ROLL NO-106122112

B.TECH. COMPUTER SCIENCE AND ENGINEERING CSLR51– Database Management Systems Laboratory #Session: 06 || Date: 05/09/2024

Question 1

a. Find the titles of courses in the CSE department that have 3 credits. `SELECT title FROM course WHERE dept_name = 'CSE' AND credits = 3;` **Output:**

```
+-------------------+
| title |
+-------------------+
| Data Structures |
| Operating Systems |
+-------------------+
```

b. Find the highest salary of any professor.

```
SELECT MAX(salary) AS highest_salary FROM professor;
```

Output:

```
+----------------+
| highest_salary |
+----------------+
| 150000 |
+----------------+
```

c. Find all professors earning the highest salary (there may be more than one with the same salary).

```
SELECT name FROM professor
WHERE salary =
(SELECT MAX(salary) FROM professor);
```

Output:
```
+----------------+
| name |
+----------------+
| Dr. Tony Stark |
+----------------+
```

d. Find the maximum enrollment, across all sections, in Fall 2020.

```
SELECT MAX(enrollment_count) FROM (
    SELECT COUNT(*) AS enrollment_count
    FROM takes
    WHERE semester = 'Fall' AND year = 2020
    GROUP BY course_id, sec_id
) AS enrollments;
```

Output:

```
+-----------------------+
| MAX(enrollment_count) |
+-----------------------+
| 1 |
+-----------------------+
```

e. Find the enrollment of each section that was offered in Spring 2019.

```
SELECT course_id, sec_id, COUNT(*) AS enrollment
FROM takes
WHERE semester = 'Spring' AND year = 2019
GROUP BY course_id, sec_id;
```

Output:

```
+-----------+--------+------------+
| course_id | sec_id | enrollment |
+-----------+--------+------------+
| CS-103 | 1 | 1 |
| ECE-201 | 1 | 1 |
| ME-301 | 1 | 1 |
+-----------+--------+------------+
```

f. Find the IDs and names of all students who have not taken any course offering before Spring 2013.

```
SELECT pID, name
FROM student
WHERE pID NOT IN (
    SELECT sID
    FROM takes
    WHERE (semester = 'Spring' AND year < 2013) OR year <
2013 );
```

Output:

```
+-----+---------+
| pID | name |
+-----+---------+
| 101 | Alice |
| 102 | Bob |
| 103 | Charlie |
| 104 | David |
| 105 | Eve |
+-----+---------+
```

g. Find the lowest, across all departments, of the per-department maximum salary computed by the preceding query.

```
SELECT MIN(max_salary)
FROM (
    SELECT MAX(salary) AS max_salary
    FROM professor
    GROUP BY dept_name
) AS dept_salaries;
```

Output:

```
+-----------------+
| MIN(max_salary) |
+-----------------+
| 110000 |
+-----------------+
```

h. Create a new course “CS-001”, titled “Weekly Seminar”, with 1 credit.

```
INSERT INTO course VALUES ('CS-001', 'Weekly Seminar', 'CSE',

1);
```

i. 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';
```

```
If you run this delete statement without first deleting related sections
(offerings) from the `section` table, it will violate the foreign key
constraint and result in an error.
```

j. Display the list of all course sections offered in Spring 2022, along with the names of the professors teaching the section. If a section has more than one professor, it should appear as many times in the result as it has professor. If it does not have any professors, it should still appear in the result with the professor name set to “-”.

```
SELECT section.course_id, section.sec_id, section.semester, section.year,
COALESCE(professor.name, '-') AS professor_name
FROM section
LEFT JOIN teaches ON section.course_id = teaches.course_id
AND section.sec_id = teaches.sec_id
 AND section.semester = teaches.semester
 AND section.year = teaches.year
LEFT JOIN professor ON teaches.pID = professor.pID
WHERE section.semester = 'Spring' AND section.year = 2022;
```

Output:

```
Empty set
```

k. Find the professor ID, name, dept name, and salary for professors whose salary is greater than 50,000.

```
SELECT pID, name, dept_name, salary
FROM professor
WHERE salary > 50000;
```

Output:

```
+-----+-----------------+----------+--------+
| pID | name | dept_name | salary |
+-----+-----------------+----------+--------+
| 1 | Dr. John Doe | CSE | 120000 |
| 2 | Dr. Jane Smith | ECE | 110000 |
| 3 | Dr. Sam Wilson | ME | 130000 |
| 4 | Dr. Bruce Banner | CE | 140000 |
| 5 | Dr. Tony Stark | EE | 150000 |
+-----+-----------------+----------+--------+
```

l. Find the names of all professors in the Chemical Engineering department together with the course id of all courses they teach.

```
SELECT professor.name, teaches.course_id
FROM professor
JOIN teaches ON professor.pID = teaches.pID
WHERE professor.dept_name = 'CE';
```

Output:

```
+-----------------+-----------+
| name | course_id |
+-----------------+-----------+
| Dr. Bruce Banner | ME-301 |
+-----------------+-----------+
```

m. Find the set of all courses taught in the Fall 2020 semester, the Spring 2021 semester, or both.

```
SELECT DISTINCT course_id
FROM section
WHERE (semester = 'Fall' AND year = 2020)
 OR (semester = 'Spring' AND year = 2021);
```

Output:

```
+-----------+
| course_id |
+-----------+
| CS-101 |
| CS-102 |
+-----------+
```

n. Find the names of all professors whose department is in the ‘ORION’ building.

```
SELECT professor.name
FROM professor
JOIN department ON professor.dept_name =
department.dept_name WHERE department.building = 'Orion';
```

Output:

```
+--------------+
| name |
+--------------+
| Dr. John Doe |
+--------------+
```

o. Find the set of all courses taught in the Fall 2020 semester, or in the Spring 2019 semester, or both.

```
SELECT DISTINCT course_id
FROM section
WHERE (semester = 'Fall' AND year = 2023)
 OR (semester = 'Spring' AND year = 2019);
```

Output:

```
+-----------+
| course_id |
+-----------+
| CS-103 |
| ECE-201 |
| ME-301 |
+-----------+
```

p. Find the set of all courses taught in the Fall 2020 semester, but not in the Spring 2019 semester.

```
SELECT DISTINCT course_id
FROM section
WHERE (semester = 'Fall' AND year = 2020)
 AND course_id NOT IN (
 SELECT course_id
 FROM section
 WHERE semester = 'Spring' AND year = 2019
);
```

Output:

```
+-----------+
| course_id |
+-----------+
| CS-101 |
| CS-102 |
+-----------+
```

q. Find the IDs of all students who were taught by an professor named Tejaswi; make sure there are no duplicates in the result.

```
SELECT DISTINCT takes.sID
FROM takes
JOIN teaches ON takes.course_id = teaches.course_id
 AND takes.sec_id = teaches.sec_id
 AND takes.semester = teaches.semester
 AND takes.year = teaches.year
JOIN professor ON teaches.pID = professor.pID
WHERE professor.name = 'Tejaswi';
```

Output:

```
Empty set
```

r. Find the names of all students who have taken at least one Computer Science course; make sure there are no duplicate names in the result.

```
SELECT DISTINCT student.name
FROM student
JOIN takes ON student.pID = takes.sID
JOIN course ON takes.course_id = course.course_id
WHERE course.dept_name = 'CSE';
```

Output:

```
+---------+
| name |
+---------+
| Alice |
| Bob |
| Charlie |
+---------+
```

s. For each department, find the maximum salary of professors in that department. You may assume that every department has at least one professor.

```
SELECT dept_name, MAX(salary) AS max_salary
FROM professor
GROUP BY dept_name;
```

Output:

```
+-----------+-----------+
| dept_name | max_salary |
+-----------+-----------+
| CE | 140000 |
| CSE | 120000 |
| ECE | 110000 |
| EE | 150000 |
| ME | 130000 |
+-----------+-----------+
```

t. Display a list of all professors, showing their ID, name, and the number of sections

that they have taught. Make sure to show the number of sections as 0 for professors who have not taught any section. Your query should use an outerjoin, and should not use scalar subqueries.

```
SELECT professor.pID, professor.name, COUNT(teaches.course_id)
AS num_sections
FROM professor
LEFT JOIN teaches ON professor.pID = teaches.pID
GROUP BY professor.pID, professor.name;
```

Output:

```
+-----+-----------------+-------------+
| pID | name | num_sections |
+-----+-----------------+-------------+
| 1 | Dr. John Doe | 0 |
| 2 | Dr. Jane Smith | 0 |
| 3 | Dr. Sam Wilson | 0 |
| 4 | Dr. Bruce Banner | 0 |
| 5 | Dr. Tony Stark | 0 |
+-----+-----------------+-------------+
```

u. Write the same query as above, but using a scalar subquery, without outer join.

```
SELECT pID, name,
 (SELECT COUNT(*)
 FROM teaches
 WHERE teaches.pID = professor.pID) AS num_sections
FROM professor;
```

Output:

```
+------+-----------------+-------------+
| pID | name | num_sections |
+------+-----------------+-------------+
| 1 | Dr. John Doe | 0 |
| 2 | Dr. Jane Smith | 0 |
| 3 | Dr. Sam Wilson | 0 |
| 4 | Dr. Bruce Banner | 0 |
| 5 | Dr. Tony Stark | 0 |
+------+-----------------+-------------+
```

v. Find all students who have taken all courses offered in the Biology department.

```
SELECT sID
FROM takes
WHERE course_id IN (
 SELECT course_id
 FROM course
 WHERE dept_name = 'Biology'
)
GROUP BY sID
HAVING COUNT(DISTINCT course_id) = (SELECT COUNT(course_id) FROM course WHERE
dept_name = 'Biology');
```

Output:

```
Empty set
```

w. Create your own query:
Find the name of the student who has the maximum number of total credits in the Computer Science department, along with their total credit count.

```
SELECT student.name, student.tot_cred
FROM student
WHERE student.dept_name = 'CSE'
ORDER BY tot_cred DESC
LIMIT 1;
```

Output:

```
+-------+----------+
 | name | tot_cred |
+-------+----------+
 | Alice | 15 |
+-------+----------+
```

x.i) Create a new user ‘testuser’ on the localhost.

```
CREATE USER 'testuser'@'localhost' IDENTIFIED BY 'password';
```

ii) Grant all privileges for the testuser on the University database you have created.

```
GRANT ALL PRIVILEGES ON University.* TO
'testuser'@'localhost';
```

iii) Revoke all the privileges given to testuser.

```
REVOKE ALL PRIVILEGES ON University.* FROM 'testuser'@'localhost';
```

y.i) Create a new user 'testuser1' on the localhost.

```
CREATE USER 'testuser1'@'localhost' IDENTIFIED BY
'password';
```

ii) Grant only select privileges for the testuser1 on the Student table.

```
GRANT SELECT ON University.student TO 'testuser1'@'localhost';
```

iii) Revoke the select privileges for the testuser1 on the Student table.

```
REVOKE SELECT ON University.student FROM
'testuser1'@'localhost';
```