**Week 2 PL/SQL Hands-On Exercises**

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE

);

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 (CustomerID, Name, DOB, Balance, LastModified)

VALUES (1, 'John Doe', TO\_DATE('1985-05-15', 'YYYY-MM-DD'), 1000, SYSDATE);

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (2, 'Jane Smith', TO\_DATE('1960-07-20', 'YYYY-MM-DD'), 1500, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (1, 1, 'Savings', 1000, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (2, 2, 'Checking', 1500, SYSDATE);

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (1, 1, SYSDATE, 200, 'Deposit');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (2, 2, SYSDATE, 300, 'Withdrawal');

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

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

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

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

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

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

**Exercise – 1:**

**Scenario 1:**

BEGIN

FOR cust IN (

SELECT CustomerID, DOB

FROM Customers

) LOOP

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

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = cust.CustomerID;

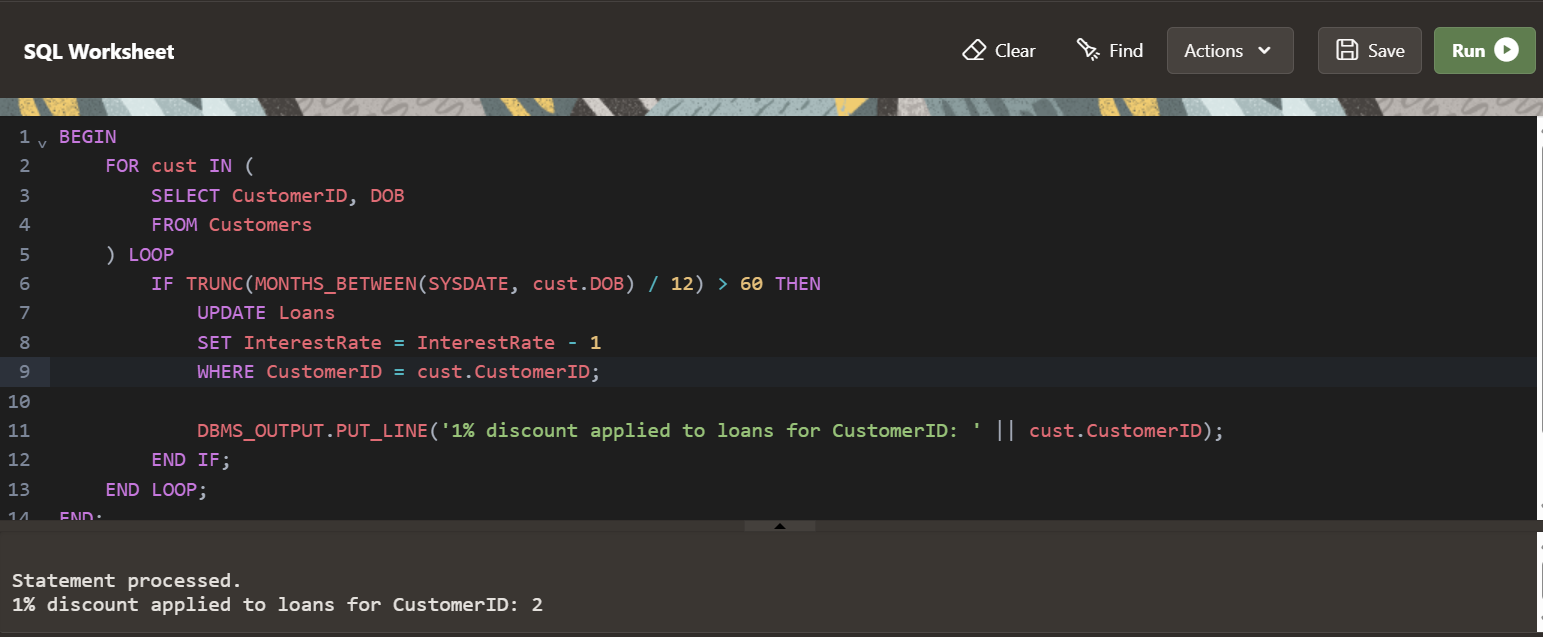
DBMS\_OUTPUT.PUT\_LINE('1% discount applied to loans for CustomerID: ' || cust.CustomerID);

END IF;

END LOOP;

END;

/



**Scenario -2**

ALTER TABLE Customers ADD IsVIP CHAR(1);

BEGIN

FOR cust IN (

SELECT CustomerID, Balance

FROM Customers

) LOOP

IF cust.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'Y'

WHERE CustomerID = cust.CustomerID;

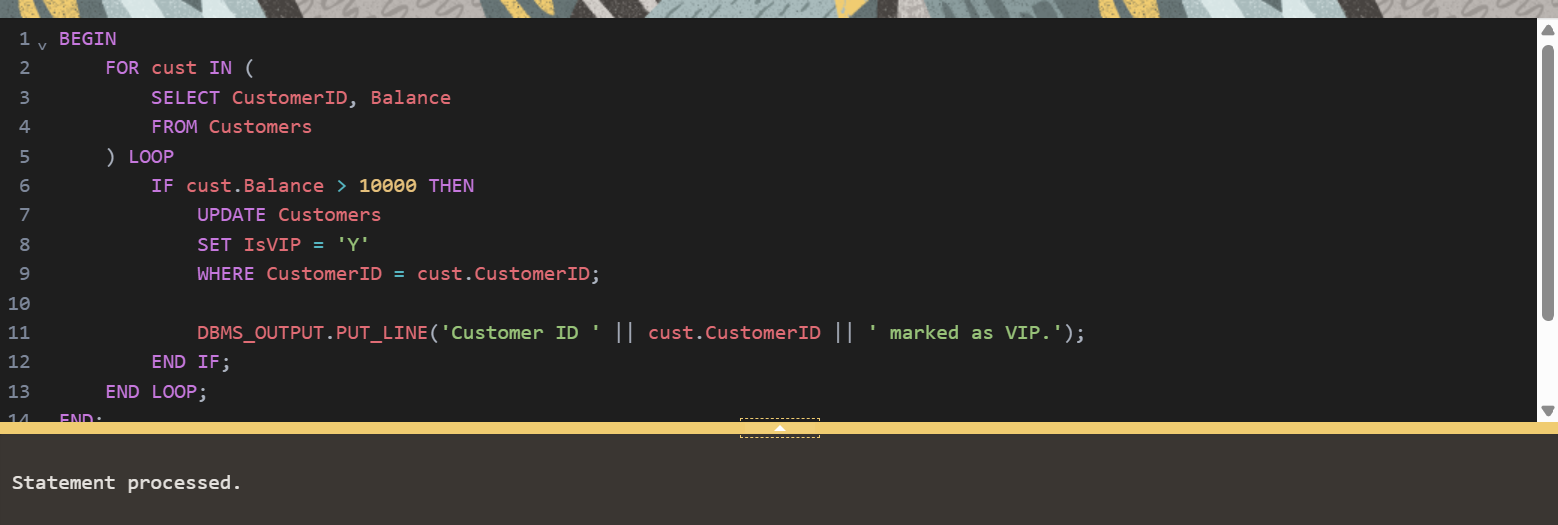
DBMS\_OUTPUT.PUT\_LINE('Customer ID ' || cust.CustomerID || ' marked as VIP.');

END IF;

END LOOP;

END;

/



**Scenario – 3**

BEGIN

FOR loan\_rec IN (

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

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Reminder: Loan ID ' || loan\_rec.LoanID || ' for customer "' || loan\_rec.Name ||

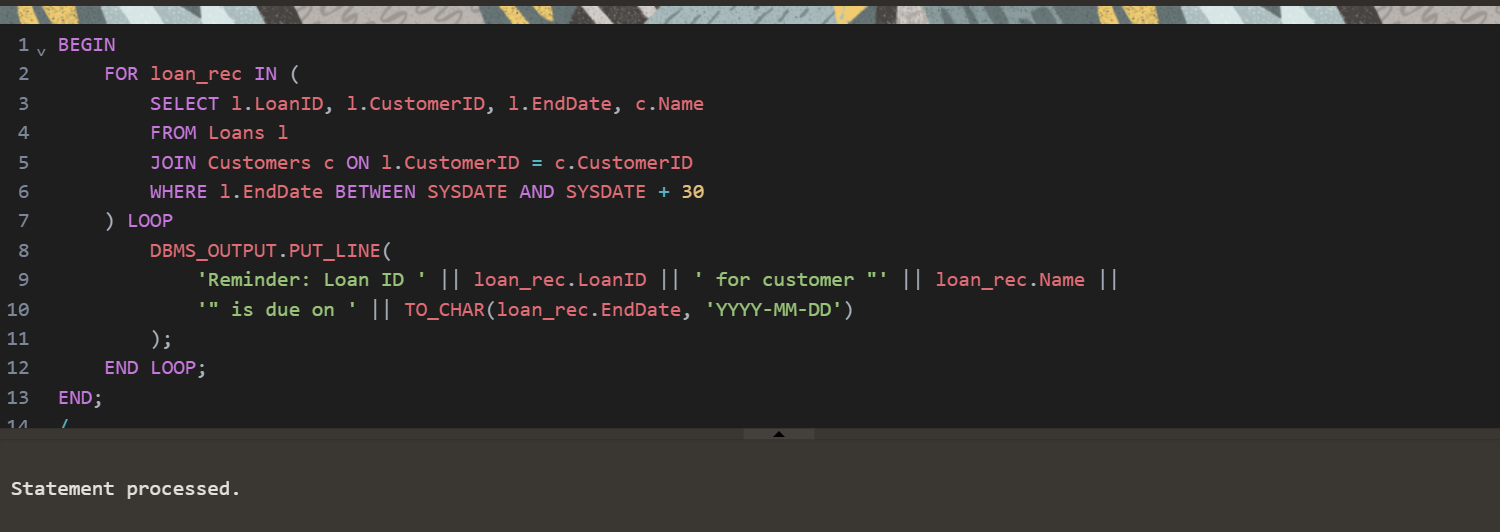
'" is due on ' || TO\_CHAR(loan\_rec.EndDate, 'YYYY-MM-DD')

);

END LOOP;

END;

/



**Exercise – 3**

**Scenario – 1**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR acc IN (

SELECT AccountID, Balance

FROM Accounts

WHERE AccountType = 'Savings'

) LOOP

UPDATE Accounts

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

LastModified = SYSDATE

WHERE AccountID = acc.AccountID;

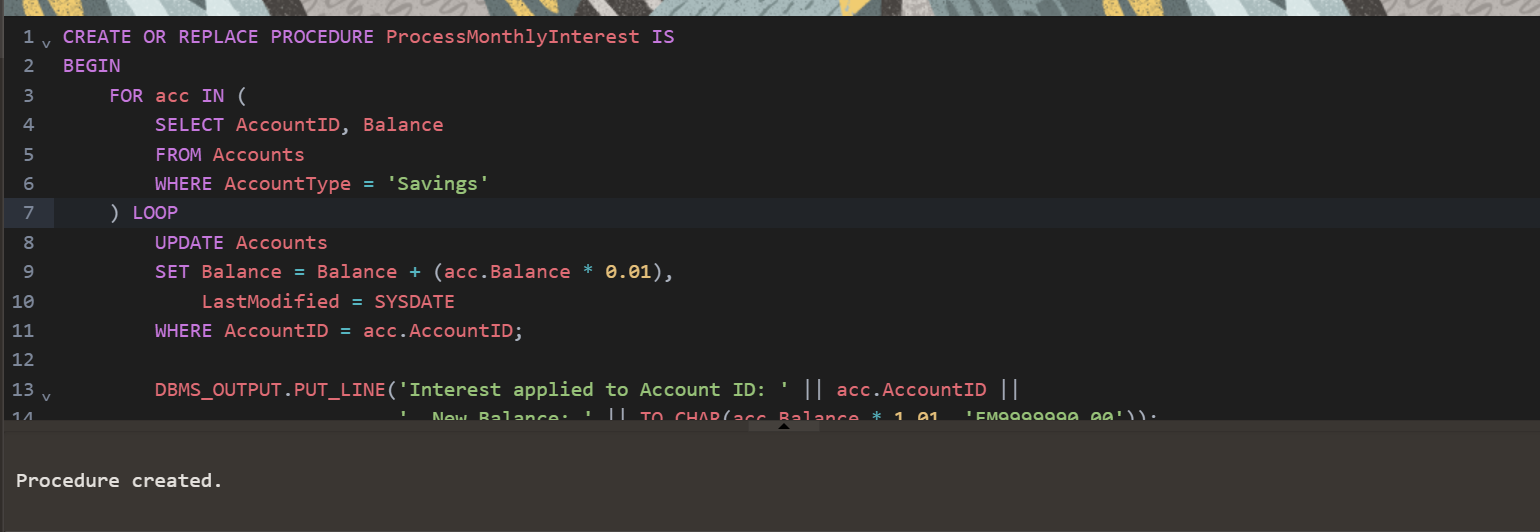
DBMS\_OUTPUT.PUT\_LINE('Interest applied to Account ID: ' || acc.AccountID ||

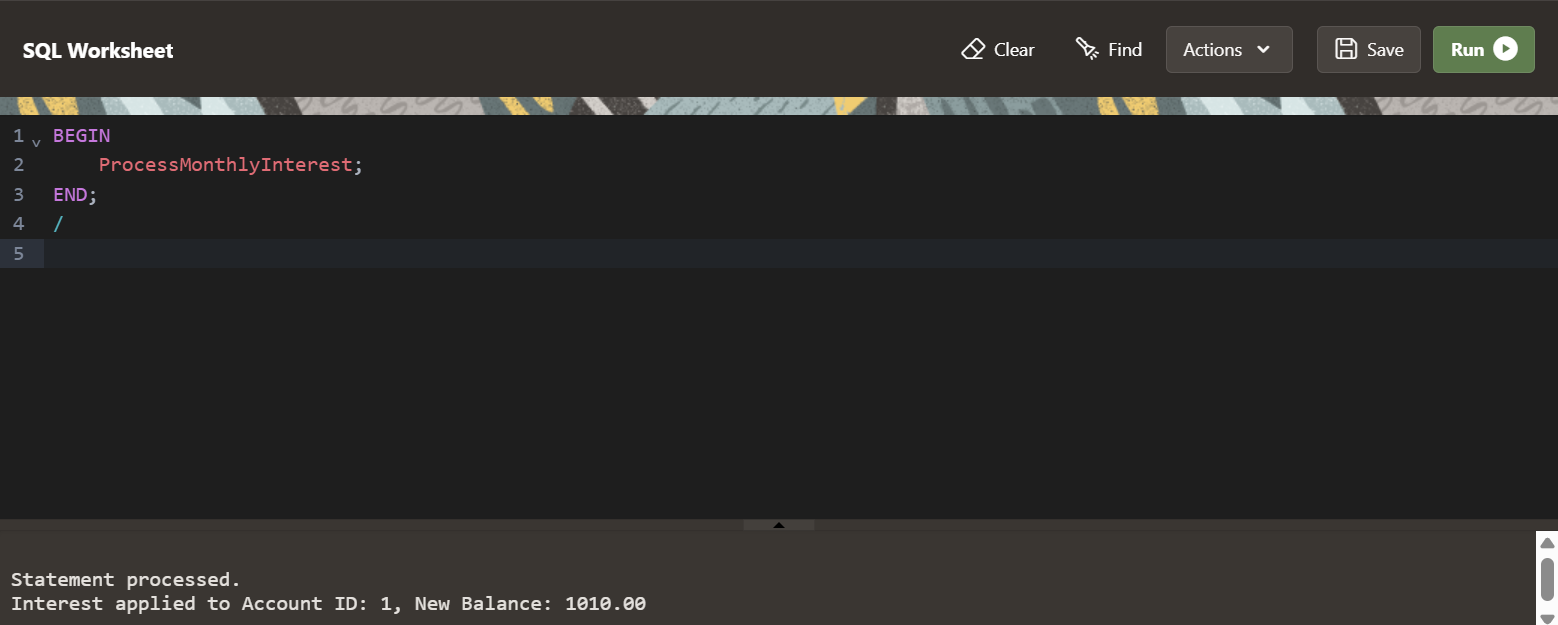
', New Balance: ' || TO\_CHAR(acc.Balance \* 1.01, 'FM9999990.00'));

END LOOP;

END;

/





**Scenario – 2**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_department IN VARCHAR2,

p\_bonus\_percent IN NUMBER -- Pass 10 for 10%

) IS

BEGIN

FOR emp IN (

SELECT EmployeeID, Salary

FROM Employees

WHERE Department = p\_department

) LOOP

UPDATE Employees

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

WHERE EmployeeID = emp.EmployeeID;

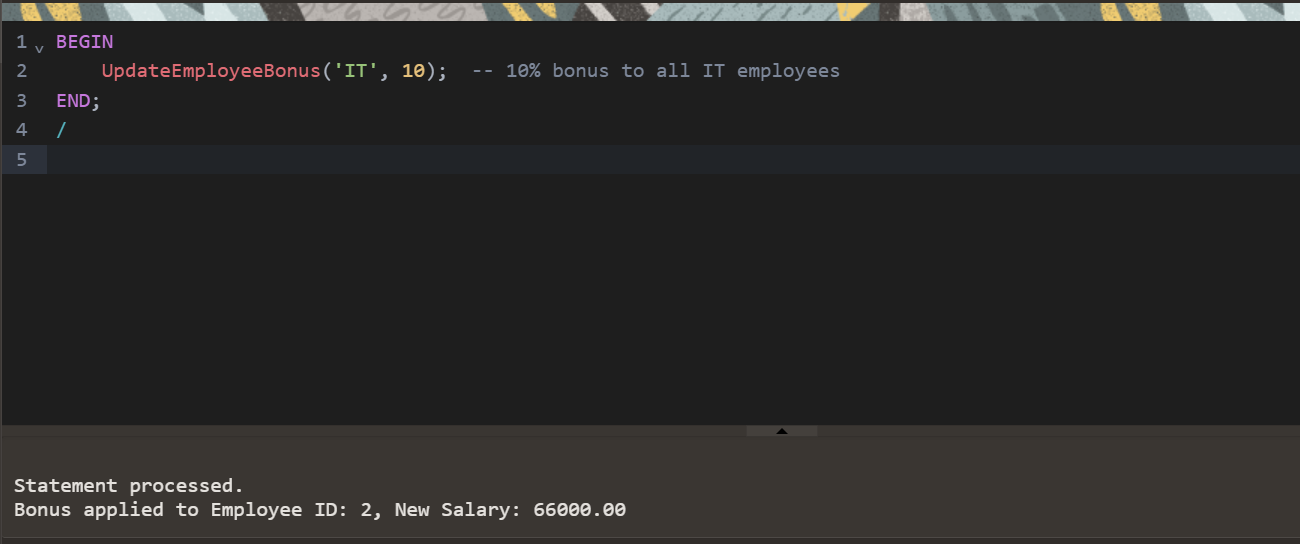
DBMS\_OUTPUT.PUT\_LINE('Bonus applied to Employee ID: ' || emp.EmployeeID ||

', New Salary: ' || TO\_CHAR(emp.Salary \* (1 + p\_bonus\_percent / 100), 'FM9999990.00'));

END LOOP;

END;

/

****

**Scenario – 3**

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_sourceAccountID IN NUMBER,

p\_targetAccountID IN NUMBER,

p\_amount IN NUMBER

) IS

v\_source\_balance NUMBER;

BEGIN

-- Get balance of source account

SELECT Balance INTO v\_source\_balance

FROM Accounts

WHERE AccountID = p\_sourceAccountID;

-- Check for sufficient funds

IF v\_source\_balance < p\_amount THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient balance in source account.');

RETURN;

END IF;

-- Deduct from source account

UPDATE Accounts

SET Balance = Balance - p\_amount,

LastModified = SYSDATE

WHERE AccountID = p\_sourceAccountID;

-- Add to target account

UPDATE Accounts

SET Balance = Balance + p\_amount,

LastModified = SYSDATE

WHERE AccountID = p\_targetAccountID;

DBMS\_OUTPUT.PUT\_LINE('Transfer of $' || p\_amount ||

' from Account ' || p\_sourceAccountID ||

' to Account ' || p\_targetAccountID || ' successful.');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: One or both accounts not found.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Transfer failed due to an unexpected error: ' || SQLERRM);

END;

/

