Exercise 1. Write a nested cursor in which the parent cursor gathers information about each section of a course. The child cursor counts the enrollment. The only output is one line for each course, with the course name, section number, and total enrollment.

SOLUTION:

```
SET SERVEROUTPUT ON
DECLARE
   CURSOR parent_cursor IS
      SELECT course_no, description
      FROM course
      WHERE course_no != 0;
   CURSOR child_cursor(param_course_no IN course.course_no%type) IS
      SELECT st.section_no, COUNT(*) total_per_course
      FROM section st, enrollment et
      WHERE st.course_no = param_course_no
                 AND st.section_id = et.section_id
      GROUP BY st.section_no;
BEGIN
   FOR i IN parent_cursor LOOP
      FOR j IN child_cursor(i.course_no) LOOP
         DBMS_OUTPUT.PUT_LINE(i.description||' ~~Section number
'||j.section_no||' has '||j.total_per_course||' students!~~');
      END LOOP;
   END LOOP;
END;
```

OUTPUT:

```
Data Bases 2 ~~Section number 10 has 2
students!~~

Statistics ~~Section number 30 has 1
students!~~

Computer Networks ~~Section number 80 has
1 students!~~

Software Engineering ~~Section number 90
has 1 students!~~
```

Exercise 2. Write an anonymous PL/SQL block that finds all the courses that have at least one section that is at its maximum enrollment. If no courses meet that criterion, pick two courses and create that situation for each.
a) For each of those courses, add another section. The instructor for the new section should be taken from the existing records in the instructor table. Use the instructor who is signed up to teach the fewest courses. Handle the fact that, during the execution of your program, the instructor teaching the most courses may change.
b) Use any exception-handling techniques you think are useful to capture error conditions.

SOLUTION:

```
SET SERVEROUTPUT ON
DECLARE
   intructor_min_courses instructor.instructor_id%TYPE;
   new_section_ID section.section_id%TYPE;
   new_number_section section.section_no%TYPE := 0;
   --create a cursor to point to the courses with at least one section at its capacity limit
   --we need to use the enrollment and section tables which have in common "section_id" column
   CURSOR courses_maxed IS
      SELECT DISTINCT st.course_no
      FROM section st
```

OUTPUT:

Either the message from the exception statement or rows are inserted in the table

```
    WHERE st.capacity =(SELECT COUNT(section_id)
          FROM enrollment et
          WHERE et.section_id = st.section_id);
BEGIN
  FOR course_maxed IN courses_maxed LOOP
    --we have a list with courses that have at least 1 full section
    --for each we will add one new section who will be taught by the instructor with less courses
    SELECT instructor_id
    INTO intructor_min_courses
    FROM instructor
    WHERE EXISTS( SELECT NULL
            FROM section
            WHERE section.instructor_id = instructor.instructor_id
            GROUP BY instructor_id
            HAVING COUNT(*) = (SELECT MIN(COUNT(*))
                      FROM section
                      WHERE instructor_id IS NOT NULL
                      GROUP BY instructor_id)
          )
          AND ROWNUM = 1;

    --we need to assign an ID to the new section that will be created
    --we can take the maximum existing ID and maximum existing number and add some numeric
integer (like 10)
    SELECT MAX(section_id)+10, MAX(section_no)+10
    INTO new_section_ID, new_number_section
    FROM section;

    INSERT INTO section(section_id, course_no, section_no, instructor_id, created_by,
created_date, modified_by, modified_date)
    VALUES (new_section_ID, course_maxed.course_no, new_number_section,
intructor_min_courses, 'Kovaci', '07-April-2020', 'Kovaci', '07-April-2020');
    COMMIT;
  END LOOP;

  EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No data!');

    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Some errors!');
END;
```

Exercise 3. Construct 3 cursors. The first one, cursor c_student takes no parameters and is a collection of students with a last name beginning with J. The second one c_course takes in the parameter of student_ID to generate a list of courses that student is taking. The third one, c_grade takes in two parameters, section_id and student_id. In this way it can generate an average of the different grade types (quizzes, homework, final, etc.) for that student for that course. Display the student name for the first coursor. The second cursor takes the parameter of student_id from the first cursor. Only the description of the course is displayed. The third cursor takes in the parameter of section_id from the second cursor and student_id from the first cursor. The grades are then displayed.

SOLUTION:

```
SET SERVEROUTPUT ON
DECLARE
  CURSOR c_student IS
    SELECT *
    FROM student
    WHERE last_name LIKE 'P%';

  CURSOR c_course(param_student_ID IN student.student_id%type) IS
    SELECT ct.description, st.section_id
    FROM course ct, section st, enrollment et
    WHERE et.student_id = param_student_ID
        AND ct.course_no = st.course_no
        AND st.section_id = et.section_id;

  CURSOR c_grade(param_section_ID IN section.section_id%type,
param_student_ID IN student.student_id%type) IS
    SELECT gtype.description grd_desc, TO_CHAR (AVG(gt.numeric_grade),
'999.99') final_grade
    FROM enrollment et, grade gt, grade_type gtype
    WHERE et.section_id = param_section_ID
        AND et.student_id = gt.student_id
        AND et.student_id = param_student_ID
        AND et.section_id = gt.section_id
        AND gt.grade_type_code = gtype.grade_type_code
    GROUP BY gtype.description;

BEGIN
  FOR i IN c_student LOOP
    DBMS_OUTPUT.PUT_LINE(i.first_name||' '||i.last_name);
    FOR j in c_course(i.student_id) LOOP
      DBMS_OUTPUT.PUT_LINE(' ~~ '||j.description);
      FOR k in c_grade(j.section_id, i.student_id) LOOP
        DBMS_OUTPUT.PUT_LINE(' ~~ '||k.final_grade);
      END LOOP;
    END LOOP;
  END LOOP;
END;
```

OUTPUT:

```
Ana Pup
 ~~ Data Bases 2
 ~~   30.00
 ~~   30.00
 ~~   40.00

Emanuel Pop
 ~~ Statistics
 ~~   30.00
 ~~   10.00
 ~~   60.00

Anita Pupu
 ~~ Data Bases 2

Emanuel Popa
 ~~ Statistics
```