# DBMS Lab - Session-5

**106122076 CSE-B**
**Lohesh S U**

**Execute the following Queries in SQL over the Flight Schema given below. Flights(flno: integer, from: string, to: string, distance: integer, departs: time, arrives: time, price: integer) Aircraft(aid: integer, aname: string, cruising range: integer) Certified(eid: integer, aid: integer) Employees(eid: integer, ename: string, salary: integer) Note: Every pilot is certified for some aircraft, and only pilots are certified to fly. cruisingrange means the maximum distance an aircraft can fly without landing say, 10000 miles. Aircraft Id(aid) is the company id of the aircraft e.g. Aircraft(101, Boeing, 1000). Employees include pilots along with Airlines(Aircraft) staff. Identify the Primary key for each table. Before inserting values, please go through the questions below which shall facilitate you to choose appropriate values for the fields in the table.**

```
CREATE TABLE Aircraft (
   aid INT PRIMARY KEY,
   aname VARCHAR(50),
   cruisingrange INT
);

CREATE TABLE Employees (
   eid INT PRIMARY KEY,
   ename VARCHAR(50),
   salary INT
);

CREATE TABLE Flights (
   flno INT PRIMARY KEY,
   `from` VARCHAR(50),
   `to` VARCHAR(50),
   distance INT,
   departs TIME,
   arrives TIME,
   price INT
);

CREATE TABLE Certified (
   eid INT,
   aid INT,
   PRIMARY KEY (eid, aid),
   FOREIGN KEY (eid) REFERENCES Employees(eid),
   FOREIGN KEY (aid) REFERENCES Aircraft(aid)
);
```

**a. Find the names of aircraft such that all pilots certified to operate them earn more than Rs. 50,000.**

```
SELECT a.aname
FROM Aircraft a
WHERE NOT EXISTS (
   SELECT *
   FROM Certified c, Employees e
   WHERE a.aid = c.aid AND c.eid = e.eid AND e.salary <= 50000
);
```

**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/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(c.aid) > 3;
```

**c. Find the names of pilots whose salary is less than the price of the cheapest route from Trichy to Agartala.**

```
SELECT e.ename
FROM Employees e
WHERE e.salary < (
   SELECT MIN(f.price)
   FROM Flights f
   WHERE f.`from` = 'Trichy' AND f.`to` = 'Agartala'
);
```

**d. 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
JOIN Certified c ON a.aid = c.aid
JOIN Employees e ON c.eid = e.eid
WHERE a.cruisingrange > 1000
GROUP BY a.aname;
```

**e. Find the names of pilots certified for some Boeing aircraft who drove the maximum distance on all flights departing from Ladakh.**

```
SELECT e.ename
FROM Employees e
JOIN Certified c ON e.eid = c.eid
JOIN Aircraft a ON c.aid = a.aid
JOIN Flights f ON f.flno = f.flno
WHERE a.aname = 'Boeing' AND f.`from` = 'Ladakh'
GROUP BY e.ename
HAVING MAX(f.distance) = (
   SELECT MAX(f.distance)
   FROM Flights f
   WHERE f.`from` = 'Ladakh'
);
```

**f. Find the aids of all aircraft that can be used on routes from Chandigarh to Surat.**

```
SELECT a.aid
FROM Aircraft a
JOIN Flights f ON f.distance <= a.cruisingrange
WHERE f.`from` = 'Chandigarh' AND f.`to` = 'Surat';
```

**g. Identify the routes that can be piloted by every pilot who makes more than 100,000.**

```
SELECT DISTINCT f.`from`, f.`to`
FROM Flights f
WHERE NOT EXISTS (
   SELECT e.eid
   FROM Employees e
   WHERE e.salary > 100000 AND NOT EXISTS (
      SELECT c.aid
      FROM Certified c
      WHERE c.eid = e.eid AND c.aid = f.flno
   )
);
```

**h. 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 e.ename
FROM Employees e
JOIN Certified c ON e.eid = c.eid
JOIN Aircraft a ON c.aid = a.aid
WHERE a.cruisingrange > 3000 AND e.eid NOT IN (
   SELECT e.eid
   FROM Employees e
   JOIN Certified c ON e.eid = c.eid
   JOIN Aircraft a ON c.aid = a.aid
   WHERE a.aname = 'Boeing'
);
```

**i. Compute the difference between the average salary of a pilot and the average salary of all employees (including pilots).**

```
SELECT
   (SELECT AVG(e1.salary)
    FROM Employees e1
    WHERE EXISTS (SELECT * FROM Certified c WHERE e1.eid = c.eid)) -
   (SELECT AVG(e2.salary) FROM Employees e2) AS salary_difference;
```

**j. 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 c.eid FROM Certified c)
AND e.salary > (
   SELECT AVG(e1.salary)
   FROM Employees e1
   JOIN Certified c ON e1.eid = c.eid
);
```

**k. Print the names of employees who are certified only on aircraft with cruising range longer than 1000 miles.**

```
SELECT e.ename
FROM Employees e
WHERE NOT EXISTS (
   SELECT c.aid
   FROM Certified c
   JOIN Aircraft a ON c.aid = a.aid
   WHERE e.eid = c.eid AND a.cruisingrange <= 1000
);
```

**l. Print the names of employees who are certified only on aircraft with cruising range shorter than 1000 miles, but on at least two such aircraft.**

```
SELECT e.ename
FROM Employees e
JOIN Certified c ON e.eid = c.eid
JOIN Aircraft a ON c.aid = a.aid
WHERE a.cruisingrange < 1000
GROUP BY e.ename
HAVING COUNT(c.aid) >= 2;
```

**m. Print the names of employees who are certified only on aircraft with cruising range longer than 1000 miles and who are certified on some Boeing aircraft.**

```
SELECT e.ename
FROM Employees e
JOIN Certified c ON e.eid = c.eid
JOIN Aircraft a ON c.aid = a.aid
WHERE a.cruisingrange > 1000
AND EXISTS (
   SELECT c1.eid
   FROM Certified c1
   JOIN Aircraft a1 ON c1.aid = a1.aid
   WHERE e.eid = c1.eid AND a1.aname = 'Boeing'
);
```

**n. Find the eids of pilots certified for some Boeing aircraft.**

```
SELECT DISTINCT e.eid
FROM Employees e
JOIN Certified c ON e.eid = c.eid
JOIN Aircraft a ON c.aid = a.aid
WHERE a.aname = 'Boeing';
```

**o. Retrieve the names of pilots certified for some Boeing aircraft.**

```
SELECT DISTINCT e.ename
FROM Employees e
JOIN Certified c ON e.eid = c.eid
JOIN Aircraft a ON c.aid = a.aid
WHERE a.aname = 'Boeing';
```

**p. Find the aids of all aircraft that can be used on non-stop flights from Kolkata to Madras.**

```
SELECT a.aid
FROM Aircraft a
JOIN Flights f ON a.cruisingrange >= f.distance
WHERE f.`from` = 'Kolkata' AND f.`to` = 'Madras';
```

**q. Identify the flights that can be piloted by every pilot whose salary is more than 70,000.**

```
SELECT f.flno
FROM Flights f
WHERE NOT EXISTS (
   SELECT e.eid
   FROM Employees e
   WHERE e.salary > 70000 AND NOT EXISTS (
      SELECT c.aid
      FROM Certified c
      WHERE c.eid = e.eid AND c.aid = f.flno
   )
);
```

**r. Find the names of pilots who can operate planes with a range greater than 3,000 miles but are not certified on any Boeing aircraft.**

```
SELECT DISTINCT e.ename
FROM Employees e
JOIN Certified c ON e.eid = c.eid
JOIN Aircraft a ON c.aid = a.aid
WHERE a.cruisingrange > 3000
AND e.eid NOT IN (
   SELECT e1.eid
   FROM Employees e1
   JOIN Certified c1 ON e1.eid = c1.eid
   JOIN Aircraft a1 ON c1.aid = a1.aid
   WHERE a1.aname = 'Boeing'
);
```

**s. Find the eids of employees who make the highest salary in every airline.**

```
SELECT e.eid
FROM Employees e
WHERE e.salary = (SELECT MAX(e1.salary) FROM Employees e1 WHERE e1.eid = e.eid);
```

**t. Retrieve the eids of employees who make the second highest salary.**

```
SELECT e.eid
FROM Employees e
WHERE e.salary = (
   SELECT MAX(e1.salary)
   FROM Employees e1
   WHERE e1.salary < (SELECT MAX(e2.salary) FROM Employees e2)
);
```

**u. Find the eids of employees who are certified for the largest number of aircraft.**

```
SELECT c.eid
FROM Certified c
GROUP BY c.eid
ORDER BY COUNT(c.aid) DESC
LIMIT 1;
```

**v. Find the eids of employees who are certified for exactly three aircraft.**

```
SELECT c.eid
FROM Certified c
GROUP BY c.eid
HAVING COUNT(c.aid) = 3;
```

**w. Find the total amount paid to pilots who drove greater than 500,000 miles together across all their journeys on the routes from Chennai to Dublin and the return route as well. You need to consider all direct flights along with connecting flights as well.**

```
SELECT SUM(e.salary) AS total_paid
FROM Employees e
JOIN Certified c ON e.eid = c.eid
JOIN Flights f ON c.aid = f.flno
WHERE (f.`from` = 'Chennai' AND f.`to` = 'Dublin') OR (f.`from` = 'Dublin' AND f.`to` = 'Chennai')
GROUP BY e.eid
HAVING SUM(f.distance) > 500000;
```

**x. Is there a sequence of flights from Tiruchirappalli to Frankfurt? Each flight in the sequence is required to depart from the city that is the destination of the previous flight; the first flight must leave Tiruchirappalli, the last flight must reach Frankfurt, and there is no restriction on the number of intermediate flights. Your query must determine whether a sequence of flights from Tiruchirappalli to Frankfurt exists for any input Flights relation instance.**

```
WITH RECURSIVE FlightPath AS (
   SELECT `from`, `to`
   FROM Flights
   WHERE `from` = 'Tiruchirappalli'

   UNION ALL

   SELECT f.`from`, f.`to`
   FROM Flights f
   JOIN FlightPath p ON f.`from` = p.`to`
)
SELECT DISTINCT `to`
FROM FlightPath
WHERE `to` = 'Frankfurt';
```

**y)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 names of pilots who are certified for both Boeing and Airbus aircraft.**

```
SELECT e.ename
FROM Employees e
JOIN Certified c1 ON e.eid = c1.eid
JOIN Certified c2 ON e.eid = c2.eid
JOIN Aircraft a1 ON c1.aid = a1.aid
JOIN Aircraft a2 ON c2.aid = a2.aid
WHERE a1.aname = 'Boeing' AND a2.aname = 'Airbus';
```

**3) Write the following as triggers on the EMPLOYEE Schema which you have already created. In each case, disallow if it does not satisfy the stated constraint. You may assume that the desired condition holds before any change to the database is attempted. Also, prefer to modify the database, even if it means inserting tuples with NULL or default values, rather than rejecting the attempted modification.**

**a. Ensure that deleting details of an employee deletes their dependent records also.**

```
CREATE TRIGGER delete_employee_dependents
BEFORE DELETE ON EMPLOYEE
FOR EACH ROW
BEGIN
   DELETE FROM DEPENDENT WHERE Essn = OLD.Ssn;
END;
```

**b. Whenever a department with exactly one project is shifted to a new location, ensure that the project is also shifted to the new location.**

```
CREATE TRIGGER shift_department_project
AFTER UPDATE ON DEPT_LOCATIONS
FOR EACH ROW
BEGIN
   DECLARE project_count INT;

   SELECT COUNT(*) INTO project_count
   FROM PROJECT
   WHERE Dnum = OLD.Dnumber;

   IF project_count = 1 THEN
      UPDATE PROJECT
      SET Plocation = NEW.Dlocation
      WHERE Dnum = OLD.Dnumber;
   END IF;
END;
```

**c. Ensure at all times that there are no departments with more than 3 projects.**

```
CREATE TRIGGER limit_department_projects
BEFORE INSERT ON PROJECT
FOR EACH ROW
BEGIN
   DECLARE project_count INT;

   SELECT COUNT(*) INTO project_count
```

```
    FROM PROJECT
    WHERE Dnum = NEW.Dnum;

    IF project_count >= 3 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'A department cannot have more
than 3 projects';
    END IF;
END;
```

**d. Ensure that no employees work for more than one department.**

```
CREATE TRIGGER enforce_single_department
BEFORE UPDATE ON EMPLOYEE
FOR EACH ROW
BEGIN
    IF NEW.Dno != OLD.Dno THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'An employee cannot work for more
than one department';
    END IF;
END;
```

**e. Whenever a project is dropped, dissociate all the employees from the particular project.**

```
CREATE TRIGGER dissociate_employees_from_project
BEFORE DELETE ON PROJECT
FOR EACH ROW
BEGIN
    DELETE FROM WORKS_ON WHERE Pno = OLD.Pnumber;
END;
```

**f. When a new department is inaugurated, ensure that it is not co-located with any other departments.**

```
CREATE TRIGGER unique_department_location
BEFORE INSERT ON DEPT_LOCATIONS
FOR EACH ROW
BEGIN
    DECLARE loc_count INT;

    SELECT COUNT(*) INTO loc_count
    FROM DEPT_LOCATIONS
    WHERE Dlocation = NEW.Dlocation;

    IF loc_count > 0 THEN
```

```
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'No two departments can be
co-located';
   END IF;
END;
```

**g. For every employee, ensure that their dependent's Birthdate is less than their Birthdate.**

```
CREATE TRIGGER validate_dependent_birthdate
BEFORE INSERT ON DEPENDENT
FOR EACH ROW
BEGIN
   DECLARE emp_bdate DATE;

   SELECT Bdate INTO emp_bdate
   FROM EMPLOYEE
   WHERE Ssn = NEW.Essn;

   IF NEW.Bdate >= emp_bdate THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Dependent birthdate must be earlier
than employee birthdate';
   END IF;
END;
```

**h. Increment 1000 rupees to the salary of those employees if any of their dependents expire.**

```
CREATE TRIGGER increment_salary_on_dependent_death
AFTER DELETE ON DEPENDENT
FOR EACH ROW
BEGIN
   UPDATE EMPLOYEE
   SET Salary = Salary + 1000
   WHERE Ssn = OLD.Essn;
END;
```