Q6) Consider the given database:
Project (project_id, proj_name, chief_arch)
Employee (emp_id, emp_name)
Assigned_To (project_id , emp_id)
Find the sqL queries for the following
Statements:
i. Get the employee number of employees working on project C353
2) Get details of employees working on project C353.
3) Obtain details of employees working on database project
4) Get the employee number of employees who are not on any project.
5) Display project_id, proj_name, emp_id and emp_name
6) Display project_id, proj_name, emp_id end emp_name who are cooking on project 'Avenue Archi'.

```
CREATE TABLE Project (
    project_id VARCHAR(10) PRIMARY KEY,
    proj_name VARCHAR(100),
    chief_arch VARCHAR(100)
);

CREATE TABLE Employee (
    emp_id INT PRIMARY KEY,
    emp_name VARCHAR(100)
);

CREATE TABLE Assigned_To (
    project_id VARCHAR(10),
    emp_id INT,
    FOREIGN KEY (project_id) REFERENCES Project(project_id),
    FOREIGN KEY (emp_id) REFERENCES Employee(emp_id)
);

INSERT INTO Project (project_id, proj_name, chief_arch) VALUES
    ('C353', 'Project C353', 'Chief Architect Alpha'),
    ('D789', 'Database Project D789', 'Chief Architect Beta'),
    ('A111', 'Avenue Archi', 'Chief Architect Sigma');

INSERT INTO Employee (emp_id, emp_name) VALUES
    (1, 'Shriyog More'),
    (2, 'Naresh Suthar'),
    (3, 'Aryan Pawar'),
    (4, 'Rushikesh Kadam');
INSERT INTO Assigned_To (project_id, emp_id) VALUES
```

```
('C353', 1),
('C353', 2),
('D789', 3),
('A111', 1),
('A111', 4);
```

1) Get the employee number of employees working on project C353

```
SELECT emp_id
FROM Assigned_To
WHERE project_id = 'C353';
```

2) Get details of employees working on project C353.

```
SELECT Employee.emp_id, emp_name
FROM Employee
JOIN Assigned_To ON Employee.emp_id = Assigned_To.emp_id
WHERE project_id = 'C353';
```

3)Obtain details of employees working on database project

```
SELECT Employee.emp_id, emp_name
FROM Employee
JOIN Assigned_To ON Employee.emp_id = Assigned_To.emp_id
JOIN Project ON Assigned_To.project_id = Project.project_id
WHERE proj_name LIKE '%database%';
```

4)Get the employee number of employees who are not on any project.

```
SELECT emp_id
FROM Employee
WHERE emp_id NOT IN (SELECT emp_id FROM Assigned_To);
```

5) Display project_id, proj_name, emp_id and emp_name

```
SELECT Project.project_id, proj_name, Employee.emp_id, emp_name
FROM Project
JOIN Assigned_To ON Project.project_id = Assigned_To.project_id
JOIN Employee ON Assigned_To.emp_id = Employee.emp_id;
```

6) Display project_id, proj_name, emp_id end emp_name who are cooking on project 'Avenue Archi'.

```
SELECT Project.project_id, proj_name, Employee.emp_id, emp_name
```

```
FROM Project
JOIN Assigned_To ON Project.project_id = Assigned_To.project_id
JOIN Employee ON Assigned_To.emp_id = Employee.emp_id
WHERE proj_name = 'Avenue Archi';
```

—---

Q8) Consider the relational database:
dept(dept_no, dname, loc, mgrcode)
emp(emp_no, ename, designation)
project(proj_no, proj_name,status)
dept and emp are related as 1 to many.
project and emp are related as 1 to many.
Write queries for the following:
1. List all employees of 'INVENTORY' department of 'PUNE' location.
2. Give the names of employees who are working on 'Blood Bank' project.
3. Given the name of managers from 'MARKETING' department.
4. Give all the employees working under status 'INCOMPLETE' projects.
5. Write a trigger before insert on emp table to auto insert the record and also update the dept table.
6. Wtite a view to display all the employee in the 'Computer' Department.

```
CREATE TABLE dept (
    dept_no INT PRIMARY KEY,
    dname VARCHAR(100),
    loc VARCHAR(100),
    mgrcode INT
);

CREATE TABLE emp (
    emp_no INT PRIMARY KEY,
    ename VARCHAR(100),
    designation VARCHAR(100),
    dept_no INT,
    FOREIGN KEY (dept_no) REFERENCES dept(dept_no)
);

CREATE TABLE project (
    proj_no INT PRIMARY KEY,
    proj_name VARCHAR(100),
    status VARCHAR(100),
    emp_no INT,
    FOREIGN KEY (emp_no) REFERENCES emp(emp_no)
);

INSERT INTO dept (dept_no, dname, loc, mgrcode) VALUES
(1, 'INVENTORY', 'PUNE', 101),
(2, 'MARKETING', 'DELHI', 102),
(3, 'Computer', 'MUMBAI', 103);

INSERT INTO emp (emp_no, ename, designation, dept_no) VALUES
(101, 'Shriyog', 'Manager', 1),
(102, 'Naresh', 'Manager', 2),
(103, 'Aryan', 'Manager', 3),
(201, 'Rishikesh', 'Developer', 3),
(202, 'Yash', 'Analyst', 3),
(203, 'Atharva', 'Tester', 3);

INSERT INTO project (proj_no, proj_name, status, emp_no) VALUES
(1, 'Blood Bank', 'INCOMPLETE', 201),
(2, 'Inventory Management', 'COMPLETE', 201),
(3, 'Marketing Campaign', 'INCOMPLETE', 202);
```

1)List all employees of 'INVENTORY' department of 'PUNE' location.

```
SELECT emp_no, ename
FROM emp
WHERE dept_no IN (SELECT dept_no FROM dept WHERE dname = 'INVENTORY' AND loc =
'PUNE');
```

2)

```
SELECT ename
FROM emp
WHERE emp_no IN (SELECT emp_no FROM project WHERE proj_name = 'Blood Bank');
```

3)

```
SELECT ename
FROM emp
WHERE emp_no = (SELECT mgrcode FROM dept WHERE dname = 'MARKETING');
```

4)

```
SELECT ename
FROM emp
WHERE emp_no IN (SELECT emp_no FROM project WHERE status = 'INCOMPLETE');
```

5)

```
DELIMITER //

CREATE TRIGGER insert_emp_trigger
BEFORE INSERT ON emp
FOR EACH ROW
BEGIN
   -- Update the dept table with new emp_no if necessary
   IF NEW.designation = 'Manager' THEN
      UPDATE dept
      SET mgrcode = NEW.emp_no
      WHERE dept_no = NEW.dept_no;
   END IF;
END;
//

DELIMITER ;


INSERT INTO emp (emp_no, ename, designation, dept_no) VALUES (301, 'Shivraj', 'Manager',
3);

SELECT * FROM emp;

SELECT * FROM dept WHERE dept_no = 3;

INSERT INTO emp (emp_no, ename, designation, dept_no) VALUES (302, 'Dinesh', 'Analyst',
3);

SELECT * FROM emp;

SELECT * FROM dept WHERE dept_no = 3;
```

6)

```
CREATE VIEW Computer_Department_Employees AS
SELECT emp.emp_no, emp.ename, emp.designation
FROM emp
JOIN dept ON emp.dept_no = dept.dept_no
WHERE dept.dname = 'Computer';
```

```
SELECT * FROM Computer_Department_Employees;
```

—--

Q9) Consider the following six relations for all order processing database application is a company.
Customer(cust #, cname, city)
order (order #, odate, cust #, ord _amt)
order_item(order #, item #, Qty)
item(item #, unit_price)
shipment(order #, waterhouse #, ship _date)
warehouse(warehouse #, city)
Here, ord _amt refers to total amount of an order; odate is the date the order was placed; ship_date is the date an order is shipped from the warehouse. Assume that an order can be shipped from several warehouses.
Specify the following SQL queries:
1. List the order # and ship_date for all orders shipped from warehouse number W2.
2. List the warehouse information from which the customer named Jose Lopez' was shipped his order.
3. List the orders that were not shipped within 30 days of ordering.
4. List the order # for orders that were shipped from all warehouse that the company has in New York.
5. Write a cursor to extract order details

```
CREATE TABLE Customer (
   cust INT PRIMARY KEY,
   cname VARCHAR(50),
   city VARCHAR(50)
);


CREATE TABLE `Order` (
   `order` INT PRIMARY KEY,
   odate DATE,
   cust INT,
   ord_amt INT,
   FOREIGN KEY (cust) REFERENCES Customer(cust)
);

CREATE TABLE Order_Item (
   `order` INT,
   item INT,
   Qty INT,
   PRIMARY KEY (`order`, item),
   FOREIGN KEY (`order`) REFERENCES `Order`(`order`)
);

CREATE TABLE Item (
   item INT PRIMARY KEY,
   unit_price DECIMAL(10,2)
);

CREATE TABLE Shipment (
   `order` INT,
   warehouse VARCHAR(10),
   ship_date DATE,
   PRIMARY KEY (`order`, warehouse),
   FOREIGN KEY (`order`) REFERENCES `Order`(`order`)
);

CREATE TABLE Warehouse (
   warehouse VARCHAR(10) PRIMARY KEY,
   city VARCHAR(50)
);


INSERT INTO Customer (cust, cname, city) VALUES
(1, 'Jose Lopez', 'New York'),
(2, 'Maria Garcia', 'Los Angeles'),
(3, 'John Smith', 'Chicago');

INSERT INTO `Order` (`order`, odate, cust, ord_amt) VALUES
(101, '2024-05-01', 1, 100),
(102, '2024-05-15', 2, 150),
(103, '2024-05-20', 3, 200);

INSERT INTO Order_Item (`order`, item, Qty) VALUES
(101, 1, 2),
(101, 2, 1),
(102, 3, 3),
(103, 1, 1),
(103, 2, 2);

INSERT INTO Item (item, unit_price) VALUES
(1, 10.00),
(2, 20.00),
(3, 30.00);

INSERT INTO Shipment (`order`, warehouse, ship_date) VALUES
(101, 'W1', '2024-05-05'),
(101, 'W2', '2024-05-07'),
(102, 'W1', '2024-05-17'),
(103, 'W2', '2024-06-10');

INSERT INTO Warehouse (warehouse, city) VALUES
('W1', 'New York'),
('W2', 'Los Angeles'),
('W3', 'Chicago');
```

1)

```
SELECT `order`, ship_date
FROM Shipment
WHERE warehouse = 'W2';
```

2)

```
SELECT w.*
FROM Warehouse w
JOIN Shipment s ON w.warehouse = s.warehouse
JOIN `Order` o ON s.`order` = o.`order`
JOIN Customer c ON o.cust = c.cust
WHERE c.cname = 'Jose Lopez';
```

3)

```
SELECT o.`order`
FROM `Order` o
LEFT JOIN Shipment s ON o.`order` = s.`order`
WHERE DATEDIFF(s.ship_date, o.odate) > 30 OR s.ship_date IS NULL;
```

4)

```
SELECT s.`order`
FROM Shipment s
JOIN Warehouse w ON s.warehouse = w.warehouse
WHERE w.city = 'New York'
GROUP BY s.`order`
HAVING COUNT(DISTINCT s.warehouse) = (
   SELECT COUNT(*)
   FROM Warehouse
   WHERE city = 'New York'
);
```

5)

```
DELIMITER //

CREATE PROCEDURE ExtractOrderDetails()
BEGIN
   DECLARE done INT DEFAULT FALSE;
   DECLARE order_number INT;
   DECLARE ship_date DATE;
   DECLARE cursor_order CURSOR FOR
      SELECT `order`, ship_date
      FROM Shipment;
   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

   OPEN cursor_order;

   read_loop: LOOP
      FETCH cursor_order INTO order_number, ship_date;
      IF done THEN
         LEAVE read_loop;
      END IF;
      -- Process order details here
      SELECT order_number, ship_date; -- Example: You can print or process the values here
   END LOOP;

   CLOSE cursor_order;
END//

DELIMITER ;

CALL ExtractOrderDetails();
```

—--

Q12) A book seller wants to keep following information
1, Name, mobile, title of books purchased, category of books.(Biographic, Cultural, Historical, Cooking, Kids)
2. List names of persons and categories of books they purchased and total cost.
3. It a person purchase books of Rs, 5000/- then he is allowed to purchase additional books of Rs. 1000/-

List of persons who purchased books more than Rs. 5000/-,
Name of person who purehased minimum amount of books and a name of person who purchased maximum amount of books.
6. Total sell (count of books and amount) during given period.
7. Count of books sold in each category.
8. Count of persons who purchased books of cost more than Rs. 3000/-.
9 List names of persons who purchased books more than 3 times in a given period.
10, List of different books purchased during given period.

```
CREATE TABLE Persons (
   person_id INT PRIMARY KEY,
   name VARCHAR(50),
   mobile VARCHAR(15)
);

CREATE TABLE Books (
   book_id INT PRIMARY KEY,
   title VARCHAR(100),
   category ENUM('Biographic', 'Cultural', 'Historical', 'Cooking', 'Kids'),
   price DECIMAL(10, 2)
);

CREATE TABLE Purchases (
   purchase_id INT PRIMARY KEY,
   person_id INT,
   book_id INT,
   purchase_date DATE,
   FOREIGN KEY (person_id) REFERENCES Persons(person_id),
   FOREIGN KEY (book_id) REFERENCES Books(book_id)
);

INSERT INTO Persons (person_id, name, mobile) VALUES
(1, 'Shriyog', '1234567890'),
(2, 'Naresh', '9876543210'),
(3, 'Aryan', '9998887776'),
(4, 'Rishikesh', '5554443332'),
(5, 'Yash', '1112223334');

INSERT INTO Books (book_id, title, category, price) VALUES
(101, 'Biography of XYZ', 'Biographic', 5000),
(102, 'Cultural Studies 101', 'Cultural', 6000),
(103, 'History of Civilization', 'Historical', 7000),
(104, 'The Art of Cooking', 'Cooking', 4000),
(105, 'Children Stories', 'Kids', 3000);

INSERT INTO Purchases (purchase_id, person_id, book_id, purchase_date) VALUES
(1, 1, 101, '2024-05-01'),
(2, 1, 102, '2024-05-03'),
(3, 2, 103, '2024-05-05'),
(4, 2, 104, '2024-05-10'),
(5, 3, 105, '2024-05-15'),
(6, 3, 101, '2024-05-20'),
(7, 3, 102, '2024-05-25'),
(8, 4, 102, '2024-05-02'),
(9, 4, 104, '2024-05-08'),
(10, 5, 101, '2024-05-10'),
(11, 5, 103, '2024-05-12'),
(12, 5, 104, '2024-05-14'),
(13, 5, 105, '2024-05-16'),
(14, 5, 101, '2024-05-18');
```

1)

```
SELECT p.name, b.category, SUM(b.price) AS total_cost
FROM Persons p
JOIN Purchases pur ON p.person_id = pur.person_id
JOIN Books b ON pur.book_id = b.book_id
GROUP BY p.name, b.category;
```

2)

```
SELECT p.name
FROM Persons p
JOIN (
   SELECT person_id, SUM(b.price) AS total_price
   FROM Purchases pur
   JOIN Books b ON pur.book_id = b.book_id
   GROUP BY person_id
) AS pur_sum ON p.person_id = pur_sum.person_id
WHERE pur_sum.total_price > 5000;
```

3)

```
SELECT p.name
FROM Persons p
JOIN (
   SELECT person_id, SUM(b.price) AS total_price
```

```
    FROM Purchases pur
    JOIN Books b ON pur.book_id = b.book_id
    GROUP BY person_id
) AS pur_sum ON p.person_id = pur_sum.person_id
WHERE pur_sum.total_price > 5000;
```

4)

```
SELECT p.name
FROM Persons p
JOIN (
    SELECT pur.person_id, SUM(b.price) AS total_price
    FROM Purchases pur
    JOIN Books b ON pur.book_id = b.book_id
    GROUP BY pur.person_id
    ORDER BY total_price ASC
    LIMIT 1
) AS min_pur ON p.person_id = min_pur.person_id
UNION
SELECT p.name
FROM Persons p
JOIN (
    SELECT pur.person_id, SUM(b.price) AS total_price
    FROM Purchases pur
    JOIN Books b ON pur.book_id = b.book_id
    GROUP BY pur.person_id
    ORDER BY total_price DESC
    LIMIT 1
) AS max_pur ON p.person_id = max_pur.person_id;
```

5)

```
SELECT
    COUNT(*) AS total_books_sold,
    SUM(b.price) AS total_amount_sold
FROM Purchases pur
JOIN Books b ON pur.book_id = b.book_id
WHERE pur.purchase_date BETWEEN '2024-05-01' AND '2024-05-25';
```

6)

```
SELECT
    category,
    COUNT(*) AS books_sold
FROM Books b
```

```
JOIN Purchases pur ON b.book_id = pur.book_id
WHERE pur.purchase_date BETWEEN '2024-05-01' AND '2024-05-25'
GROUP BY category;
```

7)

```
SELECT COUNT(DISTINCT person_id) AS persons_count
FROM (
   SELECT pur.person_id, SUM(b.price) AS total_price
   FROM Purchases pur
   JOIN Books b ON pur.book_id = b.book_id
   GROUP BY pur.person_id
) AS pur_sum
WHERE total_price > 3000;
```

8)

```
SELECT p.name
FROM Persons p
JOIN (
   SELECT person_id, COUNT(*) AS purchase_count
   FROM Purchases
   WHERE purchase_date BETWEEN '2024-05-01' AND '2024-05-25'
   GROUP BY person_id
) AS pur_count ON p.person_id = pur_count.person_id
WHERE pur_count.purchase_count > 3;
```

9)

```
SELECT DISTINCT title
FROM Books b
JOIN Purchases pur ON b.book_id = pur.book_id
WHERE pur.purchase_date BETWEEN '2024-05-01' AND '2024-05-25';
```

10)

```
SELECT p.name
FROM Persons p
JOIN (
   SELECT person_id, COUNT(*) AS purchase_count
   FROM Purchases
   WHERE purchase_date BETWEEN '2024-05-01' AND '2024-05-25'
   GROUP BY person_id
) AS pur_count ON p.person_id = pur_count.person_id
```

```
WHERE pur_count.purchase_count > 3;
```

—---

Q13) A fish shop is selling approximately 100Kg fish daily. Shop is having 10 types of different fishes in the shop. Cost of each fish is different and it is per Kg. Shop man wants to keep following record.

1. Quantity of fishes purchase each day, cost of purchase, transportation cost.
2. Quantity of fishes sold per day and selling price
3. Name and mobile number of person purchasing fish and types of fishes he purchases
4. Find frequency of a person purchasing fishes
5. Find daily sales amount and sales amount between given period
6. If any person has purchased 5 times in a month and total cost of purchase is more than Rs 1000/- then given him discount of 10% for next 3 purchases in the same month
7. If any person purchases 5Kg of costliest fish then he will be offered 1Kg of non-selling fish.
8. List name of persons purchasing costliest fish 5 times in a month.
9. List names of persons purchasing fishes 3 times in a month.
10. List names of fishes which are sold every day and which are sold once in a month.

```
CREATE TABLE PurchaseRecords (
   record_id INT PRIMARY KEY AUTO_INCREMENT,
   purchase_date DATE,
   fish_type VARCHAR(50),
   quantity DECIMAL(10, 2),
   purchase_cost DECIMAL(10, 2),
   transportation_cost DECIMAL(10, 2)
);

CREATE TABLE SalesRecords (
   record_id INT PRIMARY KEY AUTO_INCREMENT,
   sale_date DATE,
   fish_type VARCHAR(50),
   quantity DECIMAL(10, 2),
   selling_price DECIMAL(10, 2)
);

CREATE TABLE CustomerPurchases (
   purchase_id INT PRIMARY KEY AUTO_INCREMENT,
   customer_name VARCHAR(100),
   mobile_number VARCHAR(20),
   purchase_date DATE,
   fish_type VARCHAR(50),
    quantity DECIMAL(10, 2)
);


-- Insert sample values into PurchaseRecords table
INSERT INTO PurchaseRecords (purchase_date, fish_type, quantity, purchase_cost,
transportation_cost)
VALUES
    ('2024-05-01', 'Tuna', 20, 4000.00, 200.00),
    ('2024-05-01', 'Salmon', 15, 3000.00, 150.00),
    ('2024-05-02', 'Cod', 25, 2500.00, 100.00),
    ('2024-05-02', 'Trout', 30, 2400.00, 120.00),
    ('2024-05-03', 'Snapper', 18, 2700.00, 130.00);

-- Insert sample values into SalesRecords table
INSERT INTO SalesRecords (sale_date, fish_type, quantity, selling_price)
VALUES
    ('2024-05-01', 'Tuna', 18, 6000.00),
    ('2024-05-01', 'Salmon', 12, 4800.00),
    ('2024-05-02', 'Cod', 20, 4000.00),
    ('2024-05-02', 'Trout', 25, 5000.00),
    ('2024-05-03', 'Snapper', 15, 3750.00);

-- Insert sample values into CustomerPurchases table
INSERT INTO CustomerPurchases (customer_name, mobile_number, purchase_date,
fish_type, quantity)
VALUES
    ('Shriyog', '1234567890', '2024-05-01', 'Tuna', 10),
    ('Shriyog', '1234567890', '2024-05-01', 'Salmon', 8),
    ('Aryan', '9876543210', '2024-05-02', 'Cod', 12),
    ('Aryan', '9876543210', '2024-05-02', 'Trout', 15),
    ('Naresh', '5551234567', '2024-05-03', 'Snapper', 10);
```

1)

```
SELECT purchase_date, SUM(quantity) AS total_quantity, SUM(purchase_cost) AS
total_purchase_cost, SUM(transportation_cost) AS total_transportation_cost
FROM PurchaseRecords
GROUP BY purchase_date;
```

2)

```
SELECT sale_date, SUM(quantity) AS total_quantity_sold, SUM(selling_price) AS
total_sales_amount
```

```
FROM SalesRecords
GROUP BY sale_date;
```

3)

```
SELECT customer_name, mobile_number, GROUP_CONCAT(DISTINCT fish_type ORDER BY
fish_type ASC SEPARATOR ', ') AS purchased_fishes
FROM CustomerPurchases
GROUP BY customer_name, mobile_number;
```

4)

```
SELECT customer_name, mobile_number, COUNT(*) AS purchase_frequency
FROM CustomerPurchases
GROUP BY customer_name, mobile_number;
```

5)

```
SELECT sale_date, SUM(quantity * selling_price) AS daily_sales_amount
FROM SalesRecords
GROUP BY sale_date;

SELECT SUM(quantity * selling_price) AS total_sales_amount
FROM SalesRecords
WHERE sale_date BETWEEN '2024-05-01' AND '2024-05-02';
```

6)

```
SELECT customer_name, mobile_number, MONTH(purchase_date) AS month,
      COUNT(*) AS purchase_count, SUM(quantity * purchase_cost) AS total_purchase_cost
FROM CustomerPurchases
GROUP BY customer_name, mobile_number, MONTH(purchase_date)
HAVING purchase_count = 5 AND total_purchase_cost > 1000;
```

7)

```
-- Identify customers who purchased 5Kg of the costliest fish
SELECT customer_name, mobile_number
FROM (
   SELECT customer_name, mobile_number, SUM(quantity) AS total_quantity
   FROM CustomerPurchases
```

```
    WHERE fish_type = (SELECT fish_type FROM SalesRecords ORDER BY selling_price
DESC LIMIT 1)
    GROUP BY customer_name, mobile_number
) AS purchases_of_costliest_fish
WHERE total_quantity >= 5;
```

8)

```
SELECT customer_name, mobile_number
FROM (
    SELECT customer_name, mobile_number, MONTH(purchase_date) AS month, COUNT(*)
AS purchase_count
    FROM CustomerPurchases
    WHERE fish_type = (SELECT fish_type FROM SalesRecords ORDER BY selling_price
DESC LIMIT 1)
    GROUP BY customer_name, mobile_number, MONTH(purchase_date)
) AS purchases_of_costliest_fish
WHERE purchase_count >= 5;
```

9)

```
SELECT customer_name, mobile_number
FROM (
    SELECT customer_name, mobile_number, MONTH(purchase_date) AS month, COUNT(*)
AS purchase_count
    FROM CustomerPurchases
    GROUP BY customer_name, mobile_number, MONTH(purchase_date)
) AS purchases_per_month
WHERE purchase_count >= 3;
```

10)

```
-- List names of fishes sold every day
SELECT fish_type
FROM (
    SELECT fish_type, COUNT(DISTINCT sale_date) AS days_count
    FROM SalesRecords
    GROUP BY fish_type
) AS daily_sales
WHERE days_count = (SELECT COUNT(DISTINCT sale_date) FROM SalesRecords);

-- List names of fishes sold once in a month
SELECT fish_type
FROM (
    SELECT fish_type, COUNT(DISTINCT sale_date) AS days_count
    FROM SalesRecords
    GROUP BY fish_type
) AS monthly_sales
WHERE days_count = 1;
```

—---

Q15) A school wants to keep following information of their students
1. Name, distance of his house from school, own or rental house
2. Father/Mother working status, type of organization they are working
3. Count of students whose one parent working and both parent working
4. Count of students leaving in own house and rental.
5. If the student is deleted from the table then his information must be transferred to other table.
6. Count of students whose father and mother type of organization is same.
7. Count of students according to their distances from school.
8. Name of students whose mother is working and name of students whose father is working.
List name if it is coming more than once.

```
CREATE TABLE Parents (
    parent_id INT PRIMARY KEY,
    name VARCHAR(50),
    working_status ENUM('working', 'not working'),
    organization_type VARCHAR(50)
);

CREATE TABLE Students (
    student_id INT PRIMARY KEY,
    name VARCHAR(50),
    house_distance FLOAT,
    house_type ENUM('own', 'rental'),
    father_id INT,
    mother_id INT,
    FOREIGN KEY (father_id) REFERENCES Parents(parent_id),
    FOREIGN KEY (mother_id) REFERENCES Parents(parent_id)
);

CREATE TABLE HouseDetails (
    house_id INT PRIMARY KEY,
    student_id INT,
    own_or_rental ENUM('own', 'rental'),
    house_distance FLOAT,
    FOREIGN KEY (student_id) REFERENCES Students(student_id)
);

INSERT INTO Parents (parent_id, name, working_status, organization_type) VALUES
(1, 'John', 'working', 'Business'),
(2, 'Alice', 'working', 'Service'),
(3, 'David', 'not working', NULL),
(4, 'Lisa', 'working', 'Defense');


INSERT INTO Students (student_id, name, house_distance, house_type, father_id, mother_id)
VALUES
(1, 'Shriyog', 5.5, 'own', 1, 2),
(2, 'Naresh', 3.2, 'rental', 3, 4),
(3, 'Aryan', 4.8, 'own', 1, 2),
(4, 'Rishikesh', 6.1, 'rental', 1, 4),
(5, 'Yash', 2.5, 'own', 3, 2);


INSERT INTO HouseDetails (house_id, student_id, own_or_rental, house_distance) VALUES
(1, 1, 'own', 5.5),
(2, 2, 'rental', 3.2),
(3, 3, 'own', 4.8),
(4, 4, 'rental', 6.1),
(5, 5, 'own', 2.5);
```

1)

```
SELECT Name, House_Distance, House_Type
FROM Students;
```

2)

```
SELECT
    S.Name,
    F.Working_Status AS Father_Working_Status,
    F.Organization_Type AS Father_Organization_Type,
    M.Working_Status AS Mother_Working_Status,
    M.Organization_Type AS Mother_Organization_Type
FROM Students S
JOIN Parents F ON S.Father_ID = F.Parent_ID
JOIN Parents M ON S.Mother_ID = M.Parent_ID;
```

3)

```
SELECT
   SUM(CASE WHEN F.Working_Status = 'working' AND M.Working_Status = 'working' THEN 1
ELSE 0 END) AS both_parents_working,
   SUM(CASE WHEN F.Working_Status = 'working' XOR M.Working_Status = 'working' THEN 1
ELSE 0 END) AS one_parent_working
FROM Students S
JOIN Parents F ON S.Father_ID = F.Parent_ID
JOIN Parents M ON S.Mother_ID = M.Parent_ID;
```

4)

```
SELECT
   SUM(CASE WHEN S.House_Type = 'own' THEN 1 ELSE 0 END) AS own_house,
   SUM(CASE WHEN S.House_Type = 'rental' THEN 1 ELSE 0 END) AS rental_house
FROM Students S;
```

6)

```
SELECT COUNT(*) AS same_org_type_count
FROM Students S
JOIN Parents F ON S.Father_ID = F.Parent_ID
JOIN Parents M ON S.Mother_ID = M.Parent_ID
WHERE F.Organization_Type = M.Organization_Type;
```

7)

```
SELECT
   SUM(CASE WHEN House_Distance < 1 THEN 1 ELSE 0 END) AS near_school,
   SUM(CASE WHEN House_Distance BETWEEN 1 AND 5 THEN 1 ELSE 0 END) AS
moderate_distance,
   SUM(CASE WHEN House_Distance > 5 THEN 1 ELSE 0 END) AS far_from_school
FROM Students;
```

8)

```
SELECT
   F.Name AS Father_Name,
   M.Name AS Mother_Name
FROM Students S
JOIN Parents F ON S.Father_ID = F.Parent_ID
JOIN Parents M ON S.Mother_ID = M.Parent_ID
GROUP BY Father_Name, Mother_Name
HAVING COUNT(*) > 1;
```

—---

Q20) Consider the following DB
President identifier,last
name,first_name,political_party,state form)
Admin(start _date,pres
identifier, end date, VP last name, VP _first name)
State(state_ name,date _admitted,area,population,capital_ _city)
Implement the following queries
1. Which presidents were from state M.S. and also numbers of Republication party?
2. Which states were admitted when president A.P.J.K. was in office?
3. Which VP did not later become president?
4. create a view which will display the information about a president and his state information
5. Create a stored procedure which will update the details of the population when there is population count held every year.

```
CREATE TABLE President (
   identifier INT PRIMARY KEY,
   last_name VARCHAR(50),
   first_name VARCHAR(50),
   political_party VARCHAR(50),
   state_form VARCHAR(50)
);

CREATE TABLE Admin (
   start_date DATE,
   pres_identifier INT,
   end_date DATE,
   VP_last_name VARCHAR(50),
   VP_first_name VARCHAR(50),
   FOREIGN KEY (pres_identifier) REFERENCES President(identifier)
);

CREATE TABLE State (
   state_name VARCHAR(50) PRIMARY KEY,
   date_admitted DATE,
   area DECIMAL(10,2),
   population INT,
   capital_city VARCHAR(50)
);


INSERT INTO President (identifier, last_name, first_name, political_party, state_form) VALUES
```

```
(1, 'Washington', 'George', 'Independent', 'Virginia'),
(2, 'Adams', 'John', 'Federalist', 'Massachusetts'),
(3, 'Jefferson', 'Thomas', 'Democratic-Republican', 'Virginia'),
(4, 'Madison', 'James', 'Democratic-Republican', 'Virginia'),
(5, 'Monroe', 'James', 'Democratic-Republican', 'Virginia');


INSERT INTO Admin (start_date, pres_identifier, end_date, VP_last_name, VP_first_name)
VALUES
('1789-04-30', 1, '1797-03-04', NULL, NULL),
('1797-03-04', 2, '1801-03-04', NULL, 'Thomas'),
('1801-03-04', 3, '1809-03-04', 'Burr', 'Aaron'),
('1809-03-04', 4, '1817-03-04', NULL, NULL),
('1817-03-04', 5, '1825-03-04', NULL, NULL);


INSERT INTO State (state_name, date_admitted, area, population, capital_city) VALUES
('Virginia', '1788-06-25', 110.787, 8525660, 'Richmond'),
('Massachusetts', '1788-02-06', 10554, 6892503, 'Boston'),
('New York', '1788-07-26', 54556, 19491339, 'Albany'),
('California', '1850-09-09', 163696, 39538223, 'Sacramento'),
('Texas', '1845-12-29', 268596, 29145505, 'Austin');
```

1)

```
SELECT first_name, last_name
FROM President
WHERE state_form = 'M.S.' AND political_party = 'Republican';

SELECT COUNT(*)
FROM President
WHERE political_party = 'Republican';
```

2)

```
SELECT state_name
FROM State
JOIN Admin ON State.date_admitted BETWEEN Admin.start_date AND Admin.end_date
JOIN President ON Admin.pres_identifier = President.identifier
WHERE President.last_name = 'A.P.J.K.';
```

3)

```
SELECT DISTINCT VP_first_name, VP_last_name
```

```
FROM Admin
WHERE (VP_first_name, VP_last_name) NOT IN (
   SELECT first_name, last_name
   FROM President
);
```

4)

```
CREATE VIEW President_State_Info AS
SELECT President.identifier, President.last_name, President.first_name,
President.political_party, President.state_form,
     State.state_name, State.date_admitted, State.area, State.population, State.capital_city
FROM President
JOIN State ON President.state_form = State.state_name;

SELECT * FROM President_State_Info;
```

5)

```
DELIMITER //

CREATE PROCEDURE UpdatePopulation(IN state_name VARCHAR(50), IN new_population
INT)
BEGIN
   UPDATE State
   SET population = new_population
   WHERE state_name = state_name;
END //

DELIMITER ;

CALL UpdatePopulation('Virginia', 1000000);

SELECT * FROM State WHERE state_name = 'Virginia';
```

—---

Q11) A insurance company wants to keep following information
1. Name, address, phone number and email of person.
2. Type of insurance medical, accidental, death.
3. Insurance payment options one time, monthly, quarterly.
4. List of persons who claimed against their insurance.
5. Amount collected in given period

6. Name of person, amount paid as insurance, amount claimed against insurance.
7. If a person has not claimed during the year then give 10% discount on insurance amount during next year.
8. Display count of persons who paid insurance during specified period but not claimed and count of persons who claimed.
9. Display count of persons who have newly taken insurance during given period.
10. Display count of persons taken different type of insurance(medical, accidental, death

```
CREATE TABLE Persons (
   person_id INT PRIMARY KEY,
   name VARCHAR(100),
   address VARCHAR(255),
   phone_number VARCHAR(15),
   email VARCHAR(255)
);

CREATE TABLE Insurance (
   insurance_id INT PRIMARY KEY,
   person_id INT,
   insurance_type VARCHAR(50),
   payment_option VARCHAR(50),
   amount_paid DECIMAL(10, 2),
   start_date DATE,
   end_date DATE,
   FOREIGN KEY (person_id) REFERENCES Persons(person_id)
);

CREATE TABLE Claims (
   claim_id INT PRIMARY KEY,
   insurance_id INT,
   claim_amount DECIMAL(10, 2),
   claim_date DATE,
   FOREIGN KEY (insurance_id) REFERENCES Insurance(insurance_id)
);

INSERT INTO Persons (person_id, name, address, phone_number, email)
   VALUES
   (1, 'Shriyog More', 'Hadapsar, Pune', '123-456-7890', 'shriyog@gmail.com'),
   (2, 'Naresh Suthar', 'Warje, Pune', '456-789-0123', 'naresh@email.com'),
   (3, 'Aryan Pawar', 'NIBM, Pune', '789-012-3456', 'aryan@email.com'),
   (4, 'Rishikesh Kadam', 'KP, Pune', '987-654-3210', 'kadam@gmail.com'),
   (5, 'Atharva Deshmukh', 'Katraj, Pune', '654-321-0987', 'ath@rmail.com');

INSERT INTO Insurance (insurance_id, person_id, insurance_type, payment_option,
amount_paid, start_date, end_date)
VALUES
   (1, 1, 'Medical', 'Monthly', 50.00, '2024-01-01', '2024-12-31'),
   (2, 2, 'Accidental', 'One-time', 500.00, '2024-01-15', '2024-12-15'),
   (3, 3, 'Death', 'Quarterly', 200.00, '2024-02-01', '2024-11-01'),
   (4, 4, 'Medical', 'Monthly', 60.00, '2024-03-01', '2024-12-01'),
   (5, 5, 'Accidental', 'One-time', 400.00, '2024-04-01', '2024-12-01');

INSERT INTO Claims (claim_id, insurance_id, claim_amount, claim_date)
VALUES
   (1, 1, 100.00, '2024-06-15'),
   (2, 2, 1000.00, '2024-07-20'),
   (3, 3, 5000.00, '2024-08-10'),
   (4, 4, 200.00, '2024-09-05'),
   (5, 5, 800.00, '2024-10-12');
```

4)

```
SELECT p.name
FROM Persons p
JOIN Insurance i ON p.person_id = i.person_id
JOIN Claims c ON i.insurance_id = c.insurance_id;
```

5)

```
SELECT SUM(amount_paid) AS total_amount_collected
FROM Insurance
WHERE start_date BETWEEN '2024-01-01' AND '2024-12-31';
```

6)

```
SELECT p.name, i.amount_paid, c.claim_amount
FROM Persons p
JOIN Insurance i ON p.person_id = i.person_id
LEFT JOIN Claims c ON i.insurance_id = c.insurance_id;
```

7)

```
SELECT p.person_id
FROM Persons p
LEFT JOIN Insurance i ON p.person_id = i.person_id
LEFT JOIN Claims c ON i.insurance_id = c.insurance_id
WHERE YEAR(c.claim_date) = YEAR(NOW())
GROUP BY p.person_id
HAVING COUNT(c.claim_id) = 0;

UPDATE Insurance
SET amount_paid = amount_paid * 0.9 -- Applying 10% discount
WHERE person_id IN (
   SELECT person_id
   FROM (
      SELECT p.person_id
      FROM Persons p
      LEFT JOIN Insurance i ON p.person_id = i.person_id
      LEFT JOIN Claims c ON i.insurance_id = c.insurance_id
      WHERE YEAR(c.claim_date) = YEAR(NOW())
      GROUP BY p.person_id
      HAVING COUNT(c.claim_id) = 0
   ) AS temp
) AND YEAR(start_date) = YEAR(NOW()) + 1;

SELECT * FROM Insurance WHERE YEAR(start_date) = YEAR(NOW()) + 1;
```

8)

```
SELECT
   (SELECT COUNT(*) FROM Insurance WHERE start_date BETWEEN '2024-01-01' AND
'2024-12-31') AS total_paid_insurance,
   (SELECT COUNT(DISTINCT i.person_id) FROM Insurance i LEFT JOIN Claims c ON
i.insurance_id = c.insurance_id WHERE c.claim_id IS NULL AND start_date BETWEEN
'2024-01-01' AND '2024-12-31') AS not_claimed,
   (SELECT COUNT(DISTINCT i.person_id) FROM Insurance i JOIN Claims c ON
i.insurance_id = c.insurance_id WHERE c.claim_id IS NOT NULL AND start_date BETWEEN
'2024-01-01' AND '2024-12-31') AS claimed;
```

9)

```
SELECT COUNT(*) AS new_insurance_count
FROM Insurance
WHERE start_date BETWEEN '2024-01-01' AND '2024-12-31';
```

10)

```
SELECT insurance_type, COUNT(*) AS insurance_count
FROM Insurance
```

```
GROUP BY insurance_type;
```

—--

Q28) Consider the following database
Branch (branch-name, branch-city, assets)
Customer (customer-name, customer-street, customer-city)
Loan (loan-number, branch-name, amount)
Borrower (customer-name, loan-number)
Account (account-number, branch-name, balance)
Depositor (customer-name, account-number)
1. Create a trigger to check the validity of balance (should be between $90,000 and $100,000) before a row is inserted in the Account table.
2. Create a trigger which does not allow any DML operation on the loan table for Perry-ridge branch.
3. Display the customer names who have taken the loan.
4. Display the details of customer with branch name 'Tilak road pune"
5. Display the Number of customers whose branch is 'borivali.Mumbai"

```
-- Create the Branch table
CREATE TABLE Branch (
   branch_name VARCHAR(50) PRIMARY KEY,
   branch_city VARCHAR(50),
   assets DECIMAL(10, 2)
);

-- Insert sample values into the Branch table
INSERT INTO Branch (branch_name, branch_city, assets) VALUES
('Perry-ridge', 'Delhi', 150000.00),
('Tilak road pune', 'Pune', 200000.00),
('Borivali', 'Mumbai', 180000.00);

-- Create the Customer table
CREATE TABLE Customer (
   customer_name VARCHAR(50) PRIMARY KEY,
   customer_street VARCHAR(100),
   customer_city VARCHAR(50)
);

-- Insert sample values into the Customer table
INSERT INTO Customer (customer_name, customer_street, customer_city) VALUES
('Shriyog More', 'Tilak road', 'Pune'),
('Naresh Suthar', 'Tilak road', 'Pune'),
('Aryan Pawar', 'Borivali', 'Mumbai');

-- Create the Loan table
CREATE TABLE Loan (
    loan_number INT AUTO_INCREMENT PRIMARY KEY,
    branch_name VARCHAR(50),
    amount DECIMAL(10, 2)
);

-- Insert sample values into the Loan table
INSERT INTO Loan (branch_name, amount) VALUES
('Tilak road pune', 50000.00),
('Perry-ridge', 75000.00),
('Borivali', 60000.00);

-- Create the Borrower table
CREATE TABLE Borrower (
    customer_name VARCHAR(50),
    loan_number INT,
    PRIMARY KEY (customer_name, loan_number)
);

-- Insert sample values into the Borrower table
INSERT INTO Borrower (customer_name, loan_number) VALUES
('Shriyog More', 1),
('Naresh Suthar', 2),
('Aryan Pawar', 3);

-- Create the Account table
CREATE TABLE Account (
    account_number INT AUTO_INCREMENT PRIMARY KEY,
    branch_name VARCHAR(50),
    balance DECIMAL(10, 2)
);

-- Insert sample values into the Account table
INSERT INTO Account (branch_name, balance) VALUES
('Tilak road pune', 95000.00),
('Perry-ridge', 100000.00),
('Borivali', 90000.00);

-- Create the Depositor table
CREATE TABLE Depositor (
    customer_name VARCHAR(50),
    account_number INT,
    PRIMARY KEY (customer_name, account_number)
);

-- Insert sample values into the Depositor table
INSERT INTO Depositor (customer_name, account_number) VALUES
('Shriyog More', 1),
('Naresh Suthar', 2),
('Aryan Pawar', 3);
```

1)

```
DELIMITER //

CREATE TRIGGER CheckBalanceValidity
BEFORE INSERT ON Account
FOR EACH ROW
BEGIN
    IF NEW.balance < 90000 OR NEW.balance > 100000 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Balance should be between $90,000 and $100,000';
    END IF;
END;
//

DELIMITER ;

-- Attempt to insert a row with a balance less than $90,000
INSERT INTO Account (branch_name, balance) VALUES ('Tilak road pune', 80000.00);

-- Attempt to insert a row with a balance greater than $100,000
INSERT INTO Account (branch_name, balance) VALUES ('Perry-ridge', 110000.00);

INSERT INTO Account (branch_name, balance) VALUES ('Tilak road pune', 96000.00);
```

2)

```
DELIMITER //

CREATE TRIGGER RestrictLoanInsert
BEFORE INSERT ON Loan
FOR EACH ROW
BEGIN
    IF NEW.branch_name = 'Perry-ridge' THEN
```

```
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Insert operation not allowed for Perry-ridge branch';
    END IF;
END;
//

CREATE TRIGGER RestrictLoanUpdate
BEFORE UPDATE ON Loan
FOR EACH ROW
BEGIN
    IF NEW.branch_name = 'Perry-ridge' THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Update operation not allowed for Perry-ridge branch';
    END IF;
END;
//

CREATE TRIGGER RestrictLoanDelete
BEFORE DELETE ON Loan
FOR EACH ROW
BEGIN
    IF OLD.branch_name = 'Perry-ridge' THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Delete operation not allowed for Perry-ridge branch';
    END IF;
END;
//

DELIMITER ;


INSERT INTO Loan (loan_number, branch_name, amount) VALUES ('123456', 'Perry-ridge',
5000);


UPDATE Loan SET amount = 6000 WHERE loan_number = '123456' AND branch_name =
'Perry-ridge';


DELETE FROM Loan WHERE loan_number = '123456' AND branch_name = 'Perry-ridge';
```

3)

```
SELECT DISTINCT customer_name
FROM Borrower;
```

4)

```
SELECT *
FROM Customer
WHERE customer_city = 'Pune' AND customer_street = 'Tilak road';
```

5)

```
SELECT COUNT(*) AS customer_count
FROM Customer
WHERE customer_city = 'Mumbai' AND customer_street = 'Borivali';
```

—---

Q16) A TV, Fridge, washing machine, Microwave selling shop wants to keep following information

1. Name, mobile number, no, of persons in house of person purchasing any item.

2. If any person purchases 2 items then he will be given 5% discount on third item.

3. Number of persons purchasing 2 items.

4. List of combination of items purchased by people.

5. Count of people purchasing 1,2,3 items.

6. Count of people purchased microwave who are having more than 4 no. of Persons in house.

7. Count of people purchased TV having 2 people in a house.

8. No. of different items sold during given period.

9. Sales amount collected per item during given period.

10. Revenue generated during given period.

```
CREATE TABLE Customers (
   customer_id INT PRIMARY KEY AUTO_INCREMENT,
   name VARCHAR(100),
   mobile_number VARCHAR(20),
   household_size INT
);

CREATE TABLE Purchases (
   purchase_id INT PRIMARY KEY AUTO_INCREMENT,
   customer_id INT,
   item_name VARCHAR(100),
   purchase_date DATE,
   price DECIMAL(10, 2), -- Adding price column
   quantity INT, -- Adding quantity column
   FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);

-- Inserting data into Customers table
INSERT INTO Customers (name, mobile_number, household_size) VALUES
('Shriyog More', '1234567890', 4),
('Naresh Suthar', '9876543210', 5),
('Aryan Pawar', '5555555555', 2);

-- Inserting data into Purchases table
INSERT INTO Purchases (customer_id, item_name, purchase_date, price, quantity) VALUES
(1, 'TV', '2024-05-01', 500.00, 1),
(1, 'Microwave', '2024-05-05', 200.00, 1),
(2, 'Fridge', '2024-05-02', 800.00, 1),
(2, 'Washing Machine', '2024-05-03', 600.00, 1),
(3, 'TV', '2024-05-06', 500.00, 1),
(3, 'Fridge', '2024-05-07', 800.00, 1),
(3, 'Microwave', '2024-05-08', 200.00, 1),
(3, 'Washing Machine', '2024-05-09', 600.00, 1);
```

1)

```
SELECT c.name, c.mobile_number, c.household_size
FROM Customers c
JOIN Purchases p ON c.customer_id = p.customer_id;
```

2)

```
SELECT name, mobile_number, household_size,
      CASE WHEN total_items >= 2 THEN 'Yes' ELSE 'No' END AS eligible_for_discount
FROM (
   SELECT c.name, c.mobile_number, c.household_size, COUNT(p.item_name) AS total_items
   FROM Customers c
   JOIN Purchases p ON c.customer_id = p.customer_id
   GROUP BY c.customer_id
) AS customer_purchase_count;
```

3)

```
SELECT COUNT(*) AS count_of_customers_purchasing_two_items
FROM (
   SELECT customer_id
   FROM Purchases
   GROUP BY customer_id
   HAVING COUNT(item_name) = 2
) AS customers_with_two_items;
```

4)

```
SELECT GROUP_CONCAT(item_name ORDER BY item_name) AS item_combination
FROM Purchases
GROUP BY customer_id;
```

5)

```
SELECT item_count, COUNT(*) AS customer_count
FROM (
   SELECT customer_id, COUNT(item_name) AS item_count
   FROM Purchases
   GROUP BY customer_id
) AS item_counts
GROUP BY item_count;
```

6)

```
SELECT COUNT(*) AS count_of_customers
FROM Customers c
JOIN Purchases p ON c.customer_id = p.customer_id
WHERE p.item_name = 'Microwave' AND c.household_size > 4;
```

7)

```
SELECT COUNT(*) AS count_of_customers
FROM Customers c
```

```
JOIN Purchases p ON c.customer_id = p.customer_id
WHERE p.item_name = 'TV' AND c.household_size = 2;
```

8)

```
SELECT COUNT(DISTINCT item_name) AS count_of_items_sold
FROM Purchases;
```

9)

```
SELECT item_name, SUM(price * quantity) AS sales_amount
FROM Purchases
WHERE purchase_date BETWEEN '2024-05-01' AND '2024-05-31'
GROUP BY item_name;
```

10)

```
SELECT SUM(price * quantity) AS total_revenue
FROM Purchases
WHERE purchase_date BETWEEN '2024-05-01' AND '2024-05-31';
```

—--

Q23) Consider the following DB
Emp(ename,street,city)
Works(enmac,cname,sal)
Company(cname,city)
Manager (ename,mname)
Implement the following SQL queries
Find all companies located in every city in which company 'FBC" is located.
2. List all managers in of FBC' a 10% raise unless the salary become greater than Rs 10 lakhs in such cases give only 2.5% raise.
3. Find those companies whose employees earn a higher salary, on average, than the average salary at'FBC'
4. Find all the employees who work in companies located in *Pune*
5. Create a view which will display the all the information of manager, the company they work in.
6. Create a trigger which will update the salary of all employees other than manager with 8 % of their salary and for manager 9% of their salary.

```
CREATE TABLE Emp (
   ename VARCHAR(100),
   street VARCHAR(100),
    city VARCHAR(100)
);

CREATE TABLE Works (
    ename VARCHAR(100),
    cname VARCHAR(100),
    sal DECIMAL(10, 2)
);

CREATE TABLE Company (
    cname VARCHAR(100),
    city VARCHAR(100)
);

CREATE TABLE Manager (
    ename VARCHAR(100),
    mname VARCHAR(100)
);

-- Insert values into the Emp table
INSERT INTO Emp (ename, street, city)
VALUES
    ('Shriyog More', '123 Main St', 'New York'),
    ('Naresh Suthar', '456 Elm St', 'New York'),
    ('Aryan Pawar', '789 Oak St', 'San Francisco'),
    ('Rishikesh Kadam', '101 Pine St', 'San Francisco'),
    ('Atharva Deshmukh', '789 Maple St', 'Pune');

-- Insert values into the Works table
INSERT INTO Works (ename, cname, sal)
VALUES
    ('Shriyog More', 'Sigma Corp', 120000),
    ('Naresh Suthar', 'Apple Inc', 90000),
    ('Aryan Pawar', 'FBC', 95000),
    ('Rishikesh Kadma', 'Apple Inc', 85000),
    ('Chris Bumsted', 'Sigma Corp', 110000),
    ('Yash Bhosale', 'Tata Ltd', 75000);

-- Insert values into the Company table
INSERT INTO Company (cname, city)
VALUES
    ('Sigma Corp', 'Mumbai'),
    ('Apple Inc', ‘New’ York),
    ('FBC', 'New York'),
    ('Tata Ltd', 'Pune');

-- Insert values into the Manager table
INSERT INTO Manager (ename, mname)
VALUES
    ('Aryan Pawar', 'Sam Sulek'),
    ('Rishikesh Kadam', 'Ronnie Coleman');
```

1)

```
SELECT cname
FROM Company
WHERE city IN (
    SELECT DISTINCT city
    FROM Company
    WHERE cname = 'FBC'
) AND cname != 'FBC';
```

2)

```
UPDATE Works
SET sal = CASE
    WHEN sal * 1.1 > 1000000 THEN sal * 1.025
    ELSE sal * 1.1
END
WHERE cname = 'FBC' AND ename IN (
    SELECT ename
    FROM Manager
    WHERE cname = 'FBC'
);

Select * from Works;
```

3)

```
SELECT DISTINCT cname
FROM Works
GROUP BY cname
HAVING AVG(sal) > (
    SELECT AVG(sal)
    FROM Works
    WHERE cname = 'FBC'
);
```

4)

```
SELECT ename
FROM Works
WHERE cname IN (
    SELECT cname
    FROM Company
    WHERE city = 'Pune'
);
```

5)

```
CREATE VIEW Manager_Company_Info AS
SELECT Manager.*, Works.cname
FROM Manager
JOIN Works ON Manager.ename = Works.ename;

select * from Manager_Company_Info;
```

6)

```
DELIMITER //

CREATE TRIGGER UpdateSalary
BEFORE UPDATE ON Works
FOR EACH ROW
BEGIN
    IF NEW.ename NOT IN (SELECT ename FROM Manager) THEN
        SET NEW.sal = OLD.sal * 1.08;
    ELSE
        SET NEW.sal = OLD.sal * 1.09;
    END IF;
END;
//

DELIMITER ;
```

```
UPDATE Works
SET sal = sal + 1000
WHERE ename = 'Naresh Suthar';

Select * from Works;
```

—---

Q4) Implement the following mysql Queries
1. Create a collection named books.
2. Insert 5 records with field TITLE, DESCRIPTION, BY ,URL TAGS AND LIKES
3. Insert I more document in collection with additional field of user name and comments.
4. Display all the documents whose title is 'MySQL'.
5. Display all the documents written by 'john' or whose title is 'mongodb'.
Display all the documents whose title is 'mongodb' and written by 'john'.
7. Display all the documents whose like is greater than 10.
8. Display all the documents whose like is greater than 100 and whose title is either 'mongodb' or written by 'john'.
9. Update the title of 'MySQL' document to 'MySQL overview'
10. Delete the document titled 'nosql overview'.
11. Display exactly two documents written by 'john'.
12. Display the second document published by 'john'.
13. Display all the books in the sorted fashion.

```
CREATE TABLE books (
   id INT AUTO_INCREMENT PRIMARY KEY,
   title VARCHAR(255),
   description TEXT,
   by_author VARCHAR(255),
   url VARCHAR(255),
   tags VARCHAR(255),
   likes INT
);

INSERT INTO books (title, description, by_author, url, tags, likes) VALUES
('MySQL', 'Description of MySQL book', 'John', 'https://example.com/mysql', 'Database', 105),
('MongoDB', 'Description of MongoDB book', 'John', 'https://example.com/mongodb', 'NoSQL',
15),
('PostgreSQL', 'Description of PostgreSQL book', 'Alice', 'https://example.com/postgresql',
'Database', 12),
('SQL Server', 'Description of SQL Server book', 'Bob', 'https://example.com/sql-server',
'Database', 8),
('Oracle', 'Description of Oracle book', 'John', 'https://example.com/oracle', 'Database', 25);
```

3)

```
INSERT INTO books (title, description, by_author, url, tags, likes, user_name, comments)
VALUES
('NoSQL Overview', 'Description of NoSQL Overview book', 'Alice', 'https://example.com/nosql',
'NoSQL', 30, 'user1', 'Great book!');
```

4)

```
SELECT * FROM books WHERE title = 'MySQL';
```

5)

```
SELECT * FROM books WHERE by_author = 'John' OR title = 'MongoDB';
```

6)

```
SELECT * FROM books WHERE by_author = 'John' AND title = 'MongoDB';
```

7)

```
SELECT * FROM books WHERE likes > 10;
```

8)

```
SELECT * FROM books WHERE likes > 100 AND (title = 'MongoDB' OR by_author = 'John');
```

9)

```
UPDATE books SET title = 'MySQL overview' WHERE title = 'MySQL';
```

10)

```
DELETE FROM books WHERE title = 'Oracle';
```

11)

```
SELECT * FROM books WHERE by_author = 'John' LIMIT 2;
```

12)

```
SELECT * FROM books WHERE by_author = 'John' LIMIT 1,1;
```

13)

```
SELECT * FROM books ORDER BY title;
```

—---

Q17)A mobile manufacturing company wants keep following record

1. Name of distributor, mobile number, email address, state, city

2.Number of mobiles ordered by distributor

3.If a distributor has ordered 1000 mobiles and deposited amount required for those mobiles then he must be given 100 mobiles on credit.

4. If the order crosses Rs. 100000/- then the distributor is given discount of Rs. 7300/-

5. Count of distributors in each state, city.

6. Revenue generated from each distributor.

7. Revenue generated from each state, city

8. Types of models ordered by each distributor.

9. Count of models ordered by each distributor during given period.

10. List Names of models and names of distributors ordered that model.

```
CREATE TABLE Distributors (
    distributor_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255),
    mobile_number VARCHAR(15),
    email VARCHAR(255),
    state VARCHAR(255),
    city VARCHAR(255)
);

CREATE TABLE Orders (
    order_id INT AUTO_INCREMENT PRIMARY KEY,
    distributor_id INT,
    num_mobiles_ordered INT,
    amount_deposited DECIMAL(10, 2),
    order_date DATE,
    FOREIGN KEY (distributor_id) REFERENCES Distributors(distributor_id)
);

CREATE TABLE Models (
    model_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255),
    price DECIMAL(10, 2)
);

CREATE TABLE OrderDetails (
    order_id INT,
    model_id INT,
    quantity INT,
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (model_id) REFERENCES Models(model_id)
);


-- Inserting data into the Distributors table
INSERT INTO Distributors (name, mobile_number, email, state, city) VALUES
('iVenus', '1234567890', 'ivenus@email.com', 'Maharashtra', 'Mumbai'),
('Unicorn', '2345678901', 'unicorn@gmail.com', 'Gujrat', 'Surat'),
('Apple Store', '3456789012', 'appst@rmail.com', 'Maharashtra', 'Pune');

-- Inserting data into the Models table
INSERT INTO Models (name, price) VALUES
('iphone 13', 10000),
('iphone 14', 15000),
('iphone 15', 20000);

-- Inserting data into the Orders table
INSERT INTO Orders (distributor_id, num_mobiles_ordered, amount_deposited, order_date)
VALUES
(1, 800, 90000, '2024-05-01'),
(2, 1200, 120000, '2024-05-05'),
(3, 1500, 110000, '2024-05-10');

-- Inserting data into the OrderDetails table (associating models with orders)
-- For simplicity, let's assume each distributor ordered all three models
INSERT INTO OrderDetails (order_id, model_id, quantity) VALUES
(1, 1, 200),
(1, 2, 300),
(1, 3, 300),
(2, 1, 400),
(2, 2, 400),
(2, 3, 400),
(3, 1, 500),
(3, 2, 500),
(3, 3, 500);
```

2)

```
SELECT distributor_id, SUM(num_mobiles_ordered) AS total_mobiles_ordered
FROM Orders
GROUP BY distributor_id;
```

3)

```
UPDATE Orders
SET num_mobiles_ordered = num_mobiles_ordered + 100
WHERE num_mobiles_ordered >= 1000 AND amount_deposited > 0;

select * from Orders;
```

4)

```
UPDATE Orders
SET amount_deposited = amount_deposited - 7300
WHERE amount_deposited > 100000;

select * from Orders;
```

5)

```
SELECT state, city, COUNT(*) AS distributor_count
FROM Distributors
```

```
GROUP BY state, city;
```

6)

```
SELECT distributor_id, SUM(amount_deposited) AS revenue
FROM Orders
GROUP BY distributor_id;
```

7)

```
SELECT d.state, d.city, SUM(o.amount_deposited) AS revenue
FROM Distributors d
JOIN Orders o ON d.distributor_id = o.distributor_id
GROUP BY d.state, d.city;
```

8)

```
SELECT o.distributor_id, m.name AS model_name, COUNT(*) AS quantity
FROM OrderDetails od
JOIN Orders o ON od.order_id = o.order_id
JOIN Models m ON od.model_id = m.model_id
GROUP BY o.distributor_id, m.name;
```

9)

```
SELECT o.distributor_id, COUNT(*) AS num_models_ordered
FROM OrderDetails od
JOIN Orders o ON od.order_id = o.order_id
WHERE o.order_date BETWEEN '2024-05-01' AND '2024-05-05'
GROUP BY o.distributor_id;
```

10)

```
SELECT m.name AS model_name, d.name AS distributor_name
FROM OrderDetails od
JOIN Orders o ON od.order_id = o.order_id
JOIN Distributors d ON o.distributor_id = d.distributor_id
JOIN Models m ON od.model_id = m.model_id;
```

—---

Coded by: Shriyog

“Choices have consequences”