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)

Q1) 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 (<u>Bid</u> integer, brname varchar (30), brcity varchar (10))
Customer (<u>Cno</u> integer, cname varchar (20), caddr varchar (35), city varchar (15))
Loan_application (<u>Lno</u> integer, l_amt_required money, lamtapproved money, l_date date)

Relationship:

Branch, Customer, Loan_application are related with ternary relationship as follows:
Ternary (<u>Bid, Cno, Lno</u>)

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

Create a View: **[10]**

1. To display the names of customers who required loan > 2,00,000.
2. To display the branch wise name of customers.

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

1. Write a trigger before inserting record of customer in customer table. If the customer number is less than or equal to zero then display the appropriate error message. **[10]**
2. Write a cursor to display customer details along with their approved loan amount. **[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_Above_200k AS SELECT c.cname 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 > 200000;

SELECT * FROM Customers_Loan_Above_200k;

CREATE VIEW Branch_Wise_Customers AS SELECT b.brname, c.cname FROM Branch b JOIN Ternary t ON b.Bid = t.Bid JOIN Customer c ON t.Cno = c.Cno;

SELECT * FROM Branch_Wise_Customers;
```

Q.2) Using above database solve following questions:

```
CREATE OR REPLACE FUNCTION validate_customer_number()
RETURNS TRIGGER AS $$
BEGIN
   IF NEW.Cno <= 0 THEN
      RAISE EXCEPTION 'Customer number must be greater than zero';
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_validate_customer_number
BEFORE INSERT ON Customer
FOR EACH ROW
EXECUTE FUNCTION validate_customer_number();

INSERT INTO Customer VALUES (-1, 'John', '101 Main St', 'Pune');
```

```
CREATE OR REPLACE FUNCTION display_customer_loan_details()
RETURNS VOID AS $$
DECLARE
   customer_cursor CURSOR FOR
      SELECT c.cname, c.caddr, l.lamtapproved
      FROM Customer c
      JOIN Ternary t ON c.Cno = t.Cno
      JOIN Loan_application l ON t.Lno = l.Lno;

   customer_record RECORD;
BEGIN
   OPEN customer_cursor;

   LOOP
      FETCH customer_cursor INTO customer_record;

      EXIT WHEN NOT FOUND;

      RAISE NOTICE 'Customer Name: %, Address: %, Approved Loan Amount: %', customer_record.cname, customer_record.caddr, customer_record.lamtapproved;
   END LOOP;

   CLOSE customer_cursor;
END;
$$ LANGUAGE plpgsql;

SELECT display_customer_loan_details();
```