# 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.

CODE:

DECLARE

  v\_c\_id Customers.CustomerID%TYPE;

  v\_dob Customers.DOB%TYPE;

  v\_age **NUMBER**;

BEGIN

  FOR c\_rec IN (SELECT CustomerID, DOB FROM Customers) LOOP

    v\_c\_id := c\_rec.CustomerID;

    v\_dob := c\_rec.DOB;

    v\_age := FLOOR(MONTHS\_BETWEEN(**SYSDATE**, v\_dob) / 12);

    IF v\_age > 60 THEN

      UPDATE Loans

      SET InterestRate = InterestRate - 1

      WHERE CustomerID = v\_c\_id;

    END IF;

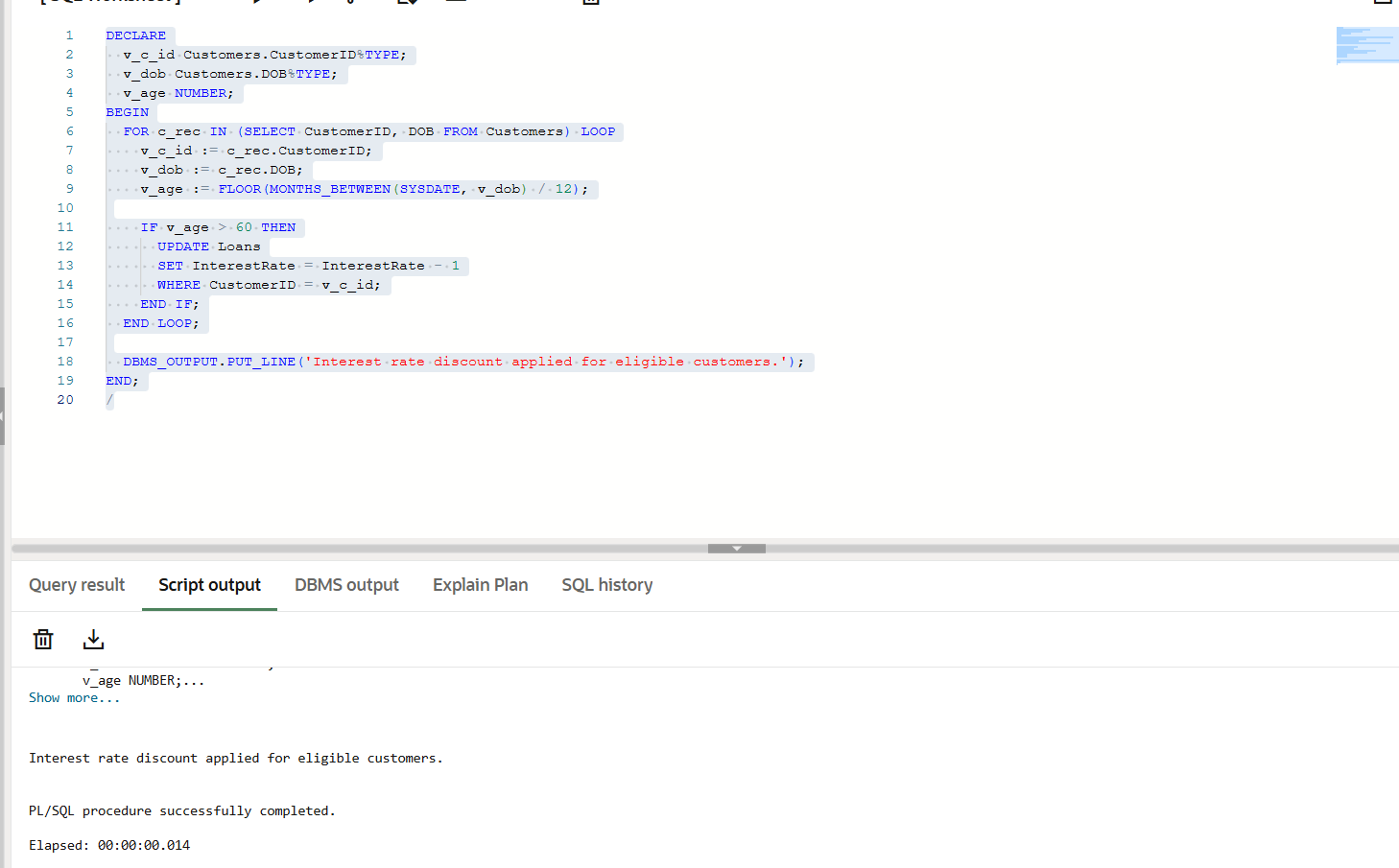
  END LOOP;

  DBMS\_OUTPUT.PUT\_LINE('Interest rate discount applied for eligible customers.');

END;

/

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.

CODE:

BEGIN

    FOR cust IN (

        SELECT CustomerID FROM Customers WHERE Balance > 10000

    ) LOOP

        UPDATE Customers

        SET IsVIP = 'TRUE'

        WHERE CustomerID = cust.CustomerID;

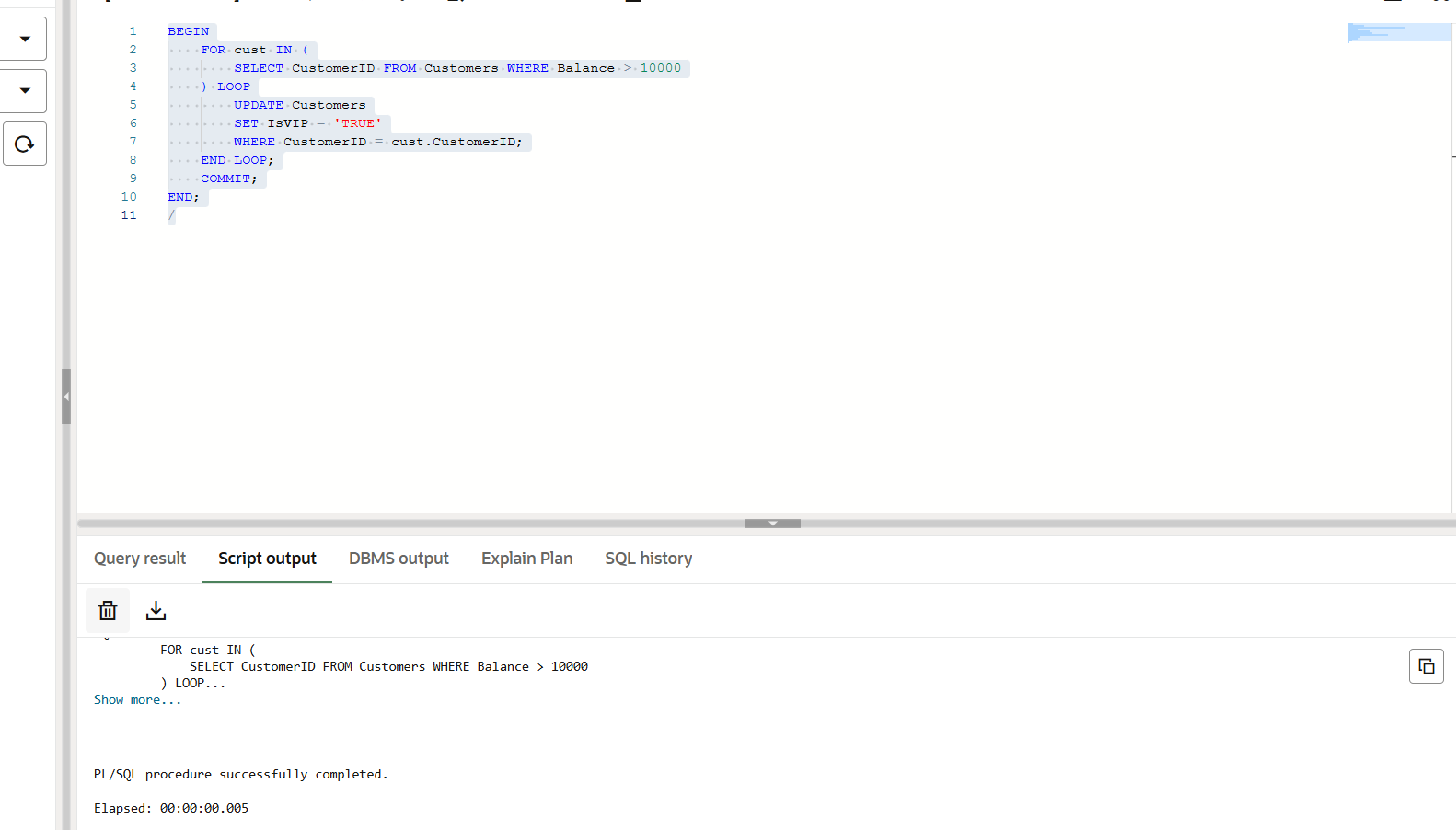
    END LOOP;

    COMMIT;

END;

/

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.

Code:

SET SERVEROUTPUT ON;

DECLARE

    v\_c\_name Customers.Name%TYPE;

    v\_c\_id Loans.CustomerID%TYPE;

    v\_l\_id Loans.LoanID%TYPE;

    v\_dd Loans.EndDate%TYPE;

    v\_amt Loans.LoanAmount%TYPE;

    CURSOR lc IS

        SELECT LoanID, CustomerID, LoanAmount, EndDate

        FROM Loans

        WHERE EndDate BETWEEN **SYSDATE** AND **SYSDATE** + 30;

BEGIN

    FOR lr IN lc LOOP

        SELECT Name INTO v\_c\_name

        FROM Customers

        WHERE CustomerID = lr.CustomerID;

        v\_c\_id := lr.CustomerID;

        v\_l\_id := lr.LoanID;

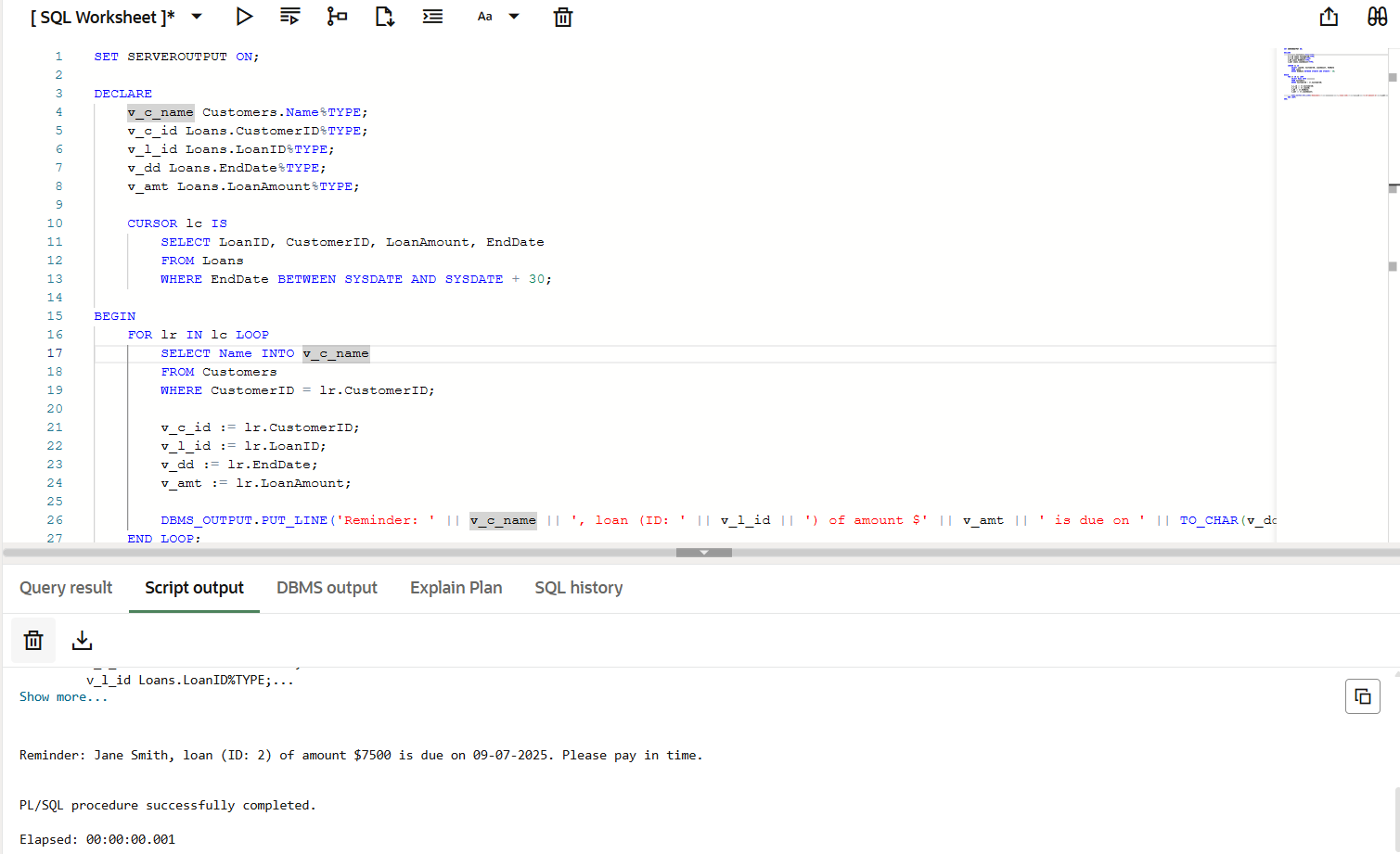
        v\_dd := lr.EndDate;

        v\_amt := lr.LoanAmount;

        DBMS\_OUTPUT.PUT\_LINE('Reminder: ' || v\_c\_name || ', loan (ID: ' || v\_l\_id || ') of amount $' || v\_amt || ' is due on ' || TO\_CHAR(v\_dd, 'DD-MM-YYYY') || '. Please pay in time.');

    END LOOP;

END;

Output:  


# 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.

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

  FOR acc IN (

    SELECT AccountID, Balance

    FROM Accounts

    WHERE AccountType = 'Savings'

  ) LOOP

    UPDATE Accounts

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

        LastModified = **SYSDATE**

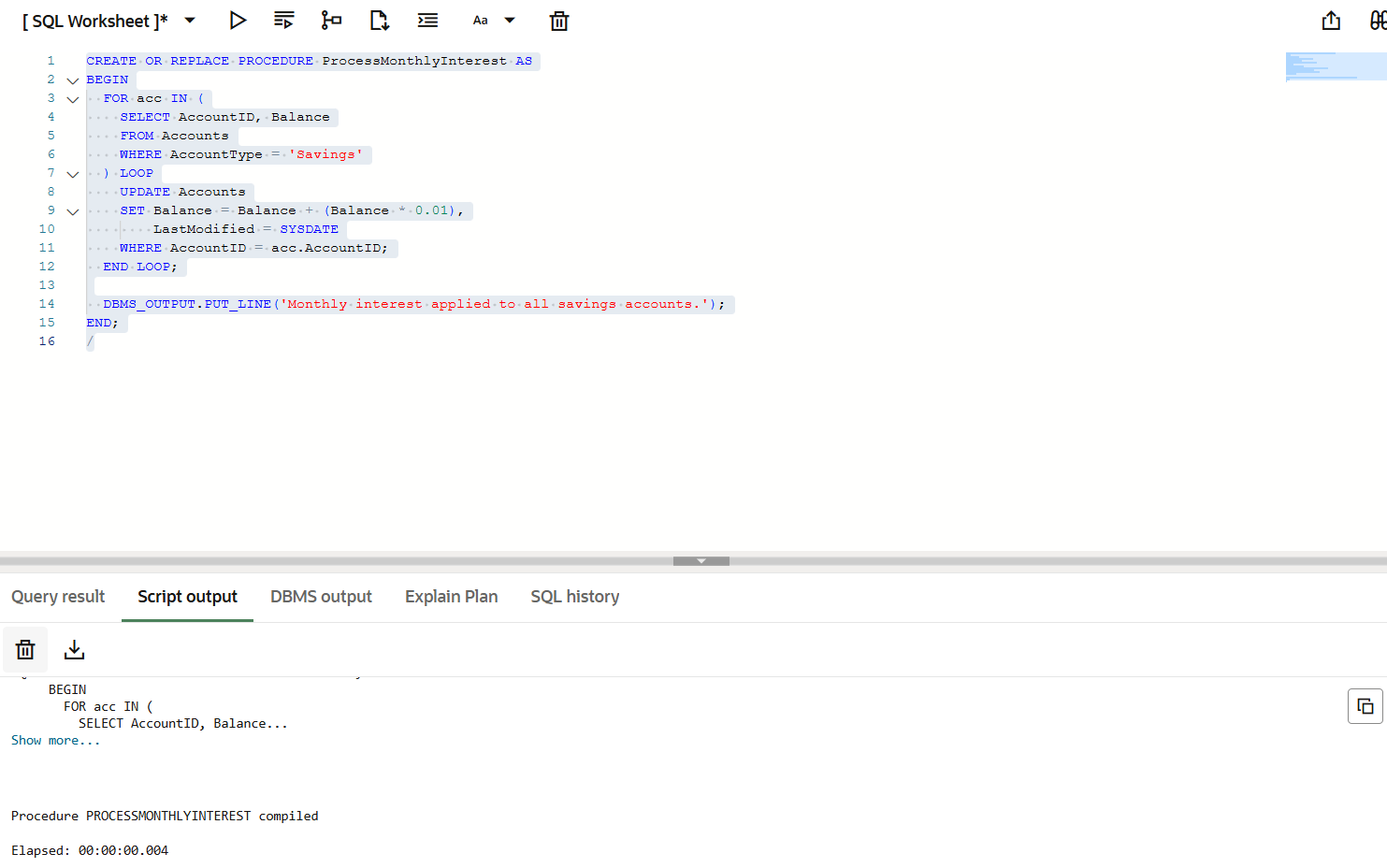
    WHERE AccountID = acc.AccountID;

  END LOOP;

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

END;

/

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.

Code:  
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

  p\_department IN Employees.Department%TYPE,

  p\_bonus\_pct IN **NUMBER**

) AS

BEGIN

  UPDATE Employees

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

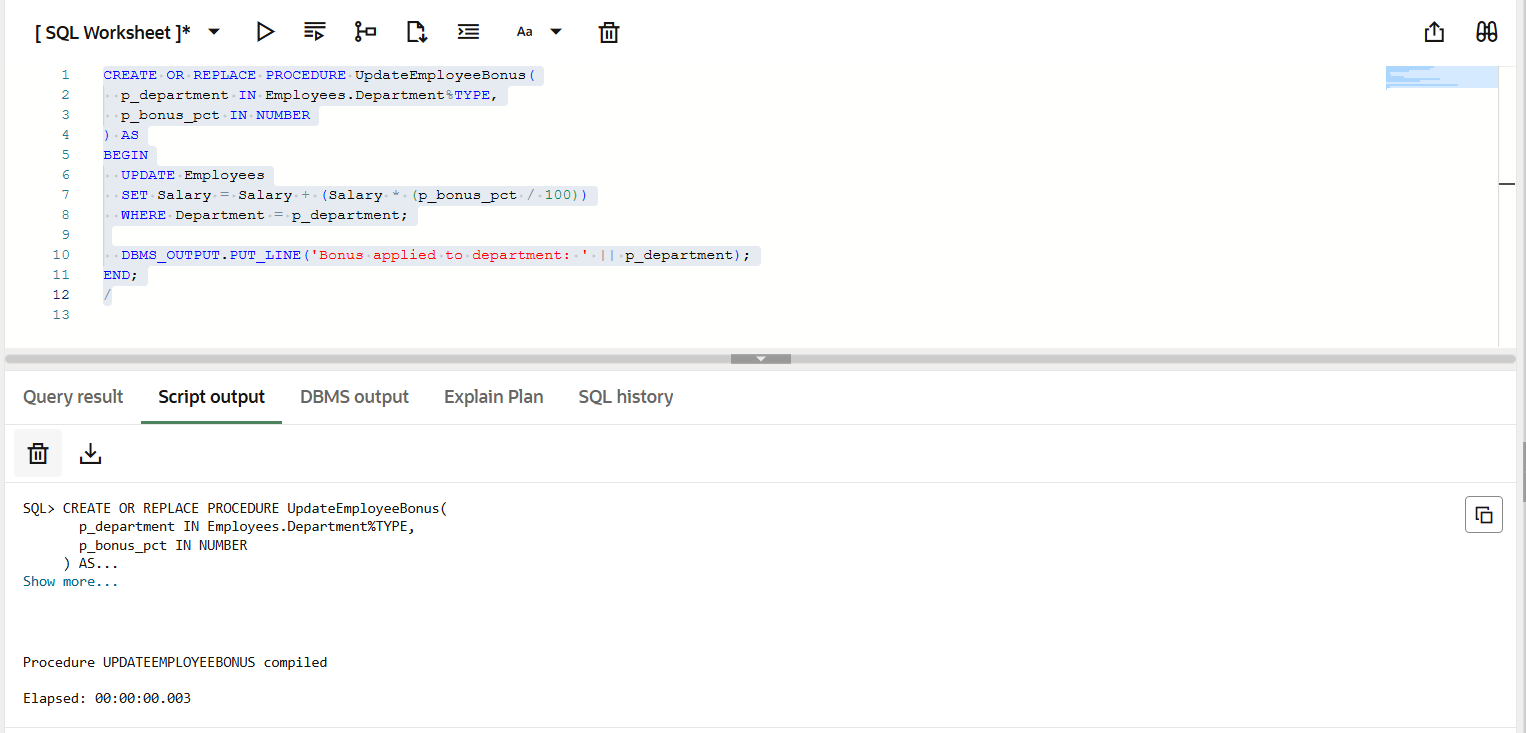
  WHERE Department = p\_department;

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

END;

/

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.

Code:

CREATE OR REPLACE PROCEDURE TransferFunds(

  p\_from\_account IN Accounts.AccountID%TYPE,

  p\_to\_account IN Accounts.AccountID%TYPE,

  p\_amount IN **NUMBER**

) AS

  v\_balance **NUMBER**;

BEGIN

  SELECT Balance INTO v\_balance

  FROM Accounts

  WHERE AccountID = p\_from\_account;

  IF v\_balance < p\_amount THEN

    DBMS\_OUTPUT.PUT\_LINE('Transfer failed: Insufficient funds.');

    RETURN;

  END IF;

  UPDATE Accounts

  SET Balance = Balance - p\_amount,

      LastModified = **SYSDATE**

  WHERE AccountID = p\_from\_account;

  UPDATE Accounts

  SET Balance = Balance + p\_amount,

      LastModified = **SYSDATE**

  WHERE AccountID = p\_to\_account;

  DBMS\_OUTPUT.PUT\_LINE('Transfer of $' || p\_amount || ' completed successfully.');

END;

/

Output:  
