Assignment 2: SQL (Spring 2019)

Instructor: Zhaonian Zou (`znzou@hit.edu.cn`)

Name: ____________ Student ID: ____________ Grade: ______

Question	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total
Score																

Notes

- **Print the assignment on A4 paper and answer the questions.**
- **Assignment due date: March 20, 2019.**

Questions

Write the following queries in SQL, using the university schema described in the attached file "`university.pdf`". We require you actually run these queries on a database, using the sample data that we provide on the Web site of the book, `https://www.db-book.com`. Instructions for setting up a database, and loading sample data, are provided on the above Web site.

1. (3 points) Find the titles of courses in the Comp. Sci. department that have 3 credits.
2. (3 points) Find the IDs of all students who were taught by an instructor named Einstein; make sure there are no duplicates in the result.
3. (3 points) Find the highest salary of any instructor.
4. (3 points) Find the enrollment of each section that was offered in Autumn 2009.
5. (3 points) Find the maximum enrollment, across all sections, in Autumn 2009.
6. (3 points) Find the IDs and names of all students who have not taken any course offering before Spring 2009.
7. (3 points) Display a list of all instructors, showing their ID, name, and the number of sections that they have taught. Make sure to show the number of sections as 0 for instructors who have not taught any section. Your query should use an outerjoin, and should not use scalar subqueries.
8. (3 points) Display the list of all course sections offered in Spring 2010, along with the names of the instructors teaching the section. If a section has more than one instructor, it should appear as many times in the result as it has instructors. If it does not have any instructor, it should still appear in the result with the instructor name set to "—".
9. (3 points) Display the list of all departments, with the total number of instructors in each department, without using scalar subqueries. Make sure to correctly handle departments with no instructors.

Write the following inserts, deletes or updates in SQL, using the university schema.

11. (2 points) Create a new course "CS-001", titled "Weekly Seminar", with 0 credits, and create a section of this course in Autumn 2009, with `sec_id` of 1.

12. (2 points) Enroll every student in the Comp. Sci. department in the above section.
13. (2 points) Delete enrollments in the above section where the student's name is Chavez.
14. (2 points) Delete the course CS-001. What will happen if you run this delete statement without first deleting offerings (sections) of this course.
15. (2 points) Increase the salary of each instructor in the Comp. Sci. department by 10%.

Answers

1. (3 points) Find the titles of courses in the Comp. Sci. department that have 3 credits.

```
SELECT title FROM course
WHERE dept_name = 'Comp. Sci.' AND credits = 3;
```

2. (3 points) Find the IDs of all students who were taught by an instructor named Einstein; make sure there are no duplicates in the result.

```
SELECT takes.ID FROM instructor NATURAL JOIN teaches
JOIN takes USING (course_id, sec_id, semester, year)
WHERE instructor.name = 'Einstein';
```

3. (3 points) Find the highest salary of any instructor.

```
SELECT MAX(salary) FROM instructor;
```

4. (3 points) Find the IDs of all instructors earning the highest salary (there may be more than one with the same salary).
   Solution 1:

```
SELECT ID FROM instructor
WHERE salary = (SELECT MAX(salary) FROM instructor);
```

   Solution 2:

```
SELECT ID FROM instructor
WHERE salary >= ALL (SELECT salary FROM instructor);
```

5. (3 points) Find the enrollment of each section that was offered in Autumn 2009.
   Solution 1:

```
SELECT course_id, sec_id, semester, year, COUNT(*) FROM takes
WHERE semester = 'Fall' and year = 2009
GROUP BY course_id, sec_id, semester, year;
```

   Solution 2:

```
SELECT course_id, sec_id, semester, year, COUNT(ID)
FROM section NATURAL LEFT JOIN takes
WHERE semester = 'Fall' and year = 2009
GROUP BY course_id, sec_id, semester, year;
```

6. (3 points) Find the maximum enrollment, across all sections, in Autumn 2009.
   Solution 1:

```
SELECT MAX(enrollment) FROM
(SELECT course_id, sec_id, semester, year, COUNT(*) AS enrollment FROM takes
WHERE semester = 'Fall' and year = 2009
GROUP BY course_id, sec_id, semester, year) AS section_enroll;
```

Solution 2:

```
SELECT MAX(enrollment) FROM
(SELECT course_id, sec_id, semester, year, COUNT(ID) AS enrollment
FROM section NATURAL LEFT JOIN takes
WHERE semester = 'Fall' and year = 2009
GROUP BY course_id, sec_id, semester, year) AS section_enroll;
```

7. (3 points) Find the IDs and names of all students who have not taken any course offering before Spring 2009. Solution 1:

```
SELECT ID, name FROM student
WHERE ID NOT IN (SELECT ID FROM takes WHERE year < 2009);
```

Solution 2:

```
SELECT ID FROM student EXCEPT
SELECT ID FROM takes WHERE year < 2009;
```

8. (3 points) Display a list of all instructors, showing their ID, name, and the number of sections that they have taught. Make sure to show the number of sections as 0 for instructors who have not taught any section. Your query should use an outerjoin, and should not use scalar subqueries.

```
SELECT ID, name, COUNT(course_id)
FROM instructor NATURAL LEFT JOIN teaches
GROUP BY ID, name;
```

9. (3 points) Display the list of all course sections offered in Spring 2010, along with the names of the instructors teaching the section. If a section has more than one instructor, it should appear as many times in the result as it has instructors. If it does not have any instructor, it should still appear in the result with the instructor name set to "—".

```
(SELECT course_id, sec_id, semester, year, name
FROM teaches NATURAL JOIN instructor
WHERE semester = 'Spring' AND year = 2010)
UNION ALL
(SELECT course_id, sec_id, semester, year, '-'
FROM section NATURAL LEFT JOIN teaches
WHERE semester = 'Spring' AND year = 2010 AND ID IS NULL);
```

10. (3 points) Display the list of all departments, with the total number of instructors in each department, without using scalar subqueries. Make sure to correctly handle departments with no instructors.

```
SELECT dept_name, COUNT(ID)
FROM department NATURAL LEFT JOIN instructor
GROUP BY dept_name;
```

11. (2 points) Create a new course "CS-001", titled "Weekly Seminar", with 0 credits, and create a section of this course in Autumn 2009, with `sec_id` of 1.

```
INSERT INTO course (course_id, title, credits)
VALUES ('CS-001', 'Weekly Seminar', 0);
INSERT INTO section (course_id, sec_id, semester, year)
VALUES ('CS-001', '1', 'Fall', 2009);
```

12. (2 points) Enroll every student in the Comp. Sci. department in the above section.

```
INSERT INTO takes (ID, course_id, sec_id, semester, year)
(SELECT ID, 'CS-001', '1', 'Fall', 2009 FROM student);
```

13. (2 points) Delete enrollments in the above section where the student's name is Chavez.

    Solution 1:

```
DELETE FROM takes WHERE 'Chavez' = (SELECT name FROM student WHERE student.ID = takes.ID);
```

Solution 2:

```
DELETE FROM takes WHERE ID IN (SELECT student.ID FROM student WHERE name = 'Chavez');
```

Solution 3:

```
DELETE FROM takes WHERE EXISTS
(SELECT * FROM student WHERE name = 'Chavez' AND student.ID = takes.ID);
```

14. (2 points) Delete the course CS-001. What will happen if you run this delete statement without first deleting offerings (sections) of this course.

```
DELETE FROM course WHERE course_id = 'CS-001';
```

The tuples in `takes` that correspond to the course CS-001 will be deleted in cascade.

15. (2 points) Increase the salary of each instructor in the Comp. Sci. department by 10%.

```
UPDATE instructor SET salary = 1.1 * salary;
```