Creating a Sample Database Structure(Exercise-1)

```
1)CREATE TABLE customers (
  customer_id       NUMBER PRIMARY KEY,
  name              VARCHAR2(100),
  age               NUMBER,
  balance           NUMBER(10,2),
  loan_interest_rate NUMBER(5,2),
  IsVIP             VARCHAR2(5) DEFAULT 'FALSE'
);

2)CREATE TABLE loans (
  loan_id       VARCHAR2(10) PRIMARY KEY,
  customer_id   NUMBER REFERENCES customers(customer_id),
  due_date      DATE
);

3)INSERT INTO customers VALUES (101, 'Sadhvik', 45, 8000.00, 10.0, 'FALSE');
INSERT INTO customers VALUES (102, 'Nani', 62, 15000.00, 12.5, 'FALSE');
INSERT INTO customers VALUES (103, 'rishi', 70, 20000.00, 11.0, 'FALSE');
INSERT INTO customers VALUES (104, 'rakshit', 59, 9000.00, 13.0, 'FALSE');

4)INSERT INTO loans VALUES ('L01', 101, SYSDATE + 30);
INSERT INTO loans VALUES ('L02', 102, SYSDATE + 10);
INSERT INTO loans VALUES ('L03', 103, SYSDATE + 45);
INSERT INTO loans VALUES ('L04', 104, SYSDATE + 5);

5)COMMIT;
```

Exercise 1: Control Structures

Scenario 1: Discount for Senior Citizens

```
DECLARE
  CURSOR cust_cursor IS
    SELECT customer_id, age, loan_interest_rate
    FROM customers
    WHERE age > 60;
```

```
BEGIN
  FOR cust_rec IN cust_cursor LOOP
    UPDATE customers
    SET loan_interest_rate = loan_interest_rate - 1
    WHERE customer_id = cust_rec.customer_id;
  END LOOP;

  COMMIT;
END;
```

Scenario 2: Promote Customers to VIP Based on Balance

```
DECLARE
  CURSOR vip_cursor IS
    SELECT customer_id, balance
    FROM customers
    WHERE balance > 10000;

BEGIN
  FOR vip_rec IN vip_cursor LOOP
    UPDATE customers
    SET IsVIP = 'TRUE'
    WHERE customer_id = vip_rec.customer_id;
  END LOOP;

  COMMIT;
END;
```

Scenario 3: Loan Due Reminders

```
DECLARE
  CURSOR due_loan_cursor IS
    SELECT loan_id, customer_id, due_date
    FROM loans
    WHERE due_date <= SYSDATE + 30;
```

```
BEGIN
  FOR loan_rec IN due_loan_cursor LOOP
    DBMS_OUTPUT.PUT_LINE('Reminder: Loan ID ' || loan_rec.loan_id ||
                         ' for Customer ID ' || loan_rec.customer_id ||
                         ' is due on ' || TO_CHAR(loan_rec.due_date, 'DD-MON-YYYY'));
  END LOOP;
END;
```

Creating Sample Tables(Exercise-3)

```
 1)CREATE TABLE accounts (
   account_id   NUMBER PRIMARY KEY,
   customer_id  NUMBER,
   balance      NUMBER(10, 2),
   account_type VARCHAR2(20)
 );

 2)CREATE TABLE employees (
   employee_id   NUMBER PRIMARY KEY,
   name          VARCHAR2(100),
   department_id NUMBER,
   salary        NUMBER(10, 2)
 );
```

Exercise 3: Stored Procedures

Scenario 1: Process Monthly Interest for Savings Accounts

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS
BEGIN
  UPDATE accounts
  SET balance = balance + (balance * 0.01)
  WHERE account_type = 'savings';

  COMMIT;
END;
```

Output:

All savings accounts have their balances increased by **1%**.

Scenario 2: Update Bonus for Employees in a Department

```
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (
  p_dept_id      IN NUMBER,
  p_bonus_pct    IN NUMBER
) IS
BEGIN
  UPDATE employees
  SET salary = salary + (salary * p_bonus_pct / 100)
  WHERE department_id = p_dept_id;

  COMMIT;
END;
```

Scenario 3: Transfer Funds Between Accounts

```
CREATE OR REPLACE PROCEDURE TransferFunds (
  p_from_account_id IN NUMBER,
  p_to_account_id   IN NUMBER,
  p_amount          IN NUMBER
) IS
  v_from_balance NUMBER;
BEGIN

  SELECT balance INTO v_from_balance
  FROM accounts
  WHERE account_id = p_from_account_id
  FOR UPDATE;

  IF v_from_balance < p_amount THEN
    RAISE_APPLICATION_ERROR(-20001, 'Insufficient balance in source account');
  END IF;


  UPDATE accounts
  SET balance = balance - p_amount
  WHERE account_id = p_from_account_id;


  UPDATE accounts
  SET balance = balance + p_amount
  WHERE account_id = p_to_account_id;

  COMMIT;
END;
```