Lab 3

1. SQL Queries Based on the Airlines Travel Schema

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

```
sql
SELECT DISTINCT a.aname
FROM Aircraft a
JOIN Certified c ON a.aid = c.aid
JOIN Employees e ON c.eid = e.eid
GROUP BY a.aname
HAVING MIN(e.salary) > 50000;
```

b. For each pilot who is certified for more than three aircraft, find the eid and the maximum cruisingrange of the aircraft for which she/he is certified.

```
sql
SELECT c.eid, MAX(a.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.

```
sql
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 cruisingrange over 1000 miles, find the name of the aircraft and the average salary of all pilots certified for this aircraft.

```
sql
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 pilot/s certified for some Boeing aircraft who drove the maximum distance on all flights departing from Ladakh.

```sql
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 = a.aid
WHERE a.aname = 'Boeing' AND f.from = 'Ladakh'
ORDER BY f.distance DESC
LIMIT 1;
```

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

```sql
SELECT DISTINCT a.aid
FROM Aircraft a
JOIN Flights f ON a.cruisingrange >= f.distance
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.

```sql
SELECT f.from, f.to
FROM Flights f
WHERE NOT EXISTS (
  SELECT c.eid
  FROM Certified c
  JOIN Employees e ON c.eid = e.eid
  WHERE e.salary > 100000
  AND NOT EXISTS (
    SELECT 1 FROM Certified c2
    WHERE c2.eid = c.eid AND c2.aid = f.flno
  )
);
```

h. Print the enames of pilots who can operate planes with cruisingrange greater than 3000 miles but are not certified on any Boeing aircraft.

```sql
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 a.aname != 'Boeing';
```

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

```sql
SELECT (AVG(e1.salary) - AVG(e2.salary)) AS salary_difference
FROM Employees e1, Employees e2
```

```
WHERE e1.eid IN (SELECT c.eid FROM Certified c)
AND e2.eid = e1.eid;
```

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

```
sql
SELECT e.ename, e.salary
FROM Employees e
WHERE e.eid NOT IN (SELECT c.eid FROM Certified c)
AND e.salary > (
  SELECT AVG(e2.salary)
  FROM Employees e2
  JOIN Certified c2 ON e2.eid = c2.eid
);
```

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

```
sql
SELECT DISTINCT e.ename
FROM Employees e
JOIN Certified c ON e.eid = c.eid
JOIN Aircraft a ON c.aid = a.aid
GROUP BY e.ename
HAVING MIN(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.

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

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.

```
sql
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 > 1000 AND a.aname = 'Boeing';
```

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

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

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

sql
```
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.

sql
```
SELECT DISTINCT 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.

sql
```
SELECT f.flno
FROM Flights f
WHERE NOT EXISTS (
  SELECT c.eid
  FROM Certified c
  JOIN Employees e ON c.eid = e.eid
  WHERE e.salary > 70000
  AND NOT EXISTS (
    SELECT 1 FROM Certified c2
    WHERE c2.eid = c.eid AND c2.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.

sql
```
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 a.aname != 'Boeing';
```

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

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

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

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

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

```sql
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.

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

w. Find the total amount paid to pilots who drove greater than 500,000 miles together across all their journey on the routes from Chennai to Dublin and return route also.

```sql
SELECT SUM(e.salary)
FROM Employees e
JOIN Certified c ON e.eid = c.eid
JOIN Flights f ON f.flno = c.aid
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?

sql
```
WITH RECURSIVE FlightPaths(f1, f2) AS (
  SELECT f.from, f.to
  FROM Flights f
  WHERE f.from = 'Tiruchirappalli'


  UNION
  SELECT fp.f1, f.to
  FROM FlightPaths fp
  JOIN Flights f ON fp.f2 = f.from
)
SELECT *
FROM FlightPaths
WHERE f2 = 'Frankfurt';
```

y. Create your own query:
Define what you want to do, then write the query.

This depends on your custom requirement. Could you provide more details?

2. Continuation to Session 04 Exercise: Execute Queries for TRIGGER, VIEW, EXCEPT, and CONTAINS

This would involve executing example queries from the specified sections of the textbook. You'll need to refer to the corresponding textbook examples for exact query structures.

3. Triggers on EMPLOYEE Schema

a. Assure that deleting details of an employee deletes his dependent records also.

sql
```
CREATE TRIGGER DeleteEmployee
AFTER DELETE ON Employees
FOR EACH ROW
BEGIN
  DELETE FROM Dependents WHERE eid = OLD.eid;
END;
```

b. Ensure that when a department with exactly one project is shifted to a new location, the project is also shifted.

sql
```
CREATE TRIGGER ShiftDepartment
AFTER UPDATE ON Department
FOR EACH ROW
WHEN (SELECT COUNT(*) FROM Project WHERE dept_id = OLD.dept_id) = 1
BEGIN
  UPDATE Project SET location = NEW.location WHERE dept_id = OLD.dept_id;
END;
```

c. Assure that no department has more than 3 projects.

sql
```
CREATE TRIGGER LimitProjects
BEFORE INSERT ON Project
FOR EACH ROW
WHEN (SELECT COUNT(*) FROM Project WHERE dept_id = NEW.dept_id) >= 3
BEGIN
  SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Department cannot have more than 3
projects';
END;
```

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

sql
```
CREATE TRIGGER OneDepartment
BEFORE INSERT ON WorksFor
FOR EACH ROW
WHEN (SELECT COUNT(*) FROM WorksFor WHERE eid = NEW.eid) > 1
BEGIN
  SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Employee cannot work for more than one
department';
END;
```

e. When a project is dropped, dissociate all employees from that project.

sql
```
CREATE TRIGGER DropProject
AFTER DELETE ON Project
FOR EACH ROW
BEGIN
  DELETE FROM WorksOn WHERE project_id = OLD.project_id;
END;
```

f. Ensure no new department is co-located with any other department.

sql
```
CREATE TRIGGER UniqueLocation
BEFORE INSERT ON Department
FOR EACH ROW
WHEN (SELECT COUNT(*) FROM Department WHERE location = NEW.location) > 0
BEGIN
  SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'No two departments can have the same
location';
END;
```

g. Ensure every employee's dependent birthdate is less than the employee's birthdate.

sql
```
CREATE TRIGGER ValidDependentBirthdate
BEFORE INSERT ON Dependents
FOR EACH ROW
WHEN NEW.birthdate >= (SELECT birthdate FROM Employees WHERE eid = NEW.eid)
BEGIN
  SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Dependent birthdate must be earlier than
employee birthdate';
```

END;

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

sql
```
CREATE TRIGGER IncrementSalaryOnDependentDeath
AFTER DELETE ON Dependents
FOR EACH ROW
BEGIN
  UPDATE Employees SET salary = salary + 1000 WHERE eid = OLD.eid;
END;
```