|  |
| --- |
| 1. For each department, find the maximum salary of instructors in that department.   SELECT dept\_name, MAX(salary) AS maximum\_salary  FROM instructor  GROUP BY dept\_name;  A black and white screen with white text  Description automatically generated |
| 1. Find the IDs of all students who were taught by an instructor named Katz; make sure there are no duplicates in the result.   SELECT DISTINCT student.ID  FROM student  JOIN takes ON student.ID = takes.ID  JOIN teaches ON teaches.course\_id = takes.course\_id  JOIN instructor ON instructor.ID = teaches.ID  WHERE instructor.name = 'Katz';  A screenshot of a black and white table  Description automatically generated |
| 1. Find the ID and title of each course in Comp. Sci. that has had at least one section with afternoon hours (i.e., ends at or after 12:00). (You should eliminate duplicates if any.)   SELECT DISTINCT c.course\_id, c.title  FROM course AS c  JOIN section AS s  ON c.course\_id = s.course\_id  JOIN time\_slot AS t  ON t.time\_slot\_id = s.time\_slot\_id  WHERE c.dept\_name = 'Comp. Sci.'  AND t.start\_hr >= 12;  A black background with white text  Description automatically generated |
| 1. Find the IDs and titles of all the courses that are (direct) prerequisite to the Robotics course.   SELECT c1.course\_id AS prerequisite\_id, c1.title AS prerequisite\_title  FROM course AS c  JOIN prereq AS p  ON p.course\_id = c.course\_id  JOIN course AS c1  ON p.prereq\_id = c1.course\_id  WHERE c.title = 'Robotics';  A black and white screen with white text  Description automatically generated |
| 1. Find the IDs and names of all instructors earning the highest salary (there may be more than one with the same salary).   SELECT ID, name  FROM instructor  WHERE salary = (SELECT MAX(salary) FROM instructor);  A black and white screen with white text  Description automatically generated |
| 1. Find the enrollment (number of students) in each section that was offered in Spring 2017. The result columns should be course id, section id, students num. You do not need to output sections with 0 students.   SELECT sec.course\_id, sec.sec\_id, COUNT(\*) AS student\_num  FROM takes AS t  INNER JOIN section AS sec  ON t.course\_id = sec.course\_id AND t.sec\_id = sec.sec\_id  WHERE sec.year = 2017 AND sec.semester = 'Spring'  GROUP BY sec.course\_id, sec.sec\_id;A black and white text  Description automatically generated |
| 1. Rewrite the preceding query, but also output sections with 0 students.   SELECT sec.course\_id, sec.sec\_id, COUNT(t.ID) AS student\_num  FROM section AS sec  LEFT JOIN takes AS t  ON sec.course\_id = t.course\_id AND sec.sec\_id = t.sec\_id  WHERE sec.year = 2017 AND sec.semester = 'Spring'  GROUP BY sec.course\_id, sec.sec\_id;A screenshot of a computer  Description automatically generated |
| 1. Find the IDs and names of all instructors who have taught at least 3 different courses.   SELECT i.ID, i.name, COUNT(DISTINCT t.course\_id) AS courses\_taught  FROM instructor AS i  JOIN teaches AS t  ON i.ID = t.ID  GROUP BY i.ID, i.name  HAVING COUNT(DISTINCT t.course\_id) >= 3; |
| 1. Find the ID and name of the student with the highest number of ’A’ grades (there may be more than one such student)  |  | | --- | | SELECT s.ID, s.name, COUNT(t.grade) AS a\_grade\_count  FROM student AS s  JOIN takes AS t ON s.ID = t.ID  WHERE t.grade = 'A'  GROUP BY s.ID, s.name  HAVING COUNT(t.grade) = (  SELECT MAX(a\_grade\_count)  FROM (  SELECT COUNT(grade) AS a\_grade\_count  FROM takes  WHERE grade = 'A'  GROUP BY ID  ) AS subquery  ); |   A screenshot of a computer  Description automatically generated |
| 1. Find the ID and name of each History student who has not taken any Music courses.   SELECT s.ID, s.name  FROM student AS s  WHERE s.dept\_name = 'History'  AND NOT EXISTS (  SELECT 1  FROM takes AS t  JOIN course AS c ON t.course\_id = c.course\_id  WHERE s.ID = t.ID  AND c.dept\_name = 'Music'  );  A black and white image of a black box with white text  Description automatically generated |
| 1. Find the ID and name of each instructor who has never given an ’A’ grade in any course she or he has taught. (Instructors who have never taught a course trivially satisfy this condition.)   SELECT DISTINCT i.ID, i.name  FROM instructor AS i  LEFT JOIN teaches AS t ON i.ID = t.ID  LEFT JOIN takes AS tk ON t.course\_id = tk.course\_id AND t.sec\_id = tk.sec\_id  WHERE i.ID NOT IN (  SELECT t.ID  FROM teaches AS t  JOIN takes AS tk ON t.course\_id = tk.course\_id AND t.sec\_id = tk.sec\_id  WHERE tk.grade = 'A' OR tk.grade = 'A-'  );  A screenshot of a computer  Description automatically generated |
| 12. Rewrite the preceding query, but also ensure that you include only instructors who have given at least one other non-null grade in some course.  SELECT DISTINCT i.ID, i.name  FROM instructor AS i  LEFT JOIN teaches AS t ON i.ID = t.ID  LEFT JOIN takes AS tk ON t.course\_id = tk.course\_id AND t.sec\_id = tk.sec\_id  WHERE i.ID NOT IN (  SELECT t.ID  FROM teaches AS t  JOIN takes AS tk ON t.course\_id = tk.course\_id AND t.sec\_id = tk.sec\_id  WHERE tk.grade = 'A' OR tk.grade = 'A-'  )  AND i.ID IN (  SELECT t.ID  FROM teaches AS t  JOIN takes AS tk ON t.course\_id = tk.course\_id AND t.sec\_id = tk.sec\_id  WHERE tk.grade IS NOT NULL  );  A black and white screen with white text  Description automatically generated |
| 1. For each student who have retaken a course at least once (i.e., the student has taken the course at least twice), show the student’s ID, name and the course ID.   SELECT s.ID, s.name, t.course\_id  FROM student AS s  JOIN takes AS t ON s.ID = t.ID  GROUP BY s.ID, s.name, t.course\_id  HAVING COUNT(t.course\_id) >= 2;  A black screen with white text  Description automatically generated |
| 1. Find the IDs of those students who have retaken at least three distinct courses at least once (i.e., the student has taken the course at least twice)   SELECT t.ID  FROM takes AS t  GROUP BY t.ID  HAVING COUNT(DISTINCT t.course\_id) >= 3 AND SUM(CASE WHEN t.course\_id IS NOT NULL THEN 1 ELSE 0 END) >= 6;  A black rectangular object with white text  Description automatically generated |
| 1. Find the IDs and names of those instructors who have taught every course in their department.   SELECT i.ID, i.name  FROM instructor AS i  JOIN teaches AS t ON i.ID = t.ID  JOIN course AS c ON t.course\_id = c.course\_id  WHERE i.dept\_name = c.dept\_name  GROUP BY i.ID, i.name  HAVING COUNT(DISTINCT c.course\_id) = (  SELECT COUNT(\*)  FROM course AS c2  WHERE c2.dept\_name = i.dept\_name  );  A screenshot of a computer  Description automatically generated |

|  |
| --- |
| 1. Create a new course ”CS-001” in the Comp. Sci. department, titled ”Weekly Seminar”, with 2 credits. |
| 1. Create a section of this course in Spring 2022, with sec id of 1, and with the location of this section not yet specified. |
|  |
| 1. Enroll every student in the Comp. Sci. department in the above section (Hint: use INSERT INTO SELECT statement).   A computer screen shot of a computer code  Description automatically generated |
| 1. Delete enrollments in the above section where the student’s ID is 12345.   A black screen with white text and yellow text  Description automatically generated |
| 5.Delete the course CS-001. What happened to the section and enrollments of this course?  A black background with white text  Description automatically generated    Since constraint on the course table have ON DELETE CASCADE constraint, the enrollment is deleted automatically with the deletion of the CS-001. |

|  |  |
| --- | --- |
| 3. Create a stored procedure that enrolls a student into a section of a course, but only if the student has completed the prerequisites for that course. If the prerequisites are not met, the procedure should return a message indicating that the student cannot be enrolled.  Call the procedure for a student who meets the prerequisites and a student who does not meet the prerequisites, and show the output in each case.   |  | | --- | | DELIMITER //  CREATE PROCEDURE enroll\_student\_in\_section (  IN input\_student\_id VARCHAR(5),  IN input\_course\_id VARCHAR(8),  IN input\_sec\_id VARCHAR(8),  IN input\_semester VARCHAR(6),  IN input\_year DECIMAL(4,0)  )  BEGIN  DECLARE prereq\_count INT;  DECLARE completed\_count INT;  DECLARE prereq\_missing INT;  DECLARE already\_enrolled INT;    SELECT COUNT(\*) INTO already\_enrolled  FROM takes  WHERE ID = input\_student\_id  AND course\_id = input\_course\_id  AND sec\_id = input\_sec\_id  AND semester = input\_semester  AND year = input\_year;    IF already\_enrolled > 0 THEN  SELECT CONCAT('Student ', input\_student\_id, ' is already enrolled in course ', input\_course\_id, ' section ', input\_sec\_id) AS result;  ELSE    SELECT COUNT(\*) INTO prereq\_count  FROM prereq  WHERE course\_id = input\_course\_id;    SELECT COUNT(\*) INTO completed\_count  FROM prereq AS p  JOIN takes AS t ON p.prereq\_id = t.course\_id  WHERE t.ID = input\_student\_id  AND t.grade IS NOT NULL  AND t.grade <> 'F'  AND p.course\_id = input\_course\_id;    IF completed\_count = prereq\_count THEN    INSERT INTO takes (ID, course\_id, sec\_id, semester, year, grade)  VALUES (input\_student\_id, input\_course\_id, input\_sec\_id, input\_semester, input\_year, NULL);    SELECT CONCAT('Student ', input\_student\_id, ' successfully enrolled in course ', input\_course\_id, ' section ', input\_sec\_id) AS result;  ELSE    SET prereq\_missing = prereq\_count - completed\_count;  SELECT CONCAT('Student ', input\_student\_id, ' cannot be enrolled in course ', input\_course\_id, ' due to missing ', prereq\_missing, ' prerequisite(s).') AS result;  END IF;  END IF;  END //  DELIMITER ; |   A screenshot of a computer  Description automatically generatedA screenshot of a computer  Description automatically generated |

e

|  |  |
| --- | --- |
| 4.Write SQL DDL commands to create a database for an insurance company according to the following schemas:  person(driver id, name, address) car(license plate, model, year) owns(driver id, license plate) accident(report number, date, location) participated(report number, license plate, driver id, damage amount)  Make any reasonable assumptions about data types, and be sure to declare primary and foreign keys.   |  | | --- | | CREATE TABLE person (  driver\_id INT PRIMARY KEY,  name VARCHAR(50) NOT NULL,  address VARCHAR(150)  );  CREATE TABLE car (  license\_plate VARCHAR(15) PRIMARY KEY,  model VARCHAR(50) NOT NULL,  year INT CHECK (year >= 1886)  );  CREATE TABLE owns (  driver\_id INT,  license\_plate VARCHAR(15),  PRIMARY KEY (driver\_id, license\_plate),  FOREIGN KEY (driver\_id) REFERENCES person(driver\_id) ON DELETE CASCADE,  FOREIGN KEY (license\_plate) REFERENCES car(license\_plate) ON DELETE CASCADE  );  CREATE TABLE accident (  report\_number INT PRIMARY KEY,  date DATE NOT NULL,  location VARCHAR(150) NOT NULL  );  CREATE TABLE participated (  report\_number INT,  license\_plate VARCHAR(15),  driver\_id INT,  damage\_amount DECIMAL(10, 2),  PRIMARY KEY (report\_number, license\_plate, driver\_id),  FOREIGN KEY (report\_number) REFERENCES accident(report\_number) ON DELETE CASCADE,  FOREIGN KEY (license\_plate) REFERENCES car(license\_plate) ON DELETE CASCADE,  FOREIGN KEY (driver\_id) REFERENCES person(driver\_id) ON DELETE CASCADE  ); |   A computer screen shot of a black screen  Description automatically generatedA computer screen shot of a black screen  Description automatically generated |