```
1  ************************************************************;
2  *  Merging Tables with Non-matching Rows                   *;
3  ************************************************************;
4  *  Syntax and Examples                                     *;
5  *                                                          *;
6  *    DATA output-table;                                    *;
7  *        MERGE input-table1(IN=variable1)                  *;
8  *              input-table2(IN=variable2) ...;             *;
9  *        BY by-variable;                                   *;
10 *        IF expression;                                    *;
11 *    RUN;                                                  *;
12 ************************************************************;
13
14 /*Include matching rows only*/
15 data class2;
16     merge pg2.class_update(in=inUpdate)
17           pg2.class_teachers(in=inTeachers);
18     by name;
19     if inUpdate=1 and inTeachers=1;
20 run;
21
22 ************************************************************;
23 *  Demo                                                    *;
24 *  1) Highlight the first PROC SORT step and run the       *;
25 *     selected code. A table named STORM_FINAL_SORT is     *;
26 *     created, arranged by Season and Name. Because some   *;
27 *     storm names have been used more than once, unique    *;
28 *     storms are identified by both Season and Name.       *;
29 *  2) Open PG2.STORM_DAMAGE. Notice that it does not       *;
30 *     include the columns Season and Name, which are in    *;
31 *     STORM_FINAL_SORT. Season and Name must be derived    *;
32 *     from the Date and Event columns.                     *;
33 *  3) Examine the DATA step that creates a temporary       *;
34 *     table named STORM_DAMAGE. SAS functions are used to  *;
35 *     create Season and Name with values that match the    *;
36 *     values in the STORM_FINAL_SORT table. Highlight the  *;
37 *     DATA step and the PROC SORT step that follows it,    *;
38 *     and run the selection.                               *;
39 *  4) Complete the final DATA step to merge the sorted     *;
40 *     tables by Season and Name. Highlight the DATA step   *;
41 *     and run the selection. Notice in the output table    *;
42 *     that row 4 is storm Allen, which is included in the  *;
43 *     STORM_DAMAGE table. Therefore, each of the columns   *;
44 *     has values read from both input tables. Most of the  *;
45 *     values in the Cost and Deaths columns are missing    *;
46 *     because those storms are not found in the            *;
47 *     STORM_DAMAGE table.                                  *;
48 *  5) Use the IN= data set option after the STORM_DAMAGE   *;
49 *     table to create a temporary variable named inDamage  *;
50 *     that flags rows where Season and Name were read      *;
51
52
```

```
*     from the STORM_DAMAGE table. Add a subsetting IF    *;
*     statement to write the 38 rows from STORM_DAMAGE    *;
*     and the corresponding data from STORM_FINAL_SORT to *;
*     the output table. Highlight the DATA step and run   *;
*     the selection.                                      *;
*********************************************************;

proc sort data=pg2.storm_final out=storm_final_sort;
    by Season Name;
run;

data storm_damage;
    set pg2.storm_damage;
    Season=Year(date);
    Name=upcase(scan(Event, -1));
    format Date date9. Cost dollar16. deaths comma5.;
    drop event;
run;

proc sort data=storm_damage;
    by Season Name;
run;

data damage_detail;
    merge storm_final_sort storm_damage(in=inDamage);
    by Season Name;
    if inDamage=1;
    keep Season Name BasinName MaxWindMPH MinPressure Cost Deaths;
run;
```