5. Consider the following relations Structured Enquiry
STUDENT(SNUM: INTEGER, SNAME: STRING, MAJOR: STRING, LEVEL: STRING, AGE: INTEGER)
CLASS(CNAME: STRING, MEETS AT: STRING, ROOM: STRING, FID: INTEGER)
ENROLLED(SNUM: INTEGER, CNAME: STRING)
FACULTY(FID: INTEGER, FNAME: STRING, DEPTID: INTEGER)
Write the following queries in SQL. No duplicates should be printed in any of the answers.

a) **Find the names of all Juniors (level = JR) who are enrolled in a class taught by Rakesh.**
```
SELECT DISTINCT S.SNAME
FROM STUDENT S, ENROLLED E, CLASS C, FACULTY F
WHERE S.LEVEL = 'JR' AND F.FNAME = 'Rakesh' AND S.SNUM = E.SNUM AND E.CNAME =
C.CNAME AND C.FID = F.FID;
```

b) **Find the age of the oldest student who is either a history major or enrolled in a course taught by Ravi.**
```
SELECT MAX(S.AGE) AS OLDEST_AGE
FROM STUDENT S, ENROLLED E, CLASS C, FACULTY F
WHERE S.MAJOR = 'History' OR F.FNAME = 'Ravi' AND S.SNUM = E.SNUM AND E.CNAME =
C.CNAME AND C.FID = F.FID;
```

c) **Find the names of all students who are enrolled in two classes that meet at the same time.**
```
SELECT DISTINCT S1.SNAME
FROM ENROLLED E1
JOIN CLASS C1 ON E1.CNAME = C1.CNAME
JOIN ENROLLED E2 ON E1.SNUM = E2.SNUM
JOIN CLASS C2 ON E2.CNAME = C2.CNAME
JOIN STUDENT S1 ON E1.SNUM = S1.SNUM
WHERE C1.MEETS_AT = C2.MEETS_AT AND C1.CNAME <> C2.CNAME;
```

d) **For each faculty member that has taught classes only in room R128, print the faculty member's name and the total number of classes she or he has taught.**
```
SELECT F.FNAME, COUNT(C.CNAME) AS TOTAL_CLASSES
FROM FACULTY F, CLASS C
WHERE C.ROOM = 'R128' AND F.FID = C.FID
GROUP BY F.FNAME
HAVING COUNT(DISTINCT C.ROOM) = 1;
```

e) **Create a view that contains the details of students along with the name of the courses enrolled.**
```
CREATE VIEW StudentCourseDetails AS
SELECT S.SNUM, S.SNAME, S.MAJOR, S.LEVEL, S.AGE, E.CNAME
FROM STUDENT S, ENROLLED E
WHERE S.SNUM = E.SNUM;
```

4. The following relations keep track of airline flight information: Exercise
FLIGHTS (FLNO: INTEGER, SOURCE: STRING, DESTINATION: STRING, DISTANCE: INTEGER, DEPARTS:TIME, ARRIVES: TIME, PRICE: INTEGER)
AIRCRAFT (AID: INTEGER, ANAME: STRING, CRUISINGRANGE: INTEGER)
CERTIFIED (EID: INTEGER, AID: INTEGER)
EMPLOYEES (EID: INTEGER, ENAME: STRING, SALARY: INTEGER)
Write SQL queries to

a) **Find the names of aircraft such that all pilots certified to operate them earn more than $80,000.**

```
SELECT A.ANAME
FROM AIRCRAFT A, CERTIFIED C, EMPLOYEES E
WHERE A.AID = C.AID AND C.EID = E.EID AND
E.SALARY > 80000;
```

b) For each pilot who is certified for more than three aircraft, find the eid and the maximum cruising range of the aircraft for which she or he is certified.

```
SELECT C.EID, MAX(A.CRUISINGRANGE) AS MAX_CRUISINGRANGE
FROM CERTIFIED C
JOIN AIRCRAFT A ON C.AID = A.AID
GROUP BY C.EID
HAVING COUNT(*) > 3;
```

c) **For all aircraft with cruising range over 1000 miles, find the name of the aircraft and the average salary of all pilots certified for this aircraft.**

```
SELECT A.ANAME, AVG(E.SALARY) AS AVG_SALARY
FROM AIRCRAFT A, CERTIFIED C, EMPLOYEES E
WHERE A.CRUISINGRANGE > 1000 AND A.AID = C.AID AND C.EID = E.EID
GROUP BY A.ANAME;
```

d) **Print the Enames of pilots who can operate planes with cruising range greater than 3000 miles but are not certified on any Boeing aircraft.**

```
SELECT DISTINCT E.ENAME
FROM EMPLOYEES E, CERTIFIED C, AIRCRAFT A
WHERE E.EID = C.EID AND C.AID = A.AID AND A.CRUISINGRANGE > 3000 AND
 E.EID NOT IN (
   SELECT EID
   FROM CERTIFIED C
   JOIN AIRCRAFT A ON C.AID = A.AID
   WHERE A.ANAME LIKE '%Boeing%');
```

e) **Print the name and salary of every non-pilot whose salary is more than the average salary for pilots.**

```
SELECT E.ENAME, E.SALARY
FROM EMPLOYEES E
WHERE E.EID NOT IN (SELECT EID FROM CERTIFIED)
AND E.SALARY > (SELECT AVG(SALARY) FROM EMPLOYEES);
```

3. Consider the schema for Company Database: Exercise
EMPLOYEE(SSN, NAME, ADDRESS, SEX, SALARY, SUPERSSN, DNO)
DEPARTMENT(DNO, DNAME, MGRSSN, MGRSTARTDATE)
DLOCATION(DNO,DLOC)
PROJECT(PNO, PNAME, PLOCATION, DNO)
WORKS_ON(SSN, PNO, HOURS)
Write SQL queries to

a) **Make a list of all project numbers for projects that involve an employee whose last name is ‘Scott’, either as a worker or as a manager of the department that Controls the project.**

```
SELECT DISTINCT P.PNO
FROM PROJECT P, DEPARTMENT D, EMPLOYEE MGR, WORKS_ON W, EMPLOYEE E
WHERE D.MGRSSN = MGR.SSN AND P.DNO = D.DNO AND P.PNO = W.PNO AND W.SSN =
E.SSN AND E.NAME LIKE '%Scott%' OR MGR.NAME LIKE '%Scott%';
```

b) **Show the resulting salaries if every employee working on the ‘IoT’ project is Given a 10 percent raise.**

```
SELECT E.SSN, E.NAME,  E.SALARY * 1.10 AS NEW_SALARY
FROM EMPLOYEE E, WORKS_ON W, PROJECT P
WHERE P.PNAME = 'IoT' AND E.SSN = W.SSN AND W.PNO = P.PNO;
```

c) **Find the sum of the salaries of all employees of the ‘Accounts’ department, as well as the maximum salary, the minimum salary, and the average salary in this department.**

```
SELECT SUM(E.SALARY) AS TOTAL_SALARY, MAX(E.SALARY) AS MAX_SALARY, MIN(E.SALARY)
AS MIN_SALARY, AVG(E.SALARY) AS AVG_SALARY
FROM EMPLOYEE E, DEPARTMENT D
WHERE D.DNAME = 'Accounts' AND E.DNO = D.DNO;
```

d) **Retrieve the name of each employee who works on the entire projects controlled by department number 5.**

```
SELECT E.NAME
FROM EMPLOYEE E JOIN WORKS_ON W ON E.SSN = W.SSN
WHERE NOT EXISTS ( SELECT P.PNO
                    FROM PROJECT P
                     WHERE P.DNO = 5)
  MINUS ( SELECT W2.SSN
          FROM WORKS_ON W2
          WHERE W2.PNO = P.PNO AND W2.SSN = E.SSN ));
```

e) **For each department that has more than five employees, retrieve the department number and the number of its employees who are making more than Rs. 6,00,000.**

```
SELECT D.DNO, COUNT(E.SSN) AS NUM_EMPLOYEES
FROM DEPARTMENT D, EMPLOYEE E
WHERE E.SALARY > 600000 AND D.DNO = E.DNO
GROUP BY D.DNO
HAVING COUNT(E.SSN) > 5;
```