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 customer details who have applied for a loan of 5, 00,000.
2. To display loan details from the 'Aundh' branch.

Q.2) Using above database solve following questions: [**Total Marks: 20]**

1. Write a trigger to validate the loan amount approved. It must be less than or equal to loan amount required. Display appropriate message. **[10]**

2. Write a stored function to count number of customers of particular branch. (Accept branch name as an input parameter). Display message for invalid branch name. **[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_Loan_500k AS SELECT c.* FROM Customer c JOIN Ternary t ON c.Cno = t.Cno JOIN Loan_application l ON t.Lno = l.Lno WHERE l.l_amt_required = 500000;

SELECT * FROM Customers_Loan_500k

CREATE VIEW Loans_From_Aundh_Branch AS SELECT l.* FROM Loan_application l JOIN Ternary t ON l.Lno = t.Lno JOIN Branch b ON t.Bid = b.Bid WHERE b.brcity = 'Aundh';

SELECT * FROM Loans_From_Aundh_Branch;
```

Q.2) Using above database solve following questions:

```
CREATE OR REPLACE FUNCTION validate_loan_amount()
RETURNS TRIGGER AS $$
BEGIN
  IF NEW.lamtapproved > NEW.l_amt_required THEN
    RAISE EXCEPTION 'Loan amount approved must be less than or equal to the loan amount required';
  END IF;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_validate_loan_amount
BEFORE INSERT OR UPDATE ON Loan_application
FOR EACH ROW
EXECUTE FUNCTION validate_loan_amount();

INSERT INTO Loan_application VALUES (104, 300000, 400000, '2024-09-15');
```

```
CREATE OR REPLACE FUNCTION count_customers_in_branch(branch_name VARCHAR)
RETURNS INT AS $$
DECLARE
  customer_count INT;
BEGIN
  SELECT COUNT(c.Cno) INTO customer_count
  FROM Customer c
  JOIN Ternary t ON c.Cno = t.Cno
  JOIN Branch b ON t.Bid = b.Bid
  WHERE b.brname = branch_name;

  IF customer_count IS NULL OR customer_count = 0 THEN
    RAISE NOTICE 'Invalid branch name';
    RETURN 0;
  ELSE
    RETURN customer_count;
  END IF;
END;
$$ LANGUAGE plpgsql;

SELECT count_customers_in_branch('Aundh');
```