**Exercise 1: Control Structures**

SET SERVEROUTPUT ON;

BEGIN

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE,

IsVIP VARCHAR2(5) DEFAULT 'FALSE'

);

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

);

INSERT INTO Customers VALUES (1, 'John Doe', TO\_DATE('1950-05-15', 'YYYY-MM-DD'), 9000, SYSDATE, 'FALSE');

INSERT INTO Customers VALUES (2, 'Jane Smith', TO\_DATE('1980-07-20', 'YYYY-MM-DD'), 15000, SYSDATE, 'FALSE');

INSERT INTO Accounts VALUES (1, 1, 'Savings', 9000, SYSDATE);

INSERT INTO Accounts VALUES (2, 2, 'Checking', 15000, SYSDATE);

INSERT INTO Transactions VALUES (1, 1, SYSDATE, 200, 'Deposit');

INSERT INTO Transactions VALUES (2, 2, SYSDATE, 300, 'Withdrawal');

INSERT INTO Loans VALUES (1, 1, 5000, 5, SYSDATE, ADD\_MONTHS(SYSDATE, 60));

INSERT INTO Loans VALUES (2, 2, 8000, 6, SYSDATE, SYSDATE + 10);

INSERT INTO Employees VALUES (1, 'Alice Johnson', 'Manager', 70000, 'HR', TO\_DATE('2015-06-15', 'YYYY-MM-DD'));

INSERT INTO Employees VALUES (2, 'Bob Brown', 'Developer', 60000, 'IT', TO\_DATE('2017-03-20', 'YYYY-MM-DD'));

BEGIN

FOR rec IN (

SELECT c.CustomerID, c.DOB, l.LoanID

FROM Customers c

JOIN Loans l ON c.CustomerID = l.CustomerID

) LOOP

IF MONTHS\_BETWEEN(SYSDATE, rec.DOB)/12 > 60 THEN

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE LoanID = rec.LoanID;

END IF;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('--- Updated Loan Interest Rates ---');

FOR r IN (SELECT LoanID, InterestRate FROM Loans) LOOP

DBMS\_OUTPUT.PUT\_LINE('Loan ' || r.LoanID || ' -> Interest: ' || r.InterestRate || '%');

END LOOP;

END;

/

BEGIN

FOR rec IN (SELECT CustomerID, Balance FROM Customers) LOOP

IF rec.Balance > 10000 THEN

UPDATE Customers SET IsVIP = 'TRUE' WHERE CustomerID = rec.CustomerID;

END IF;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('--- VIP Status Updated ---');

FOR r IN (SELECT Name, IsVIP FROM Customers) LOOP

DBMS\_OUTPUT.PUT\_LINE('Customer: ' || r.Name || ' -> VIP: ' || r.IsVIP);

END LOOP;

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('--- Loan Reminders (Due in 30 Days) ---');

FOR rec IN (

SELECT l.LoanID, l.EndDate, c.Name

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.EndDate <= SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || rec.LoanID || ' for ' || rec.Name ||

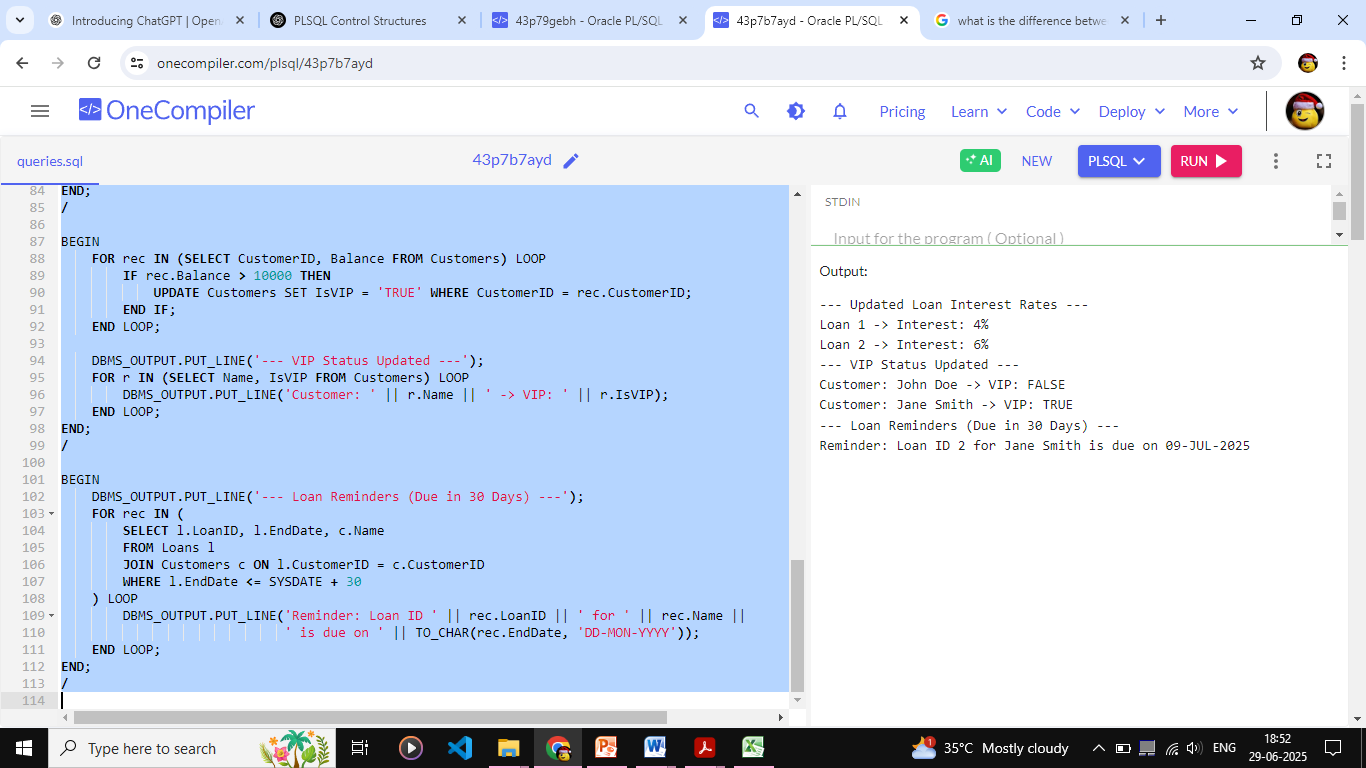
' is due on ' || TO\_CHAR(rec.EndDate, 'DD-MON-YYYY'));

END LOOP;

END;

/

**OUTPUT:**



**Exercise 3: Stored Procedures**

SET SERVEROUTPUT ON;

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE,

IsVIP VARCHAR2(5) DEFAULT 'FALSE'

);

/

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

);

/

INSERT INTO Customers VALUES (1, 'John Doe', TO\_DATE('1960-05-15', 'YYYY-MM-DD'), 9000, SYSDATE, 'FALSE');

INSERT INTO Customers VALUES (2, 'Jane Smith', TO\_DATE('1980-07-20', 'YYYY-MM-DD'), 15000, SYSDATE, 'FALSE');

/

INSERT INTO Accounts VALUES (1, 1, 'Savings', 9000, SYSDATE);

INSERT INTO Accounts VALUES (2, 2, 'Checking', 15000, SYSDATE);

/

INSERT INTO Transactions VALUES (1, 1, SYSDATE, 200, 'Deposit');

INSERT INTO Transactions VALUES (2, 2, SYSDATE, 300, 'Withdrawal');

/

INSERT INTO Loans VALUES (1, 1, 5000, 5, SYSDATE, ADD\_MONTHS(SYSDATE, 60));

INSERT INTO Loans VALUES (2, 2, 8000, 6, SYSDATE, SYSDATE + 10);

/

INSERT INTO Employees VALUES (1, 'Alice Johnson', 'Manager', 70000, 'HR', TO\_DATE('2015-06-15', 'YYYY-MM-DD'));

INSERT INTO Employees VALUES (2, 'Bob Brown', 'Developer', 60000, 'IT', TO\_DATE('2017-03-20', 'YYYY-MM-DD'));

/

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR rec IN (

SELECT AccountID, Balance FROM Accounts WHERE AccountType = 'Savings'

) LOOP

UPDATE Accounts

SET Balance = rec.Balance + (rec.Balance \* 0.01)

WHERE AccountID = rec.AccountID;

DBMS\_OUTPUT.PUT\_LINE('Interest applied to Account ID ' || rec.AccountID);

END LOOP;

END;

/

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_department IN VARCHAR2,

p\_bonus\_percent IN NUMBER

) IS

BEGIN

FOR rec IN (

SELECT EmployeeID, Salary FROM Employees WHERE Department = p\_department

) LOOP

UPDATE Employees

SET Salary = rec.Salary + (rec.Salary \* p\_bonus\_percent / 100)

WHERE EmployeeID = rec.EmployeeID;

DBMS\_OUTPUT.PUT\_LINE('Bonus added to Employee ID ' || rec.EmployeeID);

END LOOP;

END;

/

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_acc IN NUMBER,

p\_to\_acc IN NUMBER,

p\_amount IN NUMBER

) IS

v\_from\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_from\_balance

FROM Accounts

WHERE AccountID = p\_from\_acc;

IF v\_from\_balance < p\_amount THEN

DBMS\_OUTPUT.PUT\_LINE('Insufficient balance in Account ' || p\_from\_acc);

RETURN;

END IF;

UPDATE Accounts

SET Balance = Balance - p\_amount

WHERE AccountID = p\_from\_acc;

UPDATE Accounts

SET Balance = Balance + p\_amount

WHERE AccountID = p\_to\_acc;

DBMS\_OUTPUT.PUT\_LINE('Transferred ' || p\_amount || ' from Account ' || p\_from\_acc || ' to Account ' || p\_to\_acc);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('One of the account IDs does not exist.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('--- Processing Monthly Interest ---');

ProcessMonthlyInterest;

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('--- Updating Employee Bonus ---');

UpdateEmployeeBonus('IT', 10);

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('--- Transferring Funds ---');

TransferFunds(1, 2, 300);

END;

/

**OUTPUT:**

