S1.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.

CREATE TABLE Account (

Acc\_no INT PRIMARY KEY,

branch\_name VARCHAR(50) NOT NULL,

balance DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (branch\_name) REFERENCES Branch(branch\_name)

);

CREATE TABLE Branch (

branch\_name VARCHAR(50) PRIMARY KEY,

branch\_city VARCHAR(50) NOT NULL,

assets DECIMAL(15, 2) NOT NULL

);

CREATE TABLE Customer (

cust\_name VARCHAR(50) PRIMARY KEY,

cust\_street VARCHAR(100),

cust\_city VARCHAR(50) NOT NULL

);

CREATE TABLE Depositor (

cust\_name VARCHAR(50),

acc\_no INT,

PRIMARY KEY (cust\_name, acc\_no),

FOREIGN KEY (cust\_name) REFERENCES Customer(cust\_name),

FOREIGN KEY (acc\_no) REFERENCES Account(acc\_no)

);

CREATE TABLE Loan (

loan\_no INT PRIMARY KEY,

branch\_name VARCHAR(50) NOT NULL,

amount DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (branch\_name) REFERENCES Branch(branch\_name)

);

CREATE TABLE Borrower (

cust\_name VARCHAR(50),

loan\_no INT,

PRIMARY KEY (cust\_name, loan\_no),

FOREIGN KEY (cust\_name) REFERENCES Customer(cust\_name),

FOREIGN KEY (loan\_no) REFERENCES Loan(loan\_no)

);

Find the names of all branches in the loan relation.

SELECT DISTINCT branch\_name FROM Loan;

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

SELECT loan\_no FROM Loan

WHERE branch\_name = 'Wadia College' AND amount > 12000;

Find all customers who have a loan from the bank. Find their names, loan number, and loan amount.

SELECT Customer.cust\_name, Loan.loan\_no, Loan.amount

FROM Customer

JOIN Borrower ON Customer.cust\_name = Borrower.cust\_name

JOIN Loan ON Borrower.loan\_no = Loan.loan\_no;

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

SELECT DISTINCT Customer.cust\_name

FROM Customer

JOIN Borrower ON Customer.cust\_name = Borrower.cust\_name

JOIN Loan ON Borrower.loan\_no = Loan.loan\_no

WHERE Loan.branch\_name = 'Wadia College'

ORDER BY Customer.cust\_name;

Display distinct cities of branches.

SELECT DISTINCT branch\_city FROM Branch;

S2. 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

CREATE TABLE Account (

Acc\_no INT PRIMARY KEY,

branch\_name VARCHAR(50) NOT NULL,

balance DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (branch\_name) REFERENCES Branch(branch\_name)

);

CREATE TABLE Branch (

branch\_name VARCHAR(50) PRIMARY KEY,

branch\_city VARCHAR(50) NOT NULL,

assets DECIMAL(15, 2) NOT NULL

);

CREATE TABLE Customer (

cust\_name VARCHAR(50) PRIMARY KEY,

cust\_street VARCHAR(100),

cust\_city VARCHAR(50) NOT NULL

);

CREATE TABLE Depositor (

cust\_name VARCHAR(50),

acc\_no INT,

PRIMARY KEY (cust\_name, acc\_no),

FOREIGN KEY (cust\_name) REFERENCES Customer(cust\_name),

FOREIGN KEY (acc\_no) REFERENCES Account(acc\_no)

);

CREATE TABLE Loan (

loan\_no INT PRIMARY KEY,

branch\_name VARCHAR(50) NOT NULL,

amount DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (branch\_name) REFERENCES Branch(branch\_name)

);

CREATE TABLE Borrower (

cust\_name VARCHAR(50),

loan\_no INT,

PRIMARY KEY (cust\_name, loan\_no),

FOREIGN KEY (cust\_name) REFERENCES Customer(cust\_name),

FOREIGN KEY (loan\_no) REFERENCES Loan(loan\_no)

);

Find all customers who have both an account and a loan at the bank.

SELECT DISTINCT Depositor.cust\_name

FROM Depositor

INNER JOIN Borrower ON Depositor.cust\_name = Borrower.cust\_name;

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

SELECT DISTINCT cust\_name

FROM (

SELECT cust\_name FROM Depositor

UNION

SELECT cust\_name FROM Borrower

) AS all\_customers;

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

SELECT DISTINCT Depositor.cust\_name

FROM Depositor

LEFT JOIN Borrower ON Depositor.cust\_name = Borrower.cust\_name

WHERE Borrower.loan\_no IS NULL;

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

SELECT AVG(balance) AS avg\_balance

FROM Account

WHERE branch\_name = 'Wadia College';

Find the number of depositors at each branch.

SELECT branch\_name, COUNT(DISTINCT Depositor.cust\_name) AS num\_depositors

FROM Account

JOIN Depositor ON Account.Acc\_no = Depositor.acc\_no

GROUP BY branch\_name;

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.

CREATE TABLE Account (

Acc\_no INT PRIMARY KEY,

branch\_name VARCHAR(50) NOT NULL,

balance DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (branch\_name) REFERENCES Branch(branch\_name)

);

CREATE TABLE Branch (

branch\_name VARCHAR(50) PRIMARY KEY,

branch\_city VARCHAR(50) NOT NULL,

assets DECIMAL(15, 2) NOT NULL

);

CREATE TABLE Customer (

cust\_name VARCHAR(50) PRIMARY KEY,

cust\_street VARCHAR(100) NOT NULL,

cust\_city VARCHAR(50) NOT NULL

);

CREATE TABLE Depositor (

cust\_name VARCHAR(50),

acc\_no INT,

PRIMARY KEY (cust\_name, acc\_no),

FOREIGN KEY (cust\_name) REFERENCES Customer(cust\_name),

FOREIGN KEY (acc\_no) REFERENCES Account(Acc\_no)

);

CREATE TABLE Loan (

loan\_no INT PRIMARY KEY,

branch\_name VARCHAR(50) NOT NULL,

amount DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (branch\_name) REFERENCES Branch(branch\_name)

);

CREATE TABLE Borrower (

cust\_name VARCHAR(50),

loan\_no INT,

PRIMARY KEY (cust\_name, loan\_no),

FOREIGN KEY (cust\_name) REFERENCES Customer(cust\_name),

FOREIGN KEY (loan\_no) REFERENCES Loan(loan\_no)

);

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

SELECT branch\_name

FROM Account

GROUP BY branch\_name

HAVING AVG(balance) > 15000;

**Find the number of tuples in the Customer relation.**

SELECT COUNT(\*) AS num\_tuples

FROM Customer;

**Calculate the total loan amount given by the bank.**

SELECT SUM(amount) AS total\_loan\_amount

FROM Loan;

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

DELETE FROM Loan

WHERE amount BETWEEN 1300 AND 1500;

**Find the average account balance at each branch.**

SELECT branch\_name, AVG(balance) AS average\_balance

FROM Account

GROUP BY branch\_name;

**Find the name of the customer and city where the customer’s name starts with the letter 'P'.**

SELECT cust\_name, cust\_city

FROM Customer

WHERE cust\_name LIKE 'P%';

S4. 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.

CREATE TABLE Cust\_Master (

Cust\_no VARCHAR(10) PRIMARY KEY,

Cust\_name VARCHAR(50) NOT NULL,

Cust\_addr VARCHAR(100) NOT NULL

);

CREATE TABLE Order (

Order\_no INT PRIMARY KEY,

Cust\_no VARCHAR(10) NOT NULL,

Order\_date DATE NOT NULL,

Qty\_Ordered INT NOT NULL,

FOREIGN KEY (Cust\_no) REFERENCES Cust\_Master(Cust\_no)

);

CREATE TABLE Product (

Product\_no INT PRIMARY KEY,

Product\_name VARCHAR(50) NOT NULL,

Order\_no INT NOT NULL,

FOREIGN KEY (Order\_no) REFERENCES Order(Order\_no)

);

-- Inserting records into Cust\_Master

INSERT INTO Cust\_Master (Cust\_no, Cust\_name, Cust\_addr)

VALUES ('C1001', 'Alice', 'Mumbai'),

('C1002', 'Anil', 'Bangalore'),

('C1003', 'Ravi', 'Chennai'),

('C1004', 'Pankaj', 'Manglore'),

('C1005', 'Amit', 'Pune'),

('C1006', 'Pranav', 'Manglore');

-- Inserting records into Order

INSERT INTO Order (Order\_no, Cust\_no, Order\_date, Qty\_Ordered)

VALUES (101, 'C1001', '2024-10-01', 2),

(102, 'C1002', '2024-10-05', 1),

(103, 'C1003', '2024-10-10', 3),

(104, 'C1004', '2024-10-15', 5),

(105, 'C1005', '2024-10-20', 2),

(106, 'C1006', '2024-10-25', 4);

-- Inserting records into Product

INSERT INTO Product (Product\_no, Product\_name, Order\_no)

VALUES (1, 'Laptop', 101),

(2, 'Tablet', 102),

(3, 'Smartphone', 103),

(4, 'Monitor', 104),

(5, 'Keyboard', 105),

(6, 'Mouse', 106);

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

SELECT Cust\_name

FROM Cust\_Master

WHERE Cust\_name LIKE '\_A%';

Display orders from Customer no C1002, C1005, C1007, and C1008.

SELECT \*

FROM Order

WHERE Cust\_no IN ('C1002', 'C1005', 'C1007', 'C1008');

List clients who stay in either 'Bangalore' or 'Mangalore'.

SELECT Cust\_name

FROM Cust\_Master

WHERE Cust\_addr IN ('Bangalore', 'Manglore');

Display the name of customers and the product\_name they have purchased.

SELECT Cust\_Master.Cust\_name, Product.Product\_name

FROM Cust\_Master

JOIN Order ON Cust\_Master.Cust\_no = Order.Cust\_no

JOIN Product ON Order.Order\_no = Product.Order\_no;

Create view View1 consisting of Cust\_name, Product\_name.

CREATE VIEW View1 AS

SELECT Cust\_Master.Cust\_name, Product.Product\_name

FROM Cust\_Master

JOIN Order ON Cust\_Master.Cust\_no = Order.Cust\_no

JOIN Product ON Order.Order\_no = Product.Order\_no;

**Display product\_name and quantity purchased by each customer.**

SELECT Cust\_Master.Cust\_name, Product.Product\_name, Order.Qty\_Ordered

FROM Cust\_Master

JOIN Order ON Cust\_Master.Cust\_no = Order.Cust\_no

JOIN Product ON Order.Order\_no = Product.Order\_no;

Perform different join operations.

**Inner Join**: Lists only matching records.

SELECT Cust\_Master.Cust\_name, Order.Order\_no

FROM Cust\_Master

INNER JOIN Order ON Cust\_Master.Cust\_no = Order.Cust\_no;

**Left Join**: Lists all customers and their orders if any, showing NULL for customers without orders.

SELECT Cust\_Master.Cust\_name, Order.Order\_no

FROM Cust\_Master

LEFT JOIN Order ON Cust\_Master.Cust\_no = Order.Cust\_no;

**Right Join**: Lists all orders and their customers, showing NULL for orders without a matching customer (if any).

SELECT Cust\_Master.Cust\_name, Order.Order\_no

FROM Cust\_Master

RIGHT JOIN Order ON Cust\_Master.Cust\_no = Order.Cust\_no;

**Full Outer Join**: Lists all customers and all orders, with NULLs for any missing matches on either side (only works if supported by your SQL system).

SELECT Cust\_Master.Cust\_name, Order.Order\_no

FROM Cust\_Master

FULL OUTER JOIN Order ON Cust\_Master.Cust\_no = Order.Cust\_no;

S5. 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.

CREATE TABLE Employee (

emp\_id INT PRIMARY KEY,

employee\_name VARCHAR(50) NOT NULL,

street VARCHAR(100),

city VARCHAR(50)

);

CREATE TABLE Company (

company\_name VARCHAR(50) PRIMARY KEY,

city VARCHAR(50) NOT NULL

);

CREATE TABLE Works (

employee\_name VARCHAR(50) NOT NULL,

company\_name VARCHAR(50) NOT NULL,

salary DECIMAL(10, 2),

FOREIGN KEY (employee\_name) REFERENCES Employee(employee\_name),

FOREIGN KEY (company\_name) REFERENCES Company(company\_name)

);

CREATE TABLE Manages (

employee\_name VARCHAR(50) NOT NULL,

manager\_name VARCHAR(50) NOT NULL,

FOREIGN KEY (employee\_name) REFERENCES Employee(employee\_name),

FOREIGN KEY (manager\_name) REFERENCES Employee(employee\_name)

);

INSERT INTO Employee (emp\_id, employee\_name, street, city)

VALUES

(1, 'Alice', '123 Maple St', 'Mumbai'),

(2, 'Bob', '456 Oak St', 'Delhi'),

(3, 'Charlie', '789 Pine St', 'Pune'),

(4, 'Diana', '101 Birch St', 'Hyderabad'),

(5, 'Eve', '202 Cedar St', 'Bangalore');

INSERT INTO Company (company\_name, city)

VALUES

('TCS', 'Mumbai'),

('InfoSys', 'Bangalore'),

('TechM', 'Pune'),

('Wipro', 'Hyderabad');

INSERT INTO Works (employee\_name, company\_name, salary)

VALUES

('Alice', 'TCS', 12000.00),

('Bob', 'InfoSys', 15000.00),

('Charlie', 'TechM', 9500.00),

('Diana', 'TechM', 10500.00),

('Eve', 'Wipro', 13000.00);

INSERT INTO Manages (employee\_name, manager\_name)

VALUES

('Alice', 'Bob'),

('Charlie', 'Alice'),

('Diana', 'Charlie'),

('Eve', 'Alice');

SELECT employee\_name

FROM Works

WHERE company\_name = 'TCS';

SELECT employee\_name, company\_name

FROM Works

ORDER BY company\_name ASC, employee\_name DESC;

UPDATE Employee

SET city = 'Bangalore'

WHERE employee\_name IN (

SELECT employee\_name

FROM Works

WHERE company\_name = 'InfoSys'

);

SELECT e.employee\_name, e.street, e.city

FROM Employee e

JOIN Works w ON e.employee\_name = w.employee\_name

WHERE w.company\_name = 'TechM' AND w.salary > 10000;

ALTER TABLE Company

ADD COLUMN Asset DECIMAL(15, 2);

S6.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

CREATE TABLE Employee (

emp\_id INT PRIMARY KEY,

employee\_name VARCHAR(50) NOT NULL UNIQUE,

street VARCHAR(100),

city VARCHAR(50)

);

CREATE TABLE Company (

company\_name VARCHAR(50) PRIMARY KEY,

city VARCHAR(50) NOT NULL

);

CREATE TABLE Works (

employee\_name VARCHAR(50) NOT NULL,

company\_name VARCHAR(50) NOT NULL,

salary DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (employee\_name) REFERENCES Employee(employee\_name),

FOREIGN KEY (company\_name) REFERENCES Company(company\_name)

);

CREATE TABLE Manages (

employee\_name VARCHAR(50) NOT NULL,

manager\_name VARCHAR(50) NOT NULL,

FOREIGN KEY (employee\_name) REFERENCES Employee(employee\_name),

FOREIGN KEY (manager\_name) REFERENCES Employee(employee\_name)

);

**Change the city of employees working with InfoSys to 'Bangalore':**

UPDATE Employee

SET city = 'Bangalore'

WHERE employee\_name IN (

SELECT employee\_name

FROM Works

WHERE company\_name = 'InfoSys'

);

Find names of employees earning more than the average salary of employees in their company:

SELECT w.employee\_name

FROM Works w

JOIN (

SELECT company\_name, AVG(salary) AS avg\_salary

FROM Works

GROUP BY company\_name

) AS avg\_salaries

ON w.company\_name = avg\_salaries.company\_name

WHERE w.salary > avg\_salaries.avg\_salary;

Find names, street addresses, and cities for employees working for 'TechM' and earning more than $10,000:

SELECT e.employee\_name, e.street, e.city

FROM Employee e

JOIN Works w ON e.employee\_name = w.employee\_name

WHERE w.company\_name = 'TechM' AND w.salary > 10000;

Change the name of the Manages table to Management:

ALTER TABLE Manages

RENAME TO Management;

Create a simple and unique index on the Employee table:

-- Simple index on employee\_name

CREATE INDEX idx\_employee\_name ON Employee(employee\_name);

-- Unique index on emp\_id

CREATE UNIQUE INDEX idx\_emp\_id ON Employee(emp\_id);

Display index information:

SHOW INDEX FROM Employee;

S7. 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) Execute the following query: 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.

Create View1 for Customers with Loans from Pune\_Station Branch

CREATE VIEW View1 AS

SELECT c.cust\_name

FROM Customer c

JOIN Borrower b ON c.cust\_name = b.cust\_name

JOIN Loan l ON b.loan\_no = l.loan\_no

WHERE l.branch\_name = 'Pune\_Station'

ORDER BY c.cust\_name ASC;

Create View2 on Branch Table and Perform Insert, Update, Delete Operations

CREATE VIEW View2 AS

SELECT branch\_name, branch\_city

FROM Branch;

**Insert Operation:**

INSERT INTO Branch (branch\_name, branch\_city)

VALUES ('New\_Branch', 'Mumbai');

Update Operation:

UPDATE Branch

SET branch\_city = 'Pune'

WHERE branch\_name = 'New\_Branch';

Delete Operation:

DELETE FROM Branch

WHERE branch\_name = 'New\_Branch';

. Create View3 on Borrower and Depositor Tables and Perform Insert, Update, Delete Operations

CREATE VIEW View3 AS

SELECT b.cust\_name AS borrower\_name, d.acc\_no AS depositor\_acc\_no

FROM Borrower b

JOIN Depositor d ON b.cust\_name = d.cust\_name;

**Insert Operation:**

INSERT INTO Depositor (cust\_name, acc\_no)

VALUES ('New\_Customer', 12345);

INSERT INTO Borrower (cust\_name, loan\_no)

VALUES ('New\_Customer', 56789);

Update Operation:

UPDATE Depositor

SET acc\_no = 54321

WHERE cust\_name = 'New\_Customer';

Delete Operation:

DELETE FROM Borrower

WHERE cust\_name = 'New\_Customer';

DELETE FROM Depositor

WHERE cust\_name = 'New\_Customer';

Create Union of Left and Right Joins for All Customers Who Have an Account or Loan or Both

SELECT DISTINCT cust\_name

FROM Depositor

LEFT JOIN Borrower ON Depositor.cust\_name = Borrower.cust\_name

UNION

SELECT DISTINCT cust\_name

FROM Borrower

RIGHT JOIN Depositor ON Borrower.cust\_name = Depositor.cust\_name;

Create Simple and Unique Index

CREATE INDEX idx\_balance ON Account(balance);

CREATE UNIQUE INDEX idx\_cust\_name ON Customer(cust\_name);

Display Index Information

SHOW INDEX FROM Account;

SHOW INDEX FROM Customer;

SHOW INDEX FROM Branch;

SHOW INDEX FROM Loan;

SHOW INDEX FROM Depositor;

SHOW INDEX FROM Borrower;

S8. 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.

CREATE TABLE Companies (

comp\_id INT PRIMARY KEY,

name VARCHAR(50) NOT NULL,

cost DECIMAL(10, 2),

year INT

);

CREATE TABLE Orders (

comp\_id INT,

domain VARCHAR(50),

quantity INT,

FOREIGN KEY (comp\_id) REFERENCES Companies(comp\_id)

);

Inner Join: Find names, costs, domains, and quantities for companies.

SELECT c.name, c.cost, o.domain, o.quantity

FROM Companies c

INNER JOIN Orders o ON c.comp\_id = o.comp\_id;

Left Outer Join: Find names, costs, domains, and quantities for companies.

SELECT c.name, c.cost, o.domain, o.quantity

FROM Companies c

LEFT JOIN Orders o ON c.comp\_id = o.comp\_id;

Right Outer Join: Find names, costs, domains, and quantities for companie

SELECT c.name, c.cost, o.domain, o.quantity

FROM Companies c

RIGHT JOIN Orders o ON c.comp\_id = o.comp\_id;

Union Operator: Find names, costs, domains, and quantities for companies.

SELECT c.name, c.cost, o.domain, o.quantity

FROM Companies c

LEFT JOIN Orders o ON c.comp\_id = o.comp\_id

UNION

SELECT c.name, c.cost, o.domain, o.quantity

FROM Companies c

RIGHT JOIN Orders o ON c.comp\_id = o.comp\_id;

Create View1 to show company name and quantities.

CREATE VIEW View1 AS

SELECT c.name, o.quantity

FROM Companies c

JOIN Orders o ON c.comp\_id = o.comp\_id;

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

Create View2:

CREATE VIEW View2 AS

SELECT name, cost

FROM Companies;

Insert Operation:

INSERT INTO Companies (comp\_id, name, cost, year)

VALUES (4, 'NewCompany', 5000.00, 2022);

Update Operation:

UPDATE Companies

SET cost = 5500.00

WHERE name = 'NewCompany';

Delete Operation:

DELETE FROM Companies

WHERE name = 'NewCompany';

Display content of View1 and View2.

SELECT \* FROM View1;

SELECT \* FROM View2;

S9. 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.

CREATE TABLE CUSTOMERS (

CNo INT PRIMARY KEY,

Cname VARCHAR(50) NOT NULL,

Ccity VARCHAR(50),

CMobile VARCHAR(15)

);

CREATE TABLE ITEMS (

INo INT PRIMARY KEY,

Iname VARCHAR(50) NOT NULL,

Itype VARCHAR(50),

Iprice DECIMAL(10, 2),

Icount INT

);

CREATE TABLE PURCHASE (

PNo INT PRIMARY KEY,

Pdate DATE,

Pquantity INT,

CNo INT,

INo INT,

FOREIGN KEY (CNo) REFERENCES CUSTOMERS(CNo),

FOREIGN KEY (INo) REFERENCES ITEMS(INo)

);

-- Inserting sample data into CUSTOMERS table

INSERT INTO CUSTOMERS (CNo, Cname, Ccity, CMobile) VALUES

(1, 'Gopal', 'Delhi', '9876543210'),

(2, 'Maya', 'Mumbai', '9123456789'),

(3, 'Ravi', 'Delhi', '9234567890');

-- Inserting sample data into ITEMS table

INSERT INTO ITEMS (INo, Iname, Itype, Iprice, Icount) VALUES

(1, 'Pen', 'Stationary', 500, 100),

(2, 'Notebook', 'Stationary', 700, 50),

(3, 'Calculator', 'Electronics', 1500, 30),

(4, 'Eraser', 'Stationary', 300, 200);

-- Inserting sample data into PURCHASE table

INSERT INTO PURCHASE (PNo, Pdate, Pquantity, CNo, INo) VALUES

(1, '2024-10-01', 2, 1, 1),

(2, '2024-10-05', 1, 2, 2),

(3, '2024-10-07', 3, 2, 4),

(4, '2024-10-10', 2, 3, 3);

List all stationary items with a price between 400 and 1000.

SELECT Iname, Iprice

FROM ITEMS

WHERE Itype = 'Stationary' AND Iprice BETWEEN 400 AND 1000;

Change the mobile number of customer “Gopal”.

UPDATE CUSTOMERS

SET CMobile = '9999999999'

WHERE Cname = 'Gopal';

**Display the item with the maximum price.**

SELECT Iname, Iprice

FROM ITEMS

WHERE Iprice = (SELECT MAX(Iprice) FROM ITEMS);

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

SELECT \*

FROM PURCHASE

ORDER BY Pdate DESC;

Count the number of customers in every city.

SELECT Ccity, COUNT(\*) AS CustomerCount

FROM CUSTOMERS

GROUP BY Ccity;

Display all purchased quantity of Customer Maya.

SELECT p.Pquantity, i.Iname

FROM PURCHASE p

JOIN CUSTOMERS c ON p.CNo = c.CNo

JOIN ITEMS i ON p.INo = i.INo

WHERE c.Cname = 'Maya';

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

CREATE VIEW StationaryView AS

SELECT Iname, Iprice, Icount

FROM ITEMS

WHERE Itype = 'Stationary'

ORDER BY Iprice DESC;

To display the content of StationaryView:

SELECT \* FROM StationaryView;