```
-- Creating the schema for each table
CREATE TABLE classroom (
  building VARCHAR(50),
  room_number VARCHAR(10),
  capacity INT
);

CREATE TABLE department (
  dept_name VARCHAR(50),
  building VARCHAR(50),
  budget DECIMAL(10, 2)
);

CREATE TABLE course (
  course_id VARCHAR(10),
  title VARCHAR(100),
  dept_name VARCHAR(50),
  credits INT
);

CREATE TABLE professor (
  pID INT PRIMARY KEY,
  name VARCHAR(50),
  dept_name VARCHAR(50),
  salary DECIMAL(10, 2)
);

CREATE TABLE section (
  course_id VARCHAR(10),
  sec_id VARCHAR(10),
  semester VARCHAR(10),
  year INT,
  building VARCHAR(50),
  room_number VARCHAR(10),
```

```
  time_slot_id VARCHAR(10)
);

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

CREATE TABLE student (
  pID INT PRIMARY KEY,
  name VARCHAR(50),
  dept_name VARCHAR(50),
  tot_cred INT
);

CREATE TABLE takes (
  sID INT,
  course_id VARCHAR(10),
  sec_id VARCHAR(10),
  semester VARCHAR(10),
  year INT,
  grade CHAR(1)
);

CREATE TABLE guide (
  sID INT,
  pID INT
);

CREATE TABLE time_slot (
  time_slot_id VARCHAR(10),
```

```
  day VARCHAR(10),
  start_time TIME,
  end_time TIME
);

CREATE TABLE prereq (
  course_id VARCHAR(10),
  prere_id VARCHAR(10)
);

-- Populating the tables with sample data

INSERT INTO classroom VALUES
('ORION', '101', 30),
('CSE', '202', 50),
('PHYS', '303', 40),
('CHEM', '404', 35),
('BIO', '505', 25);

INSERT INTO department VALUES
('CSE', 'ORION', 50000),
('CHEM', 'ORION', 30000),
('BIO', 'ORION', 20000),
('PHYS', 'ANDROMEDA', 40000),
('MATH', 'ANDROMEDA', 60000);

INSERT INTO course VALUES
('CSE101', 'Data Structures', 'CSE', 3),
('CSE102', 'Algorithms', 'CSE', 3),
('CHEM101', 'Organic Chemistry', 'CHEM', 4),
('BIO101', 'Biology Basics', 'BIO', 2),
('PHYS101', 'Physics I', 'PHYS', 4);

INSERT INTO professor VALUES
```

```
(1, 'Tejaswi', 'CSE', 60000),
(2, 'Ramesh', 'CSE', 50000),
(3, 'Arun', 'BIO', 45000),
(4, 'Manoj', 'CHEM', 40000),
(5, 'Priya', 'PHYS', 55000);

INSERT INTO section VALUES
('CSE101', 'S1', 'Fall', 2020, 'ORION', '101', 'TS1'),
('CSE102', 'S1', 'Spring', 2019, 'CSE', '202', 'TS2'),
('CHEM101', 'S1', 'Spring', 2022', 'ORION', '404', 'TS3'),
('BIO101', 'S2', 'Fall', 2021', 'ORION', '505', 'TS4'),
('PHYS101', 'S3', 'Spring', 2022', 'PHYS', '303', 'TS5');

INSERT INTO teaches VALUES
(1, 'CSE101', 'S1', 'Fall', 2020),
(2, 'CSE102', 'S1', 'Spring', 2019),
(3, 'BIO101', 'S2', 'Fall', 2021),
(4, 'CHEM101', 'S1', 'Spring', 2022),
(5, 'PHYS101', 'S3', 'Spring', 2022);

INSERT INTO student VALUES
(101, 'John', 'CSE', 90),
(102, 'Alice', 'BIO', 85),
(103, 'Bob', 'CHEM', 80),
(104, 'Sarah', 'PHYS', 95),
(105, 'Michael', 'CSE', 75);

INSERT INTO takes VALUES
(101, 'CSE101', 'S1', 'Fall', 2020, 'A'),
(102, 'BIO101', 'S2', 'Fall', 2021, 'B'),
(103, 'CHEM101', 'S1', 'Spring', 2022, 'C'),
(104, 'PHYS101', 'S3', 'Spring', 2022, 'A'),
(105, 'CSE102', 'S1', 'Spring', 2019, 'B');
```

```
INSERT INTO guide VALUES
(101, 1),
(102, 3),
(103, 4),
(104, 5),
(105, 2);

INSERT INTO time_slot VALUES
('TS1', 'Monday', '09:00', '11:00'),
('TS2', 'Tuesday', '10:00', '12:00'),
('TS3', 'Wednesday', '14:00', '16:00'),
('TS4', 'Thursday', '09:00', '11:00'),
('TS5', 'Friday', '10:00', '12:00');

INSERT INTO prereq VALUES
('CSE102', 'CSE101'),
('BIO101', 'BIO100'),
('CHEM101', 'CHEM100'),
('PHYS101', 'PHYS100'),
('CSE101', 'MATH101');

-- 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;

-- b. Find the highest salary of any professor.
SELECT MAX(salary) AS highest_salary
FROM professor;

-- c. Find all professors earning the highest salary.
SELECT name
FROM professor
WHERE salary = (SELECT MAX(salary) FROM professor);
```

```
-- 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 subquery;

-- 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;

-- f. Find the IDs and names of all students who have not taken any course offering before Spring 2013.
SELECT s.pID, s.name
FROM student s
WHERE NOT EXISTS (SELECT 1
            FROM takes t
            WHERE s.pID = t.sID AND (t.year < 2013 OR (t.year = 2013 AND t.semester = 'Spring')));

-- g. Find the lowest, across all departments, of the per-department maximum salary.
SELECT MIN(max_salary) AS lowest_max_salary
FROM (SELECT MAX(salary) AS max_salary
    FROM professor
    GROUP BY dept_name) AS subquery;

-- h. Create a new course “CS-001”, titled “Weekly Seminar”, with 1 credit.
INSERT INTO course VALUES ('CS001', 'Weekly Seminar', 'CSE', 1);

-- i. Delete the course CS-001.
DELETE FROM course WHERE course_id = 'CS001';

-- j. Display the list of all course sections offered in Spring 2022, along with the names of the professors teaching the section.
```

```
SELECT s.course_id, s.sec_id, COALESCE(p.name, '-') AS professor_name
FROM section s
LEFT JOIN teaches t ON s.course_id = t.course_id AND s.sec_id = t.sec_id
LEFT JOIN professor p ON t.pID = p.pID
WHERE s.semester = 'Spring' AND s.year = 2022;

-- 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;

-- l. Find the names of all professors in the Chemical Engineering department together with the course id of all courses they teach.
SELECT p.name, t.course_id
FROM professor p
JOIN teaches t ON p.pID = t.pID
WHERE p.dept_name = 'CHEM';

-- 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);

-- n. Find the names of all professors whose department is in the ‘ORION’ building.
SELECT name
FROM professor
WHERE dept_name IN (SELECT dept_name FROM department WHERE building = 'ORION');

-- 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);

-- 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);


-- q. Find the IDs of all students who were taught by a professor named Tejaswi; make sure there are no duplicates in the result.
SELECT DISTINCT sID
FROM takes
WHERE course_id IN (SELECT course_id FROM teaches WHERE pID IN (SELECT pID FROM professor WHERE name = 'Tejaswi'));


-- 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 s.name
FROM student s
JOIN takes t ON s.pID = t.sID
JOIN course c ON t.course_id = c.course_id
WHERE c.dept_name = 'CSE';


-- s. For each department, find the maximum salary of professors in that department.
SELECT dept_name, MAX(salary) AS max_salary
FROM professor
GROUP BY dept_name;


-- t. Display a list of all professors, showing their ID, name, and the number of sections that they have taught.
-- Use an outer join, and make sure to show the number of sections as 0 for professors who have not taught any section.
SELECT p.pID, p.name, COUNT(t.course_id) AS num_sections
FROM professor p
LEFT JOIN teaches t ON p.pID = t.pID
GROUP BY p.pID, p.name;


-- u. Write the same query as above, but using a scalar subquery, without outer join.
SELECT pID, name,
    (SELECT COUNT(*) FROM teaches t WHERE t.pID = p.pID) AS num_sections
FROM professor p;
```

```
-- 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 = 'BIO')
GROUP BY sID
HAVING COUNT(DISTINCT course_id) = (SELECT COUNT(course_id) FROM course WHERE dept_name = 'BIO');

-- w. Create your own query: Find all students who have taken both a CSE course and a BIO course.
SELECT DISTINCT s.name
FROM student s
JOIN takes t1 ON s.pID = t1.sID
JOIN course c1 ON t1.course_id = c1.course_id
JOIN takes t2 ON s.pID = t2.sID
JOIN course c2 ON t2.course_id = c2.course_id
WHERE c1.dept_name = 'CSE' AND c2.dept_name = 'BIO';

-- x. Use the DCL commands to perform the following operations.
-- 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. Use the DCL command to revoke privilege to the user.
-- 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';
```