S1 - 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 names of all branches in loan relation.
2. Find all loan numbers for loans made at ‘Wadia College’ Branch with loan amount > 12000.
3. Find all customers who have a loan from bank. Find their names,loan_no and loan amount.
4. List all customers in alphabetical order who have loan from ‘Wadia College’ branch.
5. Display distinct cities of branch.

Creating the database

```
CREATE DATABASE Bank1;
USE Bank1;
```

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),
('Gundeti', 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 the names of all branches in loan relation.

```
SELECT DISTINCT branch_name FROM Loan;
```

2. Find all loan numbers for loans made at ‘Wadia College’ Branch with loan amount > 12000.

```
SELECT loan_no FROM Loan WHERE branch_name = 'Wadia College' AND amount > 12000;
```

3. Find all customers who have a loan from bank. Find their names,loan_no and loan amount.

```
SELECT Borrower.cust_name, Borrower.loan_no, Loan.amount FROM Borrower INNER JOIN Loan ON Borrower.loan_no = Loan.loan_no;
```

4. List all customers in alphabetical order who have loan from ‘Wadia College’ branch.

```
SELECT cust_name FROM Borrower INNER JOIN Loan on Borrower.loan_no = Loan.loan_no WHERE Loan.branch_name = 'Wadia College' ORDER BY
```

5. Display distinct cities of branch.

```
SELECT DISTINCT branch_city FROM Branch;
```

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:

```
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'),
('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.(

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.O
```

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 C
```

- 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.C
```

- 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.(
```

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;
```

S6 - 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. Change the city of employee working with InfoSys to ‘Bangalore’
2. Find the names of all employees who earn more than the average salary of all employees of their company. Assume that all people work for at most one company.
3. Find the names, street address, and cities of residence for all employees who work for 'TechM' and earn more than $10,000.
4. Change name of table Manages to Management.
5. Create Simple and Unique index on employee table.
6. Display index Information

Creating the database

```
CREATE DATABASE Companies2;
USE Companies2;
```

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'),
(6, 'Jambo', 'Street 23', 'Mumbai');

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

INSERT INTO Works VALUES
('Mehul', 'MEPA', 15000),
('Himanshu', 'TCS', 25000),
('Gundeti', 'TCS', 9000),
('Salvi', 'TechM', 8000),
('Afan', 'Infosys', 13000),
('Jambo', 'MEPA', 28000);


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

Queries

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

```
UPDATE Company SET city = "Bangalore" WHERE company_name = "Infosys";
SELECT * FROM Company;
```

2. Find the names of all employees who earn more than the average salary of all employees of their company. Assume that all people work for at most one company.

```
SELECT employee_name, salary, company_name FROM Works as W WHERE salary > (SELECT AVG(salary) FROM Works WHERE company_name = W.compa
```

3. 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, street, city FROM Employee INNER JOIN Works ON Employee.employee_name = Works.employee_name WHERE sal
```

4. Change name of table Manages to Management.

```
ALTER TABLE Manages RENAME TO Management;
SHOW TABLES;
```

5. Create Simple and Unique index on employee table.

```
-- Simple Index
CREATE INDEX emp_index ON Employee(employee_name);

-- Unique Index
CREATE UNIQUE INDEX emp_uniqueIndex ON Employee(emp_id);
```

6. Display index Information

```
SHOW INDEX FROM Employee;
```

S7 - 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)

1. Create a View1 to display List all customers in alphabetical order who have loan from Pune_Station branch.
2. Create View2 on branch table by selecting any two columns and perform insert update delete operations.
3. Create View3 on borrower and depositor table by selecting any one column from each table perform insert update delete operations.
4. Create Union of left and right joint for all customers who have an account or loan or both at bank
5. Create Simple and Unique index.
6. Display index Information.

Creating the database

```
CREATE DATABASE Bank4;
USE Bank4;
```

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
('Pune_Station', '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, 'Pune_Station', 6462);

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

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

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

Queries

1. Create a View1 to display List all customers in alphabetical order who have loan from Pune_Station branch.

```
CREATE VIEW View1 AS SELECT cust_name FROM Borrower INNER JOIN Loan ON Borrower.loan_no = Loan.loan_no WHERE branch_name = "Pune_Sta
SELECT * FROM View1;
```

2. Create View2 on branch table by selecting any two columns and perform insert update delete operations.

```
-- Creating view
CREATE VIEW View2 AS SELECT branch_name, assets FROM Branch;
SELECT * FROM View2;

-- Insert operation
INSERT INTO View2 VALUES ('Kharadi', 594000);
INSERT INTO View2 VALUES ('Yerwada', 34004);
SELECT * FROM View2;

-- Update operation
UPDATE View2 SET assets = 590000 WHERE branch_name = "Kharadi";
UPDATE View2 SET assets = 24000 WHERE branch_name = "Yerwada";
SELECT * FROM View2;

-- Delete
DELETE FROM View2 WHERE branch_name = "Yerwada";
SELECT * FROM View2;
```

3. Create View3 on borrower and depositor table by selecting any one column from each table perform insert update delete operations.

```
-- Creating view
CREATE VIEW View3 AS SELECT Borrower.cust_name, Depositor.acc_no FROM Borrower JOIN Depositor ON Borrower.cust_name = Depositor.cust_
SELECT * FROM View3;

-- Insert operation
INSERT INTO Borrower (cust_name, loan_no) VALUES ('Pintu', 205);
INSERT INTO Depositor (cust_name, acc_no) VALUES ('Pintu', 102);
SELECT * FROM View3;

-- Update operation
UPDATE View3 SET cust_name = "Piyush" WHERE cust_name = "Pintu";
SELECT * FROM View3;

-- Delete operation
DELETE FROM Borrower WHERE cust_name = 'Macho';
-- This will also remove it from View3. We cannot perform delete operation directly View3 since it is created using join clause
SELECT * FROM View3;
```

4. Create Union of left and right joint for all customers who have an account or loan or both at bank

```
SELECT Borrower.cust_name FROM Borrower LEFT JOIN Loan ON Borrower.loan_no = Loan.loan_no UNION SELECT Depositor.cust_name FROM Depos
```

5. Create Simple and Unique index.

```
-- Simple Index
CREATE INDEX loaners ON Borrower(cust_name);

-- Unique Index
CREATE UNIQUE INDEX depos ON Depositor(cust_name);
```

6. Display index Information.

```
SHOW INDEX FROM Borrower;
SHOW INDEX FROM Depositor;
```

S8 - SQL Queries (in MySQL)

Problem Statement: Consider following Relation: Companies (comp_id, name, cost, year) Orders (comp_id, domain, quantity) Execute the following query:

1. Find names, costs, domains and quantities for companies using inner join.
2. Find names, costs, domains and quantities for companies using left outer join.
3. Find names, costs, domains and quantities for companies using right outer join.
4. Find names, costs, domains and quantities for companies using Union operator.
5. Create View View1 by selecting both tables to show company name and quantities.
6. Create View View2 by selecting any two columns and perform insert update delete operations.
7. Display content of View1, View2.

Creating the database

```
CREATE DATABASE Store2;
USE Store2;
```

Creating tables:

```
CREATE TABLE Companies (
  comp_id INT,
  name VARCHAR(255),
  cost INT,
  year INT,
  PRIMARY KEY (comp_id)
);

CREATE TABLE Orders (
  comp_id INT,
  domain VARCHAR(255),
  quantity INT,
  FOREIGN KEY (comp_id) REFERENCES Companies (comp_id)
);
```

Inserting data

```
INSERT INTO Companies VALUES
(1, 'MEPA', 40500, 2024),
(2, 'Wayne Industries', 950000, 2000),
(3, 'Oscorp', 64600, 2013),
(4, 'Lex Corp', 28500, 2001),
(5, 'Vought', 77335, 2020);

INSERT INTO Orders VALUES
(1, 'Healthcare', 45),
(2, 'Kevlar', 30),
(3, 'Goblin masks', 62),
(4, 'Haircare', 23),
(5, 'Spandex', 9);
```

Queries

1. Find names, costs, domains and quantities for companies using inner join.

```
SELECT name, cost, domain, quantity FROM Companies INNER JOIN Orders ON Companies.comp_id = Orders.comp_id;
SELECT DISTINCT name, cost, domain, quantity FROM Companies, Orders;
```

2. Find names, costs, domains and quantities for companies using left outer join.

```
SELECT name, cost, domain, quantity FROM Companies LEFT JOIN Orders ON Companies.comp_id = Orders.comp_id;
```

3. Find names, costs, domains and quantities for companies using right outer join.

```
SELECT name, cost, domain, quantity FROM Companies RIGHT JOIN Orders on Companies.comp_id = Orders.comp_id;
```

4. Find names, costs, domains and quantities for companies using Union operator.

```
SELECT name AS info, cost AS value FROM Companies UNION SELECT domain AS info, quantity AS value FROM Orders;
```

5. Create View View1 by selecting both tables to show company name and quantities.

```
CREATE VIEW View1 AS SELECT name, quantity FROM Companies INNER JOIN Orders ON Companies.comp_id = Orders.comp_id;
SELECT * FROM View1;
```

6. Create View View2 by selecting any two columns and perform insert update delete operations.

```
-- Creating view
CREATE VIEW View2 AS SELECT Companies.comp_id, domain FROM Companies INNER JOIN Orders ON Companies.comp_id = Orders.comp_id;
SELECT * FROM View2;

-- Insert operation
INSERT INTO Companies VALUES (6, 'Stark Industries', 54322, 2012);
INSERT INTO Orders VALUES (6, 'Vibranium', 66);
SELECT * FROM View2;

-- Update operation
UPDATE View2 SET domain = 'Iridium' WHERE comp_id = 6;
SELECT * FROM View2;

-- Delete operation
DELETE FROM Orders WHERE comp_id = 8;
DELETE FROM Companies WHERE comp_id = 8;
SELECT * FROM View2;
```

7. Display content of View1, View2.

```
SELECT * FROM View1;
SELECT * FROM View2;
```

S9 - SQL Queries (in MySQL)

Problem Statement: SQL Queries Create following tables with suitable constraints. Insert data and solve the following queries: CUSTOMERS(*CNo*, Cname, Ccity, CMobile) ITEMS(*INo*, Iname, Itype, Iprice, Icount) PURCHASE(*PNo*, Pdate, Pquantity, Cno, INo)

1. List all stationary items with price between 400/- to 1000/-
2. Change the mobile number of customer “Gopal”
3. Display the item with maximum price
4. Display all purchases sorted from the most recent to the oldest
5. Count the number of customers in every city
6. Display all purchased quantity of Customer Maya
7. Create view which shows Iname, Price and Count of all stationary items in descending order of price.

Creating the database

```
CREATE DATABASE Store3;
USE Store3;
```

Creating tables:

```
CREATE TABLE Customers (
  CNo INT,
  Cname VARCHAR(255),
  Ccity VARCHAR(255),
  Cmobile BIGINT,
  PRIMARY KEY (CNo)
);

CREATE TABLE Items (
  INo INT,
  Iname VARCHAR(255),
  Itype VARCHAR(255),
  Iprice INT,
  Icount INT,
  PRIMARY KEY (Icount)
);

CREATE TABLE Purchase (
  PNo INT,
  Pdate DATE,
  Pquantity INT,
  Cno INT,
  INo INT,
  PRIMARY KEY (PNo),
  FOREIGN KEY (Cno) REFERENCES Customers (CNo)
);
```

> *[!WARNING] Notice inconsistent naming for columns? We're just doing it by the books. Blame the one who made these problem statements (https://git.kska.io/sppu-te-comp-content/DatabaseManagementSystems/src/branch/main/Practical/Practical%20Exam/DBMSL%20-%20Problem%20Statements%20for%20Practical%20Exam%20%28November%202024%29.pdf).*

Inserting data

```
INSERT INTO Customers VALUES
(1, 'Kalas', 'Pune', 9857265240),
(2, 'Himanshu', 'Chennai', 9857265241),
(3, 'Gopal', 'Mumbai', 9857265245),
(4, 'Maya', 'Mumbai', 9857265243),
(5, 'Mehul', 'Pune', 9857265244);

INSERT INTO Items VALUES
(101, 'Stamp', 'Collectables', 850, 10),
(102, 'Pen', 'Instruments', 549, 50),
(103, 'Sticky notes', 'Writing', 150, 200),
(104, 'Geometry box', 'Instruments', 1350, 70),
(105, 'Pencil', 'Instruments', 670, 45);

INSERT INTO Purchase VALUES
(201, '2024-11-05', 4, 1, 101),
(202, '2024-11-07', 3, 2, 102),
(203, '2024-11-08', 20, 3, 103),
(204, '2024-10-29', 1, 4, 104),
(205, '2024-11-10', 7, 5, 105);
```

Queries

1. List all stationary items with price between 400/- to 1000/-

```
SELECT Iname FROM Items WHERE Iprice BETWEEN 400 AND 1000;
```

2. Change the mobile number of customer “Gopal”

```
UPDATE Customers SET CMobile = 9857265242 WHERE Cname = "Gopal";
SELECT * FROM Customers WHERE Cname = "Gopal";
```

3. Display the item with maximum price

```
SELECT Iname FROM Items WHERE Iprice = (SELECT MAX(Iprice) FROM Items);
```

4. Display all purchases sorted from the most recent to the oldest

```
SELECT * FROM Purchase ORDER BY Pdate DESC;
```

5. Count the number of customers in every city

```
SELECT COUNT(*), Ccity FROM Customers GROUP BY Ccity;
```

6. Display all purchased quantity of Customer Maya

```
SELECT Iname, Pquantity FROM Purchase INNER JOIN Customers ON Purchase.Cno = Customers.CNo INNER JOIN Items ON Purchase.INo = Items.
```

7. Create view which shows Iname, Price and Count of all stationary items in descending order of price.

```
CREATE VIEW itemView AS SELECT Iname, Iprice, Icount FROM Items ORDER BY Iprice DESC;
SELECT * FROM itemView;
```