**Digital Nurture 4.0**

**Week 2-PL/SQL programming**

# Exercise 1: Control Structures

# Creating Tables:

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(50),

DOB DATE,

Gender VARCHAR2(10),

Email VARCHAR2(100),

Phone VARCHAR2(15),

Address VARCHAR2(100),

Balance NUMBER(10,2),

IsVIP VARCHAR2(5)

);

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

LoanAmount NUMBER(10,2),

InterestRate NUMBER(5,2),

LoanType VARCHAR2(20),

EndDate DATE,

Status VARCHAR2(20),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

# Inserting The Values To The Table:

INSERT INTO Customers VALUES (1, 'Tony Stark', TO\_DATE('1955-06-01','YYYY-MM-DD'), 'Male', 'tony@example.com', '9876543210', 'Chennai', 12000, 'FALSE');

INSERT INTO Customers VALUES (2, 'Anita Rao', TO\_DATE('1985-10-15','YYYY-MM-DD'), 'Female', 'anita@example.com', '9123456780', 'Hyderabad', 8500, 'FALSE');

INSERT INTO Customers VALUES (3, 'Rahul Verma', TO\_DATE('1948-03-22','YYYY-MM-DD'), 'Male', 'rahul@example.com', '9988776655', 'Delhi', 13500, 'FALSE');

INSERT INTO Customers VALUES (4, 'Meena Iyer', TO\_DATE('1963-09-10','YYYY-MM-DD'), 'Female', 'meena@example.com', '9012345678', 'Mumbai', 9200, 'FALSE');

INSERT INTO Customers VALUES (5, 'Suresh Kumar', TO\_DATE('1945-12-30','YYYY-MM-DD'), 'Male', 'suresh@example.com', '9000000001', 'Coimbatore', 10800, 'FALSE');

INSERT INTO Loans VALUES (101, 1, 500000, 9.5, 'Personal', SYSDATE + 10, 'Active');

INSERT INTO Loans VALUES (102, 2, 750000, 10.0, 'Home', SYSDATE + 45, 'Active');

INSERT INTO Loans VALUES (103, 3, 200000, 8.5, 'Education', SYSDATE + 20, 'Active');

INSERT INTO Loans VALUES (104, 4, 300000, 10.0, 'Personal', SYSDATE + 18, 'Active');

INSERT INTO Loans VALUES (105, 5, 600000, 11.0, 'Business', SYSDATE + 8, 'Active');

COMMIT;

# **Scenario 1:** The bank wants to apply a discount to loan interest rates for customers above 60 years old.

BEGIN

FOR c IN (

SELECT L.LoanID, L.InterestRate, C.DOB

FROM Loans L

JOIN Customers C ON L.CustomerID = C.CustomerID

) LOOP

IF FLOOR(MONTHS\_BETWEEN(SYSDATE, c.DOB) / 12) > 60 THEN

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE LoanID = c.LoanID;

DBMS\_OUTPUT.PUT\_LINE('1% discount is applied to Loan ID: ' || c.LoanID);

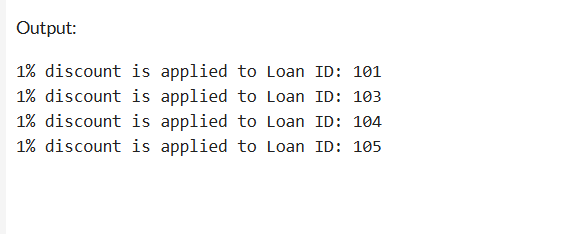
END IF;

END LOOP;

END;

/

# Output:



# **Scenario 2:** A customer can be promoted to VIP status based on their balance.

BEGIN

FOR c IN (SELECT CustomerID, Balance FROM Customers) LOOP

IF c.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = c.CustomerID;

DBMS\_OUTPUT.PUT\_LINE('VIP status granted for Customer ID: ' || c.CustomerID);

END IF;

END LOOP;

END;

/

# Output:

# **Scenario 3:** The bank wants to send reminders to customers whose loans are due within the next 30 days.

BEGIN

FOR r IN (

SELECT L.LoanID, 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 for loan due: Loan ID ' || r.LoanID ||

', Mr/Mrs ' || r.Name ||

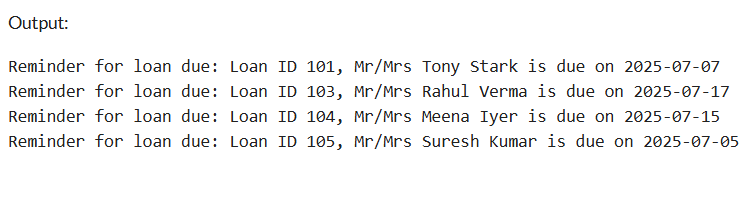
' is due on ' || TO\_CHAR(r.EndDate, 'YYYY-MM-DD'));

END LOOP;

END;

/

# Output:



# Exercise 3: Stored Procedures

# Creating Tables:

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerName VARCHAR2(50),

AccountType VARCHAR2(20),

Balance NUMBER(10,2)

);

CREATE TABLE Employees (

EmpID NUMBER PRIMARY KEY,

EmpName VARCHAR2(50),

Department VARCHAR2(30),

Salary NUMBER(10,2)

);

# Inserting The Values To The Table:

# INSERT INTO Accounts VALUES (1001, 'Hari', 'Savings', 10000);

# INSERT INTO Accounts VALUES (1002, 'Ramya', 'Savings', 15000);

# INSERT INTO Accounts VALUES (1003, 'Kumar', 'Current', 20000);

# INSERT INTO Accounts VALUES (1004, 'Priya', 'Savings', 12000);

# INSERT INTO Employees VALUES (201, 'John', 'IT', 50000);

# INSERT INTO Employees VALUES (202, 'Anu', 'HR', 45000);

# INSERT INTO Employees VALUES (203, 'Rahul', 'IT', 52000);

# INSERT INTO Employees VALUES (204, 'Sneha', 'Finance', 48000);

# COMMIT;

# **Scenario 1:** The bank needs to process monthly interest for all savings accounts.

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

v\_new\_balance NUMBER;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('--- MONTHLY INTEREST PROCESSING STARTED ---');

FOR acc IN (SELECT \* FROM Accounts WHERE AccountType = 'Savings') LOOP

v\_new\_balance := acc.Balance + (acc.Balance \* 0.01);

UPDATE Accounts

SET Balance = v\_new\_balance

WHERE AccountID = acc.AccountID;

DBMS\_OUTPUT.PUT\_LINE('Interest applied to AccountID: ' || acc.AccountID || ' | New Balance: ' || TO\_CHAR(v\_new\_balance, '99999.99'));

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('--- PROCESSING COMPLETED ---');

END;

/

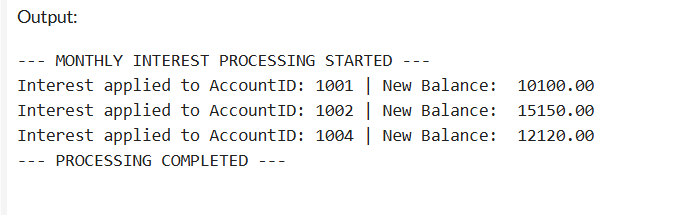
BEGIN

ProcessMonthlyInterest;

END;

/

# Output:



# **Scenario 2:** The bank wants to implement a bonus scheme for employees based on their performance.

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_dept IN VARCHAR2,

p\_bonus\_percent IN NUMBER

) IS

v\_new\_salary NUMBER;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('--- BONUS UPDATE STARTED ---');

DBMS\_OUTPUT.PUT\_LINE('Department : ' || p\_dept);

DBMS\_OUTPUT.PUT\_LINE('Bonus Percentage : ' || p\_bonus\_percent || '%');

DBMS\_OUTPUT.PUT\_LINE('-----------------------------------------');

FOR emp IN (SELECT \* FROM Employees WHERE Department = p\_dept) LOOP

v\_new\_salary := emp.Salary + (emp.Salary \* p\_bonus\_percent / 100);

UPDATE Employees

SET Salary = v\_new\_salary

WHERE EmpID = emp.EmpID;

DBMS\_OUTPUT.PUT\_LINE('Bonus applied to: ' || emp.EmpName || ' (EmpID: ' || emp.EmpID || ') | New Salary: ' || TO\_CHAR(v\_new\_salary, '99999.99'));

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('--- BONUS UPDATE COMPLETED ---');

END;

/

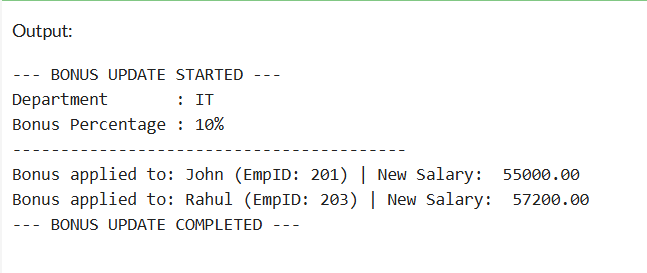
BEGIN

UpdateEmployeeBonus('IT', 10);

END;

/

# Output:



# **Scenario 3:** Customers should be able to transfer funds between their accounts.

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance FROM Accounts WHERE AccountID = p\_from\_account;

DBMS\_OUTPUT.PUT\_LINE('--- INITIATING FUNDS TRANSFER ---');

DBMS\_OUTPUT.PUT\_LINE('From Account : ' || p\_from\_account);

DBMS\_OUTPUT.PUT\_LINE('To Account : ' || p\_to\_account);

DBMS\_OUTPUT.PUT\_LINE('Amount : Rs.' || p\_amount);

IF v\_balance < p\_amount THEN

DBMS\_OUTPUT.PUT\_LINE(' Transfer failed: Insufficient funds in Account ID ' || p\_from\_account);

ELSE

UPDATE Accounts SET Balance = Balance - p\_amount WHERE AccountID = p\_from\_account;

UPDATE Accounts SET Balance = Balance + p\_amount WHERE AccountID = p\_to\_account;

DBMS\_OUTPUT.PUT\_LINE(' Transfer successful!');

DBMS\_OUTPUT.PUT\_LINE('Rs.' || p\_amount || ' transferred from Account ' || p\_from\_account || ' to ' || p\_to\_account);

END IF;

DBMS\_OUTPUT.PUT\_LINE('--- TRANSFER PROCESS COMPLETED ---');

END;

/

BEGIN

TransferFunds(1001, 1002, 2000);

END;

/

# Output:

