**WEEK2\_PLSQL\_HANDSON**

**Exercise 1: 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 Is VIP 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.

**Step1: Create the CUSTOMERS table:**

CREATE TABLE CUSTOMERS\_FINAL (

    CustID NUMBER,

    Name VARCHAR2(50),

    Age NUMBER,

    Balance NUMBER,

    IsVIP VARCHAR2(5),

    InterestRate NUMBER

);

**Step 2: Create Loan Table:**

Create new LOAN table

CREATE TABLE LOANS\_FINAL (

    LoanID NUMBER,

    CustID NUMBER,

    DueDate DATE

);

**Step3: Insert sample data:**

 INSERT INTO CUSTOMERS\_FINAL VALUES (1, 'Deepika', 20, 20000, 'FALSE', 10);

    INSERT INTO CUSTOMERS\_FINAL VALUES (2, 'Nandini', 22, 8000, 'FALSE', 12);

    INSERT INTO CUSTOMERS\_FINAL VALUES (3, 'Suvarna', 65, 20000, 'FALSE', 9);

    INSERT INTO CUSTOMERS\_FINAL VALUES (4, 'Ravi', 70, 15000, 'FALSE', 11);

    INSERT INTO LOANS\_FINAL VALUES (101, 1, SYSDATE + 15);

    INSERT INTO LOANS\_FINAL VALUES (102, 2, SYSDATE + 40);

    INSERT INTO LOANS\_FINAL VALUES (103, 3, SYSDATE + 5);

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

BEGIN

   FOR rec IN (SELECT CustID FROM CUSTOMERS\_FINAL WHERE Age > 60) LOOP

      UPDATE CUSTOMERS\_FINAL

      SET InterestRate = InterestRate - 1

      WHERE CustID = rec.CustID;

   END LOOP;

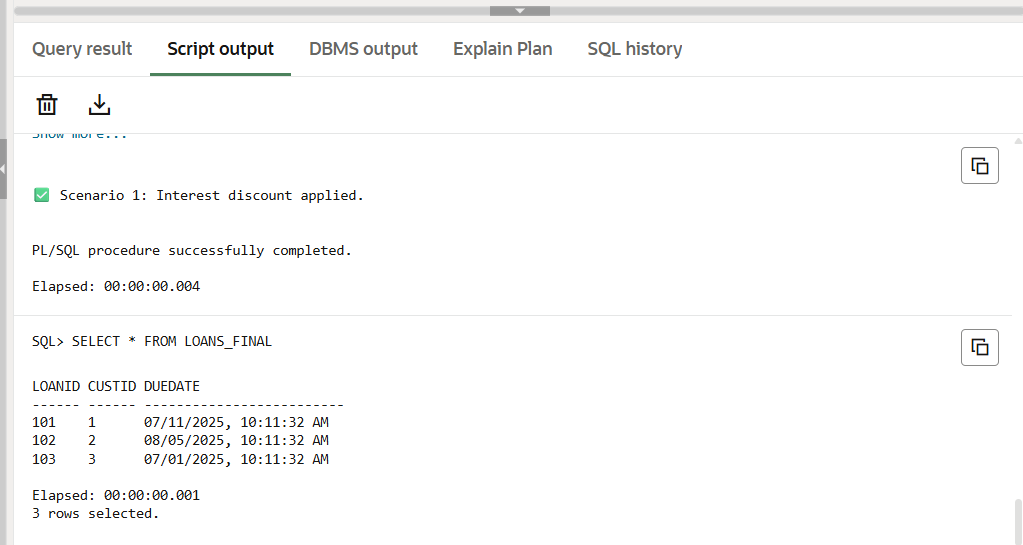
   DBMS\_OUTPUT.PUT\_LINE('Scenario 1: Interest discount applied.');

END;

/

SELECT \* FROM LOANS\_FINAL;

**Output for Scenario 1:**

****

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

BEGIN

   FOR rec IN (SELECT CustID FROM CUSTOMERS\_FINAL WHERE Balance > 10000) LOOP

      UPDATE CUSTOMERS\_FINAL

      SET IsVIP = 'TRUE'

      WHERE CustID = rec.CustID;

   END LOOP;

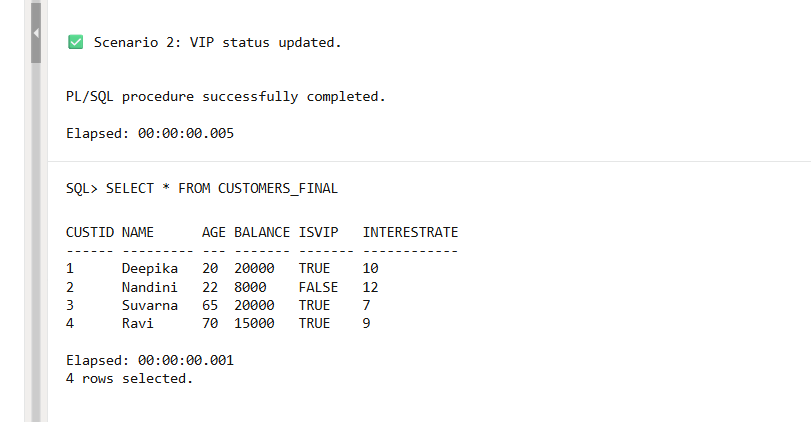
   DBMS\_OUTPUT.PUT\_LINE('✅ Scenario 2: VIP status updated.');

END;

/

SELECT \* FROM CUSTOMERS\_FINAL;

**Output for Scenario2:**



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

BEGIN

FOR rec IN (

SELECT L.LoanID, C.Name, L.DueDate

FROM LOANS\_FINAL L

JOIN CUSTOMERS\_FINAL C ON C.CustID = L.CustID

WHERE L.DueDate <= SYSDATE + 30

) LOOP

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

' for customer ' || rec.Name ||

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

END LOOP;

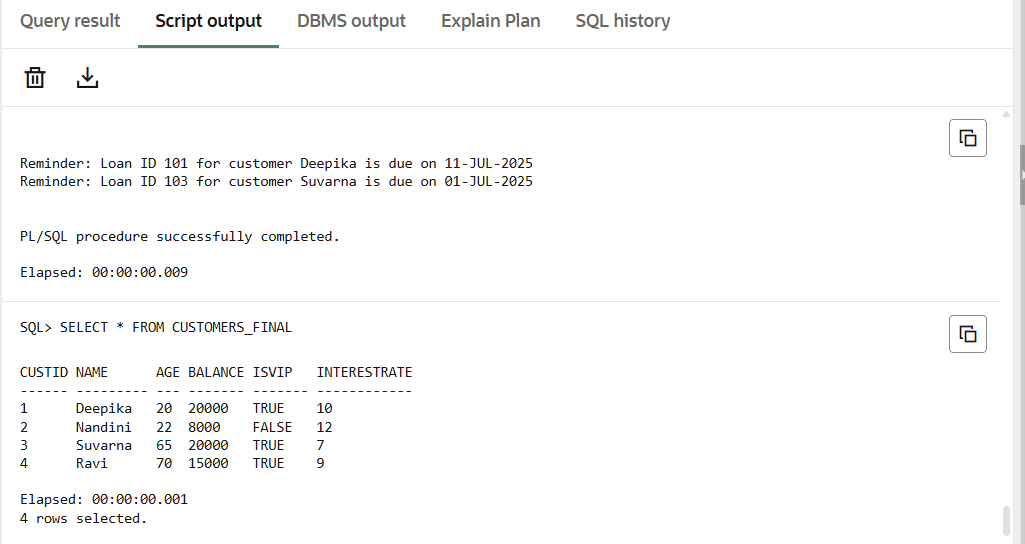
END;

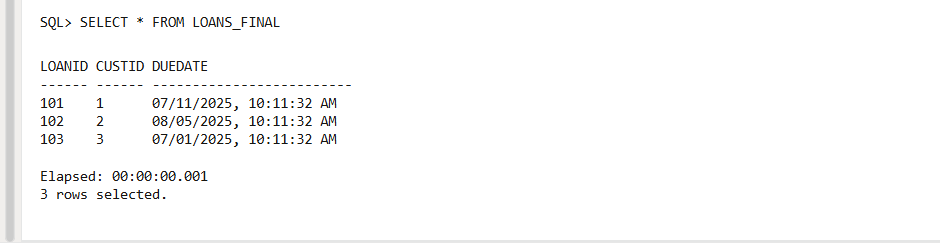
/

SELECT \* FROM CUSTOMERS\_FINAL;

SELECT \* FROM LOANS\_FINAL;

**Output for Scenario3:**





**Exercise 3: 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.

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.

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.

**STEP1: Creating Tables :**

**-- Table for savings accounts**

CREATE TABLE ACCOUNTS\_PROC (

AccID NUMBER,

CustName VARCHAR2(50),

Balance NUMBER

);

/

**-- Table for employees**

CREATE TABLE EMPLOYEES\_PROC (

EmpID NUMBER,

Name VARCHAR2(50),

Department VARCHAR2(30),

Salary NUMBER

);

/

**STEP2:** **Insert Sample Data:**

BEGIN

-- Accounts

INSERT INTO ACCOUNTS\_PROC VALUES (1, 'Deepika', 10000);

INSERT INTO ACCOUNTS\_PROC VALUES (2, 'Nandini', 20000);

INSERT INTO ACCOUNTS\_PROC VALUES (3, 'Suvarna', 15000);

-- Employees

INSERT INTO EMPLOYEES\_PROC VALUES (101, 'Asha', 'IT', 50000);

INSERT INTO EMPLOYEES\_PROC VALUES (102, 'Bhavna', 'HR', 40000);

INSERT INTO EMPLOYEES\_PROC VALUES (103, 'Chetan', 'IT', 55000);

END;

/

**Scenario 1: Procedure to Apply Monthly Interest:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR rec IN (SELECT AccID, Balance FROM ACCOUNTS\_PROC) LOOP

UPDATE ACCOUNTS\_PROC

SET Balance = Balance + (Balance \* 0.01)

WHERE AccID = rec.AccID;

END LOOP;

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

END;

/

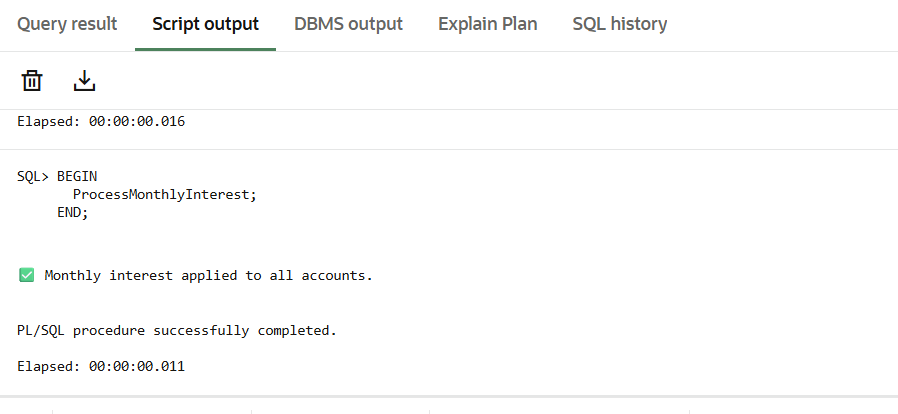
BEGIN

ProcessMonthlyInterest;

END;

/

**Output for Scenario1:**



**Scenario 2: Procedure to Apply Bonus by Department:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_dept IN VARCHAR2,

p\_bonus\_percent IN NUMBER

) IS

BEGIN

UPDATE EMPLOYEES\_PROC

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

WHERE Department = p\_dept;

DBMS\_OUTPUT.PUT\_LINE('✅ Bonus applied to department: ' || p\_dept);

END;

/

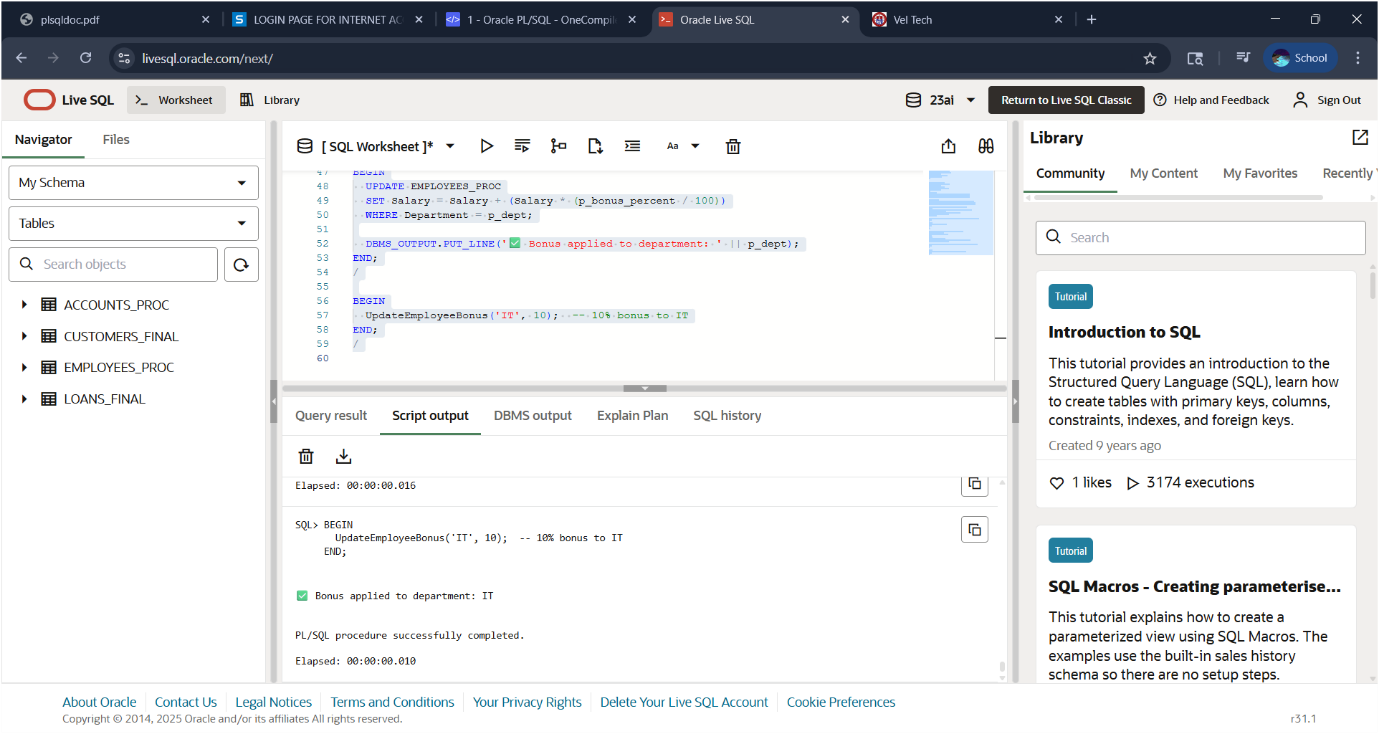
BEGIN

UpdateEmployeeBonus('IT', 10); -- 10% bonus to IT

END;

/

**Output for Scenario 2:**



**Scenario 3: Transfer Funds Between Accounts**

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_from\_acc IN NUMBER,

p\_to\_acc IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

-- Check source balance

SELECT Balance INTO v\_balance FROM ACCOUNTS\_PROC WHERE AccID = p\_from\_acc;

IF v\_balance < p\_amount THEN

DBMS\_OUTPUT.PUT\_LINE('❌ Insufficient balance in source account.');

ELSE

UPDATE ACCOUNTS\_PROC SET Balance = Balance - p\_amount WHERE AccID = p\_from\_acc;

UPDATE ACCOUNTS\_PROC SET Balance = Balance + p\_amount WHERE AccID = p\_to\_acc;

DBMS\_OUTPUT.PUT\_LINE('✅ Transferred ₹' || p\_amount ||

' from Account ' || p\_from\_acc ||

' to Account ' || p\_to\_acc);

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('❌ One or both account IDs not found.');

END;

/

BEGIN

TransferFunds(1, 2, 3000); -- Transfer ₹3000 from Acc 1 to Acc 2

END;

/

SELECT \* FROM ACCOUNTS\_PROC;

SELECT \* FROM EMPLOYEES\_PROC;

**Output for Scenario3:**

