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
);
```

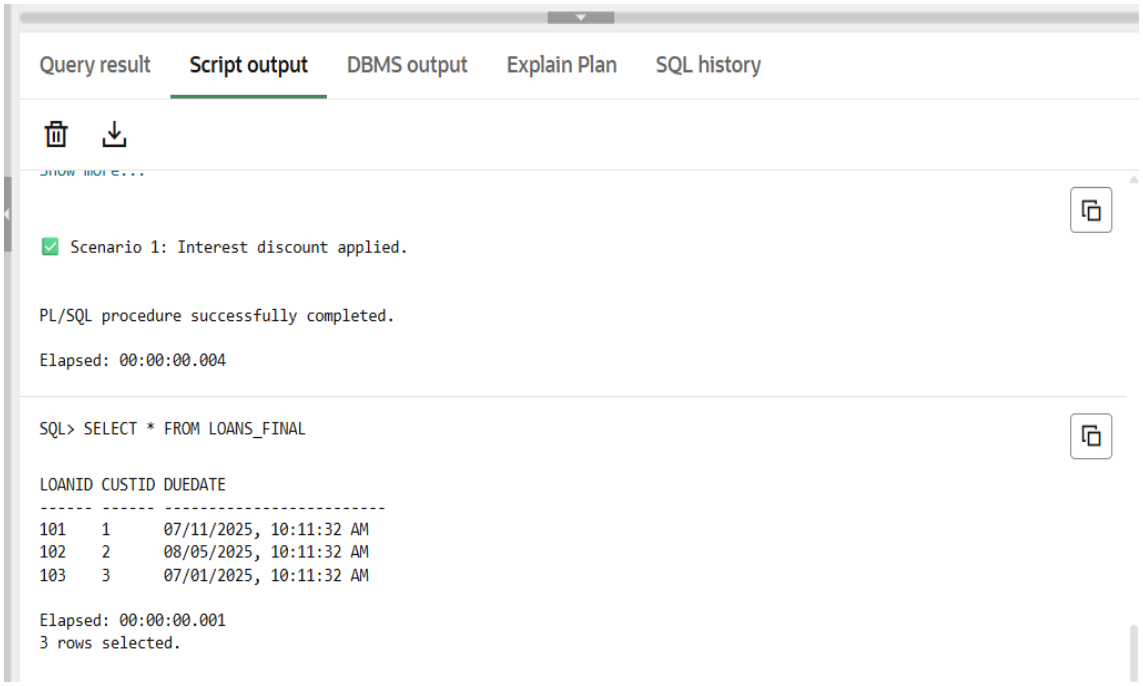
Step3: Insert sample data:

```
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 2: VIP status updated.

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.005

SQL> SELECT * FROM CUSTOMERS_FINAL

CUSTID NAME      AGE BALANCE ISVIP   INTERESTRATE
------ --------- --- ------- ------- ------------
1      Deepika   20  20000   TRUE    10
2      Nandini   22  8000    FALSE   12
3      Suvarna   65  20000   TRUE    7
4      Ravi      70  15000   TRUE    9

Elapsed: 00:00:00.001
4 rows selected.
```

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:

Query result | **Script output** | DBMS output | Explain Plan | SQL history

```
Reminder: Loan ID 101 for customer Deepika is due on 11-JUL-2025
Reminder: Loan ID 103 for customer Suvarna is due on 01-JUL-2025


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.009
```

```
SQL> SELECT * FROM CUSTOMERS_FINAL

CUSTID NAME      AGE BALANCE ISVIP   INTERESTRATE
------ --------- --- ------- ------- ------------
1      Deepika   20  20000   TRUE    10
2      Nandini   22  8000    FALSE   12
3      Suvarna   65  20000   TRUE    7
4      Ravi      70  15000   TRUE    9

Elapsed: 00:00:00.001
4 rows selected.
```

```
SQL> SELECT * FROM LOANS_FINAL

LOANID CUSTID DUEDATE
------ ------ -------------------------
101    1      07/11/2025, 10:11:32 AM
102    2      08/05/2025, 10:11:32 AM
103    3      07/01/2025, 10:11:32 AM

Elapsed: 00:00:00.001
3 rows selected.
```

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:

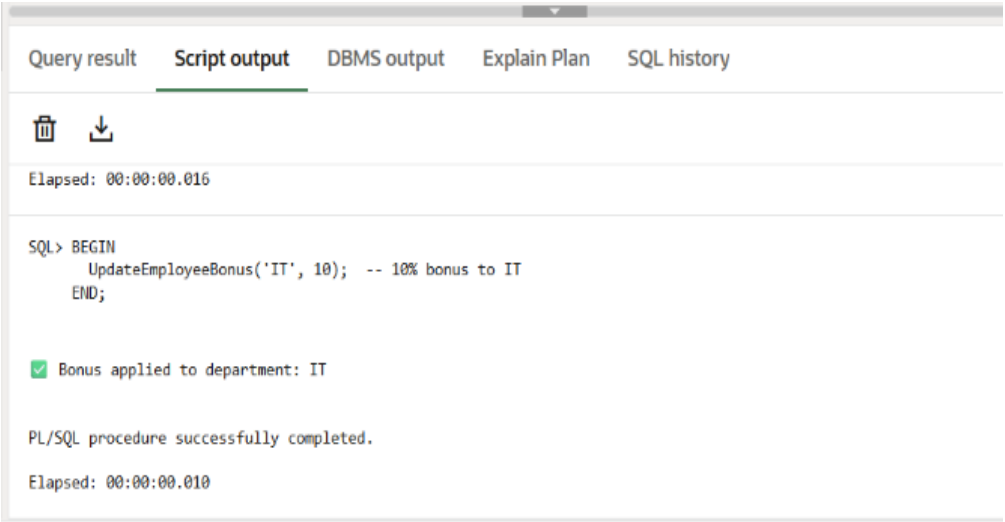
Query result | **Script output** | DBMS output | Explain Plan | SQL history

```
Elapsed: 00:00:00.016

SQL> BEGIN
       ProcessMonthlyInterest;
     END;


✅ Monthly interest applied to all accounts.


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.011
```

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: