SLIP NO : 1

Section II: Database Management Systems [15 Marks]

Q2) Consider the following database (primary keys are underlined).

Room (roomno, room_name, room_type, charges)
Guest (Guestcode, Gname, city)

The relationship is: **Room–Guest: one-to-one.**
`room_type` can have values ‘AC’ or ‘NonAC’.

A. Create the above database in PostgreSQL and insert sufficient records. [5 marks]

```
CREATE TABLE Room(
    roomno INT PRIMARY KEY,
    room_name VARCHAR(50),
    room_type VARCHAR(10) CHECK (room_type IN ('AC','NonAC')),
    charges INT
);

CREATE TABLE Guest(
    Guestcode INT PRIMARY KEY,
    Gname VARCHAR(50),
    city VARCHAR(30),
    roomno INT UNIQUE,
    FOREIGN KEY (roomno) REFERENCES Room(roomno)
);

-- Insert Records
INSERT INTO Room VALUES
(101,'Lotus','AC',8000),
(102,'Rose','NonAC',6000),
(103,'Jasmine','AC',12000),
(104,'Tulip','NonAC',9500);

INSERT INTO Guest VALUES
(1,'Amit','Pune',101),
(2,'Sneha','Mumbai',102),
(3,'Ravi','Nashik',103),
(4,'Neha','Delhi',104);
```

B. Execute the following queries in PostgreSQL (any 3). [6 marks]

i. List the details of the rooms having charges between 5000 and 10000.

```
SELECT * FROM Room
WHERE charges BETWEEN 5000 AND 10000;
```

ii. List the names of the guests in the sorted order by city name.

```
SELECT Gname, city FROM Guest
```

```
ORDER BY city;
```

iii. List the minimum charges of a room.

```
SELECT MIN(charges) AS Minimum_Charges FROM Room;
```

iv. Increase the charges of all AC rooms by 15%.

```
UPDATE Room
SET charges = charges + (charges * 0.15)
WHERE room_type = 'AC';
```

v. List the names of all the NONAC rooms whose charges are more than 10000.

```
SELECT room_name FROM Room
WHERE room_type='NonAC' AND charges > 10000;
```

C. Write a query to list the name of the guest to whom the room with highest charges is allotted. [4 marks]

```
SELECT G.Gname
FROM Guest G
JOIN Room R ON G.roomno = R.roomno
WHERE R.charges = (SELECT MAX(charges) FROM Room);
```

OR

C. Create a view to list the names of all the NonAC rooms which have charges greater than at least one of the 'AC' rooms.

```
CREATE VIEW NonAC_HigherThan_AC AS
SELECT room_name, charges
FROM Room
WHERE room_type = 'NonAC'
AND charges > ANY (SELECT charges FROM Room WHERE room_type='AC');
```

SLIP NO : 2

Section II: Database Management Systems [15 Marks]

Q2) Consider the following database (primary keys are underlined).

Employee (eno, ename, designation, salary)
Department (dno, dname, location)

The relationship is: **Employee–Department: many-to-one.**

A. Create the above database in PostgreSQL and insert sufficient records. [5 marks]

```
CREATE TABLE Department(
    dno INT PRIMARY KEY,
    dname VARCHAR(50),
    location VARCHAR(50)
);

CREATE TABLE Employee(
    eno INT PRIMARY KEY,
    ename VARCHAR(50),
    designation VARCHAR(50),
    salary NUMERIC(10,2),
    dno INT,
    FOREIGN KEY (dno) REFERENCES Department(dno)
);

-- Insert records into Department
INSERT INTO Department VALUES
(1,'HR','Delhi'),
(2,'IT','Pune'),
(3,'Finance','Nashik'),
(4,'Marketing','Mumbai');

-- Insert records into Employee
INSERT INTO Employee VALUES
(101,'Amit','Manager',50000,1),
(102,'Sneha','Analyst',30000,2),
(103,'Ravi','Clerk',25000,3),
(104,'Neha','Executive',35000,2),
(105,'Anita','HR','28000',1);
```

B. Execute the following queries in PostgreSQL (any 3). [6 marks]

i. Give a 5% raise in salary to all the employees

```
UPDATE Employee
SET salary = salary + (salary * 0.05);
```

ii. Display average salary of an employee

```
SELECT AVG(salary) AS Average_Salary FROM Employee;
```

iii. List the details of all the departments located at ‘Nashik’

```
SELECT * FROM Department
WHERE location = 'Nashik';
```

iv. Display the details of employees whose names end with an alphabet ‘a’

```
SELECT * FROM Employee
WHERE ename LIKE '%a';
```

v. Display the location of ‘HR’ department

```
SELECT location FROM Department
WHERE dname = 'HR';
```

C. Write a query to list the details of employees who do not work in any of the departments located at ‘Delhi’. [4 marks]

```
SELECT *
FROM Employee
WHERE dno NOT IN (
    SELECT dno
    FROM Department
    WHERE location = 'Delhi'
);
```

OR (Alternative C)

Create a view to list the names of employees whose salary is greater than all the employees working in ‘HR’ department

```
CREATE VIEW Employees_HigherThan_HR AS
SELECT ename, salary
FROM Employee
WHERE salary > ALL (
    SELECT salary
    FROM Employee E
    JOIN Department D ON E.dno = D.dno
    WHERE D.dname = 'HR'
);
```

SLIP NO : 3

Section II: Database Management Systems [15 Marks]

Q2) Consider the following database (primary keys are underlined).

Person (pnumber, pname, birthdate, income)
Area (aname, area_type, pincode)

The relationship is: **Person–Area: many-to-one.**
`area_type` can have values ‘urban’ or ‘rural’.

A. Create the above database in PostgreSQL and insert sufficient records. [5 marks]

```
CREATE TABLE Area(
    aname VARCHAR(50) PRIMARY KEY,
    area_type VARCHAR(10) CHECK (area_type IN ('urban','rural')),
    pincode INT
);

CREATE TABLE Person(
    pnumber INT PRIMARY KEY,
    pname VARCHAR(50),
    birthdate DATE,
    income NUMERIC(10,2),
    aname VARCHAR(50),
    FOREIGN KEY (aname) REFERENCES Area(aname)
);

-- Insert records into Area
INSERT INTO Area VALUES
('Camp','urban',411001),
('Kalyaninagar','urban',411014),
('Pimpri','rural',412017),
('Hadapsar','urban',411028);

-- Insert records into Person
INSERT INTO Person VALUES
(101,'Ravi','1990-07-12',50000,'Camp'),
(102,'Sneha','1992-05-23',40000,'Kalyaninagar'),
(103,'Ramesh','1988-07-05',60000,'Camp'),
(104,'Neha','1995-09-18',35000,'Pimpri'),
(105,'Rajesh','1991-11-30',45000,'Hadapsar');
```

B. Execute the following queries in PostgreSQL (any 3). [6 marks]

i. List the details of all people whose name starts with the alphabet ‘R’

```
SELECT * FROM Person
WHERE pname LIKE 'R%';
```

ii. List the names of all people whose birthday falls in the month of ‘July’

```
SELECT pname, birthdate FROM Person
WHERE EXTRACT(MONTH FROM birthdate) = 7;
```

iii. Display the details of people in the sorted order of their income

```
SELECT * FROM Person
ORDER BY income;
```

iv. Display the count of areas of ‘urban’ type

```
SELECT COUNT(*) AS Urban_Area_Count
FROM Area
WHERE area_type='urban';
```

v. Change the pincode of ‘Kalyaninagar’ to 411036

```
UPDATE Area
SET pincode = 411036
WHERE aname='Kalyaninagar';
```

C. Write a query to list the names of people who live in ‘Camp’ area and have income less than at least one person who lives in ‘Kalyaninagar’ area. [4 marks]

```
SELECT pname
FROM Person
WHERE aname = 'Camp'
AND income < ANY (
    SELECT income
    FROM Person
    WHERE aname='Kalyaninagar'
);
```

OR (Alternative C)

Create a view to list the details of the person with second maximum income

```
CREATE VIEW SecondHighestIncome AS
SELECT *
FROM Person
WHERE income = (
    SELECT MAX(income)
    FROM Person
    WHERE income < (SELECT MAX(income) FROM Person)
);
```

SLIP NO : 4

Section II: Database Management Systems [15 Marks]

Q2) Consider the following database (primary keys are underlined).

Policy (pno, pname, premium_amt, policy_type)
Customer (cno, cname, city, agent_name)

The relationship is: **Policy–Customer: many-to-one**
`policy_type` can have values: ‘Yearly’, ‘Half-yearly’, ‘Monthly’.

A. Create the above database in PostgreSQL and insert sufficient records [5 marks]

```
CREATE TABLE Policy(
    pno INT PRIMARY KEY,
    pname VARCHAR(50),
    premium_amt NUMERIC(10,2),
    policy_type VARCHAR(15) CHECK (policy_type IN ('Yearly','Half-
yearly','Monthly'))
);

CREATE TABLE Customer(
    cno INT PRIMARY KEY,
    cname VARCHAR(50),
    city VARCHAR(50),
    agent_name VARCHAR(50),
    pno INT,
    FOREIGN KEY (pno) REFERENCES Policy(pno)
);

-- Insert records into Policy
INSERT INTO Policy VALUES
(101,'Jeevan Anand',12000,'Yearly'),
(102,'New India Policy',6000,'Half-yearly'),
(103,'Life Secure',1000,'Monthly'),
(104,'Jeevan Anand',12000,'Yearly'),
(105,'Monthly Saver',1200,'Monthly');

-- Insert records into Customer
INSERT INTO Customer VALUES
(1,'Amit','Pune','Ramesh',101),
(2,'Sneha','Mumbai','Neha',102),
(3,'Ravi','Pune','Anita',103),
(4,'Neha','Delhi','Ramesh',104),
(5,'Rajesh','Pune','Neha',105);
```

B. Execute the following queries in PostgreSQL (any 3) [6 marks]

i. List the details of all customers who live in a city

```
SELECT * FROM CustomerWHERE city IS NOT NULL;
```

ii. Display the average premium amount

```
SELECT AVG(premium_amt) AS Average_Premium
FROM Policy;
```

iii. Count the number of customers who have taken ‘Jeevan Anand’ policy

```
SELECT COUNT(*) AS Jeevan_Anand_Customers
FROM Customer C
JOIN Policy P ON C.pno = P.pno
WHERE P.pname = 'Jeevan Anand';
```

iv. Increase the premium amount for ‘Monthly’ policies by 10%

```
UPDATE Policy
SET premium_amt = premium_amt + (premium_amt * 0.10)
WHERE policy_type = 'Monthly';
```

v. Display the policy_type wise count of policies

```
SELECT policy_type, COUNT(*) AS Policy_Count
FROM Policy
GROUP BY policy_type;
```

C. Write a query to list the policy_type with highest average premium amount [4 marks]

```
SELECT policy_type
FROM Policy
GROUP BY policy_type
ORDER BY AVG(premium_amt) DESC
LIMIT 1;
```

OR (Alternative C)

Create a view to list the name of the customer who has taken the highest number of policies

```
CREATE VIEW TopCustomer AS
SELECT cname
FROM Customer
GROUP BY cname
ORDER BY COUNT(pno) DESC
LIMIT 1;
```

SLIP NO : 5

Section II: Database Management Systems [15 Marks]

Q2) Consider the following database (primary keys are underlined).

Doctor (dno, dname, city)
Patient (pno, pat_name, city, disease)

The relationship is: **Doctor–Patient: many-to-many with `Fees` as a descriptive attribute.**

A. Create the above database in PostgreSQL and insert sufficient records [5 marks]

For **many-to-many**, we need a **junction table** (Doctor_Patient) to store the fees.

```
CREATE TABLE Doctor(
    dno INT PRIMARY KEY,
    dname VARCHAR(50),
    city VARCHAR(50)
);

CREATE TABLE Patient(
    pno INT PRIMARY KEY,
    pat_name VARCHAR(50),
    city VARCHAR(50),
    disease VARCHAR(50)
);

CREATE TABLE Doctor_Patient(
    dno INT,
    pno INT,
    fees NUMERIC(10,2),
    PRIMARY KEY(dno, pno),
    FOREIGN KEY(dno) REFERENCES Doctor(dno),
    FOREIGN KEY(pno) REFERENCES Patient(pno)
);

-- Insert records into Doctor
INSERT INTO Doctor VALUES
(1,'Dr. Kumar','Nagpur'),
(2,'Dr. Patil','Mumbai'),
(3,'Dr. Singh','Nagpur');

-- Insert records into Patient
INSERT INTO Patient VALUES
(101,'Amit','Nagpur','Cancer'),
(102,'Sneha','Pune','Asthma'),
(103,'Anita','Mumbai','Flu'),
(104,'Ravi','Nagpur','Cancer'),
(105,'Ananya','Nagpur','Asthma');

-- Insert records into Doctor_Patient
INSERT INTO Doctor_Patient VALUES
(1,101,500),
(1,102,200),
(2,103,300),
```

```
(2,104,400),
(1,105,200),
(3,101,450);
```

B. Execute the following queries in PostgreSQL (any 3) [6 marks]

i. List the details of all the doctors from 'Nagpur' city

```
SELECT * FROM Doctor
WHERE city='Nagpur';
```

ii. Display count of patients

```
SELECT COUNT(*) AS Total_Patients
FROM Patient;
```

iii. List the names of the patients suffering from 'Cancer' disease

```
SELECT pat_name
FROM Patient
WHERE disease='Cancer';
```

iv. Change the city of 'Dr. Patil' to Pune

```
UPDATE Doctor
SET city='Pune'
WHERE dname='Dr. Patil';
```

v. List the names of patients that start with alphabet 'A'

```
SELECT pat_name
FROM Patient
WHERE pat_name LIKE 'A%';
```

C. Write a query to find the number of patients suffering from "Asthma" and charged a fee of Rs. 200 by "Dr. Kumar" [4 marks]

```
SELECT COUNT(*) AS Asthma_Patients
FROM Patient P
JOIN Doctor_Patient DP ON P.pno = DP.pno
JOIN Doctor D ON DP.dno = D.dno
WHERE P.disease='Asthma' AND DP.fees=200 AND D.dname='Dr. Kumar';
```

OR (Alternative C)

Create a view to find the name of a doctor who treats maximum number of patients

```
CREATE VIEW TopDoctor AS
SELECT D.dname
FROM Doctor D
JOIN Doctor_Patient DP ON D.dno = DP.dno
GROUP BY D.dname
ORDER BY COUNT(DP.pno) DESC
LIMIT 1;
```

Slip 6 – Student-Teacher (Many-to-Many with Subject)

Q2) Consider the following database (primary keys underlined).
Student (rno, name, city)
Teacher (tno, tname, phone_no, salary)
Relationship: Student–Teacher: many-to-many with subject as a descriptive attribute.

A. Create the above database in PostgreSQL and insert sufficient records

```
CREATE TABLE Student(
    rno INT PRIMARY KEY,
    name VARCHAR(50),
    city VARCHAR(50)
);

CREATE TABLE Teacher(
    tno INT PRIMARY KEY,
    tname VARCHAR(50),
    phone_no VARCHAR(15),
    salary NUMERIC(10,2)
);

CREATE TABLE Student_Teacher(
    rno INT,
    tno INT,
    subject VARCHAR(50),
    PRIMARY KEY(rno, tno, subject),
    FOREIGN KEY(rno) REFERENCES Student(rno),
    FOREIGN KEY(tno) REFERENCES Teacher(tno)
);

-- Insert Sample Data
INSERT INTO Student VALUES
(1,'Ravi','Ahilyanagar'),
(2,'Sneha','Pune'),
(3,'Neha','Ahilyanagar');

INSERT INTO Teacher VALUES
(101,'Prof. Patil','9822000000',50000),
(102,'Mr. Sharma','9822000001',45000),
(103,'Ms. Deshmukh','9822000002',40000);
```

B. Execute the following queries in PostgreSQL (any 3)

i. List the names of students from ‘Ahilyanagar’ city

```
SELECT name FROM Student WHERE city='Ahilyanagar';
```

ii. Display the count of teachers whose salary is between 40000 to 60000

```
SELECT COUNT(*) AS Teacher_Count FROM Teacher WHERE salary BETWEEN 40000 AND
60000;
```

iii. Change the phone number of ‘Prof. Patil’ to ‘9822131226’

```
UPDATE Teacher SET phone_no='9822131226' WHERE tname='Prof. Patil';
```

iv. List the details of the teachers in the sorted order of their name

```
SELECT * FROM Teacher ORDER BY tname;
```

v. List the names of the teachers who have salary less than 50000

```
SELECT tname FROM Teacher WHERE salary < 50000;
```

C. Write a query

Teachers who do not teach any subject taught by 'Mr. Patil'

```
SELECT tname FROM Teacher
WHERE tno NOT IN (
    SELECT tno FROM Student_Teacher WHERE subject IN (
        SELECT subject FROM Student_Teacher WHERE tno = (SELECT tno FROM Teacher
WHERE tname='Mr. Patil')
    )
);
```

OR View – Students who have taken 'DBMS' subject

```
CREATE VIEW DBMS_Students AS
SELECT S.rno, S.name, S.city
FROM Student S
JOIN Student_Teacher ST ON S.rno = ST.rno
WHERE ST.subject='DBMS';
```

Slip 7 – Item-Supplier (Many-to-Many with Rate)

Q2) Consider the following database (primary keys underlined).
Item (itemno, name, quantity)
Supplier (sno, name, city)
Relationship: Item–Supplier: many-to-many with rate as a descriptive attribute.

A. Create the above database in PostgreSQL and insert sufficient records

```
CREATE TABLE Item(
    itemno INT PRIMARY KEY,
    name VARCHAR(50),
    quantity INT
);

CREATE TABLE Supplier(
    sno INT PRIMARY KEY,
    name VARCHAR(50),
    city VARCHAR(50)
);

CREATE TABLE Item_Supplier(
    itemno INT,
```

```
    sno INT,
    rate NUMERIC(10,2),
    PRIMARY KEY(itemno, sno),
    FOREIGN KEY(itemno) REFERENCES Item(itemno),
    FOREIGN KEY(sno) REFERENCES Supplier(sno)
);

-- Sample Data
INSERT INTO Item VALUES
(1,'Mouse',50),
(2,'Keyboard',30),
(3,'Monitor',20);

INSERT INTO Supplier VALUES
(101,'Mahalaxmi Traders','Pune'),
(102,'Mahindra Supply','Mumbai'),
(103,'ABC Suppliers','Pune');

INSERT INTO Item_Supplier VALUES
(1,101,250),
(1,102,240),
(2,103,500),
(3,101,5000);
```

B. Execute the following queries in PostgreSQL (any 3)

i. Change the quantity of item ‘Mouse’ to 80

```
UPDATE Item SET quantity=80 WHERE name='Mouse';
```

ii. List the details of the suppliers whose name begins with the alphabet ‘M’

```
SELECT * FROM Supplier WHERE name LIKE 'M%';
```

iii. Display the total count of items

```
SELECT COUNT(*) AS Total_Items FROM Item;
```

iv. List the names of suppliers who do not live in Pune city

```
SELECT name FROM Supplier WHERE city<>'Pune';
```

v. List the names of items with quantity less than 10

```
SELECT name FROM Item WHERE quantity<10;
```

C. Write a query

Display the name of supplier who supplies Item with minimum rate

```
SELECT s.name
FROM Supplier s
JOIN Item_Supplier i_s ON s.sno = i_s.sno
WHERE i_s.rate = (SELECT MIN(rate) FROM Item_Supplier);
```

OR View – Supplier-wise list of items

```
CREATE VIEW Supplier_Items AS
SELECT s.name AS Supplier, i.name AS Item, i_s.rate
FROM Supplier s
JOIN Item_Supplier i_s ON s.sno = i_s.sno
JOIN Item i ON i.itemno = i_s.itemno;
```

Slip 8 – Student-Teacher (Many-to-Many with Subject)

Q2) Consider the following database (primary keys underlined).
Student (sno, s_name, s_class)
Teacher (tno, t_name, yrs_experience)
Relationship: Student–Teacher: many-to-many with descriptive attribute Subject.
Constraint: s_class = "FY", "SY", "TY".

A. Create Database

```
CREATE TABLE Student(
    sno INT PRIMARY KEY,
    s_name VARCHAR(50),
    s_class VARCHAR(2) CHECK (s_class IN ('FY','SY','TY'))
);

CREATE TABLE Teacher(
    tno INT PRIMARY KEY,
    t_name VARCHAR(50),
    yrs_experience INT
);

CREATE TABLE Student_Teacher(
    sno INT,
    tno INT,
    subject VARCHAR(50),
    PRIMARY KEY(sno, tno, subject),
    FOREIGN KEY(sno) REFERENCES Student(sno),
    FOREIGN KEY(tno) REFERENCES Teacher(tno)
);

-- Sample Data
INSERT INTO Student VALUES
(101,'Ravi','FY'),
(102,'Sneha','TY'),
(103,'Neha','TY'),
(104,'Amit','SY');

INSERT INTO Teacher VALUES
(201,'Prof. Patil',10),
(202,'Mr. Sharma',8),
(203,'Ms. Deshmukh',5);
```

B. Queries (any 3)

```
-- i. Class-wise number of students
SELECT s_class, COUNT(*) AS No_of_Students
FROM Student
GROUP BY s_class;
```

```
-- ii. Students in class 'TY'
SELECT * FROM Student WHERE s_class='TY';

-- iii. Count of students who have taken 'Maths'
SELECT COUNT(*) AS Maths_Students
FROM Student_Teacher
WHERE subject='Maths';

-- iv. Delete student sno 105
DELETE FROM Student WHERE sno=105;

-- v. Add designation column to Teacher
ALTER TABLE Teacher ADD COLUMN designation VARCHAR(50);
```

C. Queries

Teachers with subjects and total students

```
SELECT t.t_name, st.subject, COUNT(st.sno) AS Total_Students
FROM Teacher t
JOIN Student_Teacher st ON t.tno = st.tno
GROUP BY t.t_name, st.subject;
```

OR View – Teacher teaching maximum subjects

```
CREATE VIEW Max_Subject_Teacher AS
SELECT t_name
FROM Teacher t
JOIN Student_Teacher st ON t.tno = st.tno
GROUP BY t_name
ORDER BY COUNT(DISTINCT st.subject) DESC
LIMIT 1;
```

Slip 9 – Person-Area (One-to-Many)

Q2) Consider the following database (primary keys underlined).
Person (pnumber, pname, birthdate, income)
Area (aname, area_type)
Relationship: Person–Area: many-to-one.
Constraint: area_type = ‘urban’ or ‘rural’.

A. Create Database

```
CREATE TABLE Area(
    aname VARCHAR(50) PRIMARY KEY,
    area_type VARCHAR(10) CHECK (area_type IN ('urban','rural'))
);

CREATE TABLE Person(
    pnumber INT PRIMARY KEY,
    pname VARCHAR(50),
    birthdate DATE,
    income NUMERIC(10,2),
```

```
    aname VARCHAR(50),
    FOREIGN KEY(aname) REFERENCES Area(aname)
);

-- Sample Data
INSERT INTO Area VALUES
('Camp','urban'),
('Kalyaninagar','urban'),
('Wadgaon','rural');

INSERT INTO Person VALUES
(1,'Ramesh','1990-07-15',12000,'Camp'),
(2,'Sita','1992-02-10',15000,'Kalyaninagar'),
(3,'Anil','1995-07-20',9000,'Camp');
```

B. Queries (any 3)

```
-- i. Persons with income > 10000
SELECT * FROM Person WHERE income > 10000;

-- ii. Transfer all people from 'Pune' to 'Delhi'
UPDATE Person SET aname='Delhi' WHERE aname='Pune';

-- iii. Count number of urban areas
SELECT COUNT(*) AS Urban_Areas FROM Area WHERE area_type='urban';

-- iv. Count of people born in February
SELECT COUNT(*) AS Feb_Birthdays
FROM Person
WHERE EXTRACT(MONTH FROM birthdate)=2;

-- v. Names of all people living in rural area
SELECT pname FROM Person P JOIN Area A ON P.aname=A.aname WHERE
A.area_type='rural';
```

C. Queries

Count of people in urban area with income equal to average income

```
SELECT COUNT(*) FROM Person
WHERE aname IN (SELECT aname FROM Area WHERE area_type='urban')
AND income = (SELECT AVG(income) FROM Person);
```

OR View – Person details with area name

```
CREATE VIEW Person_Area_View AS
SELECT p.pnumber, p.pname, p.income, a.aname, a.area_type
FROM Person p
JOIN Area a ON p.aname=a.aname;
```

Slip 10 – Account-Customer (One-to-Many)

Q2) Consider the following database (primary keys underlined).
Account (acctno, acct_type, balance, branch_name)
Customer (custno, cust_name, cust_city)

Relationship: Customer–Account: One-to-Many
Constraint: acct_type = “saving” or “current”, balance > 0

A. Create Database

```
CREATE TABLE Customer(
    custno INT PRIMARY KEY,
    cust_name VARCHAR(50),
    cust_city VARCHAR(50)
);

CREATE TABLE Account(
    acctno INT PRIMARY KEY,
    acct_type VARCHAR(10) CHECK(acct_type IN ('saving','current')),
    balance NUMERIC(10,2) CHECK(balance>0),
    branch_name VARCHAR(50),
    custno INT,
    FOREIGN KEY(custno) REFERENCES Customer(custno)
);

-- Sample Data
INSERT INTO Customer VALUES
(1,'Ramnath','Pune'),
(2,'Sanjay','Mumbai'),
(3,'Meena','Pune');

INSERT INTO Account VALUES
(101,'saving',6000,'M.G.Road',1),
(102,'current',15000,'M.G.Road',2),
(103,'saving',4000,'Kothrud',3);
```

B. Queries (any 3)

```
-- i. Display all saving accounts with balance > 5000
SELECT * FROM Account WHERE acct_type='saving' AND balance>5000;

-- ii. Count customers in city 'Pune'
SELECT COUNT(*) AS Pune_Customers FROM Customer WHERE cust_city='Pune';

-- iii. Total balance at branch 'M.G.Road'
SELECT SUM(balance) AS Total_Balance FROM Account WHERE branch_name='M.G.Road';

-- iv. Delete record of customer 'Ramnath'
DELETE FROM Customer WHERE cust_name='Ramnath';

-- v. Change city of customer 'Sanjay' to 'Pune'
UPDATE Customer SET cust_city='Pune' WHERE cust_name='Sanjay';
```

C. Queries

Number of account holders for each city

```
SELECT cust_city, COUNT(acctno) AS Total_Accounts
FROM Customer c
JOIN Account a ON c.custno=a.custno
```

```
GROUP BY cust_city;
```

OR View – Account details with customer and branch info

```
CREATE VIEW Account_Cust_View AS
SELECT a.acctno, a.acct_type, a.balance, a.branch_name, c.cust_name, c.cust_city
FROM Account a
JOIN Customer c ON a.custno=c.custno;
```

Slip 11 – Emp–Project (Many-to-Many)

Q2) Consider the following database (primary keys underlined).
Emp (eno, ename, salary)
Project (pno, pname, budget)
Relationship: Emp–Project: Many-to-Many with descriptive attribute total_hours

---

A. Create Database

```
CREATE TABLE Emp(
    eno INT PRIMARY KEY,
    ename VARCHAR(50),
    salary NUMERIC(10,2)
);

CREATE TABLE Project(
    pno INT PRIMARY KEY,
    pname VARCHAR(50),
    budget NUMERIC(10,2)
);

CREATE TABLE Emp_Project(
    eno INT,
    pno INT,
    total_hours INT,
    PRIMARY KEY(eno, pno),
    FOREIGN KEY(eno) REFERENCES Emp(eno),
    FOREIGN KEY(pno) REFERENCES Project(pno)
);

-- Sample Data
INSERT INTO Emp VALUES
(1,'Ramesh',50000),
(2,'Sita',45000),
(3,'Anil',40000);

INSERT INTO Project VALUES
(101,'ERP',100000),
(102,'Website',50000);

INSERT INTO Emp_Project VALUES
(1,101,120),
(2,102,80);
```

B. Queries (any 3)

```
-- i. Maximum budget
SELECT MAX(budget) AS Max_Budget FROM Project;

-- ii. Increase salary of all employees by 10%
UPDATE Emp SET salary = salary * 1.10;

-- iii. Count projects with duration > 100 hrs
SELECT COUNT(*) AS Projects_Over_100_Hrs FROM Emp_Project WHERE total_hours>100;

-- iv. Employees whose name ends with 'a'
SELECT * FROM Emp WHERE ename LIKE '%a';

-- v. Add column contact_number to Emp
ALTER TABLE Emp ADD COLUMN contact_number VARCHAR(15);
```

C. Queries

Employees not working on any project

```
SELECT * FROM Emp
WHERE eno NOT IN (SELECT eno FROM Emp_Project);
```

OR View – Project details with maximum hours

```
CREATE VIEW Max_Hours_Project AS
SELECT p.pno, p.pname, MAX(ep.total_hours) AS Max_Hours
FROM Project p
JOIN Emp_Project ep ON p.pno = ep.pno
GROUP BY p.pno, p.pname;
```

Slip 12 – Sales_order–Client (One-to-Many)

Q2) Consider the following database (primary keys underlined).
Sales_order (sorderno, s_order_date, order_amt)
Client (clientno, name, address)
Relationship: Client–Sales_order: One-to-Many
Constraint: order_amt > 0

A. Create Database

```
CREATE TABLE Client(
    clientno INT PRIMARY KEY,
    name VARCHAR(50),
    address VARCHAR(100)
);

CREATE TABLE Sales_order(
    sorderno INT PRIMARY KEY,
    s_order_date DATE,
    order_amt NUMERIC(10,2) CHECK(order_amt>0),
    clientno INT,
    FOREIGN KEY(clientno) REFERENCES Client(clientno)
);
```

```
-- Sample Data
INSERT INTO Client VALUES
(1,'Ramesh','Nagpur'),
(2,'Sita','Pune');

INSERT INTO Sales_order VALUES
(101,'2024-02-20',5000,1),
(102,'2024-02-22',10000,2);
```

B. Queries (any 3)

```
-- i. Update client address from 'Nagpur' to 'Kolhapur'
UPDATE Client SET address='Kolhapur' WHERE address='Nagpur';

-- ii. Delete sales order where clientno=30
DELETE FROM Sales_order WHERE clientno=30;

-- iii. Display all sales orders before '2024-02-23'
SELECT * FROM Sales_order WHERE s_order_date<'2024-02-23';

-- iv. Display sales order with maximum amount
SELECT * FROM Sales_order ORDER BY order_amt DESC LIMIT 1;

-- v. Add column order_status
ALTER TABLE Sales_order ADD COLUMN order_status VARCHAR(20);
```

C. Queries

Client details having maximum sales orders

```
SELECT c.clientno, c.name, COUNT(s.sorderno) AS Total_Orders
FROM Client c
JOIN Sales_order s ON c.clientno=s.clientno
GROUP BY c.clientno, c.name
ORDER BY Total_Orders DESC
LIMIT 1;
```

OR View – Clients with more than 2 sales orders

```
CREATE VIEW Top_Clients AS
SELECT c.clientno, c.name, COUNT(s.sorderno) AS Total_Orders
FROM Client c
JOIN Sales_order s ON c.clientno=s.clientno
GROUP BY c.clientno, c.name
HAVING COUNT(s.sorderno)>2;
```

Slip 13 – Bus-Route (Many-to-One)

Q2) Consider the following database (primary keys underlined).
Bus (Busno, capacity, depot_name)
Route (Routeno, source, destination, no_of_stations)
Relationship: Bus–Route: Many-to-One
Constraint: Bus capacity not null

A. Create Database

```
CREATE TABLE Route(
    Routeno INT PRIMARY KEY,
    source VARCHAR(50),
    destination VARCHAR(50),
    no_of_stations INT
);

CREATE TABLE Bus(
    Busno INT PRIMARY KEY,
    capacity INT NOT NULL,
    depot_name VARCHAR(50),
    Routeno INT,
    FOREIGN KEY(Routeno) REFERENCES Route(Routeno)
);

-- Sample Data
INSERT INTO Route VALUES
(41,'Pune station','Hadapsar',12),
(42,'Swargate','Shivajinagar',8);

INSERT INTO Bus VALUES
(201,50,'Swargate',42),
(204,40,'Pimpri',41);
```

B. Queries (any 3)

```
-- i. All buses at depot 'Swargate'
SELECT * FROM Bus WHERE depot_name='Swargate';

-- ii. Delete Bus no 204
DELETE FROM Bus WHERE Busno=204;

-- iii. List all buses on route 41
SELECT * FROM Bus WHERE Routeno=41;

-- iv. Routes with stations > 10
SELECT * FROM Route WHERE no_of_stations>10;

-- v. Routes starting from 'Pune station'
SELECT * FROM Route WHERE source='Pune station';
```

C. Queries

Delete buses on routes with stations < 3

```
DELETE FROM Bus WHERE Routeno IN (SELECT Routeno FROM Route WHERE
no_of_stations<3);
```

OR View – Buses and route details from 'Pimpri' to 'Hadapsar'

```
CREATE VIEW Pimpri_Hadapsar_Buses AS
SELECT b.Busno, b.capacity, b.depot_name, r.source, r.destination,
r.no_of_stations
```

```
FROM Bus b
JOIN Route r ON b.Routeno=r.Routeno
WHERE r.source='Pimpri' AND r.destination='Hadapsar';
```

Slip 14 – Doctor–Patient (One-to-Many)

Q2) Consider the following database (primary keys underlined).
Doctor (dcode, doctor_name, specialization, address, phone_no, fees)
Patient (pcode, patient_name, symptoms)
Relationship: Doctor–Patient: One-to-Many

A. Create Database

```
CREATE TABLE Doctor(
    dcode INT PRIMARY KEY,
    doctor_name VARCHAR(50),
    specialization VARCHAR(50),
    address VARCHAR(100),
    phone_no VARCHAR(15),
    fees NUMERIC(10,2)
);

CREATE TABLE Patient(
    pcode INT PRIMARY KEY,
    patient_name VARCHAR(50),
    symptoms VARCHAR(100),
    dcode INT,
    FOREIGN KEY(dcode) REFERENCES Doctor(dcode)
);

-- Sample Data
INSERT INTO Doctor VALUES
(1,'Dr. Patil','Neurologist','Sadashiv Peth','9876543210',500),
(2,'Dr. Kumar','Orthopedic','Shivajinagar','9876543220',400);

INSERT INTO Patient VALUES
(101,'Manish','Fever',1),
(102,'Meena','Asthama',2),
(103,'Mohan','Fever',1);
```

B. Queries (any 3)

```
-- i. Patients whose name starts with 'M'
SELECT * FROM Patient WHERE patient_name LIKE 'M%';

-- ii. Count doctors who are Neurologists
SELECT COUNT(*) AS Neurologists FROM Doctor WHERE specialization='Neurologist';

-- iii. List all patients suffering from 'Fever'
SELECT * FROM Patient WHERE symptoms='Fever';

-- iv. Specialization and phone numbers of doctors from 'Sadashiv Peth'
SELECT specialization, phone_no FROM Doctor WHERE address='Sadashiv Peth';

-- v. Change address of Dr. Patil to 'Camp'
```

```
UPDATE Doctor SET address='Camp' WHERE doctor_name='Dr. Patil';
```

C. Queries

Patients treated by doctors in 'Shivajinagar' with specialization 'Orthopedic'

```
SELECT p.patient_name, d.doctor_name, d.specialization
FROM Patient p
JOIN Doctor d ON p.dcode=d.dcode
WHERE d.address='Shivajinagar' AND d.specialization='Orthopedic';
```

OR View – Patients treated by doctor with minimum fees

```
CREATE VIEW Min_Fees_Patients AS
SELECT p.patient_name, d.doctor_name, d.fees
FROM Patient p
JOIN Doctor d ON p.dcode=d.dcode
WHERE d.fees = (SELECT MIN(fees) FROM Doctor);
```

Slip 15 – Game–Player (One-to-Many)

Q2) Consider the following database (primary keys underlined).
Game (gname, noofplayers, coachname, captain_name)
Player (pno, pname)
Relationship: Game–Player: One-to-Many
Constraint: captain_name should be uppercase

A. Create Database

```
CREATE TABLE Game(
    gname VARCHAR(50) PRIMARY KEY,
    noofplayers INT,
    coachname VARCHAR(50),
    captain_name VARCHAR(50) CHECK(captain_name = UPPER(captain_name))
);

CREATE TABLE Player(
    pno INT PRIMARY KEY,
    pname VARCHAR(50),
    gname VARCHAR(50),
    FOREIGN KEY(gname) REFERENCES Game(gname)
);

-- Sample Data
INSERT INTO Game VALUES
('Cricket',11,'Dhoni','VIROSH'),
('Hockey',16,'Singh','RAJ');

INSERT INTO Player VALUES
(1,'Ramesh','Cricket'),
(2,'Sita','Hockey'),
(3,'Anil','Hockey');
```

B. Queries (any 3)

```
-- i. Names of players playing 'Hockey'
SELECT pname FROM Player WHERE gname='Hockey';

-- ii. Average number of players
SELECT AVG(noofplayers) AS Avg_Players FROM Game;

-- iii. Delete records of players playing 'Kho Kho'
DELETE FROM Player WHERE gname='Kho Kho';

-- iv. Names of players not playing 'Cricket'
SELECT pname FROM Player WHERE gname<>'Cricket';

-- v. Update coach name from 'Dhoni' to 'Tendulkar' for 'Cricket'
UPDATE Game SET coachname='Tendulkar' WHERE gname='Cricket';
```

C. Queries

Game details with maximum number of players

```
SELECT * FROM Game ORDER BY noofplayers DESC LIMIT 1;
```

OR View – Game-wise list of players along with game name

```
CREATE VIEW Game_Player_View AS
SELECT g.gname, g.coachname, g.captain_name, p.pname
FROM Game g
JOIN Player p ON g.gname = p.gname;
```

Slip 16

Q2) Database: Student-Teacher
Tables:

- Student(sno, s_name, s_class)
- Teacher(tno, t_name, yrs_experience)
  Relationship: Many-to-Many with descriptive attribute Subject.
  Constraint: s_class = "FY", "SY", "TY"

A. Create database and insert records

```
CREATE TABLE Student(
    sno INT PRIMARY KEY,
    s_name VARCHAR(50),
    s_class VARCHAR(10) CHECK(s_class IN ('FY','SY','TY'))
);

CREATE TABLE Teacher(
    tno INT PRIMARY KEY,
    t_name VARCHAR(50),
    yrs_experience INT
);

CREATE TABLE Student_Teacher(
    sno INT REFERENCES Student(sno),
    tno INT REFERENCES Teacher(tno),
```

```
    subject VARCHAR(50),
    PRIMARY KEY(sno,tno,subject)
);

-- Insert sample data
INSERT INTO Student VALUES (1,'Ravi','SY'),(2,'Anita','FY'),(3,'Meena','TY');
INSERT INTO Teacher VALUES (1,'Dr. Pawar',15),(2,'Dr. Wani',10);
INSERT INTO Student_Teacher VALUES (1,1,'Data
Structure'),(2,2,'DBMS'),(3,2,'Java');
```

B. Queries (any 3)

```
-- i. Teachers with more than 5 years of experience
SELECT * FROM Teacher WHERE yrs_experience > 5;

-- ii. Count all students who have taken subject "Data Structure"
SELECT COUNT(sno) FROM Student_Teacher WHERE subject='Data Structure';

-- iii. Names of all students from SY class
SELECT s_name FROM Student WHERE s_class='SY';

-- iv. Change experience of teacher named "Pawar" to 20
UPDATE Teacher SET yrs_experience=20 WHERE t_name='Dr. Pawar';

-- v. Display classwise details of students
SELECT s_class, COUNT(*) AS total_students FROM Student GROUP BY s_class;
```

C. Query options

```
-- 1. List all teachers with their subjects and student names
SELECT T.t_name, ST.subject, S.s_name
FROM Teacher T
JOIN Student_Teacher ST ON T.tno = ST.tno
JOIN Student S ON S.sno = ST.sno;

-- 2. Create view of teacher teaching maximum subjects
CREATE VIEW MaxSubjectsTeacher AS
SELECT tno, t_name, COUNT(subject) AS total_subjects
FROM Student_Teacher ST JOIN Teacher T ON ST.tno=T.tno
GROUP BY tno,t_name
ORDER BY total_subjects DESC
LIMIT 1;
```

Slip 17

Q2) Database: Property-Owner
Tables:

- Property(Pno, area_Sqft, location, city)
- Owner(oname, address, phone)
  Relationship: Many-to-One

A. Create database and insert records

```
CREATE TABLE Owner(
    oname VARCHAR(50) PRIMARY KEY,
    address VARCHAR(100),
    phone VARCHAR(15)
```

```
);

CREATE TABLE Property(
    pno INT PRIMARY KEY,
    area_sqft INT,
    location VARCHAR(50),
    city VARCHAR(50),
    oname VARCHAR(50) REFERENCES Owner(oname)
);

-- Sample Inserts
INSERT INTO Owner VALUES ('Mr.Rahane','Street 1','9876543210');
INSERT INTO Owner VALUES ('Mr.Baravkar','Street 2','9876543211');

INSERT INTO Property VALUES (1,2500,'Kothrud','Pune','Mr.Baravkar'),
                            (2,1500,'Deolali','Nashik','Mr.Rahane');
```

B. Queries (any 3)

```
-- i. Properties in Nashik
SELECT * FROM Property WHERE city='Nashik';

-- ii. Names of owners with property > 2000 sqft
SELECT DISTINCT oname FROM Property WHERE area_sqft>2000;

-- iii. Count properties owned by Mr.Rahane
SELECT COUNT(*) FROM Property WHERE oname='Mr.Rahane';

-- iv. Count properties in Pune
SELECT COUNT(*) FROM Property WHERE city='Pune';

-- v. Delete properties in Pune owned by Mr.Baravkar
DELETE FROM Property WHERE city='Pune' AND oname='Mr.Baravkar';
```

C. Query options

```
-- 1. Owners who own property in Pune, Kothrud >10000 sqft
SELECT DISTINCT oname FROM Property WHERE city='Pune' AND location='Kothrud' AND
area_sqft>10000;

-- 2. Create view for owner-wise list of properties
CREATE VIEW OwnerProperties AS
SELECT oname, pno, area_sqft, location, city FROM Property;
```

Slip 18

Q2) Database: Student-Teacher
Tables:

- Student(rno,name,city)
- Teacher(tno,tname,phone_no,salary)
  Relationship: Many-to-Many with descriptive attribute Subject

A. Create database and insert records

```
CREATE TABLE Student(rno INT PRIMARY KEY, name VARCHAR(50), city VARCHAR(50));
CREATE TABLE Teacher(tno INT PRIMARY KEY, tname VARCHAR(50), phone_no
VARCHAR(15), salary INT);
```

```
CREATE TABLE Student_Teacher(rno INT REFERENCES Student(rno), tno INT REFERENCES
Teacher(tno), subject VARCHAR(50), PRIMARY KEY(rno,tno,subject));

-- Sample Inserts
INSERT INTO Student VALUES (1,'Ravi','Sangamner'),(2,'Meena','Pune');
INSERT INTO Teacher VALUES (1,'Dr. Mulay','9876543210',50000),(2,'Dr.
Wani','9876543211',60000);
INSERT INTO Student_Teacher VALUES (1,1,'IKS'),(2,2,'Electronics');
```

B. Queries (any 3)

```
-- i. Students from Sangamner
SELECT * FROM Student WHERE city='Sangamner';

-- ii. Maximum salary of teachers
SELECT MAX(salary) FROM Teacher;

-- iii. Change phone number of Dr. Mulay
UPDATE Teacher SET phone_no='9834233235' WHERE tname='Dr. Mulay';

-- iv. Teacher details sorted by name
SELECT * FROM Teacher ORDER BY tname;

-- v. Count of students who took subject IKS
SELECT COUNT(*) FROM Student_Teacher WHERE subject='IKS';
```

C. Query options

```
-- 1. Students taught by Dr. Wani
SELECT S.name FROM Student S
JOIN Student_Teacher ST ON S.rno = ST.rno
JOIN Teacher T ON T.tno = ST.tno
WHERE T.tname='Dr. Wani';

-- 2. Create view of students who have not taken Electronics
CREATE VIEW Students_Not_Electronics AS
SELECT S.* FROM Student S
WHERE S.rno NOT IN (
  SELECT rno FROM Student_Teacher WHERE subject='Electronics'
);
```

Slip 19

Q2) Database: Order-Client
Tables:

- Order(orderno, s_order_date, amount)
- Client(cno, name, address)
  Relationship: One-to-Many (one client → many orders)
  Constraint: amount > 0

A. Create database and insert records

```
CREATE TABLE Client(
    cno INT PRIMARY KEY,
    name VARCHAR(50),
    address VARCHAR(100)
);
```

```
CREATE TABLE Order(
    orderno INT PRIMARY KEY,
    s_order_date DATE,
    amount INT CHECK(amount>0),
    cno INT REFERENCES Client(cno)
);

-- Sample Inserts
INSERT INTO Client VALUES (1,'Ravi','Mumbai'),(2,'Anita','Pune');
INSERT INTO Order VALUES (101,'2019-10-10',35000,1),(102,'2019-10-15',45000,2);
```

B. Queries (any 3)

```
-- i. Change order date of client number 04
UPDATE Order SET s_order_date='2019-04-12' WHERE cno=4;

-- ii. Display all orders given in October
SELECT * FROM Order WHERE EXTRACT(MONTH FROM s_order_date)=10;

-- iii. Display all orders in descending order of amount
SELECT * FROM Order ORDER BY amount DESC;

-- iv. Delete all clients of Mumbai
DELETE FROM Client WHERE address='Mumbai';

-- v. Count orders having amount > 40000
SELECT COUNT(*) FROM Order WHERE amount>40000;
```

C. Query options

```
-- 1. Client name who placed maximum orders
SELECT name FROM Client C
JOIN Order O ON C.cno=O.cno
GROUP BY C.name
ORDER BY COUNT(O.orderno) DESC
LIMIT 1;

-- 2. View of clients having more than 2 sales orders
CREATE VIEW Client_More2Orders AS
SELECT C.name, COUNT(O.orderno) AS total_orders
FROM Client C
JOIN Order O ON C.cno=O.cno
GROUP BY C.name
HAVING COUNT(O.orderno)>2;
```

Slip 20

Q2) Database: Customer-Account
Tables:

- Customer(cust_no, cust_name, cust_city)
- Account(acc_no, acc_type, balance)
  Relationship: One-to-Many
  Constraint: acc_type = 'Saving' or 'Current'

A. Create database and insert records

```
CREATE TABLE Customer(
```

```
    cust_no INT PRIMARY KEY,
    cust_name VARCHAR(50),
    cust_city VARCHAR(50)
);

CREATE TABLE Account(
    acc_no INT PRIMARY KEY,
    acc_type VARCHAR(10) CHECK(acc_type IN ('Saving','Current')),
    balance INT CHECK(balance>0),
    cust_no INT REFERENCES Customer(cust_no)
);

-- Sample Inserts
INSERT INTO Customer VALUES (101,'Ravi','Delhi'),(102,'Anita','Mumbai');
INSERT INTO Account VALUES
(1001,'Saving',5000,101),(1002,'Current',1500000,102);
```

B. Queries (any 3)

```
-- i. Customers in Delhi
SELECT cust_name FROM Customer WHERE cust_city='Delhi';

-- ii. Total balance of all Saving accounts
SELECT SUM(balance) FROM Account WHERE acc_type='Saving';

-- iii. Maximum balance
SELECT MAX(balance) FROM Account;

-- iv. Increase balance of account no 101123 by 10000
UPDATE Account SET balance = balance + 10000 WHERE acc_no=101123;

-- v. Add column AADHAR number
ALTER TABLE Customer ADD COLUMN aadhar_no VARCHAR(12);
```

C. Query options

```
-- 1. Current account customers with balance > 1000000
SELECT cust_name, cust_city FROM Customer C
JOIN Account A ON C.cust_no=A.cust_no
WHERE A.acc_type='Current' AND balance>1000000;

-- 2. View: Customers with saving accounts not in Ahilyanagar
CREATE VIEW SavingNotAhilyanagar AS
SELECT C.* FROM Customer C
JOIN Account A ON C.cust_no=A.cust_no
WHERE A.acc_type='Saving' AND C.cust_city<>'Ahilyanagar';
```

Slip 21

Q2) Database: Bus-Driver
Tables:

- Bus(bno, capacity, depot_name)
- Driver(dno, driver_name, license_no, address, age, salary)
  Relationship: Many-to-Many with descriptive attribute Date_of_duty

A. Create database and insert records

```
CREATE TABLE Bus(
    bno INT PRIMARY KEY,
    capacity INT NOT NULL,
    depot_name VARCHAR(50)
);

CREATE TABLE Driver(
    dno INT PRIMARY KEY,
    driver_name VARCHAR(50),
    license_no VARCHAR(15),
    address VARCHAR(50),
    age INT,
    salary INT
);

CREATE TABLE Bus_Driver(
    bno INT REFERENCES Bus(bno),
    dno INT REFERENCES Driver(dno),
    date_of_duty DATE,
    PRIMARY KEY(bno,dno,date_of_duty)
);

-- Sample Inserts
INSERT INTO Bus VALUES (101,30,'Kothrud Depot'),(102,20,'Swargate');
INSERT INTO Driver VALUES (1,'Swapnil','DL1234','Pune',24,20000);
INSERT INTO Bus_Driver VALUES (101,1,'2009-02-12');
```

B. Queries (any 3)

```
-- i. Number of buses with capacity >20
SELECT COUNT(*) FROM Bus WHERE capacity>20;

-- ii. Drivers who did duty on 15/01/2017
SELECT COUNT(*) FROM Bus_Driver WHERE date_of_duty='2017-01-15';

-- iii. License numbers of drivers starting with 'S'
SELECT license_no FROM Driver WHERE driver_name LIKE 'S%';

-- iv. Average salary of drivers
SELECT AVG(salary) FROM Driver;

-- v. Details of driver Swapnil
SELECT * FROM Driver WHERE driver_name='Swapnil';
```

C. Query options

```
-- 1. Drivers driving buses from Kothrud Depot on 12/02/2009
SELECT D.driver_name FROM Driver D
JOIN Bus_Driver BD ON D.dno=BD.dno
JOIN Bus B ON B.bno=BD.bno
WHERE B.depot_name='Kothrud Depot' AND BD.date_of_duty='2009-02-12';

-- 2. View of buses driven by drivers aged <25
CREATE VIEW YoungDriverBuses AS
SELECT B.* FROM Bus B
JOIN Bus_Driver BD ON B.bno=BD.bno
JOIN Driver D ON D.dno=BD.dno
WHERE D.age<25;
```

Slip 22

Q2) Database: Employee-Department
Tables:

- Employee(eno, ename, designation, salary)
- Department(dno, dname, location)
  Relationship: One-to-Many
  Constraint: salary > 5000

A. Create database and insert records

```
CREATE TABLE Department(
    dno INT PRIMARY KEY,
    dname VARCHAR(50),
    location VARCHAR(50)
);

CREATE TABLE Employee(
    eno INT PRIMARY KEY,
    ename VARCHAR(50),
    designation VARCHAR(50),
    salary INT CHECK(salary>5000),
    dno INT REFERENCES Department(dno)
);

-- Sample Inserts
INSERT INTO Department VALUES (1,'HR','Nanded'),(2,'IT','Pune');
INSERT INTO Employee VALUES
(101,'Sagar','Clerk',15000,1),(102,'Anita','Staff',12000,2);
```

B. Queries (any 3)

```
-- i. Change designation of Sagar
UPDATE Employee SET designation='Manager' WHERE ename='Sagar';

-- ii. Minimum salary
SELECT MIN(salary) FROM Employee;

-- iii. Departments located at Nanded
SELECT * FROM Department WHERE location='Nanded';

-- iv. Employee names starting with A
SELECT * FROM Employee WHERE ename LIKE 'A%';

-- v. Department-wise count of employees
SELECT dno, COUNT(*) AS total_emp FROM Employee GROUP BY dno;
```

C. Query options

```
-- 1. Department name-wise total salary
SELECT D.dname, SUM(E.salary) AS total_salary
FROM Employee E JOIN Department D ON E.dno=D.dno
GROUP BY D.dname;

-- 2. View: Department with maximum employees
CREATE VIEW MaxEmployeesDept AS
SELECT dno, COUNT(*) AS total_emp
FROM Employee
GROUP BY dno
ORDER BY total_emp DESC
LIMIT 1;
```

Slip 23

Q2) Database: Customer-Account
Tables:

- Customer(cno, cust_name, cust_city)
- Account(ano, acc_type, balance)
  Relationship: One-to-Many
  Constraint: acc_type = 'Saving' or 'Current'

A. Create database and insert records

```
CREATE TABLE Customer(
    cno INT PRIMARY KEY,
    cust_name VARCHAR(50),
    cust_city VARCHAR(50)
);

CREATE TABLE Account(
    ano INT PRIMARY KEY,
    acc_type VARCHAR(10) CHECK(acc_type IN ('Saving','Current')),
    balance INT,
    cno INT REFERENCES Customer(cno)
);

-- Sample Inserts
INSERT INTO Customer VALUES (101,'Sahil','Pune'),(102,'Ravi','Mumbai');
INSERT INTO Account VALUES
(1001,'Saving',50000,101),(1002,'Current',150000,102);
```

B. Queries (any 3)

```
-- i. Customers with name containing "in"
SELECT * FROM Customer WHERE cust_name LIKE '%in%';

-- ii. Average balance of Current accounts
SELECT AVG(balance) FROM Account WHERE acc_type='Current';

-- iii. Account details sorted by balance
SELECT * FROM Account ORDER BY balance;

-- iv. Increase balance of Sahil by 10000
UPDATE Account SET balance=balance+10000 WHERE cno=(SELECT cno FROM Customer
WHERE cust_name='Sahil');

-- v. Add phone_no column
ALTER TABLE Customer ADD COLUMN phone_no VARCHAR(15);
```

C. Query options

```
-- 1. City-wise customer names having minimum balance
SELECT cust_city, cust_name FROM Customer C
JOIN Account A ON C.cno=A.cno
WHERE balance = (SELECT MIN(balance) FROM Account WHERE cno=C.cno);

-- 2. View: Customers in Pune having maximum saving balance
CREATE VIEW PuneMaxSaving AS
```

```
SELECT C.*, A.balance FROM Customer C
JOIN Account A ON C.cno=A.cno
WHERE C.cust_city='Pune' AND A.acc_type='Saving'
ORDER BY A.balance DESC
LIMIT 1;
```

Slip 24

Q2) Database: Bus-Route
Tables:

- Bus(Busno, capacity, depot_name)
- Route(Routeno, source, destination, no_of_stations)
  Relationship: Many-to-One
  Constraint: Bus capacity NOT NULL

A. Create database and insert records

```
CREATE TABLE Route(
    Routeno INT PRIMARY KEY,
    source VARCHAR(50),
    destination VARCHAR(50),
    no_of_stations INT
);

CREATE TABLE Bus(
    Busno INT PRIMARY KEY,
    capacity INT NOT NULL,
    depot_name VARCHAR(50),
    Routeno INT REFERENCES Route(Routeno)
);

-- Sample Inserts
INSERT INTO Route VALUES
(41,'Pimpri','Vimannagar',10),(42,'Sangamner','Baramati',5);
INSERT INTO Bus VALUES (101,30,'Depot1',41),(102,15,'Depot2',42);
```

B. Queries (any 3)

```
-- i. Count total buses at Sangamner
SELECT COUNT(*) FROM Bus WHERE depot_name='Sangamner';

-- ii. Delete buses with capacity <20
DELETE FROM Bus WHERE capacity<20;

-- iii. List all buses on route 41
SELECT * FROM Bus WHERE Routeno=41;

-- iv. Maximum number of stations
SELECT MAX(no_of_stations) FROM Route;

-- v. Routes from Pimpri to Vimannagar
SELECT * FROM Route WHERE source='Pimpri' AND destination='Vimannagar';
```

C. Query options

```
-- 1. Delete all buses with destination Baramati
```

```
DELETE FROM Bus WHERE Routeno IN (SELECT Routeno FROM Route WHERE
destination='Baramati');

-- 2. View: Route with maximum stations and buses on that route
CREATE VIEW MaxStationRoute AS
SELECT R.Routeno, R.source, R.destination, R.no_of_stations, B.Busno
FROM Route R
JOIN Bus B ON R.Routeno=B.Routeno
ORDER BY R.no_of_stations DESC
LIMIT 1;
```

Slip 25

Q2) Database: Employee-Project
Tables:

- Employee(empno, emp_name, city, designation, salary)
- Project(projno, proj_name, status, start_date)
  Relationship: Many-to-One
  Constraint: status = "Complete" or "In progress"

A. Create database and insert records

```
CREATE TABLE Project(
    projno INT PRIMARY KEY,
    proj_name VARCHAR(50),
    status VARCHAR(20) CHECK(status IN ('Complete','In progress')),
    start_date DATE
);

CREATE TABLE Employee(
    empno INT PRIMARY KEY,
    emp_name VARCHAR(50),
    city VARCHAR(50),
    designation VARCHAR(50),
    salary INT,
    projno INT REFERENCES Project(projno)
);

-- Sample Inserts
INSERT INTO Project VALUES (10,'Robotics','In progress','2025-01-
01'),(105,'AI','Complete','2024-05-01');
INSERT INTO Employee VALUES
(101,'Ravi','Pune','Developer',50000,10),(102,'Anita','Mumbai','Tester',45000,10
5);
```

B. Queries (any 3)

```
-- i. Employees working on project 105
SELECT emp_name FROM Employee WHERE projno=105;

-- ii. Count of projects “In progress”
SELECT COUNT(*) FROM Project WHERE status='In progress';

-- iii. Change start date of Robotics to 15/12/2022
UPDATE Project SET start_date='2022-12-15' WHERE proj_name='Robotics';

-- iv. Increase salaries of employees on project 10 by 5%
UPDATE Employee SET salary=salary*1.05 WHERE projno=10;
```

```
-- v. Delete completed projects
DELETE FROM Project WHERE status='Complete';
```

C. Query options

```
-- 1. Projects with more than 3 employees
SELECT proj_name FROM Project P
JOIN Employee E ON P.projno=E.projno
GROUP BY P.projno,P.proj_name
HAVING COUNT(E.empno)>3;

-- 2. View: Employees working on project started on 1/4/2025 and in progress
CREATE VIEW ProjectStart040125 AS
SELECT * FROM Employee E
JOIN Project P ON E.projno=P.projno
WHERE P.start_date='2025-04-01' AND P.status='In progress';
```