WEEK-2 :- PL/SQL PROGRAMMING

HANDS-ON – 1 :- CONTROL STRUCTURES

**CODE :-**

SET SERVEROUTPUT ON SIZE 1000000;

-- Create Customers Table

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Age NUMBER,

Balance NUMBER,

IsVIP VARCHAR2(5) DEFAULT 'FALSE'

);

-- Create Loans Table

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

InterestRate NUMBER,

DueDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

-- Insert Sample Customers

INSERT INTO Customers VALUES (1, 'Alice', 65, 12000, 'FALSE');

INSERT INTO Customers VALUES (2, 'Bob', 58, 8000, 'FALSE');

INSERT INTO Customers VALUES (3, 'Charlie', 70, 5000, 'FALSE');

INSERT INTO Customers VALUES (4, 'David', 45, 15000, 'FALSE');

-- Insert Sample Loans

INSERT INTO Loans VALUES (101, 1, 7.5, SYSDATE + 10); -- Due in 10 days

INSERT INTO Loans VALUES (102, 2, 8.0, SYSDATE + 40); -- Due in 40 days

INSERT INTO Loans VALUES (103, 3, 6.5, SYSDATE + 5); -- Due in 5 days

INSERT INTO Loans VALUES (104, 4, 7.0, SYSDATE + 50); -- Due in 50 days

COMMIT;

-- Scenario 1: Interest Discount

BEGIN

FOR rec IN (SELECT CustomerID FROM Customers WHERE Age > 60) LOOP

UPDATE Loans

SET InterestRate = InterestRate \* 0.9

WHERE CustomerID = rec.CustomerID;

END LOOP;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Interest discounts applied to eligible customers.');

END;

-- Scenario 2: VIP Status Update

BEGIN

FOR rec IN (SELECT CustomerID FROM Customers WHERE Balance > 10000) LOOP

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = rec.CustomerID;

END LOOP;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('VIP status updated for eligible customers.');

END;

-- Display Updated Customers

SELECT \* FROM Customers;

-- Scenario 3: Loan Reminder

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Starting Scenario 3...');

FOR rec IN (

SELECT c.Name, l.LoanID, l.DueDate

FROM Loans l

JOIN Customers c ON c.CustomerID = l.CustomerID

WHERE l.DueDate <= SYSDATE + 30

) LOOP

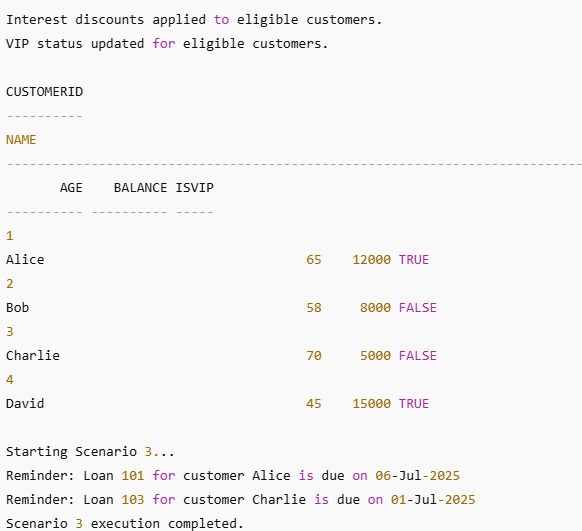
DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ' || rec.LoanID || ' for customer ' || rec.Name || ' is due on ' || TO\_CHAR(rec.DueDate, 'DD-MON-YYYY'));

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Scenario 3 execution completed.');

END;

/

**OUTPUT:-**

HANDS-ON – 2 :- STORED PROCEDURES

**CODE :-**

**SET SERVEROUTPUT ON;**

**-- Create Accounts Table**

**CREATE TABLE Accounts (**

**AccountID NUMBER PRIMARY KEY,**

**CustomerName VARCHAR2(100),**

**Balance NUMBER**

**);**

**-- Create Employees Table**

**CREATE TABLE Employees (**

**EmployeeID NUMBER PRIMARY KEY,**

**Name VARCHAR2(100),**

**Department VARCHAR2(100),**

**Salary NUMBER**

**);**

**-- Insert Sample Accounts**

**INSERT INTO Accounts VALUES (1, 'Alice', 10000);**

**INSERT INTO Accounts VALUES (2, 'Bob', 5000);**

**INSERT INTO Accounts VALUES (3, 'Charlie', 2000);**

**-- Insert Sample Employees**

**INSERT INTO Employees VALUES (1, 'David', 'Sales', 4000);**

**INSERT INTO Employees VALUES (2, 'Eva', 'Sales', 4500);**

**INSERT INTO Employees VALUES (3, 'Frank', 'HR', 5000);**

**COMMIT;**

**-- Scenario 1: Process Interest**

**CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS**

**BEGIN**

**FOR rec IN (SELECT AccountID, Balance FROM Accounts) LOOP**

**UPDATE Accounts**

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

**WHERE AccountID = rec.AccountID;**

**END LOOP;**

**COMMIT;**

**DBMS\_OUTPUT.PUT\_LINE('Monthly interest applied to all accounts.');**

**END;**

**/**

**-- Scenario 2: Update Employee Bonus Procedure**

**CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (**

**dept\_name IN VARCHAR2,**

**bonus\_pct IN NUMBER**

**) AS**

**BEGIN**

**UPDATE Employees**

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

**WHERE Department = dept\_name;**

**COMMIT;**

**DBMS\_OUTPUT.PUT\_LINE('Bonus applied to ' || dept\_name || ' department employees.');**

**END;**

**-- Scenario 3: Transfer Funds Procedure**

**CREATE OR REPLACE PROCEDURE TransferFunds (**

**from\_account IN NUMBER,**

**to\_account IN NUMBER,**

**amount IN NUMBER**

**) AS**

**insufficient\_funds EXCEPTION;**

**BEGIN**

**DECLARE**

**from\_balance NUMBER;**

**BEGIN**

**SELECT Balance INTO from\_balance FROM Accounts WHERE AccountID = from\_account;**

**IF from\_balance < amount THEN**

**RAISE insufficient\_funds;**

**ELSE**

**UPDATE Accounts**

**SET Balance = Balance - amount**

**WHERE AccountID = from\_account;**

**UPDATE Accounts**

**SET Balance = Balance + amount**

**WHERE AccountID = to\_account;**

**COMMIT;**

**DBMS\_OUTPUT.PUT\_LINE('Transfer of ' || amount || ' from Account ' || from\_account || ' to Account ' || to\_account || ' completed.');**

**END IF;**

**EXCEPTION**

**WHEN insufficient\_funds THEN**

**DBMS\_OUTPUT.PUT\_LINE('Error: Insufficient funds in source account.');**

**END;**

**END;**

**/**

**-- Execute All Scenarios**

**BEGIN**

**-- Scenario 1**

**ProcessMonthlyInterest;**

**-- Scenario 2**

**UpdateEmployeeBonus('Sales', 10);**

**-- Scenario 3**

**TransferFunds(1, 2, 3000);**

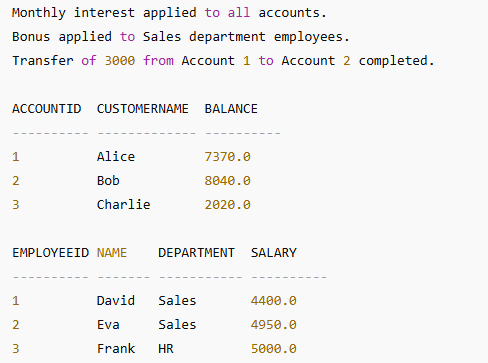
**END;**

**/**

**SELECT \* FROM Accounts;**

**SELECT \* FROM Employees;**

**OUTPUT:-**

****