DATA Step Information - DO Loops

Within a DATA step operation, one or more do loops can be included. The syntax of a DO loop:

DO < options > ;
    SAS statements
END ;

That is, each DO statement must have an END statement that "closes" the loop. DO loops can be nested within each other, or can be encountered sequentially.

DO statements can have the following formats:

1. DO *index-variable = specification-1, specification-2, . . . , specification-n* ;
   Example:
```
DO i = 8;
DO day = "Mon" , "Wed", "Fri" ;
DO month = 3, 6, 9, 12 ;
```

2. DO *index-variable = start* TO *stop* ;
   Example:
```
DO k = 1 to 100 ;
```

3. DO *index-variable = start* TO *stop* BY *increment*;
   (If unspecified the increment is 1, as in the above example.)
   Example:
```
DO k = 1 to 100 BY 5;
DO m = 75 to 50 BY -1;
```

4. DO WHILE (*expression*) ;
   Example: `DO WHILE (n lt 5) ;`

5. DO UNTIL (*expression*) ;
   Example: `DO UNTIL (n >= 5) ;`

OUTPUT Statement

With many DO loops, you will need to use an OUTPUT statement. The OUTPUT statement tells SAS to write the current observation to the data set immediately rather than at the end of the DATA step.

The format of the OUTPUT statement can take one of the following forms:

1. OUTPUT ;
2. OUTPUT *dataset* ;
3. IF *condition* THEN OUTPUT <dataset> ;

Objective 1 – Rather than enter all of the variables in a data set, it may sometimes be appropriate to enter response variables only. When the data are in order, DO loops can assign the treatment and replication number.

```
DATA a ;

DO trt=1 TO 6 ;
   DO rep = 1 TO 2 ;
      INPUT y @@;
      OUTPUT ;
   END;
END;

DATALINES;

12 14 18 16 12 11 17 19 20 11 13 15

PROC PRINT DATA=a;

RUN;
QUIT;
```

Objective 2 - Run the above program without the OUTPUT statement and note the difference in the output.

Objective 3 – Create two data sets. The first data set, B, contains only information from the first three treatments. The second data set, C, contains only information from treatment levels 4, 5, and 6.
Modify the above program to read as follows:

```
DATA b c ;

DO trt=1 TO 6 ;
   DO rep = 1 TO 2 ;
      INPUT Y @@;
      IF trt > 3 THEN OUTPUT c;
      ELSE OUTPUT b ;
   END;
END;

DATALINES;

12 14 18 16 12 11 17 19 20 11 13 15

PROC PRINT DATA=b;

PROC PRINT DATA=c;

RUN;
QUIT;
```

Objective 4 – Run the following program with and without the OUTPUT statement and compare the results. Then run the program a third time using the OUTPUT statement and changing i<21 to i<=21. Note the difference.

```
DATA a;
i=12;

DO WHILE (i<21) ;
      i+1;
      OUTPUT;
END;

PROC PRINT DATA=a;
RUN;
QUIT;
```

Objective 5 – Compare this output with the output from objective 4. Note DO UNTIL evaluates the condition at the bottom of the loop, and the DO WHILE evaluates the condition at the top of the loop.

```
DATA a;
i=12;

DO UNTIL (i=21) ;
      i+1;
      OUTPUT;
END;

PROC PRINT DATA=a;
RUN;
QUIT;
```

Objective 6 – This program illustrates how a DO loop can be invoked using an IF – THEN statement.

```
DATA a ;
INPUT x @@;
IF x <= 3 THEN
    DO;
      remain = 3 - x;
    END;
ELSE overdue = x - 3 ;
OUTPUT;
DATALINES;
1 2 3 4 5 6
PROC PRINT DATA=a;
RUN;
QUIT;
```