Data Manipulation

When SAS creates a dataset, it performs the following actions for each observation in the input data set:

1) SAS reads an observation using the INPUT stmt

2) SAS executes programming stmts, using data values in the current observation

3) SAS "outputs" (adds) the obs. to the SAS data set

4) Repeat steps 1-3 until all data in the input (raw) data set has been read into SAS
DATA datasetname;
INPUT — —;
programming statements go here
DATALINES;
data records go here
;
SAS procedures and additional DATA steps go here

Note: SAS program statements are "carried out" (executed) once for each obs.
Assignment Statements

An Assignment Statement
1) evaluates an expression
2) assign the resulting value to a variable
3) has general form
   
   \text{variable} = \text{expression} ;

4) can be used to modify the value of an existing variable
5) can be used to create a new variable
A SAS expression contains operands and operators that form a set of instructions that produce a value.

Operands can be:
1) Variables
2) Constants

Operators are:
1) Symbols that indicate arithmetic calculations
2) SAS built-in functions

Operators that perform basic arithmetic calculations include
Order of Operations

G E M D A S

add
subtract
multiply
divide
exponent
negative prefix

Grouping Symbols
Create a New Variable

1) Choose name for new variable
   Place on left-side of = in an assignment stmt.

2) Write "formula" for computing value of new variable
   using operands and operators
   Place formula on right side of = in an assignment stmt.

EX: Weight_lbs = Weight_kg * 2.2;
    Totals_day = Lastday - Firstday;
    Celsius = (5/9) * (F - 32);
Modify an Existing Variable Value

SAS assignment stmt can be used to modify the value of a variable that exists in your data set.

The name of the variable we wish to modify will appear on both sides of =.

EX: Height = Height / 12;
Weight = Weight / 2.2;

The = sign says "assign computed value on right of = sign to the variable on left of = sign."
Conditional Execution

You may want SAS to execute a (group of) statement(s), but only for those observations that satisfy (a) specific condition(s).

IF-THEN statement

1) used for condition processing
2) select rows to include in a SAS data set
3) has general form

   IF condition THEN statement;

If the condition is evaluated as True, then the statement is executed using the data values
of the current obs.
The "condition" is actually an expression that contains operands and operators. This expression is actually a set of instructions that produce a value.

Operands can be:
1) variables
2) constants

Operators can be:
1) comparison operators
   > GT
   >= GE
   < LT
   <= LE
   = EQ
   ≠ NE
   ~ NE
2) logical operators
   AND
   OR
   NOT

3) arithmetic expressions

4) SAS built-in functions

Ex: IF Age <= 12 then
    Stage = "Child";

IF Age > 12 and Age <= 19
    THEN stage = "teenage";

IF Age > 19 then
    stage = "adult";
ELSE statement

Can use ELSE stmt in conjunction with IF-THEN stmt.

Whenever the IF condition is not satisfied, the statement following ELSE is executed.

EX: IF age < 30 THEN
    status = "Young" ;
ELSE status = "Old" ;
NESTED IF-THEN ... ELSE stmts
An IF-THEN stmt can be part of an ELSE stmt.

EX: IF age <=12 THEN stage = "child";
    ELSE IF 12 < age <= 19
    THEN stage = "teenage";
    ELSE IF age > 19
    THEN stage = "adult";
    ELSE stage = . ;
Data Quality

IF age < 0 THEN LIST;
asks SAS to list in the job log
any observation with a possible
error for the variable age.

IF age < 0 THEN DELETE;
asks SAS to exclude from the
SAS dataset any obs which has
a negative or missing value
for age.
Subsetting

IF-THEN...ELSE stmts can be used to create subsets of a data set.

IF sex='M' THEN DELETE;
asks SAS to delete all Males from the current SAS data set.

IF sex='F';
asks SAS to keep only Females in the data set SAS creates.

IF sex NE 'F';
asks SAS to keep anyone whose gender is not female, i.e.,
Delete all the females, which we could also do as

IF SEX EQ 'F' THEN DELETE;

Upper case vs. Lower Case in Data

Could write:

IF SEX='M' or SEX='m'
    THEN OUTPUT;

IF SEX='M' or SEX='m';

IF UPCASE(SEX)='M';

SAS built-in function UPCASE converts letters to all uppercase.
DO groups

You may want to execute a group of statements whenever some condition is satisfied.

Ex: IF age > 18 then stmt1;
    IF age > 18 then stmt2;
    IF age > 18 then stmt3;

Can be handled more efficiently by using a DO group, along with an END stmt.

IF condition THEN DO;
    statements
END;
Ex: IF age > 18 THEN DO;
        stmt 1;
        stmt 2;
        stmt 3;
        END;

A DO-group can also be used with the ELSE stmt.

IF ~ THEN ~;
ELSE DO;
        ~;
        ~;
        ~;
        END;
SELECT statement

Can perform function similar to IF-THEN...ELSE

General Form

SELECT;

WHEN (condition1) Assignment1;

... Otherwise assignmentn;

END;

RUN;

Can be more efficient (less typing) than IF-THEN...ELSE
EX: SELECT;
    WHEN (0 <= Age <= 12)
        stage = "child";
    WHEN (12 < Age <= 19)
        stage = "teen";
    WHEN (Age > 19)
        stage = "adult";
    OTHERWISE stage = "";
END;
null string
PROC FORMAT;
The PUT function can be used to "change" a numeric value to a character value.

General Form

char_var = PUT(variable, format);

Ex: DATA;
    INPUT SSN;
    DATALINES;
    456 879113
    ;
    ;
    DATA; SET work1;
    SS = PUT(SSN, SSN111.)
    RUN;
creates character variable SS
with length 11 and format
XXX-XX-XXXX
from the numeric variable SSN