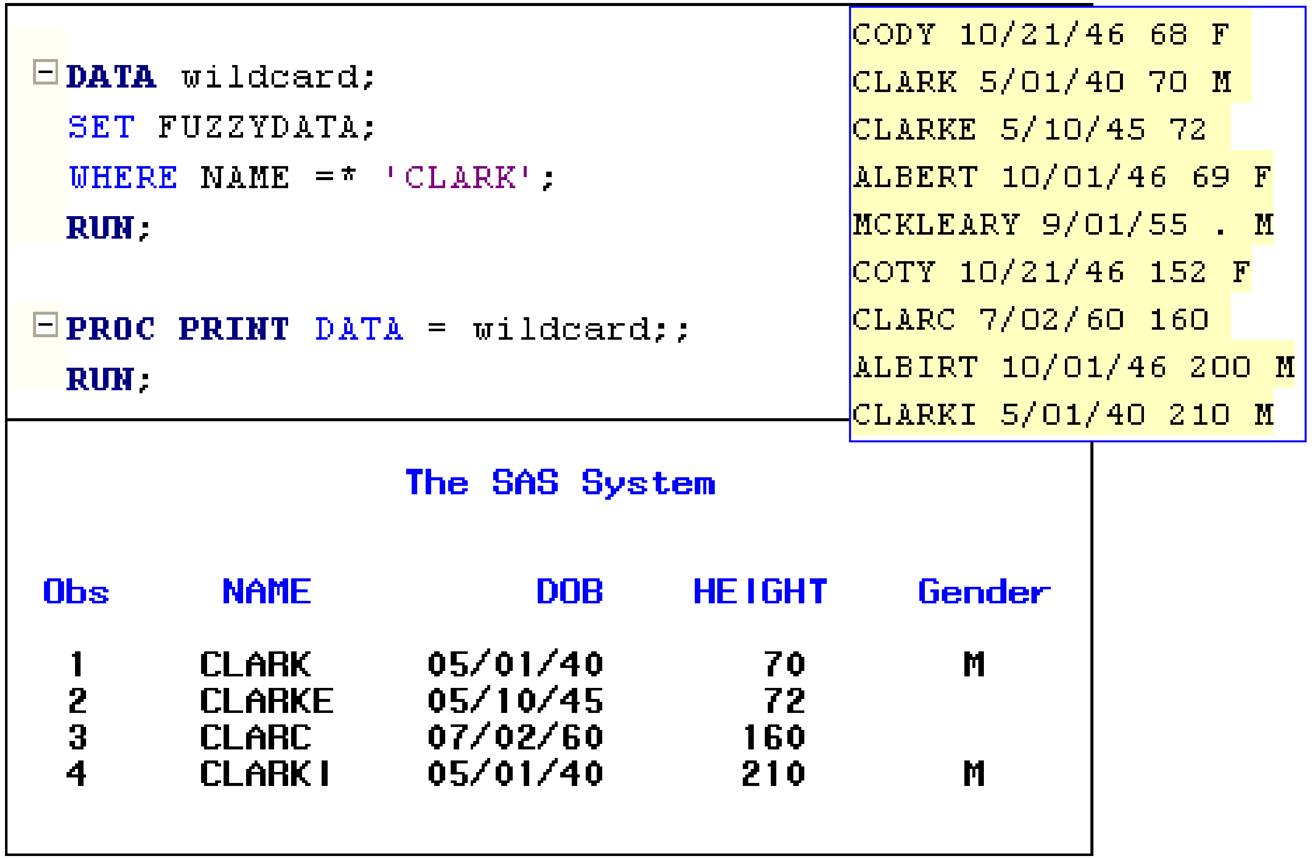


```
DATA like;
SET FUZZYDATA;
WHERE NAME LIKE 'CO%';
RUN;

PROC PRINT DATA = like;
RUN;
```

```
CODY 10/21/46 68 F
CLARK 5/01/40 70 M
CLARKE 5/10/45 72
ALBERT 10/01/46 69 F
MCKLEARY 9/01/55 . M
COTY 10/21/46 152 F
CLARC 7/02/60 160
ALBIRT 10/01/46 200 M
CLARKI 5/01/40 210 M
```

```
                    The SAS System

Obs     NAME           DOB    HEIGHT    Gender

 1      CODY      10/21/46       68       F
 2      COTY      10/21/46      152       F
```

# WHERE Statement Operators



```
DATA wildcard;
SET FUZZYDATA;
WHERE NAME =* 'CLARK';
RUN;

PROC PRINT DATA = wildcard;;
RUN;
```

```
CODY 10/21/46 68 F
CLARK 5/01/40 70 M
CLARKE 5/10/45 72
ALBERT 10/01/46 69 F
MCKLEARY 9/01/55 . M
COTY 10/21/46 152 F
CLARC 7/02/60 160
ALBIRT 10/01/46 200 M
CLARKI 5/01/40 210 M
```

```
                    The SAS System

Obs      NAME           DOB     HEIGHT    Gender

 1       CLARK     05/01/40        70       M
 2       CLARKE    05/10/45        72
 3       CLARC     07/02/60       160
 4       CLARKI    05/01/40       210       M
```