CSLR-51 DBMS - Session 6

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 the 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.

Schema:

```
CREATE TABLE classroom(
   building VARCHAR(15),
   room_number INT,
   capacity INT,
   PRIMARY KEY(building,room_number)
);

CREATE TABLE department(
   dept_name VARCHAR(30) PRIMARY KEY,
   building VARCHAR(15),
   budget INT
);

CREATE TABLE course(
   course_id VARCHAR(15) PRIMARY KEY,
   title VARCHAR(20),
```

```
    dept_name VARCHAR(30),
    credits INT
);

CREATE TABLE professor(
    pID VARCHAR(15) PRIMARY KEY,
    name VARCHAR(15),
    dept_name VARCHAR(30),
    salary INT
);

CREATE TABLE section(
    course_id VARCHAR(15),
    sec_id VARCHAR(15),
    semester VARCHAR(10),
    year INT,
    building VARCHAR(15),
    room_number INT,
    time_slot_id INT,
    PRIMARY KEY(course_id,sec_id,semester,year)
);

CREATE TABLE teaches(
    pID VARCHAR(15),
    course_id VARCHAR(15),
    sec_id VARCHAR(15),
    semester VARCHAR(10),
    year INT,
    PRIMARY KEY(pID,course_id,sec_id,semester,year)
);

CREATE TABLE student(
    pID VARCHAR(15) PRIMARY KEY,
    name VARCHAR(25),
    dept_name VARCHAR(30),
    tot_cred INT
);

CREATE TABLE takes(
    sID VARCHAR(15),
    course_id VARCHAR(15),
    sec_id VARCHAR(15),
    semester VARCHAR(10),
    year INT,
```

```
    grade CHAR,
    PRIMARY KEY(sID,course_id,sec_id,semester,year)
);

CREATE TABLE guide(
    sID VARCHAR(15),
    pID VARCHAR(15),
    PRIMARY KEY(sID,pID)
);

CREATE TABLE time_slot(
    time_slot_id INT,
    day VARCHAR(15),
    start_time TIME,
    end_time TIME,
    PRIMARY KEY(time_slot_id,day,start_time)
);

CREATE TABLE prereq(
    course_id VARCHAR(15),
    prere_id VARCHAR(15),
    PRIMARY KEY(course_id,prere_id)
);
```

Values:

```
INSERT INTO classroom VALUES
('ORION',101,100),
('ORION',102,50),
('ORION',103,75),
('LHC',101,75),
('LHC',102,60);

INSERT INTO department VALUES
('Computer Science','ORION',50000),
('Chemical Engineering','ORION',60000),
('Mechanical Engineering','LHC',45000),
('Electrical Engineering','ORION',55000),
('Production Engineering','LHC',75000),
('Biology Department','GJCH',150000);

INSERT INTO course VALUES
('CS-001','DBMS','Computer Science',4),
('CS-002','CN','Computer Science',3),
('CS-003','DSA','Computer Science',4),
```

('CS-004','OS','Computer Science',4),
('CE-001','Chemicals','Chemical Engineering',3),
('CE-002','Polymers','Chemical Engineering',2),
('ME-001','Engines','Mechanical Engineering',4),
('ME-002','Machines','Mechanical Engineering',3),
('PE-001','Drawing','Production Engineering',3),
('PE-002','Manufacturing','Production Engineering',3),
('EE-001','Conductors','Electrical Engineering',4),
('EE-002','Circuits','Electrical Engineering',3),
('BD-001','Botany','Biology Department',4),
('BD-002','Zoology','Biology Department',3),
('BD-003','Pathology','Biology Department',4);

INSERT INTO professor VALUES
('CS-P01','Albert','Computer Science',150000),
('CS-P02','Bob','Computer Science',500000),
('CS-P03','Jordan','Computer Science',350000),
('CE-P01','Clair','Chemical Engineering',30000),
('CE-P02','Ronald','Chemical Engineering',75000),
('CE-P03','Sara','Chemical Engineering',88000),
('ME-P01','Donald','Mechanical Engineering',75000),
('ME-P02','Jonkins','Mechanical Engineering',90000),
('PE-P01','Ernst','Production Engineering',20000),
('PE-P02','Gary','Production Engineering',80000),
('EE-P01','George','Electrical Engineering',100000),
('EE-P02','Tronton','Electrical Engineering',150000),
('BD-P01','Jonathon','Biology Department',75000),
('BD-P02','Christ','Biology Department',100000);

INSERT INTO section VALUES
('CS-001','SEC-01','Spring',2022,'ORION',1,1001),
('CS-002','SEC-02','Fall',2020,'ORION',2,1002),
('CS-003','SEC-03','Fall',2012,'ORION',3,1006),
('CE-001','SEC-01','Spring',2022,'LHC',1,1003),
('CE-002','SEC-02','Winter',2021,'LHC',2,1002),
('ME-001','SEC-01','Fall',2021,'ORION',4,1004),
('ME-002','SEC-02','Spring',2022,'ORION',5,1003),
('PE-001','SEC-01','Spring',2019,'LHC',3,1005),
('EE-001','SEC-02','Fall',2010,'ORION',3,1002),
('PE-002','SEC-02','Winter',2012,'LHC',4,1001),
('CS-004','SEC-04','Spring',2022,'ORION',3,1003),
('BD-001','SEC-02','Winyer',2021,'ORION',4,1001),
('BD-002','SEC-01','Fall',2022,'LHC',3,1004),
('BD-003','SEC-03','Spring',2023,'ORION',4,1002);

```
INSERT INTO teaches VALUES
('CS-P01','CS-001','SEC-01','Spring',2022),
('CS-P02','CS-004','SEC-02','Fall',2020),
('CS-P03','CS-002','SEC-01','Fall',2023),
('CE-P01','CE-001','SEC-01','Spring',2022),
('CE-P02','CE-002','SEC-02','Fall',2021),
('CE-P02','CS-002','SEC-01','Fall',2021),
('CE-P01','CE-002','SEC-03','Spring',2019),
('ME-P01','ME-001','SEC-01','Fall',2021),
('ME-P02','ME-002','SEC-02','Spring',2021),
('PE-P01','PE-001','SEC-01','Spring',2019),
('EE-P01','EE-001','SEC-03','Fall',2021),
('PE-P02','PE-002','SEC-02','Spring',2012);

INSERT INTO student VALUES
('CS-P01','Sam','Computer Science',13),
('CS-P02','Mary','Computer Science',9),
('CE-P01','Roger','Chemical Engineering',15),
('ME-P01','Turing','Mechanical Engineering',12),
('PE-P01','Gregory','Production Engineering',10),
('PE-P02','Mat','Production Engineering',11),
('EE-P01','David','Electrical Engineering',9),
('BD-P01','Martin','Biology Department',12),
('BD-P02','Joseph','Biology Department',13),
('BD-P03','Beck','Biology Department',15),
('BD-P04','Watson','Biology Department',13);

INSERT INTO takes VALUES
('CS-S001','CS-001','SEC-01','Spring',2022,'S'),
('CS-S002','CS-002','SEC-02','Fall',2020,'A'),
('CE-S001','CE-001','SEC-01','Spring',2022,'S'),
('ME-S001','ME-001','SEC-01','Fall',2021,'B'),
('ME-S002','ME-002','SEC-02','Spring',2022,'A'),
('CS-S003','CS-004','SEC-04','Spring',2022,'S'),
('PE-S001','PE-001','SEC-01','Spring',2019,'C'),
('PE-S002','PE-002','SEC-02','Spring',2012,'C'),
('EE-S001','EE-001','SEC-02','Spring',2012,'B'),
('BD-S001','BD-001','SEC-02','Spring',2022,'A'),
('BD-S001','BD-002','SEC-01','Fall',2021,'S'),
('BD-S001','BD-003','SEC-03','Spring',2020,'B'),
('BD-S002','BD-001','SEC-02','Winter',2022,'A'),
('BD-S002','BD-002','SEC-01','Fall',2020,'S'),
('BD-S002','BD-003','SEC-03','Spring',2023,'S'),
```

('BD-S003','BD-002','SEC-02','Winter',2022,'A'),
('BD-S004','BD-003','SEC-03','Spring',2021,'B');

INSERT INTO guide VALUES
('CS-S001','CS-P01'),
('CS-S003','CS-P01'),
('CS-S002','CS-P02'),
('CE-S001','CE-P01'),
('ME-S001','ME-P01'),
('ME-S002','ME-P02'),
('PE-S001','PE-P01'),
('PE-S002','PE-P02'),
('EE-S001','EE-P01'),
('BD-S001','BD-P01'),
('BD-S002','BD-P02'),
('BD-S003','BD-P01'),
('BD-S004','BD-P02');

INSERT INTO time_slot VALUES
(1001,'Monday','08:00:00','10:00:00'),
(1002,'Tuesday','08:00:00','12:00:00'),
(1003,'Monday','10:00:00','14:30:00'),
(1004,'Wednesday','14:00:00','18:30:00'),
(1005,'Thursday','09:30:00','11:45:00'),
(1006,'Friday','15:30:00','18:45:00');

INSERT INTO prereq VALUES
('CS-001','CS-003'),
('CS-002','CS-001'),
('CE-001','CE-002'),
('PE-001','PE-002'),
('ME-001','ME-002');

1. **Find the titles of courses in the CSE department that have 3 credits.**
   Query:
   SELECT title FROM course WHERE dept_name = 'Computer Science' AND credits = 3;

   Output:

```
mysql> SELECT title FROM course WHERE dept_name = 'Computer Science' AND
 credits = 3;
+--------+
| title |
+--------+
| CN    |
+--------+
1 row in set (0.00 sec)
```

2. **Find the highest salary of any professor.**
   Query:
   SELECT MAX(salary) FROM professor;

   Output:

```
mysql> SELECT MAX(salary) FROM professor;
+--------------+
| MAX(salary) |
+--------------+
|       500000 |
+--------------+
1 row in set (0.00 sec)
```

3. **Find all professors earning the highest salary (there may be more than one with the same salary).**
   Query:
   SELECT name FROM professor WHERE salary = (SELECT MAX(salary) FROM professor);

   Output:

```
mysql> SELECT name FROM professor WHERE salary = (SELECT MAX(salary) FRO
M professor);
+------+
| name |
+------+
| Bob  |
+------+
1 row in set (0.00 sec)
```

4. **Find the maximum enrollment, across all sections, in Fall 2020.**
   Query:
   SELECT COUNT(*) FROM section WHERE semester = 'Fall' AND year = 2020;

   Output:

```
mysql> SELECT COUNT(*) FROM section WHERE semester = 'Fall' AND year = 2
020;
+----------+
| COUNT(*) |
+----------+
|        1 |
+----------+
1 row in set (0.00 sec)
```

5. **Find the enrollment of each section that was offered in Spring 2022.**
   Query:
   SELECT sec_id,COUNT(*) FROM section WHERE semester = 'Spring' AND year = 2022 GROUP BY sec_id;

   Output:

```
mysql> SELECT sec_id,COUNT(*) FROM section WHERE semester = 'Spring' AND
 year = 2022 GROUP BY sec_id;
+--------+----------+
| sec_id | COUNT(*) |
+--------+----------+
| SEC-01 |        2 |
| SEC-04 |        1 |
| SEC-02 |        1 |
+--------+----------+
3 rows in set (0.01 sec)
```

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

   Query:

   SELECT sID,name FROM student NATURAL JOIN guide WHERE sID NOT IN (SELECT sID FROM student NATURAL JOIN guide NATURAL JOIN takes WHERE year <= 2013);

   Output:

```
mysql> SELECT sID,name FROM student NATURAL JOIN guide WHERE sID NOT IN
(SELECT sID FROM student NATURAL JOIN guide NATURAL JOIN takes WHERE yea
r <= 2013);
+---------+---------+
| sID     | name    |
+---------+---------+
| BD-S001 | Martin  |
| BD-S002 | Joseph  |
| BD-S003 | Martin  |
| BD-S004 | Joseph  |
| CE-S001 | Roger   |
| CS-S001 | Sam     |
| CS-S002 | Mary    |
| CS-S003 | Sam     |
| ME-S001 | Turing  |
| PE-S001 | Gregory |
+---------+---------+
10 rows in set (0.00 sec)
```

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

   Query:

   SELECT MAX(salary) FROM professor WHERE dept_name IN (SELECT dept_name FROM student NATURAL JOIN guide WHERE sID NOT IN (SELECT sID FROM student NATURAL JOIN guide NATURAL JOIN takes WHERE year <= 2013)) GROUP BY dept_name ORDER BY MAX(salary) LIMIT 1;

   Output:

```
mysql> SELECT MAX(salary) FROM professor WHERE dept_name IN (SELECT dept_name FR
OM student NATURAL JOIN guide WHERE sID NOT IN (SELECT sID FROM student NATURAL
JOIN guide NATURAL JOIN takes WHERE year <= 2013)) GROUP BY dept_name ORDER BY M
AX(salary) LIMIT 1;
+-------------+
| MAX(salary) |
+-------------+
|       80000 |
+-------------+
1 row in set (0.00 sec)
```

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

Query:

INSERT INTO course VALUE ('CS-004','Weekly Seminar','Computer Science',1);

Output:

```
mysql> INSERT INTO course VALUE ('CS-004','Weekly Seminar','Computer Science',1);
Query OK, 1 row affected (0.00 sec)
```

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

Query:

DELETE FROM course WHERE course_id = 'CS-004';

Output:

```
mysql> DELETE FROM course WHERE course_id = 'CS-004';
Query OK, 1 row affected (0.01 sec)
```

10. **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 t0 ‘-’.**

Query:

SELECT DISTINCT section.sec_id,section.course_id,COALESCE(name,'-') AS prof_name FROM section LEFT OUTER JOIN takes ON section.sec_id = takes.sec_id AND section.course_id = takes.course_id LEFT OUTER JOIN guide ON takes.sID = guide.sID LEFT OUTER JOIN professor ON guide.pID = professor.pID WHERE section.semester = "Spring" AND section.year = 2022;

Output:

```
mysql> SELECT DISTINCT section.sec_id,section.course_id,COALESCE(name,'-') AS pro
f_name FROM section LEFT OUTER JOIN takes ON section.sec_id = takes.sec_id AND se
ction.course_id = takes.course_id LEFT OUTER JOIN guide ON takes.sID = guide.sID
LEFT OUTER JOIN professor ON guide.pID = professor.pID WHERE section.semester = "
Spring" AND section.year = 2022;
+--------+-----------+-----------+
| sec_id | course_id | prof_name |
+--------+-----------+-----------+
| SEC-01 | CE-001    | Clair     |
| SEC-01 | CS-001    | Albert    |
| SEC-04 | CS-004    | Albert    |
| SEC-02 | ME-002    | Jonkins   |
+--------+-----------+-----------+
4 rows in set (0.00 sec)
```

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

Query:

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

Output:

```
mysql> SELECT pID,name,dept_name FROM professor WHERE salary > 50000;
+--------+----------+------------------------+
| pID    | name     | dept_name              |
+--------+----------+------------------------+
| BD-P01 | Jonathon | Biology Department     |
| BD-P02 | Christ   | Biology Department     |
| CE-P02 | Ronald   | Chemical Engineering   |
| CE-P03 | Sara     | Chemical Engineering   |
| CS-P01 | Albert   | Computer Science       |
| CS-P02 | Bob      | Computer Science       |
| CS-P03 | Jordan   | Computer Science       |
| EE-P01 | George   | Electrical Engineering |
| EE-P02 | Tronton  | Electrical Engineering |
| ME-P01 | Donald   | Mechanical Engineering |
| ME-P02 | Jonkins  | Mechanical Engineering |
| PE-P02 | Gary     | Production Engineering |
+--------+----------+------------------------+
12 rows in set (0.00 sec)
```

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

Query:

SELECT name, course_id FROM professor NATURAL JOIN teaches WHERE dept_name = 'Chemical Engineering';

Output:

```
mysql> SELECT name, course_id FROM professor NATURAL JOIN teaches WHERE dept_name
 = 'Chemical Engineering';
+--------+-----------+
| name   | course_id |
+--------+-----------+
| Clair  | CE-001    |
| Clair  | CE-002    |
| Ronald | CE-002    |
| Ronald | CS-002    |
+--------+-----------+
4 rows in set (0.00 sec)
```

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

Query:

SELECT course_id, title FROM course NATURAL JOIN teaches WHERE semester = 'Fall' AND year = 2021 UNION SELECT course_id,title FROM course NATURAL JOIN teaches WHERE semester = 'Spring' AND year = 2021;

Output:

```
mysql> SELECT course_id, title FROM course NATURAL JOIN teaches WHERE semester =
'Fall' AND year = 2021 UNION SELECT course_id,title FROM course NATURAL JOIN teac
hes WHERE semester = 'Spring' AND year = 2021;
+-----------+------------+
| course_id | title      |
+-----------+------------+
| CE-002    | Polymers   |
| CS-002    | CN         |
| EE-001    | Conductors |
| ME-001    | Engines    |
| ME-002    | Machines   |
+-----------+------------+
5 rows in set (0.00 sec)
```

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

Query:

SELECT name FROM professor NATURAL JOIN department WHERE building = 'ORION';

Output:

```
mysql> SELECT name FROM professor NATURAL JOIN department WHERE building = 'ORION';
+---------+
| name    |
+---------+
| Clair   |
| Ronald  |
| Sara    |
| Albert  |
| Bob     |
| Jordan  |
| George  |
| Tronton |
+---------+
8 rows in set (0.00 sec)
```

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

Query:

SELECT course_id, title FROM course NATURAL JOIN teaches WHERE semester = 'Fall' AND year = 2023 UNION SELECT course_id,title FROM course NATURAL JOIN teaches WHERE semester = 'Spring' AND year = 2022;

Output:

```
mysql> SELECT course_id, title FROM course NATURAL JOIN teaches WHERE semester = 'F
all' AND year = 2023 UNION SELECT course_id,title FROM course NATURAL JOIN teaches
WHERE semester = 'Spring' AND year = 2022;
+-----------+-----------+
| course_id | title     |
+-----------+-----------+
| CS-002    | CN        |
| CE-001    | Chemicals |
| CS-001    | DBMS      |
+-----------+-----------+
3 rows in set (0.00 sec)
```

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

Query:

SELECT course_id, title FROM course NATURAL JOIN teaches WHERE semester = 'Fall' AND year = 2021 EXCEPT SELECT course_id,title FROM course NATURAL JOIN teaches WHERE semester = 'Spring' AND year = 2019;

Output:

```
mysql> SELECT course_id, title FROM course NATURAL JOIN teaches WHERE semester = 'F
all' AND year = 2021 EXCEPT SELECT course_id,title FROM course NATURAL JOIN teaches
 WHERE semester = 'Spring' AND year = 2019;
+-----------+------------+
| course_id | title      |
+-----------+------------+
| CS-002    | CN         |
| EE-001    | Conductors |
| ME-001    | Engines    |
+-----------+------------+
3 rows in set (0.00 sec)
```

17. **Find the IDs of all students who were taught by a professor named Albert; make sure there are no duplicates in the result.**

Query:

SELECT DISTINCT sID FROM guide NATURAL JOIN professor WHERE name = 'Albert';

Output:

```
mysql> SELECT DISTINCT sID FROM guide NATURAL JOIN professor WHERE name = 'Albert';
+---------+
| sID     |
+---------+
| CS-S001 |
| CS-S003 |
+---------+
2 rows in set (0.00 sec)
```

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

Query:

SELECT DISTINCT name FROM student NATURAL JOIN guide NATURAL JOIN takes WHERE dept_name = 'Computer Science';

Output:

```
mysql> SELECT DISTINCT name FROM student NATURAL JOIN guide NATURAL JOIN takes WHER
E dept_name = 'Computer Science';
+------+
| name |
+------+
| Sam  |
| Mary |
+------+
2 rows in set (0.00 sec)
```

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

Query:

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

Output:

```
mysql> SELECT dept_name,MAX(salary) FROM professor GROUP BY dept_name;
+------------------------+-------------+
| dept_name              | MAX(salary) |
+------------------------+-------------+
| Biology Department     |      100000 |
| Chemical Engineering   |       88000 |
| Computer Science       |      500000 |
| Electrical Engineering |      150000 |
| Mechanical Engineering |       90000 |
| Production Engineering |       80000 |
+------------------------+-------------+
6 rows in set (0.00 sec)
```

20. **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.**

Query:

SELECT professor.pID,name,COUNT(DISTINCT sec_id) AS no_of_sections FROM professor LEFT OUTER JOIN teaches ON professor.pID = teaches.pID GROUP BY pID;

Output:

```
mysql> SELECT professor.pID,name,COUNT(DISTINCT sec_id) AS no_of_sections FROM prof
essor LEFT OUTER JOIN teaches ON professor.pID = teaches.pID GROUP BY pID;
+--------+----------+----------------+
| pID    | name     | no_of_sections |
+--------+----------+----------------+
| BD-P01 | Jonathon |              0 |
| BD-P02 | Christ   |              0 |
| CE-P01 | Clair    |              2 |
| CE-P02 | Ronald   |              2 |
| CE-P03 | Sara     |              0 |
| CS-P01 | Albert   |              1 |
| CS-P02 | Bob      |              1 |
| CS-P03 | Jordan   |              1 |
| EE-P01 | George   |              1 |
| EE-P02 | Tronton  |              0 |
| ME-P01 | Donald   |              1 |
| ME-P02 | Jonkins  |              1 |
| PE-P01 | Ernst    |              1 |
| PE-P02 | Gary     |              1 |
+--------+----------+----------------+
14 rows in set (0.00 sec)
```

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

Query:

SELECT professor.pID,name,(SELECT COUNT(DISTINCT sec_id) FROM teaches WHERE teaches.pID = professor.pID) AS no_of_sections FROM professor;

Output:

```
mysql> SELECT professor.pID,name,(SELECT COUNT(DISTINCT sec_id) FROM teaches WHERE
teaches.pID = professor.pID) AS no_of_sections FROM professor;
+--------+----------+----------------+
| pID    | name     | no_of_sections |
+--------+----------+----------------+
| BD-P01 | Jonathon |              0 |
| BD-P02 | Christ   |              0 |
| CE-P01 | Clair    |              2 |
| CE-P02 | Ronald   |              2 |
| CE-P03 | Sara     |              0 |
| CS-P01 | Albert   |              1 |
| CS-P02 | Bob      |              1 |
| CS-P03 | Jordan   |              1 |
| EE-P01 | George   |              1 |
| EE-P02 | Tronton  |              0 |
| ME-P01 | Donald   |              1 |
| ME-P02 | Jonkins  |              1 |
| PE-P01 | Ernst    |              1 |
| PE-P02 | Gary     |              1 |
+--------+----------+----------------+
14 rows in set (0.01 sec)
```

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

Query:

SELECT DISTINCT student.name FROM takes AS t1 NATURAL JOIN guide NATURAL JOIN student WHERE NOT EXISTS ((SELECT course_id FROM course WHERE dept_name = 'Biology Department') EXCEPT (SELECT course_id FROM course NATURAL JOIN takes AS t2 WHERE t1.sID = t2.sID AND course.dept_name = 'Biology Department'));

Output:

```
mysql> SELECT DISTINCT student.name FROM takes AS t1 NATURAL JOIN guide NATURAL JOI
N student WHERE NOT EXISTS ( (SELECT course_id FROM course WHERE dept_name = 'Biolo
gy Department') EXCEPT (SELECT course_id FROM course NATURAL JOIN takes AS t2 WHERE
 t1.sID = t2.sID AND course.dept_name = 'Biology Department'));
+--------+
| name   |
+--------+
| Martin |
| Joseph |
+--------+
2 rows in set (0.00 sec)
```

23. 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.

Question:

Find the names of professors who have taught greater than equal to 1 different courses in the "Fall" semester of 2021, along with the number of different courses they have taught.

Query:

SELECT p.name AS professor_name, COUNT(DISTINCT t.course_id) AS course_count FROM professor p JOIN teaches t ON p.pID = t.pID WHERE t.semester = 'Fall' AND t.year =2021 GROUP BY p.name HAVING COUNT(DISTINCT t.course_id) >= 1;

Output:

```
mysql> SELECT p.name AS professor_name, COUNT(DISTINCT t.course_id) AS course_count
 FROM professor p JOIN teaches t ON p.pID = t.pID WHERE t.semester = 'Fall' AND t.y
ear =2021 GROUP BY p.name HAVING COUNT(DISTINCT t.course_id) >= 1;
+----------------+--------------+
| professor_name | course_count |
+----------------+--------------+
| Donald         |            1 |
| George         |            1 |
| Ronald         |            2 |
+----------------+--------------+
3 rows in set (0.01 sec)
```

24. Use the DCL commands to perform the following operations.

a. Create a new user 'testuser' on the localhost.

Query:

CREATE USER testuser@localhost IDENTIFIED BY 'Test@user#1';

Output:

```
mysql> CREATE USER testuser@localhost IDENTIFIED BY 'Test@user#1';
Query OK, 0 rows affected (0.02 sec)

mysql> select user from mysql.user;
+------------------+
| user             |
+------------------+
| debian-sys-maint |
| mysql.infoschema |
| mysql.session    |
| mysql.sys        |
| root             |
| testuser         |
| testuser1        |
+------------------+
7 rows in set (0.00 sec)
```

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

Query:

GRANT ALL PRIVILEGES ON University TO testuser@localhost;

Output:

```
mysql> GRANT ALL PRIVILEGES ON University TO testuser@localhost;
Query OK, 0 rows affected (0.01 sec)

mysql> SHOW GRANTS FOR testuser@localhost;
+-----------------------------------------------------------------------------+
| Grants for testuser@localhost                                               |
+-----------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `testuser`@`localhost`                                |
| GRANT ALL PRIVILEGES ON `University`.`University` TO `testuser`@`localhost` |
+-----------------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

c. **Revoke all the privileges given to the testuser.**

Query:

REVOKE ALL PRIVILEGES ON University FROM testuser@localhost;

Output:

```
mysql> REVOKE ALL PRIVILEGES ON University FROM testuser@localhost;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS FOR testuser@localhost;
+----------------------------------------------+
| Grants for testuser@localhost                |
+----------------------------------------------+
| GRANT USAGE ON *.* TO `testuser`@`localhost` |
+----------------------------------------------+
1 row in set (0.00 sec)
```

25. Use the DCL command to revoke privilege to the user.

a. **Create a new user ‘testuser1’ on the localhost.**

Query:

CREATE USER testuser1@localhost IDENTIFIED BY 'Test@user#1';

Output:

```
mysql> CREATE USER testuser1@localhost IDENTIFIED BY 'Test@user#1';
Query OK, 0 rows affected (0.01 sec)

mysql> select user from mysql.user;
+------------------+
| user             |
+------------------+
| debian-sys-maint |
| mysql.infoschema |
| mysql.session    |
| mysql.sys        |
| root             |
| testuser         |
| testuser1        |
+------------------+
7 rows in set (0.00 sec)
```

b. **Grant only select privileges for the testuser1 on the Student table.**

Query:

GRANT SELECT ON University.student TO testuser1@localhost;

Output:

```
mysql> GRANT SELECT ON University.student TO testuser1@localhost;
Query OK, 0 rows affected (0.05 sec)

mysql> SHOW GRANTS FOR testuser1@localhost;
+-------------------------------------------------------------------+
| Grants for testuser1@localhost                                    |
+-------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `testuser1`@`localhost`                     |
| GRANT SELECT ON `University`.`student` TO `testuser1`@`localhost` |
+-------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

c. **Revoke the select privileges for the testuser1 on the Student table.**

Query:

REVOKE SELECT ON University.student FROM testuser1@localhost;

Output: