PL/SQL

EXERCISE 5

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 5: Triggers

Scenario 1: Automatically update the last modified date when a customer's record is updated.

Question: Write a trigger **UpdateCustomerLastModified** that updates the LastModified column of the Customers table to the current date whenever a customer's record is updated.

```
CREATE OR REPLACE TRIGGER UpdateCustomerLastModified

AFTER UPDATE ON Customers

FOR EACH ROW
```

```
BEGIN
    :NEW.LastModified := SYSDATE;
END;
/
```

Scenario 2: Maintain an audit log for all transactions.

Question: Write a trigger **LogTransaction** that inserts a record into an AuditLog table whenever a transaction is inserted into the Transactions table.

```
CREATE OR REPLACE TRIGGER LogTransaction
AFTER INSERT ON Transactions
FOR EACH ROW
BEGIN
    INSERT INTO AuditLog (LogID, TransactionID, LogDate, Message)
    VALUES (
        AuditLog_seq.NEXTVAL,
        :NEW.TransactionID,
        SYSDATE,
        'Transaction logged: ID ' || :NEW.TransactionID
    );
END;
/
```

Scenario 3: Enforce business rules on deposits and withdrawals.

Question: Write a trigger **CheckTransactionRules** that ensures withdrawals do not exceed the balance and deposits are positive before inserting a record into the Transactions table.

```
CREATE OR REPLACE TRIGGER CheckTransactionRules
BEFORE INSERT ON Transactions
FOR EACH ROW
BEGIN
   -- Ensure withdrawal does not exceed balance
   IF :NEW.TransactionType = 'Withdrawal' THEN
      DECLARE
         v_balance Accounts.Balance%TYPE;
      BEGIN
         SELECT Balance INTO v_balance FROM Accounts WHERE AccountID = :NEW.AccountID;
         IF :NEW.Amount > v_balance THEN
            RAISE_APPLICATION_ERROR(-20002, 'Insufficient balance for withdrawal.');
         END IF;
      END;
   END IF;

   -- Ensure deposit amount is positive
   IF :NEW.TransactionType = 'Deposit' AND :NEW.Amount <= 0 THEN
```

```
        RAISE_APPLICATION_ERROR(-20003, 'Deposit amount must
be positive.');
    END IF;
END;
/
```