S2 - SQL Queries (in MySQL)

Problem Statement: Consider following Relation Account (Acc_no, branch_name,balance) Branch(branch_name,branch_city,assets) Customer(cust_name,cust_street,cust_city) Depositor(cust_name,acc_no) Loan(loan_no,branch_name,amount) Borrower(cust_name,loan_no) Create above tables with appropriate constraints like primary key, foreign key, not null etc.

1. Find all customers who have both account and loan at bank.
2. Find all customers who have an account or loan or both at bank.
3. Find all customers who have account but no loan at the bank.
4. Find average account balance at 'Wadia College' branch.
5. Find no. of depositors at each branch

Creating the database

```
CREATE DATABASE Bank2;
USE Bank2;
```

Creating tables:

```
CREATE TABLE Account (
  acc_no INT,
  branch_name VARCHAR(255),
  balance INT,
  PRIMARY KEY (acc_no)
);

CREATE TABLE Branch (
  branch_name VARCHAR(255),
  branch_city VARCHAR(255),
  assets INT,
  PRIMARY KEY (branch_name)
);

CREATE TABLE Customer (
  cust_name VARCHAR(255),
  cust_street VARCHAR(255),
  cust_city VARCHAR(255),
  PRIMARY KEY (cust_name)
);

CREATE TABLE Depositor (
  cust_name VARCHAR(255),
  acc_no INT
);

CREATE TABLE Loan (
  loan_no INT,
  branch_name VARCHAR(255),
  amount INT,
  PRIMARY KEY (loan_no)
);

CREATE TABLE Borrower (
  cust_name VARCHAR(255),
  loan_no INT
);
```

Declaring foreign keys

```
ALTER TABLE Account ADD FOREIGN KEY (branch_name) REFERENCES Branch (branch_name);
ALTER TABLE Depositor ADD FOREIGN KEY (cust_name) REFERENCES Customer (cust_name);
ALTER TABLE Depositor ADD FOREIGN KEY (acc_no) REFERENCES Account (acc_no);
ALTER TABLE Loan ADD FOREIGN KEY (branch_name) REFERENCES Branch (branch_name);
ALTER TABLE Borrower ADD FOREIGN KEY (cust_name) REFERENCES Customer (cust_name);
ALTER TABLE Borrower ADD FOREIGN KEY (loan_no) REFERENCES Loan (loan_no);
```

Inserting data

```
INSERT INTO Branch VALUES
('Wadia College', 'Pune', 50000),
('PES', 'Pune', 65000),
('Lohegaon', 'Pune', 350000),
('Viman Nagar', 'Pune', 850000);

INSERT INTO Customer VALUES
('Kalas', 'Street 12', 'Pune'),
('Himanshu', 'Street 15', 'Pune'),
('Mehul', 'Street 29', 'Pune'),
('Macho', 'Street 59', 'Mumbai'),
('Gundeti', 'Street 40', 'Mumbai'),
('Salvi', 'Street 8', 'Pune');

INSERT INTO Account VALUES
(101, 'Lohegaon', 5500),
(102, 'PES', 4324),
(103, 'PES', 5467),
(104, 'Viman Nagar', 5433),
(105, 'Wadia College', 6462);

INSERT INTO Depositor VALUES
('Kalas', 101),
('Macho', 104),
('Gundeti', 105),
('Salvi', 105);

INSERT INTO Loan VALUES
(201, 'Wadia College', 18000),
(202, 'PES', 8500),
(203, 'PES', 15000),
(204, 'Wadia College', 5322);

INSERT INTO Borrower VALUES
('Macho', 201),
('Mehul', 202),
('Himanshu', 203),
('Salvi', 204);
```

Queries

1. Find all customers who have both account and loan at bank.

```
SELECT cust_name FROM Depositor INTERSECT SELECT cust_name FROM Borrower;
```

2. Find all customers who have an account or loan or both at bank.

```
SELECT cust_name FROM Depositor UNION SELECT cust_name FROM Borrower;
```

3. Find all customers who have account but no loan at the bank.

```
SELECT cust_name FROM Depositor WHERE cust_name NOT IN (SELECT cust_name FROM Borrower);
```

4. Find average account balance at ‘Wadia College’ branch.

```
SELECT AVG(balance) FROM Account WHERE branch_name = 'Wadia College';
```

5. Find no. of depositors at each branch

```
SELECT Account.branch_name, COUNT(*) AS total FROM Account INNER JOIN Depositor ON Account.acc_no = Depositor.acc_no GROUP BY branch_
```

S3 - SQL Queries (in MySQL)

Problem Statement: Consider following Relation Account (Acc_no, branch_name,balance) Branch(branch_name,branch_city,assets) Customer(cust_name,cust_street,cust_city) Depositor(cust_name,acc_no) Loan(loan_no,branch_name,amount) Borrower(cust_name,loan_no) Create above tables with appropriate constraints like primary key, foreign key, not null etc.

1. Find the branches where average account balance > 15000.
2. Find number of tuples in customer relation.
3. Calculate total loan amount given by bank.
4. Delete all loans with loan amount between 1300 and 1500.
5. Find the average account balance at each branch
6. Find name of Customer and city where customer name starts with Letter P.

Creating the database

```
CREATE DATABASE Bank3;
USE Bank3;
```

Creating tables:

```sql
CREATE TABLE Account (
  acc_no INT,
  branch_name VARCHAR(255),
  balance INT,
  PRIMARY KEY (acc_no)
);

CREATE TABLE Branch (
  branch_name VARCHAR(255),
  branch_city VARCHAR(255),
  assets INT,
  PRIMARY KEY (branch_name)
);

CREATE TABLE Customer (
  cust_name VARCHAR(255),
  cust_street VARCHAR(255),
  cust_city VARCHAR(255),
  PRIMARY KEY (cust_name)
);

CREATE TABLE Depositor (
  cust_name VARCHAR(255),
  acc_no INT
);

CREATE TABLE Loan (
  loan_no INT,
  branch_name VARCHAR(255),
  amount INT,
  PRIMARY KEY (loan_no)
);

CREATE TABLE Borrower (
  cust_name VARCHAR(255),
  loan_no INT
);
```

Declaring foreign keys

```sql
ALTER TABLE Account ADD FOREIGN KEY (branch_name) REFERENCES Branch (branch_name);
ALTER TABLE Depositor ADD FOREIGN KEY (cust_name) REFERENCES Customer (cust_name);
ALTER TABLE Depositor ADD FOREIGN KEY (acc_no) REFERENCES Account (acc_no);
ALTER TABLE Loan ADD FOREIGN KEY (branch_name) REFERENCES Branch (branch_name);
ALTER TABLE Borrower ADD FOREIGN KEY (cust_name) REFERENCES Customer (cust_name);
ALTER TABLE Borrower ADD FOREIGN KEY (loan_no) REFERENCES Loan (loan_no);
```

Inserting data

```
INSERT INTO Branch VALUES
('Wadia College', 'Pune', 50000),
('PES', 'Pune', 65000),
('Lohegaon', 'Pune', 350000),
('Viman Nagar', 'Pune', 850000);

INSERT INTO Customer VALUES
('Kalas', 'Street 12', 'Pune'),
('Himanshu', 'Street 15', 'Pune'),
('Mehul', 'Street 29', 'Pune'),
('Macho', 'Street 59', 'Mumbai'),
('Gundeti', 'Street 40', 'Mumbai'),
('Salvi', 'Street 8', 'Pune'),
('Pintu', 'Street 55', 'Ahemadnagar'),
('Piyush', 'Street 21', 'Assam');

INSERT INTO Account VALUES
(101, 'Lohegaon', 67000),
(102, 'PES', 4324),
(103, 'PES', 54670),
(104, 'Viman Nagar', 5433),
(105, 'Wadia College', 6462);

INSERT INTO Depositor VALUES
('Kalas', 101),
('Macho', 104),
('Gundeti', 105),
('Salvi', 105);

INSERT INTO Loan VALUES
(201, 'Wadia College', 1800),
(202, 'PES', 8500),
(203, 'PES', 15000),
(204, 'Wadia College', 5322),
(205, 'Viman Nagar', 1300),
(206, 'Lohegaon', 1450);

INSERT INTO Borrower VALUES
('Macho', 201),
('Mehul', 202),
('Himanshu', 203),
('Salvi', 204);
```

Queries

1. Find the branches where average account balance > 15000.

```
SELECT branch_name FROM Account GROUP BY branch_name HAVING AVG(balance) > 15000 ;
```

2. Find number of tuples in customer relation.

```
SELECT COUNT(*) FROM Customer;
```

3. Calculate total loan amount given by bank.

```
SELECT SUM(amount) FROM Loan;
```

4. Delete all loans with loan amount between 1300 and 1500.

```
DELETE FROM Loan WHERE amount BETWEEN 1300 AND 1500;
```

5. Find the average account balance at each branch

```
SELECT branch_name, AVG(balance) FROM Account GROUP BY branch_name;
```

6. Find name of Customer and city where customer name starts with letter P.

```
SELECT cust_name, cust_city FROM Customer WHERE cust_name LIKE "P%";
```

S4 - SQL Queries (in MySQL)

Problem Statement: SQL Queries: Create following tables with suitable constraints (primary key, foreign key, not null etc). Insert record and solve the following queries: Create table Cust_Master(Cust_no, Cust_name, Cust_addr) Create table Order(Order_no, Cust_no, Order_date, Qty_Ordered) Create Product (Product_no, Product_name, Order_no)

1. List names of customers having 'A' as second letter in their name.
2. Display order from Customer no C1002, C1005, C1007 and C1008
3. List Clients who stay in either 'Banglore or 'Manglore'
4. Display name of customers& the product_name they have purchase
5. Create view View1 consisting of Cust_name, Product_name.
6. Disply product_name and quantity purchase by each customer
7. Perform different joint operation.

Creating the database

```
CREATE DATABASE Store1;
USE Store1;
```

Creating tables:

```
CREATE TABLE Cust_Master (
  Cust_no VARCHAR(255) NOT NULL,
  Cust_name VARCHAR(255),
  Cust_addr VARCHAR(255),
  PRIMARY KEY (cust_no)
);

CREATE TABLE Orders (
  -- Cannot have 'Order' as table name since it is a keyword reserved for 'ORDER BY' cause
  Order_no INT,
  Cust_no VARCHAR(255),
  Order_date DATE,
  Qty_Ordered INT,
  PRIMARY KEY (Order_no)
);

CREATE TABLE Product (
  Product_no INT,
  Product_name VARCHAR(255),
  Order_no INT
);
```

Declaring foreign keys

```
ALTER TABLE Orders ADD FOREIGN KEY (Cust_no) REFERENCES Cust_Master (Cust_no);
ALTER TABLE Product ADD FOREIGN KEY (Order_no) REFERENCES Orders (Order_no);
```

Inserting data

```
INSERT INTO Cust_Master VALUES
('C1001', 'Kalas', 'Pune'),
('C1002', 'Macho', 'Banglore'),
('C1003', 'Gundeti', 'Chennai'),
('C1005', 'Salvi',  'Manglore'),
('C1006', 'Kshitij', 'Assam'),
('C1007', 'Himashu', 'Banglore'),
('C1008', 'Mehul', 'Mumbai');

INSERT INTO Orders VALUES
(1, 'C1001', '2024-11-09', 10),
(2, 'C1003', '2024-11-01', 5),
(3, 'C1005', '2024-11-05', 45),
(4, 'C1002', '2024-10-29', 3),
(5, 'C1007', '2024-10-15', 2),
(6, 'C1008', '2024-11-10', 7),
(7, 'C1006', '2024-11-09', 1);

INSERT INTO Product VALUES
('101', 'Political Stamps', 1),
('204', 'Fashion Accessory', 2),
('438', 'Complan', 3),
('327', 'ID Card Strap', 4),
('243', 'Face and Hair Wash', 5),
('373', 'Fat Reducer 6000', 6),
('327', 'Personality', 7);
```

Queries

1. List names of customers having 'A' as second letter in their name.

```
SELECT Cust_name FROM Cust_Master WHERE Cust_name LIKE "_a%";
```

2. Display order from Customer no C1002, C1005, C1007 and C1008

```
SELECT * FROM Orders WHERE Cust_no IN ('C1002', 'C1005', 'C1007', 'C1008');
```

3. List Clients who stay in either 'Banglore or 'Manglore'

```
SELECT Cust_name FROM Cust_Master WHERE Cust_addr = 'Banglore' OR Cust_addr = 'Manglore';
```

4. Display name of customers & the product_name they have purchase

```
SELECT Cust_Master.Cust_name, Product.Product_name FROM Product INNER JOIN Orders ON Product.Order_no = Orders.Order_no INNER JOIN Cu
```

5. Create view View1 consisting of Cust_name, Product_name.

```
CREATE VIEW View1 AS SELECT Cust_Master.Cust_name, Product.Product_name FROM Product INNER JOIN Orders ON Product.Order_no = Orders.O

SELECT * FROM View1;
```

6. Disply product_name and quantity purchase by each customer

```
SELECT Cust_Master.Cust_name, Product.Product_name, Orders.Qty_Ordered FROM Product INNER JOIN Orders ON Product.Order_no = Orders.Or
```

7. Perform different joint operation.

- INNER JOIN:

```
SELECT Cust_Master.Cust_name, Product.Product_name FROM Product INNER JOIN Orders ON Product.Order_no = Orders.Order_no INNER JOIN Cu
```

- OUTER LEFT JOIN:

```
SELECT Cust_Master.Cust_name, Orders.Order_no, Orders.Order_date FROM Cust_Master LEFT JOIN Orders ON Cust_Master.Cust_no = Orders.Cu
```

- OUTER RIGHT JOIN:

```
SELECT Orders.Order_no, Orders.Order_date, Cust_Master.Cust_name FROM Orders RIGHT JOIN Cust_Master ON Orders.Cust_no = Cust_Master.C
```

S5 - SQL Queries (in MySQL)

Problem Statement: Consider following Relation Employee(emp_id,employee_name,street,city) Works(employee_name,company_name,salary) Company(company_name,city) Manages(employee_name,manager_name) Create above tables with appropriate constraints like primary key, foreign key, not null etc.

1. Find the names of all employees who work for ‘TCS’.
2. Find the names and company names of all employees sorted in ascending order of company name and descending order of employee names of that company.
3. Change the city of employee working with InfoSys to ‘Bangalore’
4. Find the names, street address, and cities of residence for all employees who work for 'TechM' and earn more than $10,000.
5. Add Column Asset to Company table.

Creating the database

```
CREATE DATABASE Companies1;
USE Companies1;
```

Creating tables:

```
CREATE TABLE Employee (
  emp_id INT UNIQUE NOT NULL, -- can be set to auto increment using AUTO_INCREMENT
  employee_name VARCHAR(255),
  street VARCHAR(255),
  city VARCHAR(255),
  PRIMARY KEY (employee_name)
);

CREATE TABLE Works (
  employee_name VARCHAR(255),
  company_name VARCHAR(255),
  salary INT -- use FLOAT if you are feeling fancy and pay your employees in Paise
);

CREATE TABLE Company (
  company_name VARCHAR(255),
  city VARCHAR(255),
  PRIMARY KEY (company_name)
);

CREATE TABLE Manages (
  employee_name VARCHAR(255),
  manager_name VARCHAR(255)
);
```

Declaring foreign keys

```
ALTER TABLE Works ADD FOREIGN KEY (employee_name) REFERENCES Employee (employee_name);
ALTER TABLE Works ADD FOREIGN KEY (company_name) REFERENCES Company (company_name);
ALTER TABLE Manages ADD FOREIGN KEY (employee_name) REFERENCES Employee (employee_name);
```

Inserting data

```
INSERT INTO Employee VALUES
(1, 'Mehul', 'Street 42', 'Pune'),
(2, 'Himanshu', 'Street 74', 'Mumbai'),
(3, 'Gundeti', 'Street 14', 'Pune'),
(4, 'Salvi', 'Street 38', 'Pune'),
(5, 'Afan', 'Steet 98', 'Pune');

INSERT INTO Company VALUES
('TCS', 'Pune'),
('Infosys', 'Mumbai'),
('TechM', 'Pune'),
('MEPA', 'Pune');

INSERT INTO Works VALUES
('Mehul', 'MEPA', 15000),
('Himanshu', 'TCS', 25000),
('Gundeti', 'TCS', 21500),
('Salvi', 'TechM', 11000),
('Afan', 'Infosys', 13000);


INSERT INTO Manages VALUES
('Mehul', 'Kalas'),
('Himanshu', 'Kshitij'),
('Gundeti', 'Macho'),
('Salvi', 'Kshitij'),
('Afan', 'Kalas');
```

Queries

1. Find the names of all employees who work for ‘TCS’.

```
SELECT employee_name FROM Works WHERE company_name = "TCS";
```

2. Find the names and company names of all employees sorted in ascending order of company name and descending order of employee names of that company.

```
SELECT company_name, employee_name FROM Works ORDER BY company_name ASC, employee_name DESC;
```

3. Change the city of employee working with InfoSys to ‘Bangalore’

```
update Employee set city = "Banglore" where employee_name in (select employee_name from Works where company_name = "Infosys");
select * from Employee;
```

4. Find the names, street address, and cities of residence for all employees who work for 'TechM' and earn more than $10,000.

```
SELECT Employee.employee_name, Employee.street, Employee.city FROM Employee INNER JOIN Works ON Employee.employee_name = Works.emplo
```

5. Add Column Asset to Company table.

```
ALTER TABLE Company ADD assets INT;
DESCRIBE Company;
```