PL/SQL

EXERCISE 6

TABLE CREATION AND DATA INSERTION:

-- Create the Customers table

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

-- Create the Accounts table

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

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

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

-- Create the Employees table
```

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

-- Create the AuditLog table

```
CREATE TABLE AuditLog (
    LogID NUMBER PRIMARY KEY,
    TransactionID NUMBER,
    LogDate DATE,
    Message VARCHAR2(255),
    FOREIGN KEY (TransactionID) REFERENCES Transactions(TransactionID)
);
```

-- Insert sample data into the Customers table

```
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 sample data into the Accounts table

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 sample data into the Transactions table

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 sample data into the Loans table

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (1, 1, 5000, 5, SYSDATE, ADD_MONTHS(SYSDATE, 60));

-- Insert sample data into the Employees table

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'));

Exercise 6: Cursors

Scenario 1: Generate monthly statements for all customers.

Question: Write a PL/SQL block using an explicit cursor **GenerateMonthlyStatements** that retrieves all transactions for the current month and prints a statement for each customer.

```
DECLARE
    CURSOR transaction_cursor IS
```

```
        SELECT c.CustomerID, c.Name, t.TransactionDate, t.Amount,
t.TransactionType
        FROM Customers c
        JOIN Accounts a ON c.CustomerID = a.CustomerID
        JOIN Transactions t ON a.AccountID = t.AccountID
        WHERE EXTRACT(MONTH FROM t.TransactionDate) =
EXTRACT(MONTH FROM SYSDATE)
          AND EXTRACT(YEAR FROM t.TransactionDate) =
EXTRACT(YEAR FROM SYSDATE);

    v_customer_id Customers.CustomerID%TYPE;
    v_name Customers.Name%TYPE;
    v_transaction_date Transactions.TransactionDate%TYPE;
    v_amount Transactions.Amount%TYPE;
    v_transaction_type Transactions.TransactionType%TYPE;
BEGIN
    FOR transaction_record IN transaction_cursor LOOP
        v_customer_id := transaction_record.CustomerID;
        v_name := transaction_record.Name;
        v_transaction_date := transaction_record.TransactionDate;
        v_amount := transaction_record.Amount;
        v_transaction_type := transaction_record.TransactionType;

        -- Print statement for each transaction
        DBMS_OUTPUT.PUT_LINE('Customer: ' || v_name || ' (' ||
v_customer_id || ')');
```

```
        DBMS_OUTPUT.PUT_LINE('Transaction Date: ' ||
v_transaction_date);
        DBMS_OUTPUT.PUT_LINE('Amount: ' || v_amount || ' Type: '
|| v_transaction_type);
        DBMS_OUTPUT.PUT_LINE('-------------------------');
    END LOOP;
END;
/
```

__**Scenario 2:** Apply annual fee to all accounts.__

Question: Write a PL/SQL block using an explicit cursor **ApplyAnnualFee** that deducts an annual maintenance fee from the balance of all accounts.

```
DECLARE
    CURSOR account_cursor IS
        SELECT AccountID, Balance
        FROM Accounts;

    v_account_id Accounts.AccountID%TYPE;
    v_balance Accounts.Balance%TYPE;
    v_annual_fee CONSTANT NUMBER := 50; -- Define annual fee
BEGIN
    FOR account_record IN account_cursor LOOP
        v_account_id := account_record.AccountID;
        v_balance := account_record.Balance;

        -- Deduct annual fee
        UPDATE Accounts
        SET Balance = v_balance - v_annual_fee
        WHERE AccountID = v_account_id;
    END LOOP;

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Annual fees applied to all accounts.');
END;
/
```

Scenario 3: Update the interest rate for all loans based on a new policy.

Question: Write a PL/SQL block using an explicit cursor **UpdateLoanInterestRates** that fetches all loans and updates their interest rates based on the new policy.

```
DECLARE
    CURSOR loan_cursor IS
        SELECT LoanID, InterestRate
        FROM Loans;

    v_loan_id Loans.LoanID%TYPE;
    v_interest_rate Loans.InterestRate%TYPE;
    v_new_interest_rate NUMBER;
BEGIN
    FOR loan_record IN loan_cursor LOOP
        v_loan_id := loan_record.LoanID;
        v_interest_rate := loan_record.InterestRate;

        -- Calculate new interest rate based on policy
        v_new_interest_rate := v_interest_rate * 0.95; -- Example:
decrease by 5%

        -- Update interest rate
        UPDATE Loans
        SET InterestRate = v_new_interest_rate
        WHERE LoanID = v_loan_id;
    END LOOP;

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Loan interest rates updated based
on new policy.');
END;
/
```