**Digital Nurture 4.0 Java FSE**

**Week 2 PL/SQL Exercises**

**(Mandatory)**

**Exercise 01: Control Structures**

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

**Question:** Write a PL/SQL block that loops through all customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

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

**Question:** Write a PL/SQL block that iterates through all customers and sets a flag IsVIP to TRUE for those with a balance over $10,000.

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

**Question:** Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

**//Creating CUSTOMER\_NEW table//**

CREATE TABLE CUSTOMER\_NEW (

CID NUMBER PRIMARY KEY,

CUSTOMER\_NAME VARCHAR2(100),

AGE NUMBER,

BALANCE NUMBER(10,2),

IS\_VIP CHAR(1),

LOAN\_INTEREST NUMBER(5,2)

);

**//Creating LOAN\_NEW table//**

CREATE TABLE LOAN\_NEW (

LOAN\_ID NUMBER PRIMARY KEY,

CID NUMBER REFERENCES CUSTOMER\_NEW(CID),

DUE\_DATE DATE,

INTEREST\_RATE NUMBER(5,2)

);

**//Insert values//**

INSERT INTO CUSTOMER\_NEW VALUES (1, 'Arjun', 65, 12000, 'N', 9.5);

INSERT INTO CUSTOMER\_NEW VALUES (2, 'Deshva', 45, 8000, 'N', 8.0);

INSERT INTO CUSTOMER\_NEW VALUES (3, 'Kishonika', 71, 15000, 'N', 9.2);

INSERT INTO CUSTOMER\_NEW VALUES (4, 'Dhasvika', 62, 3000, 'N', 8.9);

INSERT INTO LOAN\_NEW VALUES (101, 1, SYSDATE + 10, 9.5);

INSERT INTO LOAN\_NEW VALUES (102, 2, SYSDATE + 40, 8.0);

INSERT INTO LOAN\_NEW VALUES (103, 3, SYSDATE + 25, 9.2);

INSERT INTO LOAN\_NEW VALUES (104, 4, SYSDATE + 5, 8.9);

COMMIT;

**//Enabling DBMS\_OUTPUT//**

SET SERVEROUTPUT ON;

**//PL/SQL block//**

BEGIN

**//Scenario 1: Apply 1% discount on interest rate for age > 60//**

FOR c IN (

SELECT CID, CUSTOMER\_NAME

FROM CUSTOMER\_NEW

WHERE AGE > 60

) LOOP

UPDATE LOAN\_NEW

SET INTEREST\_RATE = INTEREST\_RATE - 1

WHERE CID = c.CID;

DBMS\_OUTPUT.PUT\_LINE('Interest discount applied for Customer: ' || c.CUSTOMER\_NAME);

END LOOP;

**//Scenario 2: Promote to VIP if balance > 10000//**

FOR c IN (

SELECT CID, CUSTOMER\_NAME

FROM CUSTOMER\_NEW

WHERE BALANCE > 10000

) LOOP

UPDATE CUSTOMER\_NEW

SET IS\_VIP = 'Y'

WHERE CID = c.CID;

DBMS\_OUTPUT.PUT\_LINE('Promoted to VIP: ' || c.CUSTOMER\_NAME);

END LOOP;

**//Scenario 3: Reminder for loans due in next 30 days//**

FOR l IN (

SELECT L.LOAN\_ID, L.DUE\_DATE, C.CUSTOMER\_NAME

FROM LOAN\_NEW L

JOIN CUSTOMER\_NEW C ON C.CID = L.CID

WHERE L.DUE\_DATE <= SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: ' || l.CUSTOMER\_NAME ||

', your loan ID ' || l.LOAN\_ID ||

' is due on ' || TO\_CHAR(l.DUE\_DATE, 'DD-MON-YYYY'));

END LOOP;

COMMIT;

END;

/

**Output:**

Interest discount applied for Customer: Arjun

Interest discount applied for Customer: Kishonika

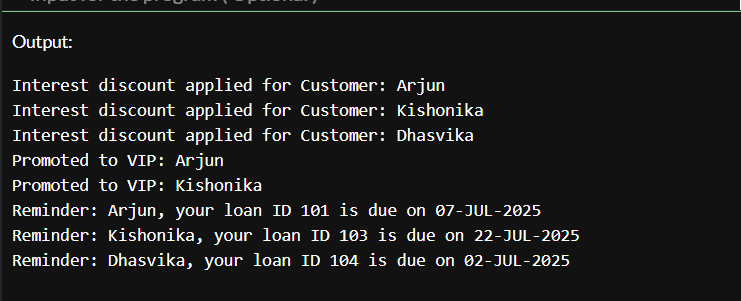
Interest discount applied for Customer: Dhasvika

Promoted to VIP: Arjun

Promoted to VIP: Kishonika

Reminder: Arjun, your loan ID 101 is due on 07-JUL-2025

Reminder: Kishonika, your loan ID 103 is due on 22-JUL-2025

Reminder: Dhasvika, your loan ID 104 is due on 02-JUL-2025

**Exercise 03: Stored Procedures**

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

**Question:** Write a stored procedure ProcessMonthlyInterest that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.

**Code:**

SET SERVEROUTPUT ON;

EXEC ProcessMonthlyInterest;

CREATE TABLE SavingsAccounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

Balance NUMBER(12,2),

AccountType VARCHAR2(20)

);

INSERT INTO SavingsAccounts VALUES (101, 1, 50000, 'SAVINGS');

INSERT INTO SavingsAccounts VALUES (102, 2, 35000, 'SAVINGS');

INSERT INTO SavingsAccounts VALUES (103, 3, 20000, 'CHECKING');

INSERT INTO SavingsAccounts VALUES (104, 4, 60000, 'SAVINGS');

COMMIT;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

UPDATE SavingsAccounts

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

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

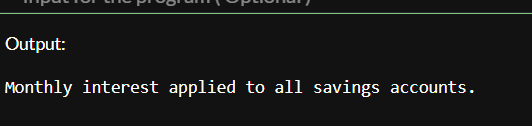
COMMIT;

END;

/

**Output:**

Monthly interest applied to all savings accounts.



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

**Question:** Write a stored procedure **UpdateEmployeeBonus** that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

**Code:**

SET SERVEROUTPUT ON;

EXEC UpdateEmployeeBonus(101, 10);

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DepartmentID NUMBER,

Salary NUMBER(10,2),

Performance VARCHAR2(20)

);

INSERT INTO Employees VALUES (1, 'Arjun', 101, 50000, 'Excellent');

INSERT INTO Employees VALUES (2, 'Deshva', 102, 45000, 'Good');

INSERT INTO Employees VALUES (3, 'Kishonika', 101, 52000, 'Excellent');

INSERT INTO Employees VALUES (4, 'Dhasvika', 103, 40000, 'Average');

COMMIT;

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_dept\_id IN NUMBER,

p\_bonus\_percent IN NUMBER

) AS

BEGIN

UPDATE Employees

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

WHERE DepartmentID = p\_dept\_id;

DBMS\_OUTPUT.PUT\_LINE('Bonus of ' || p\_bonus\_percent || '% applied to Department ' || p\_dept\_id);

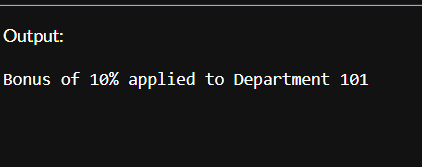
COMMIT;

END;

/

**Output:**

Bonus of 10% applied to Department 101



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

**Question:** Write a stored procedure **TransferFunds** that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.

**Code:**

SET SERVEROUTPUT ON;

EXEC TransferFunds(2001, 2002, 5000);

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

Balance NUMBER(12,2)

);

INSERT INTO Accounts VALUES (2001, 1, 15000);

INSERT INTO Accounts VALUES (2002, 2, 8000);

INSERT INTO Accounts VALUES (2003, 3, 12000);

INSERT INTO Accounts VALUES (2004, 4, 5000);

COMMIT;

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) AS

v\_balance NUMBER;

BEGIN

**//Get source account balance//**

SELECT Balance INTO v\_balance

FROM Accounts

WHERE AccountID = p\_from\_account;

-- Check sufficient balance

IF v\_balance < p\_amount THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient balance.');

END IF;

**//Debit from source//**

UPDATE Accounts

SET Balance = Balance - p\_amount

WHERE AccountID = p\_from\_account;

**// Credit to target//**

UPDATE Accounts

SET Balance = Balance + p\_amount

WHERE AccountID = p\_to\_account;

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

COMMIT;

END;

/

**Output:**

???5000 transferred from 2001 to 2002

