Savitribai Phule Pune University

S.Y. B.C.A. (Science) (Semester-III) Practical Examination

BCA 235: s(Database Management Systems II Laboratory)

Duration: 3Hrs. **Max Marks: 35+15=50**

Note: - 1. Read the questions carefully and insert data in the database accordingly.
2. Insert sufficient number of records in the database.
3. No query should generate empty output.
4. For count queries output should be more than 2 records. (If asked)

Q.1) Create the following database in 3NF using PostgresSQL. [Total Marks: 10]
Consider the following Bank database which maintains information about its branches, customers and their loan applications.
Branch (Bid integer, brname varchar (30), brcity varchar (10))
Customer (Cno integer, cname varchar (20), caddr varchar (35), city varchar (15))
Loan_application (Lno integer, l_amt_required money, lamtapproved money, l_date date)

Relationship:
Branch, Customer, Loan_application are related with ternary relationship as follows:
Ternary (Bid, Cno, Lno)

Constraints: Primary key, l_amt_required should be greater than zero.

Create a View: [10]
1. To display names of customers for the 'Pimpri' branch.
2. To display names of customers who have taken loan from the branch in the same city theylive.

Q.2) Using above database solve following questions: **[Total Marks: 20]**
1. Write a trigger which will execute when you update customer number from customer table. Display message "You can't change existing customer number". **[10]**
2. Write a stored function to accept branch name as an input parameter and display loan information of that branch. **[10]**

Q.3) External Viva [05]

Q.4) Internal Evaluation [15]

BANK DATABASE

```
CREATE TABLE Branch (Bid INTEGER PRIMARY KEY, brname VARCHAR(30) NOT NULL, brcity VARCHAR(10) NOT NULL);

CREATE TABLE Customer (Cno INTEGER PRIMARY KEY, cname VARCHAR(20) NOT NULL, caddr VARCHAR(35), city VARCHAR(15));

CREATE TABLE Loan_application (Lno INTEGER PRIMARY KEY, l_amt_required INT CHECK (l_amt_required > 0), lamtapproved INT, l_date DATE);

CREATE TABLE Ternary (Bid INTEGER, Cno INTEGER, Lno INTEGER, PRIMARY KEY (Bid, Cno, Lno), FOREIGN KEY (Bid) REFERENCES Branch(Bid), FOREIGN KEY (Cno) REFERENCES Customer(Cno), FOREIGN KEY (Lno) REFERENCES Loan_application(Lno));
```

```
INSERT INTO Branch (Bid, brname, brcity) VALUES (1, 'Pimpri', 'Pimpri'), (2, 'Aundh', 'Aundh');

INSERT INTO Customer (Cno, cname, caddr, city) VALUES (1, 'Rahul', '123 Street', 'Pimpri'), (2, 'Neha', '456 Avenue', 'Aundh'), (3, 'Raj', '789 Boulevard', 'Pune');

INSERT INTO Loan_application (Lno, l_amt_required, lamtapproved, l_date) VALUES (101, 500000, 450000, '2024-09-01'), (102, 200000, 150000, '2024-09-05'), (103, 600000, 550000, '2024-09-10');

INSERT INTO Ternary (Bid, Cno, Lno) VALUES (1, 1, 101), (2, 2, 102), (2, 3, 103);
```

Q.1) Create a View:

```
CREATE VIEW Customers_Pimpri_Branch AS SELECT c.cname FROM Customer c JOIN Ternary t ON c.Cno = t.Cno JOIN Branch b ON t.Bid = b.Bid WHERE b.brcity = 'Pimpri';

SELECT * FROM Customers_Pimpri_Branch;

CREATE VIEW Customers_Loan_Same_City AS SELECT c.cname FROM Customer c JOIN Ternary t ON c.Cno = t.Cno JOIN Branch b ON t.Bid = b.Bid WHERE c.city = b.brcity;

SELECT * FROM Customers_Loan_Same_City;
```

Q.2) Using above database solve following questions:

```
CREATE OR REPLACE FUNCTION prevent_customer_number_update()
RETURNS TRIGGER AS $$
BEGIN
   RAISE EXCEPTION 'You can’t change existing customer number';
   RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_prevent_customer_number_update
BEFORE UPDATE OF Cno ON Customer
FOR EACH ROW
EXECUTE FUNCTION prevent_customer_number_update();

UPDATE Customer SET Cno = 2 WHERE Cno = 1
```

```
CREATE OR REPLACE FUNCTION get_loan_info_by_branch(branch_name VARCHAR)
RETURNS VOID AS $$
BEGIN
   FOR rec IN
      SELECT l.Lno, l.l_amt_required, l.lamtapproved, l.l_date
      FROM Loan_application l
      JOIN Ternary t ON l.Lno = t.Lno
      JOIN Branch b ON t.Bid = b.Bid
      WHERE b.brname = branch_name
   LOOP
      RAISE NOTICE 'Loan No: %, Amount Required: %, Approved Amount: %, Date: %', rec.Lno, rec.l_amt_required, rec.lamtapproved, rec.l_date;
   END LOOP;
END;
$$ LANGUAGE plpgsql;


SELECT get_loan_info_by_branch('Pimpri');
```