# **PL/SQL PROGRAMMING**

## Exercise 1: Control Structures

### CODE:

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

Age NUMBER,

InterestRate NUMBER,

Balance NUMBER,

IsVIP VARCHAR2(5)

);

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

DueDate DATE

);

INSERT INTO Customers VALUES (1, 'Alice', 65, 10.0, 15000, 'FALSE');

INSERT INTO Customers VALUES (2, 'Bob', 45, 11.0, 8000, 'FALSE');

INSERT INTO Customers VALUES (3, 'Carol', 70, 9.5, 12000, 'FALSE');

INSERT INTO Loans VALUES (101, 1, SYSDATE + 10); -- due in 10 days

INSERT INTO Loans VALUES (102, 2, SYSDATE + 40); -- due in 40 days

INSERT INTO Loans VALUES (103, 3, SYSDATE + 5); -- due in 5 days

COMMIT;

BEGIN

FOR rec IN (SELECT CustomerID, Age FROM Customers)

LOOP

IF rec.Age > 60 THEN

UPDATE Customers

SET InterestRate = InterestRate - 1

WHERE CustomerID = rec.CustomerID;

END IF;

END LOOP;

COMMIT;

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;

COMMIT;

END;

/

BEGIN

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

FOR rec IN (

SELECT LoanID, CustomerID, DueDate

FROM Loans

WHERE DueDate BETWEEN SYSDATE AND SYSDATE + 30

)

LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Reminder: Customer ' || rec.CustomerID ||

' has loan ' || rec.LoanID ||

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

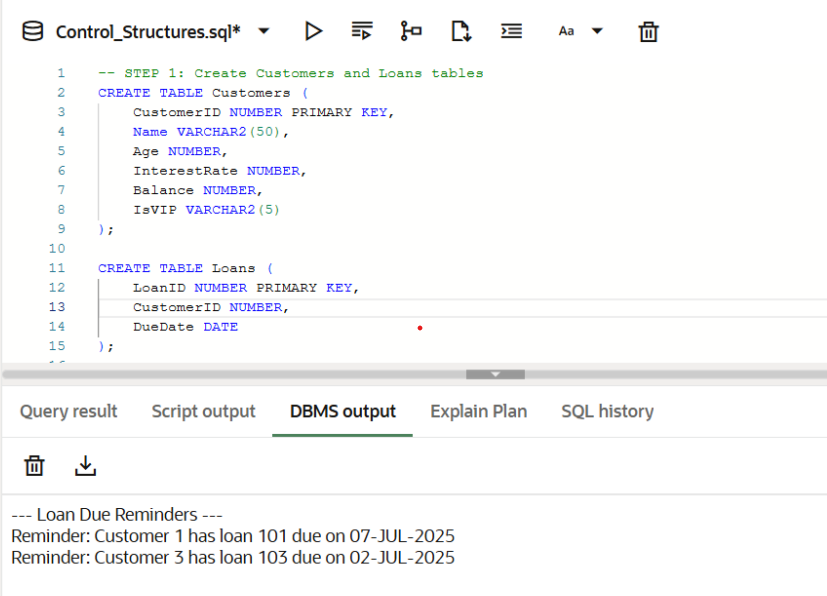
);

END LOOP;

END;

/

### OUTPUT:



## Exercise 3: Stored Procedures

### CODE:

CREATE TABLE SavingsAccounts (

AccountID NUMBER PRIMARY KEY,

CustomerName VARCHAR2(50),

Balance NUMBER

);

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

Department VARCHAR2(50),

Salary NUMBER

);

CREATE TABLE Accounts (

AccountNumber NUMBER PRIMARY KEY,

CustomerName VARCHAR2(50),

Balance NUMBER

);

INSERT INTO SavingsAccounts VALUES (1, 'Alice', 10000);

INSERT INTO SavingsAccounts VALUES (2, 'Bob', 20000);

INSERT INTO Employees VALUES (1, 'John', 'Sales', 50000);

INSERT INTO Employees VALUES (2, 'Sara', 'HR', 45000);

INSERT INTO Employees VALUES (3, 'Mike', 'Sales', 55000);

INSERT INTO Accounts VALUES (101, 'Krishika', 15000);

INSERT INTO Accounts VALUES (102, 'Krishika', 10000);

COMMIT;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR acc IN (SELECT AccountID FROM SavingsAccounts)

LOOP

UPDATE SavingsAccounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountID = acc.AccountID;

END LOOP;

COMMIT;

END;

/

BEGIN

ProcessMonthlyInterest;

END;

/

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept IN VARCHAR2,

bonus\_percent IN NUMBER

) IS

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* bonus\_percent / 100)

WHERE Department = dept;

COMMIT;

END;

/

BEGIN

UpdateEmployeeBonus('Sales', 10);

END;

/

CREATE OR REPLACE PROCEDURE TransferFunds (

from\_acc IN NUMBER,

to\_acc IN NUMBER,

amount IN NUMBER

) IS

insufficient\_balance EXCEPTION;

from\_balance NUMBER;

BEGIN

SELECT Balance INTO from\_balance FROM Accounts WHERE AccountNumber = from\_acc;

IF from\_balance < amount THEN

RAISE insufficient\_balance;

END IF;

UPDATE Accounts

SET Balance = Balance - amount

WHERE AccountNumber = from\_acc;

UPDATE Accounts

SET Balance = Balance + amount

WHERE AccountNumber = to\_acc;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Transfer successful: ₹' || amount);

EXCEPTION

WHEN insufficient\_balance THEN

DBMS\_OUTPUT.PUT\_LINE('Error: Insufficient balance for transfer.');

WHEN OTHERS THEN

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

END;

/

BEGIN

TransferFunds(101, 102, 2000);

END;

/

SELECT \* FROM SavingsAccounts;

SELECT \* FROM Employees;

SELECT \* FROM Accounts;

### OUTPUT:

