WEEK 2
PL/SQL

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 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.

```
DECLARE
    CURSOR customer_cursor IS
        SELECT LoanID, InterestRate
```

```
        FROM Loans l
        JOIN Customers c ON l.CustomerID = c.CustomerID
        WHERE EXTRACT(YEAR FROM SYSDATE) -
EXTRACT(YEAR FROM DOB) > 60;


    v_loan_id Loans.LoanID%TYPE;
    v_interest_rate Loans.InterestRate%TYPE;
BEGIN
    FOR customer_record IN customer_cursor LOOP
        v_loan_id := customer_record.LoanID;
        v_interest_rate := customer_record.InterestRate;


        -- Apply 1% discount to the current loan interest rate
        UPDATE Loans
        SET InterestRate = v_interest_rate - 1
        WHERE LoanID = v_loan_id;
    END LOOP;


    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Discount applied to eligible
customers.');
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.

```
DECLARE
   CURSOR customer_cursor IS
      SELECT CustomerID
      FROM Customers
      WHERE Balance > 10000;

   v_customer_id Customers.CustomerID%TYPE;
BEGIN
   FOR customer_record IN customer_cursor LOOP
      v_customer_id := customer_record.CustomerID;

      -- Set IsVIP flag to TRUE
      UPDATE Customers
      SET IsVIP = TRUE
      WHERE CustomerID = v_customer_id;
   END LOOP;

   COMMIT;
   DBMS_OUTPUT.PUT_LINE('VIP status updated for eligible customers.');
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.

```
DECLARE
   CURSOR loan_cursor IS
      SELECT CustomerID, LoanID, EndDate
      FROM Loans
      WHERE EndDate BETWEEN SYSDATE AND SYSDATE + 30;

   v_customer_id Customers.CustomerID%TYPE;
   v_loan_id Loans.LoanID%TYPE;
   v_end_date Loans.EndDate%TYPE;
BEGIN
   FOR loan_record IN loan_cursor LOOP
      v_customer_id := loan_record.CustomerID;
      v_loan_id := loan_record.LoanID;
      v_end_date := loan_record.EndDate;

      -- Print reminder message
      DBMS_OUTPUT.PUT_LINE('Reminder: Loan ' || v_loan_id || ' for customer ' || v_customer_id || ' is due on ' || v_end_date);
   END LOOP;
END;
/
```