**CTS DIGITAL NURTURE - 4.0 JAVA FSE**

**WEEK 2 – PL \ SQL**

MANDATORY HANDS-ON 1:

**Exercise 1: Control Structures**

PROGRAM:

**TABLE CREATION:**

SQL> CREATE TABLE Customers (

2 CustomerID NUMBER PRIMARY KEY,

3 Name VARCHAR2(100),

4 Age NUMBER,

5 Balance NUMBER,

6 InterestRate NUMBER,

7 IsVIP VARCHAR2(5)

8 );

Table created.

SQL> CREATE TABLE Loans (

2 LoanID NUMBER PRIMARY KEY,

3 CustomerID NUMBER,

4 DueDate DATE,

5 FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

6 );

Table created.

**INSERT DATA TO TABLE:**

SQL> INSERT INTO Customers VALUES (1, 'Ramya', 65, 15000, 8.5, 'FALSE');

1 row created.

SQL> INSERT INTO Customers VALUES (2, 'Kiran', 45, 9000, 9.0, 'FALSE');

1 row created.

SQL> INSERT INTO Loans VALUES (101, 1, SYSDATE + 10);

1 row created.

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

1 row created.

**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.**

PROGRAM:

SQL> BEGIN

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

3 UPDATE Customers

4 SET InterestRate = InterestRate - 1

5 WHERE CustomerID = rec.CustomerID;

6 END LOOP;

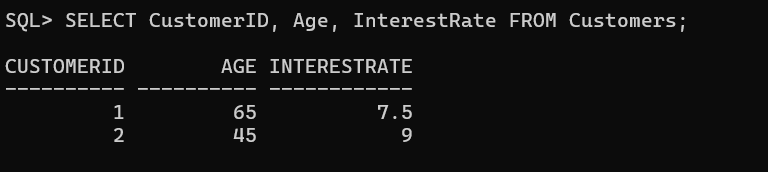
7 COMMIT;

8 END;

9 /

PL/SQL procedure successfully completed.

OUTPUT:



**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.**

PROGRAM:

SQL> BEGIN

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

3 UPDATE Customers

4 SET IsVIP = 'TRUE'

5 WHERE CustomerID = rec.CustomerID;

6 END LOOP;

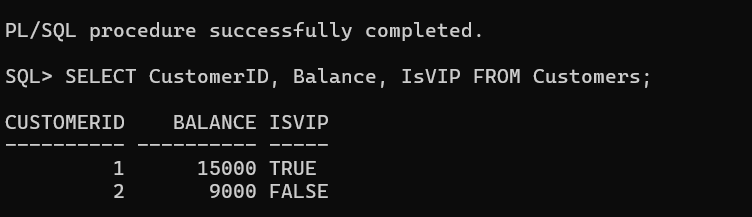
7 COMMIT;

8 END;

9 /

PL/SQL procedure successfully completed.

OUTPUT:



**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.**

PROGRAM:

SQL> SET SERVEROUTPUT ON;

SQL> BEGIN

2 FOR rec IN (

3 SELECT CustomerID, LoanID, DueDate

4 FROM Loans

5 WHERE DueDate BETWEEN SYSDATE AND SYSDATE + 30

6 ) LOOP

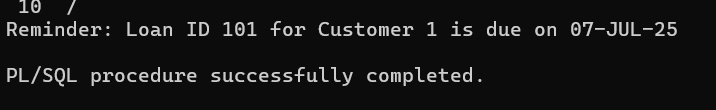
7 DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || rec.LoanID || ' for Customer ' || rec.CustomerID || ' is due on ' || rec.DueDate);

8 END LOOP;

9 END;

10 /

OUTPUT:



MANDATORY HANDS-ON 2:

QUESTION:

**Exercise 3: Stored Procedures**

**TABLE CREATION**

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

AccountType VARCHAR2(20),

Balance NUMBER

);

Table created.

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DepartmentID NUMBER,

Salary NUMBER

);

Table created.

**DATA INSERTION**

INSERT INTO Accounts VALUES (1001, 'Savings', 5000);

1 row created.

INSERT INTO Accounts VALUES (1002, 'Savings', 8000);

1 row created.

INSERT INTO Accounts VALUES (1003, 'Current', 12000);

1 row created

INSERT INTO Employees VALUES (1, 'John', 10, 40000);

1 row created.

INSERT INTO Employees VALUES (2, 'Alice', 10, 45000);

1 row created.

INSERT INTO Employees VALUES (3, 'Mark', 20, 50000);

1 row created.

**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.**

PROGRAM:

SQL> CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

2 BEGIN

3 FOR rec IN (SELECT AccountID, Balance FROM Accounts WHERE AccountType = 'Savings') LOOP

4 UPDATE Accounts

5 SET Balance = rec.Balance + (rec.Balance \* 0.01)

6 WHERE AccountID = rec.AccountID;

7 END LOOP;

8 COMMIT;

9 END;

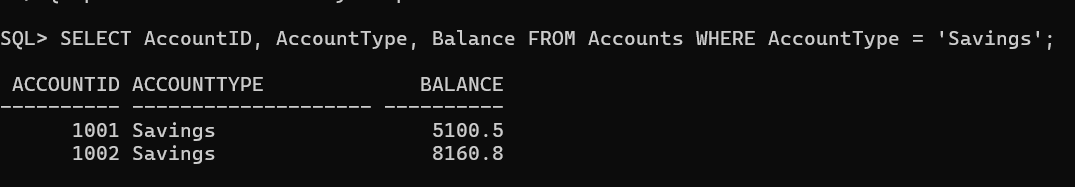
10 /

Procedure created.

SQL> EXEC ProcessMonthlyInterest;

PL/SQL procedure successfully completed.

OUTPUT:



**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.**

PROGRAM:

SQL> CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

2 dept\_id IN NUMBER,

3 bonus\_pct IN NUMBER

4 ) IS

5 BEGIN

6 UPDATE Employees

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

8 WHERE DepartmentID = dept\_id;

9 COMMIT;

10 END;

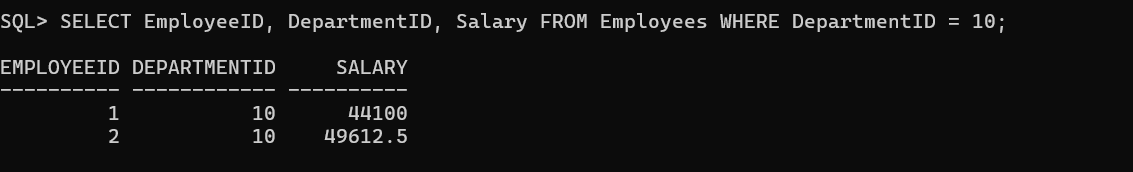
11 /

Procedure created.

SQL> EXEC UpdateEmployeeBonus(10, 5);

PL/SQL procedure successfully completed.

OUTPUT:



**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.**

PROGRAM:

SQL> CREATE OR REPLACE PROCEDURE TransferFunds (

2 from\_acc IN NUMBER,

3 to\_acc IN NUMBER,

4 amount IN NUMBER

5 ) IS

6 from\_balance NUMBER;

7 BEGIN

8 -- Get balance of source account

9 SELECT Balance INTO from\_balance FROM Accounts WHERE AccountID = from\_acc;

10

11 -- Check sufficient balance

12 IF from\_balance >= amount THEN

13 -- Deduct from source

14 UPDATE Accounts

15 SET Balance = Balance - amount

16 WHERE AccountID = from\_acc;

17

18 -- Add to destination

19 UPDATE Accounts

20 SET Balance = Balance + amount

21 WHERE AccountID = to\_acc;

22

23 COMMIT;

24 ELSE

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

26 END IF;

27 END;

28 /

Procedure created.

SQL> EXEC TransferFunds(1001, 1002, 500);

PL/SQL procedure successfully completed.

OUTPUT:

