1)
```
-- Creating the schema

CREATE TABLE Flights (
  flno INTEGER PRIMARY KEY,
  from_city VARCHAR(50),
  to_city VARCHAR(50),
  distance INTEGER,
  departs TIME,
  arrives TIME,
  price INTEGER
);

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

CREATE TABLE Certified (
  eid INTEGER,
  aid INTEGER,
  PRIMARY KEY (eid, aid),
  FOREIGN KEY (aid) REFERENCES Aircraft(aid)
);

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

-- Inserting sample data

-- Flights
INSERT INTO Flights VALUES (1, 'Trichy', 'Agartala', 2300, '10:00:00', '13:00:00', 4000);
INSERT INTO Flights VALUES (2, 'Chandigarh', 'Surat', 1500, '09:00:00', '11:30:00', 3500);
INSERT INTO Flights VALUES (3, 'Ladakh', 'Delhi', 500, '12:00:00', '13:30:00', 3000);

-- Aircraft
INSERT INTO Aircraft VALUES (101, 'Boeing 747', 12000);
INSERT INTO Aircraft VALUES (102, 'Airbus A320', 6000);
INSERT INTO Aircraft VALUES (103, 'Boeing 737', 7000);
INSERT INTO Aircraft VALUES (104, 'Cessna 172', 800);

-- Employees
INSERT INTO Employees VALUES (201, 'John Doe', 55000);
INSERT INTO Employees VALUES (202, 'Jane Smith', 60000);
INSERT INTO Employees VALUES (203, 'Michael Brown', 45000);
INSERT INTO Employees VALUES (204, 'Emily Davis', 70000);
```

```
INSERT INTO Employees VALUES (205, 'David Wilson', 50000);

-- Certified
INSERT INTO Certified VALUES (201, 101);
INSERT INTO Certified VALUES (202, 101);
INSERT INTO Certified VALUES (202, 102);
INSERT INTO Certified VALUES (203, 103);
INSERT INTO Certified VALUES (204, 102);
INSERT INTO Certified VALUES (204, 104);
INSERT INTO Certified VALUES (205, 104);
```

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

```
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 cruising range of the aircraft for which she/he is certified.

```
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.

```
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)
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 DISTINCT 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 a.aid = f.flno
WHERE a.aname LIKE '%Boeing%'
AND f.from = 'Ladakh'
AND f.distance = (
  SELECT MAX(f2.distance)
  FROM Flights f2
  WHERE f2.from = 'Ladakh'
);
```

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

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

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

```
SELECT f.flno, 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.eid
    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 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 c2.eid
 FROM Certified c2
 JOIN Aircraft a2 ON c2.aid = a2.aid
 WHERE a2.aname LIKE '%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 e1.eid IN (SELECT c.eid FROM Certified c)
     ) -
     (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(e2.salary)
  FROM Employees e2
  WHERE e2.eid IN (SELECT c2.eid FROM Certified c2)
);
```

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 e.eid IN (
  SELECT c.eid
  FROM Certified c
  JOIN Aircraft a ON c.aid = a.aid
  GROUP BY c.eid
  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.

```
SELECT e.ename
FROM Employees e
WHERE e.eid IN (
  SELECT c.eid
  FROM Certified c
  JOIN Aircraft a ON c.aid = a.aid
  GROUP BY c.eid
  HAVING MAX(a.cruisingrange) < 1000 AND 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
WHERE e.eid IN (
  SELECT c.eid
  FROM Certified c
  JOIN Aircraft a ON c.aid = a.aid
  GROUP BY c.eid
  HAVING MIN(a.cruisingrange) > 1000 AND COUNT(DISTINCT CASE WHEN a.aname LIKE '%Boeing%' THEN 1 END) > 0
);
```

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

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

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

```
SELECT DISTINCT a.aid
FROM Aircraft a
JOIN Flights f ON a.aid = f.flno
```

```
WHERE f.from = 'Kolkata' AND f.to = 'Madras' AND a.cruisingrange >= f.distance;
```

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

```
SELECT f.flno, f.from, f.to
FROM Flights f
WHERE NOT EXISTS (
  SELECT e.eid
  FROM Employees e
  WHERE e.salary > 70000
  AND NOT EXISTS (
    SELECT c.eid
    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 3000 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 c2.eid
  FROM Certified c2
  JOIN Aircraft a2 ON c2.aid = a2.aid
  WHERE a2.aname LIKE '%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(e2.salary)
  FROM Employees e2
);
```

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

```
SELECT e.eid
FROM Employees e
```

```
WHERE e.salary = (
  SELECT MAX(e2.salary)
  FROM Employees e2
  WHERE e2.salary < (
    SELECT MAX(e3.salary)
    FROM Employees e3
  )
);
```

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
HAVING COUNT(c.aid) = (
  SELECT MAX(counts)
  FROM (SELECT COUNT(c2.aid) AS counts
      FROM Certified c2
      GROUP BY c2.eid) AS subquery
);
```

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 return route.

```
SELECT SUM(e.salary)
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? Your query must determine whether a sequence exists for any input Flights relation instance.

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

```
  UNION
  SELECT fp.from, f.to
  FROM FlightPath fp
  JOIN Flights f ON fp.to = f.from
)
SELECT 'Yes, a sequence exists' AS result
FROM FlightPath
WHERE to = 'Frankfurt'
LIMIT 1;
```

2) EXCEPT:

mysql> select Fname from Employee where not exists((select Pnumber from Project where Dnum=5) except (select Pno from Works_on where Essn=Ssn));

```
+----------+
| Fname |
+----------+
| Alam XYZ |
+----------+
```

1 row in set (0.05 sec)

VIEW:

mysql> create view Works_on1 as select Fname,Lname,Pname,Hours from Employee ,Project,Works_on where Ssn=Essn and Pno=Pnumber;

Query OK, 0 rows affected (0.03 sec)

Retrieve the last name and first name of all employees who work on the ‘ProjectX’ project,

mysql> select Fname, Lname from Works_on1 where Pname="ProjectX";

```
+----------+--------+
| Fname | Lname |
+----------+--------+
| Alam XYZ | Marini |
| Mukesh | Ragav |
| Andrea | Khan |
+----------+--------+
```

3 rows in set (0.00 sec)

mysql> create view Dept_info(Dept_name,No_of_emp,Total_sal) as select Dname, count(*),sum(Salary) from Department,Employee where Dno=Dnumber group by Dno;

Query OK, 0 rows affected (0.01 sec)

mysql> select * from Dept_info;

```
+----------------+-----------+-----------+
| Dept_name | No_of_emp | Total_sal |
+----------------+-----------+-----------+
| CS | 2 | 141000.00 |
| IT | 6 | 228000.00 |
| Headquarters | 1 | 39000.00 |
| Administration | 1 | 41000.00 |
| Research | 1 | 30000.00 |
```

```
| Clinic | 2 | 130000.00 |
+----------------+-----------+-----------+
6 rows in set (0.01 sec)
mysql> drop view Works_on1;
Query OK, 0 rows affected (0.02 sec)
mysql> update Works_on1 set Fname="Allen" where Lname="Mar";
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0
mysql> create view Dept5emp as select * from Employee where Dno=5;
Query OK, 0 rows affected (0.01 sec)
mysql> create view basic_emp_detail as select Fname,Lname,Address from Emplo
yee;
Query OK, 0 rows affected (0.01 sec)
TRIGGER:
mysql> create table log(Super_ssn int,Ssn int);
Query OK, 0 rows affected (0.05 sec)
mysql> delimiter //
mysql> create procedure inform_supervisor(super_ssn int ,ssn int)
-> begin
-> insert into log values(super_ssn,ssn);
-> end//
Query OK, 0 rows affected (0.01 sec)
mysql> delimiter //
mysql> create trigger Salary_violation1
-> before insert on Employee
-> for each row
-> begin
-> if NEW.Salary>(select Salary from Employee where Ssn=NEW.Super_ssn)
-> then call inform_supervisor(NEW.Super_ssn,NEW.Ssn);
-> end if;
-> end//
Query OK, 0 rows affected (0.01 sec)
mysql> delimiter ;
mysql> create trigger Salary_violation_update
-> before update on Employee
-> for each row
-> begin
-> if NEW.Salary>(select Salary from Employee where NEW.Super_ssn=Ssn)
-> then call inform_supervisor(NEW.Super_ssn,NEW.Ssn);
-> end if;
-> end//
Query OK, 0 rows affected (0.02 sec)
mysql> insert into Employee values("Rahul","R","Anand",'653298699','1962-12-30','177 Oak
Forest,Katy,TX','M',80000,'653298665',5);
Query OK, 1 row affected (0.01 sec)
mysql> select * from log;
+-----------+-----------+
| Super_ssn | Ssn |
+-----------+-----------+
```

```
| 653298665 | 653298699 |
+-----------+-----------+
1 row in set (0.00 sec)
```

3) A) Assure that deleting details of an employee deletes his dependent records also.

```
CREATE OR REPLACE TRIGGER trg_delete_employee
BEFORE DELETE ON EMPLOYEE
FOR EACH ROW
BEGIN
  DELETE FROM DEPENDENT WHERE EMPLOYEE_ID = :OLD.EMPLOYEE_ID;
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 OR REPLACE TRIGGER trg_update_dept_location
BEFORE UPDATE OF LOCATION ON DEPARTMENT
FOR EACH ROW
DECLARE
  proj_count INTEGER;
BEGIN
  SELECT COUNT(*) INTO proj_count FROM PROJECT WHERE DEPARTMENT_ID = :OLD.DEPARTMENT_ID;

  IF proj_count = 1 THEN
    UPDATE PROJECT SET LOCATION = :NEW.LOCATION WHERE DEPARTMENT_ID = :OLD.DEPARTMENT_ID;
  END IF;
END;
/
```

C) Assure at all times that there are no departments with more than 3 projects.

```
CREATE OR REPLACE TRIGGER trg_check_project_count
BEFORE INSERT OR UPDATE ON PROJECT
FOR EACH ROW
DECLARE
  proj_count INTEGER;
BEGIN
  SELECT COUNT(*) INTO proj_count FROM PROJECT WHERE DEPARTMENT_ID = :NEW.DEPARTMENT_ID;

  IF proj_count >= 3 THEN
    RAISE_APPLICATION_ERROR(-20001, 'A department cannot have more than 3 projects.');
  END IF;
END;
/
```

D) Assure that no employees work for more than one department.

```
CREATE OR REPLACE TRIGGER trg_check_employee_department
BEFORE INSERT OR UPDATE ON WORKS_FOR
FOR EACH ROW
```

```
DECLARE
  dept_count INTEGER;
BEGIN
  SELECT COUNT(*) INTO dept_count FROM WORKS_FOR WHERE EMPLOYEE_ID = :NEW.EMPLOYEE_ID;

  IF dept_count >= 1 THEN
    RAISE_APPLICATION_ERROR(-20002, '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 OR REPLACE TRIGGER trg_delete_project
BEFORE DELETE ON PROJECT
FOR EACH ROW
BEGIN
  DELETE FROM WORKS_ON WHERE PROJECT_ID = :OLD.PROJECT_ID;
END;
/
```

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

```
CREATE OR REPLACE TRIGGER trg_check_department_location
BEFORE INSERT ON DEPARTMENT
FOR EACH ROW
DECLARE
  loc_count INTEGER;
BEGIN
  SELECT COUNT(*) INTO loc_count FROM DEPARTMENT WHERE LOCATION = :NEW.LOCATION;

  IF loc_count > 0 THEN
    RAISE_APPLICATION_ERROR(-20003, 'Another department is already located at this location.');
  END IF;
END;
/
```

G) For every employee, ensure that his dependent's Birthdate is less than his Birthdate.

```
CREATE OR REPLACE TRIGGER trg_check_dependent_birthdate
BEFORE INSERT OR UPDATE ON DEPENDENT
FOR EACH ROW
DECLARE
  emp_birthdate DATE;
BEGIN
  SELECT BIRTHDATE INTO emp_birthdate FROM EMPLOYEE WHERE EMPLOYEE_ID = :NEW.EMPLOYEE_ID;

  IF :NEW.BIRTHDATE >= emp_birthdate THEN
    RAISE_APPLICATION_ERROR(-20004, 'Dependent birthdate must be less than employee birthdate.');
  END IF;
END;
/
```

H) Increment 1000 rupees to the salary for those employees if any of his/her dependent expire.

```
CREATE OR REPLACE TRIGGER trg_increment_salary_on_dependent_expire
AFTER DELETE ON DEPENDENT
FOR EACH ROW
BEGIN
  UPDATE EMPLOYEE
  SET SALARY = SALARY + 1000
  WHERE EMPLOYEE_ID = :OLD.EMPLOYEE_ID;
END;
/
```