NOTE #2: DATA MANIPULATIONS I

2.1 SUBSETTING DATA

IF: You can use IF statement to subset your data. For example, for a grade roster data if you want to choose senior students data only, then after you define *Class* variable in INPUT statement use

```
IF Class = 'senior';
```

Note that the equal sign (=) can be replaced with `EQ`. That is, `IF Class EQ 'senior';`

More than one conditions can be chosen by using *`AND/OR/IN`*. For example,

```
IF Class = 'senior' and Gender = 'Female';
IF Class = 'senior' or Class = 'junior';
IF Class in ('senior','junior');
```

The second and the third statements will return the same result. "*&*" can replace "*and*" and "*|*" can replace "*or*". For subsetting using a numeric variable, the following mnemonic or symbolic expression can be used:

```
Equal to:               EQ    =
Not equal to:           NE    ~=
Less than:              LT    <
Greater than            GT    >
Less than or equal:     LE    <=
Greater than or equal:  GE    >=
```

For example,

```
IF 70 <= Average < 90;
IF Average GE 70 & Average LT 90;
```

IF-OUTPUT: You can create multiple sub-datasets using IF-THEN-OUTPUT statement. For example,

```
IF Gender = 'M' THEN OUTPUT MALES;
ELSE IF Gender = 'F' THEN OUTPUT FEMALES;
ELSE Missing(Gender) THEN OUTPUT MISSINGDATA;
```

This will generate three SAS data sets; males, females, and missingdata.

WHERE: Subsetting also can be done using WHERE statement. Simply replace IF with WHERE.

```
WHERE Class = 'senior' and Gender = 'Female';
WHERE Class = 'senior' or Class = 'junior';
WHERE Class in ('senior','junior');
```

Note that WHERE statement can be used in SAS PROC while IF can only used in DATA step. Also note that WHERE has a larger choice of operators.

2.2 CONDITIONAL PROCESS

IF and ELSE IF: IF-THEN statement can be used to apply a statement to some part or subset of observations.

```
   IF conditions THEN statements;
or
   IF conditions THEN DO;
         Statements;
   END;
```

Multiple conditions can be combined with *AND* (&) or *OR* (|). To apply separate statements for more than one groups of observations ELSE IF statement can be used:

```
   IF conditions THEN statements;
   ELSE IF conditions THEN statements;
   ELSE IF conditions THEN statements;
   …
```

For example,

```
   IF missing(Score) THEN Grade = .;
   ELSE IF Score <= 50 THEN Grade = "D";
   ELSE IF Score <= 70 THEN Grade = "C";
   ELSE IF Score <= 90 THEN Grade = "B";
   ELSE IF Score > 90 THEN Grade = "A";
```

IF-THEN statements can be used with INPUT statement.

```
   IF conditions THEN INPUT variable list;
   ELSE IF conditions THEN INPUT variable list;
```

SELECT-WHEN: Series of IF and ELSE IF statements can be simplified using SELECT-WHEN statement.

```
   SELECT (varname);
     WHEN (conditions or values) statement;
     …
     Otherwise statement;
   END;
```

Don't forget `END;` at the end. More than one values in when statement can be separated by a comma. For example, `WHEN (1,2,3).` Above example can be replaced with

```
   SELECT (Score);
     WHEN (missing(Score))Grade = .;
     WHEN (Score LE 50) Grade = "D";
     WHEN (Score LE 70) Grade = "C";
     WHEN (Score LE 90) Grade = "B";
     OTHEWISE Grade = "A";
   END;
```

2.3 LOOPING

DO loop: Iterative loop for sequential calculations or input. For data having each observation line represents values from a group or a variable it is useful to use Do loop with INPUT statement. For example, consider the data

```
87 74 89 70 74
69 68 77 80 54
98 96 97 84 82
```

Here, scores in each line are test scores from five students for three different subjects (for example, first line is English, second is Math, and the third is History score). Then we can write

```
DATA score;
  Do Subj = 'English','Math','History';
       DO Student = 1 to 5;
              INPUT Score @;      * @ is a pointer holder;
              OUTPUT;    *to write out an observation to the output data;
       END;               * See example2_3 for more about OUTPUT;
  END;

DATALINES;
87 74 89 70 74
69 68 77 80 54
98 96 97 84 82
; RUN;
```

This example used both character values (English, Math, History) and numeric values (1,…,5) for DO loop indices. More examples will follow.

Other form of DO loop: DO WHILE, DO UNTIL

ARRAY **:** In case you have many variables to transform, you may simply your program using array. After you define the variables in INPUT statement you can write an array statement as follows (see example below): `ARRAY` *`arrayname`* `(n) $` *`variable list`*

```
/* Example2_1 */

/* DATA: 'E\SASCLASS\example2_1.dat'

STAT101 Elementary Statistic
Fall 2010 Grade Roster
 C Gonzalez   J  82 60 78
 T Chung      Sr 92 58 96
 I Irwin      F  52 69 77
 C Reid       S  98 92 89
 P Lee        S  60 48 70
 B Washington J  47    63
 K Piazza     F  87 72 90
 M Crafton    F 100 74100
 A Shulz      S     47 61
 S Olson      F  74 77 75
 C Chen       S  90 73 68
 H Wood       F    100 98
*/

LIBNAME SASPRAC 'E:\SASCLASS';

DATA SASPRAC.roster;
 INFILE 'E\SASCLASS\example2_1.dat' FIRSTOBS=3 MISSOVER;
  INPUT NAME $2-14  Class $15-16 Class. (Score1 Score2 Score3)(3.);

  ARRAY Test (3) Score1-Score3;
  DO i = 1 to 3;
      If missing(Test(i)) then Test(i) =0;
  END;
 DROP i;

Average = ROUND(MEAN(Score1,Score2,Score3),1);
If Average  GE 90 then Grade = "A";
   ELSE if Average >= 80 then Grade = "B";
   ELSE if Average >= 70 then Grade = "C";
   ELSE if Average >= 50 then Grade = "D";
   ELSE if Average < 50 then Grade = "F";

SELECT (Grade);
   WHEN ("A") GP = 4;
   WHEN ("B") GP = 3;
   WHEN ("C") GP = 2;
   WHEN ("D") GP = 1;
   otherwise GP = 0;
END;

LABEL Score1='Midterm I' Score2='Midterm II' Score3='Final' GP='Grade
Point';
RUN;

PROC FORMAT;
 VALUE $class 'F'= 'Freshman' 'S'= 'Sophomore'
              'J'= 'Junior' 'Sr'='Seniour';
RUN;

PROC Print LABEL NOOBS U data=SASPRAC.roster;
     * NOOBS will remove Obs # and U will remove title, date, and page #;
  FORMAT Class $class.;
  TITLE "STAT101 Final Grade Roster";
RUN;
```

```
/* Example2_2 */


LIBNAME SASPRAC 'J:\SASCLASS';

DATA SASPRAC.sales (DROP = type p);

INPUT @1 type $ @;
  IF type="@" THEN DO;
     INPUT @3 Store $10.;
        Delete;
        END;
  RETAIN Store;
  ELSE IF type ne "@" THEN
     INPUT p $1-3 d 4-6 unit 7-11 @12 price Dollar7.2;

SELECT (p);
  WHEN ("011") prod = "CDR50";
  WHEN ("012") prod = "DVDR-";
  WHEN ("014") prod = "DVDDL";
  WHEN ("017") prod = "CDR100";
  WHEN ("020") prod = "USB2G";
  WHEN ("021") prod = "USB8G";
  OTHERWISE;
END;

sales=unit*price/d;

FORMAT price Dollar7.2 sales Dollar8.2;
LABEL prod="Product Name"  d="Survey Duration" unit="Unit sold"
price="Unit price" sales="Daily Sales";

DATALINES;
@ Kenwood
0110300023601200
0120600065203650
0140300102504190
0170600150702160
0201200056203650
0210900023410100
@ Westside
0110300017801290
0120600025603470
0140300087204090
0170600180701960
0201200114803290
0210900040209900
@ SouthHill
0110300030700900
0120600037403750
0140300087404290
0170600099802045
0201200078403880
0210900041509990
;
RUN;

PROC PRINT LABEL U NOOBS  DATA= SASPRAC.sales;
RUN;
```

```
/* Example2_3 (OUTPUT statement) */

DATA e2_3;
INFILE DATALINES MISSOVER;
INPUT (Name subj room) ($) @; OUTPUT;
INPUT (subj room) ($) @; OUTPUT;
INPUT (subj room) ($) ; OUTPUT;

DATALINES;
LEE          MATH101  LA5-100  MATH201  LA2-203  MED112   AS301
WOOD         STAT102  LA1-309  MATH358  AS402
BROWN        STAT440  LA5-307  MATH445  CBA-144  STAT590  CBA-159
SCOTT        MATH801  LA5-101  MATH560  ENG-200  STAT600  ENG-300
;
PROC PRINT; RUN;

DATA e2_3; SET e2_3;
IF missing(subj) THEN delete; RUN;
PROC PRINT; RUN;


/*********************************************************************
IN-CLASS PRACTICE

Write a DATA step which calculate the balance for next 30 years using
   (a) compound interest rate of 5% and compounding every year
   (b) compound interest rate of 5% and compounding every month
   (c) fixed annual interest rate of 5%
Use the principal of $5000.
*********************************************************************/
```