**Exercise 1: Control Structures**

-- Step 1: Create Tables

-----------------------------

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Loans';

EXECUTE IMMEDIATE 'DROP TABLE Customers';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Age NUMBER,

Balance NUMBER(10,2),

IsVIP CHAR(1) DEFAULT 'N'

);

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

InterestRate NUMBER(5,2),

DueDate DATE

);

-- Step 2: Insert Sample Data

-------------------------------------

INSERT INTO Customers VALUES (1, 'Ravi', 65, 12000, 'N');

INSERT INTO Customers VALUES (2, 'Meena', 45, 8000, 'N');

INSERT INTO Customers VALUES (3, 'Kumar', 70, 15000, 'N');

INSERT INTO Loans VALUES (101, 1, 10.5, SYSDATE + 10);

INSERT INTO Loans VALUES (102, 2, 11.0, SYSDATE + 40);

INSERT INTO Loans VALUES (103, 3, 12.0, SYSDATE + 5);

COMMIT;

-- Scenario 1: Apply 1% Interest Discount to Customers Over 60

------------------------------------------------------------------------------------

BEGIN

FOR rec IN (

SELECT CustomerID FROM Customers WHERE Age > 60

) LOOP

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = rec.CustomerID;

END LOOP;

COMMIT;

END;

/

-- Scenario 2: Promote to VIP if Balance > 10000

----------------------------------------------------------------

BEGIN

FOR rec IN (

SELECT CustomerID FROM Customers WHERE Balance > 10000

) LOOP

UPDATE Customers

SET IsVIP = 'Y'

WHERE CustomerID = rec.CustomerID;

END LOOP;

COMMIT;

END;

/

-- Scenario 3: Reminders for Loans Due in Next 30 Days

------------------------------------------------------------------------

BEGIN

FOR rec IN (

SELECT L.LoanID, C.Name, L.DueDate

FROM Loans L

JOIN Customers C ON L.CustomerID = C.CustomerID

WHERE L.DueDate BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Reminder: Loan ' || rec.LoanID || ' for ' || rec.Name ||

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

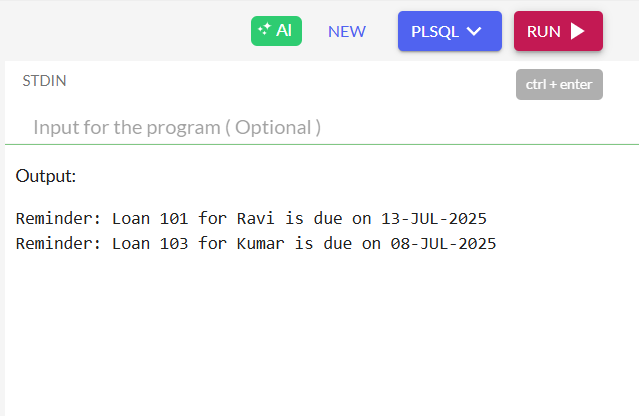
);

END LOOP;

END;

/

OUTPUT:



**Exercise 3: Stored Procedures**

Drop old tables (if any)

-------------------------------

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE SavingsAccounts';

EXECUTE IMMEDIATE 'DROP TABLE Employees';

EXECUTE IMMEDIATE 'DROP TABLE Accounts';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

Create Tables  
-----------------

CREATE TABLE SavingsAccounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

Balance NUMBER(10,2)

);

CREATE TABLE Employees (

EmpID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DepartmentID NUMBER,

Salary NUMBER(10,2)

);

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

Balance NUMBER(10,2)

);

/

Insert Sample Data

---------------------------

INSERT INTO SavingsAccounts VALUES (101, 1, 10000);

INSERT INTO SavingsAccounts VALUES (102, 2, 20000);

INSERT INTO Employees VALUES (1, 'John', 10, 50000);

INSERT INTO Employees VALUES (2, 'Priya', 10, 55000);

INSERT INTO Employees VALUES (3, 'Sam', 20, 60000);

INSERT INTO Accounts VALUES (201, 1, 15000); -- Source

INSERT INTO Accounts VALUES (202, 1, 5000); -- Target

COMMIT;

Procedure 1: ProcessMonthlyInterest

--------------------------------------------------

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;

/

- Procedure 2: UpdateEmployeeBonus

---------------------------------------------------

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_dept\_id IN NUMBER,

p\_bonus\_percent IN NUMBER

) IS

BEGIN

UPDATE Employees

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

WHERE DepartmentID = p\_dept\_id;

COMMIT;

END;

/

Procedure 3: TransferFunds

-------------------------------------

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_source\_id IN NUMBER,

p\_target\_id IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance

FROM Accounts

WHERE AccountID = p\_source\_id;

IF v\_balance < p\_amount THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient balance in source account');

END IF;

UPDATE Accounts

SET Balance = Balance - p\_amount

WHERE AccountID = p\_source\_id;

UPDATE Accounts

SET Balance = Balance + p\_amount

WHERE AccountID = p\_target\_id;

COMMIT;

END;

/

Call All Procedures

---------------------------

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Adding 1% interest to savings accounts...');

ProcessMonthlyInterest;

DBMS\_OUTPUT.PUT\_LINE(' Giving 10% bonus to department 10...');

UpdateEmployeeBonus(10, 10);

DBMS\_OUTPUT.PUT\_LINE('Transferring ₹3000 from Account 201 to 202...');

TransferFunds(201, 202, 3000);

END;

/

Show Final Output

-----------------------------

BEGIN

DBMS\_OUTPUT.PUT\_LINE(' Final Savings Account Balances:');

FOR rec IN (SELECT \* FROM SavingsAccounts) LOOP

DBMS\_OUTPUT.PUT\_LINE('Account ' || rec.AccountID || ': ₹' || rec.Balance);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Final Employee Salaries:');

FOR rec IN (SELECT \* FROM Employees) LOOP

DBMS\_OUTPUT.PUT\_LINE(rec.Name || ' (Dept ' || rec.DepartmentID || '): ₹' || rec.Salary);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Final Account Balances:');

FOR rec IN (SELECT \* FROM Accounts) LOOP

DBMS\_OUTPUT.PUT\_LINE('Account ' || rec.AccountID || ': ₹' || rec.Balance);

END LOOP;

END;

/

OUTPUT

-----------

