Name:- VIDYA CS TRACK:JAVA FSE

Exercise 1: Control Structures

Scenario 1: The bank wants to apply a discount to loan interest rates for customers above 60 years old.
Question: Write a PL/SQL block that loops through all customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

Pl/SQL code:-

```
BEGIN
   FOR cust_rec IN (SELECT CustomerID, DOB FROM Customers) LOOP
      IF MONTHS_BETWEEN(SYSDATE, cust_rec.DOB) / 12 > 60 THEN
         UPDATE Loans
         SET InterestRate = InterestRate - 1
         WHERE CustomerID = cust_rec.CustomerID;
      END IF;
   END LOOP;
   COMMIT;
END;
```

Scenario 2: A customer can be promoted to VIP status based on their balance.
Question: Write a PL/SQL block that iterates through all customers and sets a flag IsVIP to TRUE for those with a balance over $10,000.

Pl/SQL code:-

```
ALTER TABLE Customers ADD IsVIP CHAR(1);  -- 'Y' or 'N'
BEGIN
FOR cust_rec IN (SELECT CustomerID, Balance FROM Customers) LOOP
IF cust_rec.Balance > 10000 THEN
UPDATE Customers
SET IsVIP = 'Y'
WHERE CustomerID = cust_rec.CustomerID;
ELSE
UPDATE Customers
SET IsVIP = 'N'
```

```
WHERE CustomerID = cust_rec.CustomerID;
END IF;
END LOOP;
COMMIT;
END;
```

Scenario 3: The bank wants to send reminders to customers whose loans are due within the next 30 days.
Question: Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

Pl/SQL code:-

```
DECLARE
   v_name Customers.Name%TYPE;
BEGIN
   FOR loan_rec IN (
      SELECT LoanID, CustomerID, EndDate
      FROM Loans
      WHERE EndDate BETWEEN SYSDATE AND SYSDATE + 30
   ) LOOP
      SELECT Name INTO v_name
      FROM Customers
      WHERE CustomerID = loan_rec.CustomerID;

      DBMS_OUTPUT.PUT_LINE('Reminder: Loan ID ' || loan_rec.LoanID ||
                          ' for customer ' || v_name ||
                          ' is due on ' || TO_CHAR(loan_rec.EndDate, 'YYYY-MM-DD'));
   END LOOP;
END;
```

Exercise 3: Stored Procedures

Scenario 1: The bank needs to process monthly interest for all savings accounts.
Question: Write a stored procedure **ProcessMonthlyInterest** that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.

Pl/SQL code:-

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS
BEGIN
   UPDATE Accounts
   SET Balance = Balance + (Balance * 0.01)
   WHERE AccountType = 'Savings';

   COMMIT;
END;
```

Scenario 2: The bank wants to implement a bonus scheme for employees based on their performance.
Question: Write a stored procedure **UpdateEmployeeBonus** that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

Pl/SQL code:-

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS
BEGIN
   UPDATE Accounts
   SET Balance = Balance + (Balance * 0.01)
   WHERE AccountType = 'Savings';

   COMMIT;
END;
```

Usage Example:-

```
BEGIN
   UpdateEmployeeBonus('IT', 10);
END;
```

Scenario 3: Customers should be able to transfer funds between their accounts.
Question: Write a stored procedure **TransferFunds** that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.

Pl/SQL code:-

```
CREATE OR REPLACE PROCEDURE TransferFunds(
   source_account_id IN NUMBER,
   dest_account_id IN NUMBER,
   amount IN NUMBER
) IS
   source_balance NUMBER;
BEGIN
   -- Get source account balance
   SELECT Balance INTO source_balance
   FROM Accounts
   WHERE AccountID = source_account_id
   FOR UPDATE;

   IF source_balance < amount THEN
      RAISE_APPLICATION_ERROR(-20001, 'Insufficient funds in source account.');
   END IF;

   -- Deduct from source
   UPDATE Accounts
   SET Balance = Balance - amount
   WHERE AccountID = source_account_id;

   -- Add to destination
   UPDATE Accounts
   SET Balance = Balance + amount
   WHERE AccountID = dest_account_id;

   COMMIT;
END;
```

Usage Example:-

```
BEGIN
   TransferFunds(1, 2, 500);  -- Transfer 500 from AccountID 1 to 2
END;
```

Schema to Created

```
CREATE TABLE Customers (
   CustomerID NUMBER PRIMARY KEY,
   Name VARCHAR2(100),
   DOB DATE,
   Balance NUMBER,
   LastModified DATE
);

CREATE TABLE Accounts (
   AccountID NUMBER PRIMARY KEY,
   CustomerID NUMBER,
   AccountType VARCHAR2(20),
   Balance NUMBER,
   LastModified DATE,
   FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

CREATE TABLE Transactions (
   TransactionID NUMBER PRIMARY KEY,
   AccountID NUMBER,
   TransactionDate DATE,
   Amount NUMBER,
   TransactionType VARCHAR2(10),
   FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)
);

CREATE TABLE Loans (
   LoanID NUMBER PRIMARY KEY,
   CustomerID NUMBER,
   LoanAmount NUMBER,
   InterestRate NUMBER,
   StartDate DATE,
   EndDate DATE,
   FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

CREATE TABLE Employees (
   EmployeeID NUMBER PRIMARY KEY,
   Name VARCHAR2(100),
   Position VARCHAR2(50),
   Salary NUMBER,
   Department VARCHAR2(50),
    HireDate DATE
);
```

Scripts Data Inserted:-

```
INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)
VALUES (1, 'John Doe', TO_DATE('1985-05-15', 'YYYY-MM-DD'), 1000, SYSDATE);

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)
VALUES (2, 'Jane Smith', TO_DATE('1990-07-20', 'YYYY-MM-DD'), 1500, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)
VALUES (1, 1, 'Savings', 1000, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)
VALUES (2, 2, 'Checking', 1500, SYSDATE);

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)
VALUES (1, 1, SYSDATE, 200, 'Deposit');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)
VALUES (2, 2, SYSDATE, 300, 'Withdrawal');

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)
VALUES (1, 1, 5000, 5, SYSDATE, ADD_MONTHS(SYSDATE, 60));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)
VALUES (1, 'Alice Johnson', 'Manager', 70000, 'HR', TO_DATE('2015-06-15', 'YYYY-MM-DD'));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)
VALUES (2, 'Bob Brown', 'Developer', 60000, 'IT', TO_DATE('2017-03-20', 'YYYY-MM-DD'));
```