DBMS Lab - Session-6

106122076 CSE-B
Lohesh S U

1. Relational Database Design – University Schema
Execute the following Queries in SQL over the University Schema given below.

classroom(building, room_number, capacity)
department(dept_name, building, budget)
course(course id, title, dept_name, credits)
professor(pID, name, dept_name, salary)
section(course_id, sec_id, semester, year, building, room_number, time_slot_id)
teaches(pID, course_id, sec_id, semester, year)
student(pID, name, dept_name, tot_cred)
takes(sID, course_id, sec_id, semester, year, grade)
guide(sID, pID)
time_slot(time_slot_id, day, start_time, end_time)
prereq(course_id, prere_ id)

Before inserting values, please go through the questions below which shall facilitate you to choose appropriate values for the fields in the table. Populate each table with a minimum of 5 records and ensure an empty set is not returned for any query. State and Make Valid Assumptions where required. Use of Views is not permitted unless its explicitly mentioned in the Query.

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                    |
+--------------------------+
| Introduction to Programming|
+--------------------------+
```

b. Find the highest salary of any professor.

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

Output:-

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

```
|        95000  |
+---------------+
```

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

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

Output:-

```
+-----------+--------+
| name      | salary |
+-----------+--------+
| Dr. Williams | 95000 |
+-----------+--------+
```

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

```
SELECT MAX(enrollment) AS max_enrollment
FROM (
   SELECT COUNT(sID) AS enrollment
   FROM takes
   WHERE semester = 'Fall' AND year = 2020
   GROUP BY course_id, sec_id
) AS section_enrollments;
```

Output:-

```
+----------------+
| max_enrollment |
+----------------+
|              3 |
+----------------+
```

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

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

Output:-

```
+-----------+--------+------------+
| course_id | sec_id | enrollment |
+-----------+--------+------------+
```

```
| CS-101    |     1 |         2 |
| EE-101    |     1 |         1 |
+-----------+-------+-----------+
```

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

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

Output:-

```
+------+---------+
| pID  | name    |
+------+---------+
| 1001 | Alice   |
| 1002 | Bob     |
| 1003 | Charlie |
| 1004 | David   |
+------+---------+
```

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

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

Output:-

```
+------------------+
| lowest_max_salary |
+------------------+
|            80000 |
+------------------+
```

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

```
INSERT INTO course (course_id, title, dept_name, credits)
VALUES ('CS-001', 'Weekly Seminar', 'CSE', 1);
```

Output:-

```
Query OK, 1 row affected (0.01 sec)
```

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';
```

Output:-

```
Query OK, 1 row affected (0.01 sec)
```

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, 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 (0.00 sec)
```

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. Smith | CSE       | 90000  |
|    2 | Dr. Johnson | ECE     | 80000  |
```

```
|    3 | Dr. Williams | MECH   | 95000  |
|    4 | Dr. Brown | CSE       | 90000  |
+------+-----------+-----------+--------+
```

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 = 'Chemical Engineering';
```

Output:-

Empty set (0.00 sec)

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

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

Output:-

Empty set (0.00 sec)

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:-

Empty set (0.00 sec)

o. Find the set of all courses taught in the Fall 2023 semester, or in the Spring 2022 semester, or both.

```
SELECT DISTINCT course_id
FROM section
WHERE (semester = 'Fall' AND year = 2023)
   OR (semester = 'Spring' AND year = 2022);
```

Empty set (0.00 sec)

p. Find the set of all courses taught in the Fall 2021 semester, but not in the Spring 2019 semester.

```
SELECT DISTINCT course_id
FROM section
WHERE semester = 'Fall' AND year = 2021
AND course_id NOT IN (
   SELECT course_id
   FROM section
   WHERE semester = 'Spring' AND year = 2019
);
```

Empty set (0.00 sec)

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';
```

Empty set (0.00 sec)

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';
```

```
+---------+
| name    |
+---------+
| Alice   |
| David   |
+---------+
```

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;
```

```
+-----------+------------+
| dept_name | max_salary |
+-----------+------------+
| CSE       |      90000 |
| ECE       |      80000 |
| MECH      |      95000 |
+-----------+------------+
```

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 outer join, 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;
```

```
+------+-----------+--------------+
| pID  | name      | num_sections |
+------+-----------+--------------+
|    1 | Dr. Smith |            2 |
|    2 | Dr. Johnson |          1 |
|    3 | Dr. Williams |        0  |
|    4 | Dr. Brown |            0 |
+------+-----------+--------------+
```

u. Write the same query as above, but using a scalar subquery, without outerjoin. v. Find all students who have taken all courses offered in the Biology department. w. Create your own query: define what you want to do in English, then write the query in SQL. Make it as difficult as you wish, the harder the better.

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

```
+------+-----------+--------------+
```

```
| pID  | name      | num_sections |
+------+-----------+--------------+
|    1 | Dr. Smith |            2 |
|    2 | Dr. Johnson |          1 |
|    3 | Dr. Williams |        0  |
|    4 | Dr. Brown |            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(DISTINCT course_id)
   FROM course
   WHERE dept_name = 'Biology'
);

Empty set (0.00 sec)
```

w. Create your own query: define what you want to do in English, then write the query in SQL. Make it as difficult as you wish, the harder the better. Find the number of students enrolled in courses taught by professors with salaries greater than 80,000.

```
SELECT COUNT(DISTINCT takes.sID) AS num_students
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.salary > 80000;

+--------------+
| num_students |
+--------------+
|            3 |
+--------------+
```