Exercises for Chapter 3: SQL in PL/SQL

The Labs below provide you with exercises and suggested answers with discussion related to how those answers resulted. The most important thing to realize is whether your answer works. You should figure out the implications of the answers here and what the effects are from any different answers you may come up with.

In the chapter discussion, you learned how to use numerous SQL techniques in a PL/SQL block. First, you learned how to use `SELECT INTO` to generate values for a variable. Then you learned the various DML methods, including the use of a sequence. Finally, you learned how to manage transactions by using savepoints. Complete the following projects by writing the code for each step and running it and then going on to the next step.

1. Create a table called `CHAP4` with two columns; one is ID (a number) and the second is `NAME`, which is a `VARCHAR2 (20)`.
   Answer: The answer should look similar to the following:

```
PROMPT Creating Table 'CHAP4'
CREATE TABLE chap4
   (id   NUMBER,
    name VARCHAR2(20));
```

2. Create a sequence called `CHAP4_SEQ` that increments by units of 5.
   Answer: The answer should look similar to the following:

```
PROMPT Creating Sequence 'CHAP4_SEQ'
CREATE SEQUENCE chap4_seq
   NOMAXVALUE
   NOMINVALUE
   NOCYCLE
   NOCACHE;
```

3. Write a PL/SQL block that performs the following in this order:
   a. Declares two variables, one for the `v_name` and one for `v_id`. The `v_name` variable can be used throughout the block for holding the name that will be inserted, realize that the value will change in the course the block.
   b. The block then inserts into the table the name of the student that is enrolled in the most classes and uses a sequence for the ID; afterward there is `SAVEPOINT A`.
   c. Then the student with the least enrollments is inserted; afterward there is `SAVEPOINT B`.

d. Then the instructor who is teaching the maximum number of courses is inserted in the same way. Afterward there is `SAVEPOINT C`.

e. Using a `SELECT INTO` statement, hold the value of the instructor in the variable `v_id`.

f. Undo the instructor insert by the use of rollback.

g. Insert the instructor teaching the least amount of courses, but do not use the sequence to generate the ID; instead use the value from the first instructor, whom you have since undone.

h. Now insert the instructor teaching the most number of courses and use the sequence to populate his ID.

i. Add `DBMS_OUTPUT` throughout the block to display the values of the variables as they change. (This is a good practice for debugging.)

Answer: The script should look similar to the following:

```
DECLARE
    v_name student.last_name%TYPE;
    v_id   student.student_id%TYPE;
BEGIN
   BEGIN
      -- A second block is used to capture the possibility of
      -- multiple students meeting this requirement.
      -- The exception section will handles this situation
      SELECT s.last_name
        INTO v_name
        FROM student s, enrollment e
       WHERE s.student_id = e.student_id
      HAVING COUNT(*) = (SELECT MAX(COUNT(*))
                           FROM student s, enrollment e
                          WHERE s.student_id = e.student_id
                         GROUP BY s.student_id)
      GROUP BY s.last_name;
   EXCEPTION
      WHEN TOO_MANY_ROWS THEN
         v_name := 'Multiple Names';
   END;

   INSERT INTO CHAP4
   VALUES (CHAP4_SEQ.NEXTVAL, v_name);
   SAVEPOINT A;

   BEGIN
      SELECT s.last_name
        INTO v_name
        FROM student s, enrollment e
       WHERE s.student_id = e.student_id
      HAVING COUNT(*) = (SELECT MIN(COUNT(*))
                           FROM student s, enrollment e
                          WHERE s.student_id = e.student_id
                         GROUP BY s.student_id)
      GROUP BY s.last_name;
```

```
EXCEPTION
   WHEN TOO_MANY_ROWS THEN
      v_name := 'Multiple Names';
END;

INSERT INTO CHAP4
VALUES (CHAP4_SEQ.NEXTVAL, v_name);
SAVEPOINT B;

BEGIN
   SELECT i.last_name
     INTO v_name
     FROM instructor i, section s
    WHERE s.instructor_id = i.instructor_id
   HAVING COUNT(*) = (SELECT MAX(COUNT(*))
                        FROM instructor i, section s
                       WHERE s.instructor_id = i.instructor_id
                      GROUP BY i.instructor_id)
   GROUP BY i.last_name;
EXCEPTION
   WHEN TOO_MANY_ROWS THEN
      v_name := 'Multiple Names';
END;

SAVEPOINT C;

BEGIN
   SELECT instructor_id
     INTO v_id
     FROM instructor
    WHERE last_name = v_name;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      v_id := 999;
END;

INSERT INTO CHAP4
VALUES (v_id, v_name);
ROLLBACK TO SAVEPOINT B;

BEGIN
   SELECT i.last_name
     INTO v_name
     FROM instructor i, section s
    WHERE s.instructor_id = i.instructor_id
   HAVING COUNT(*) = (SELECT MIN(COUNT(*))
                        FROM instructor i, section s
                       WHERE s.instructor_id = i.instructor_id
                      GROUP BY i.instructor_id)
   GROUP BY i.last_name;
EXCEPTION
  WHEN TOO_MANY_ROWS THEN
    v_name := 'Multiple Names';
END;
```

```
  INSERT INTO CHAP4
  VALUES (v_id, v_name);

  BEGIN
     SELECT i.last_name
       INTO v_name
       FROM instructor i, section s
      WHERE s.instructor_id = i.instructor_id
     HAVING COUNT(*) = (SELECT MAX(COUNT(*))
                          FROM instructor i, section s
                         WHERE s.instructor_id = i.instructor_id
                        GROUP BY i.instructor_id)
     GROUP BY i.last_name;
  EXCEPTION
     WHEN TOO_MANY_ROWS THEN
        v_name := 'Multiple Names';
  END;

  INSERT INTO CHAP4
  VALUES (CHAP4_SEQ.NEXTVAL, v_name);
END;
```