Exercise – 1

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Age NUMBER,

Balance NUMBER,

IsVIP VARCHAR2(5)

);

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

InterestRate NUMBER(5,2),

DueDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

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

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

INSERT INTO Customers VALUES (3, 'Swathi', 70, 15000, 'FALSE');

INSERT INTO Loans VALUES (101, 1, 7.50, SYSDATE + 15);

INSERT INTO Loans VALUES (102, 2, 8.25, SYSDATE + 40);

INSERT INTO Loans VALUES (103, 3, 6.75, SYSDATE + 10);

COMMIT;

BEGIN

FOR cust IN (

SELECT CustomerID, Name FROM Customers WHERE Age > 60

) LOOP

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = cust.CustomerID;

DBMS\_OUTPUT.PUT\_LINE('Applied 1% interest discount for senior customer: ' || cust.Name);

END LOOP;

FOR cust IN (

SELECT CustomerID, Name FROM Customers WHERE Balance > 10000

) LOOP

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = cust.CustomerID;

DBMS\_OUTPUT.PUT\_LINE('Customer ' || cust.Name || ' has been promoted to VIP.');

END LOOP;

FOR loan IN (

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

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.DueDate <= SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || loan.LoanID ||

' for customer ' || loan.Name ||

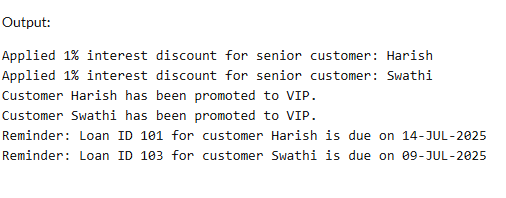
' is due on ' || TO\_CHAR(loan.DueDate, 'DD-MON-YYYY'));

END LOOP;

COMMIT;

END;

/



Exercise – 3

Scenario – 1

CREATE TABLE SavingsAccounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

Balance NUMBER(12, 2)

);

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

INSERT INTO SavingsAccounts VALUES (2, 102, 15000);

COMMIT;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR acc IN (SELECT AccountID, Balance FROM SavingsAccounts) LOOP

UPDATE SavingsAccounts

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

WHERE AccountID = acc.AccountID;

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

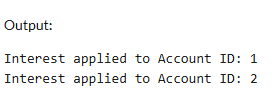
END LOOP;

COMMIT;

END;

/

EXEC ProcessMonthlyInterest;



Scenario – 2

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DepartmentID NUMBER,

Salary NUMBER(10, 2)

);

INSERT INTO Employees VALUES (1, 'Alice', 1, 50000);

INSERT INTO Employees VALUES (2, 'Bob', 2, 60000);

INSERT INTO Employees VALUES (3, 'Charlie', 2, 65000);

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

dept\_id IN NUMBER,

bonus\_percent IN NUMBER

) IS

BEGIN

FOR emp IN (

SELECT EmployeeID, Salary FROM Employees WHERE DepartmentID = dept\_id

) LOOP

UPDATE Employees

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

WHERE EmployeeID = emp.EmployeeID;

DBMS\_OUTPUT.PUT\_LINE('Bonus applied to Employee ID: ' || emp.EmployeeID);

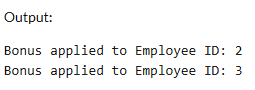
END LOOP;

COMMIT;

END;

/

EXEC UpdateEmployeeBonus(2, 10);



Scenario – 3

CREATE TABLE BankAccounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

Balance NUMBER(12, 2)

);

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

INSERT INTO BankAccounts VALUES (102, 2, 5000);

COMMIT;

CREATE OR REPLACE PROCEDURE TransferFunds(

source\_id IN NUMBER,

target\_id IN NUMBER,

amount IN NUMBER

) IS

source\_balance NUMBER;

BEGIN

SELECT Balance INTO source\_balance FROM BankAccounts WHERE AccountID = source\_id;

IF source\_balance < amount THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient balance in Account ID ' || source\_id);

RETURN;

END IF;

UPDATE BankAccounts

SET Balance = Balance - amount

WHERE AccountID = source\_id;

UPDATE BankAccounts

SET Balance = Balance + amount

WHERE AccountID = target\_id;

DBMS\_OUTPUT.PUT\_LINE('Transferred ' || amount || ' from Account ' || source\_id ||

' to Account ' || target\_id);

COMMIT;

END;

/

EXEC TransferFunds(101, 102, 500);

