**TASK 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.

**THIS IS MY CODE :**

BEGIN

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

IF MONTHS\_BETWEEN(SYSDATE, c.DOB) / 12 > 60 THEN

UPDATE Loans

SET InterestRate = InterestRate - 1

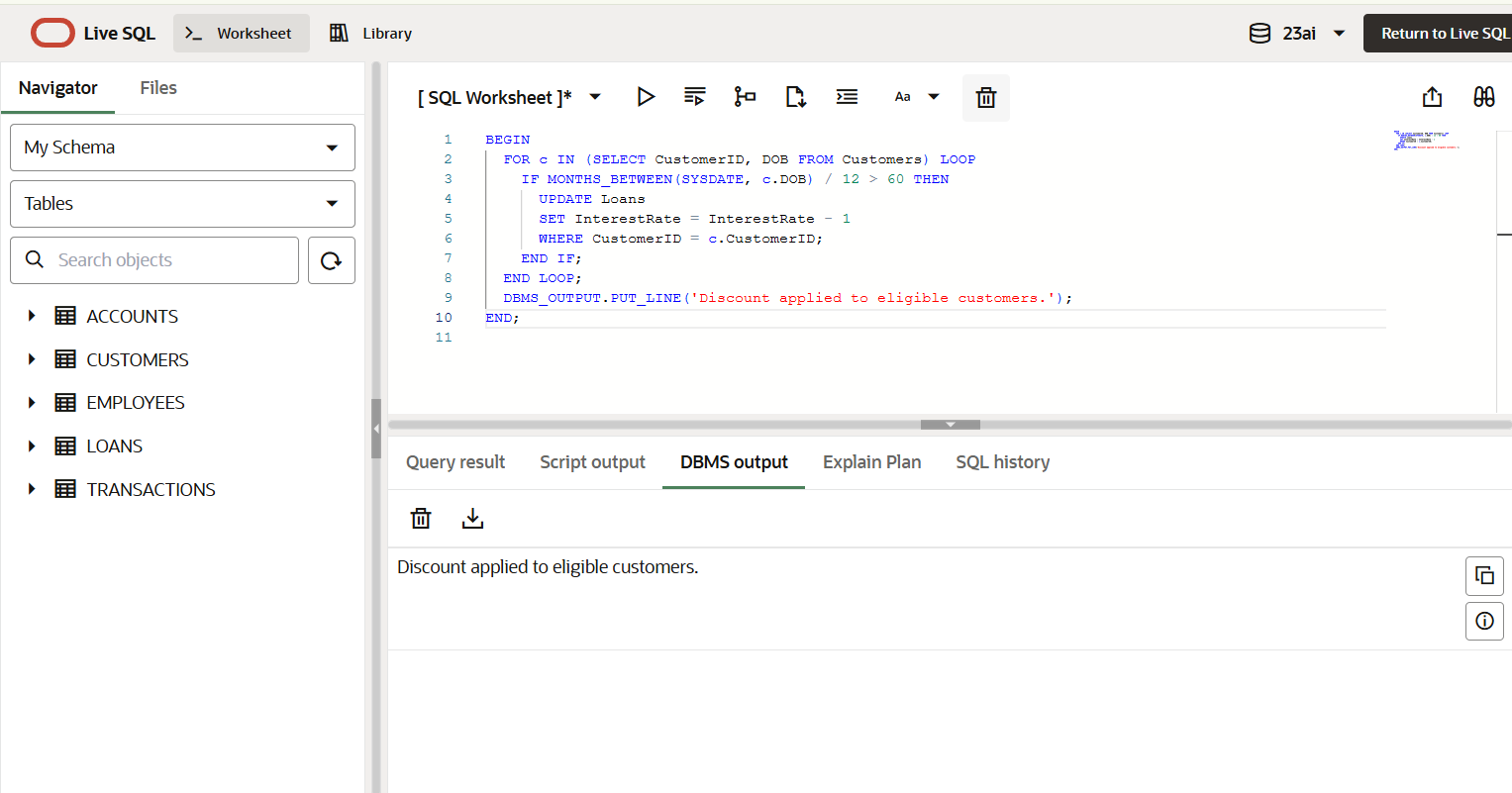
WHERE CustomerID = c.CustomerID;

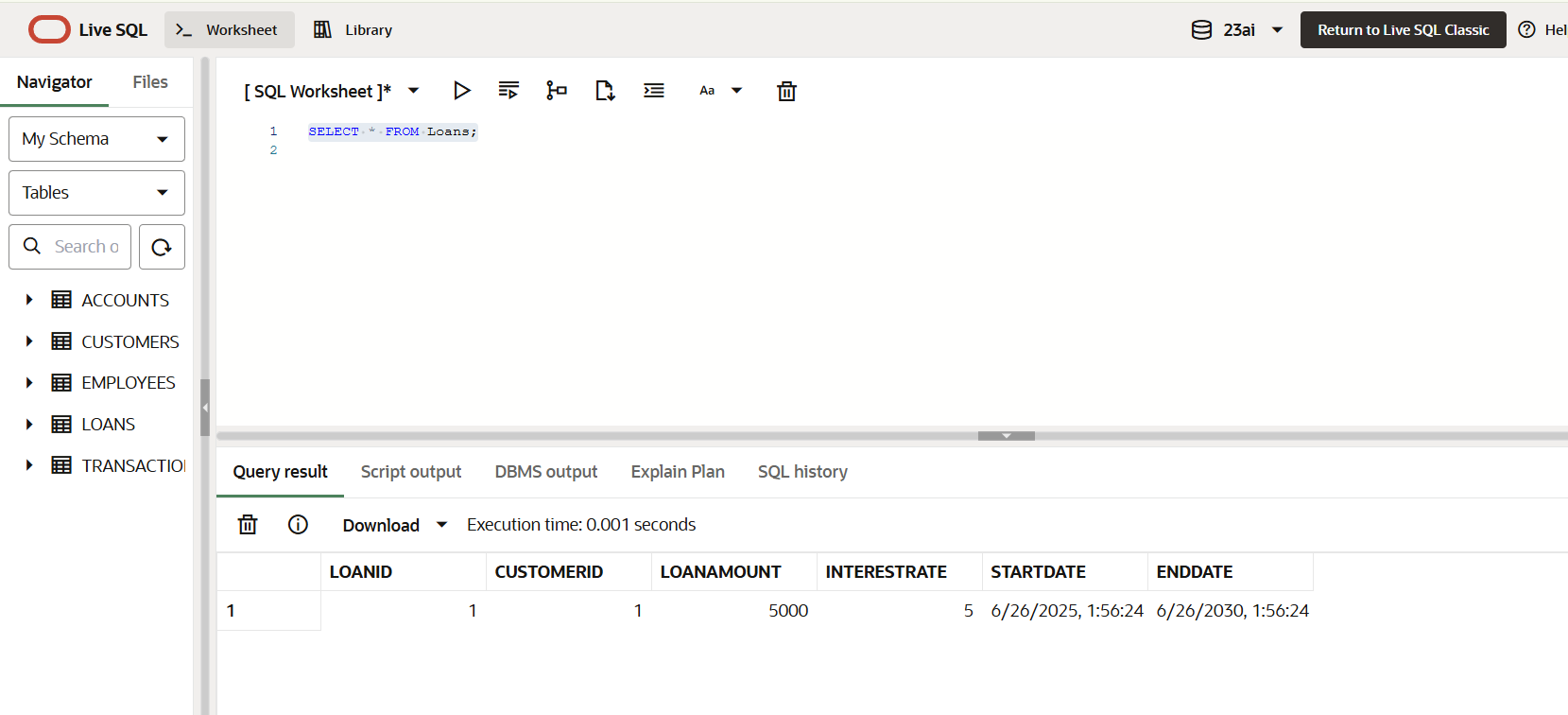
END IF;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Discount applied to eligible customers.');

END;

**THIS IS MY 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.

**THIS IS MY CODE :**

BEGIN

  FOR cust IN (SELECT CustomerID, Balance FROM Customers) LOOP

    IF cust.Balance > 10000 THEN

      UPDATE Customers

      SET IsVIP = 'TRUE'

      WHERE CustomerID = cust.CustomerID;

    END IF;

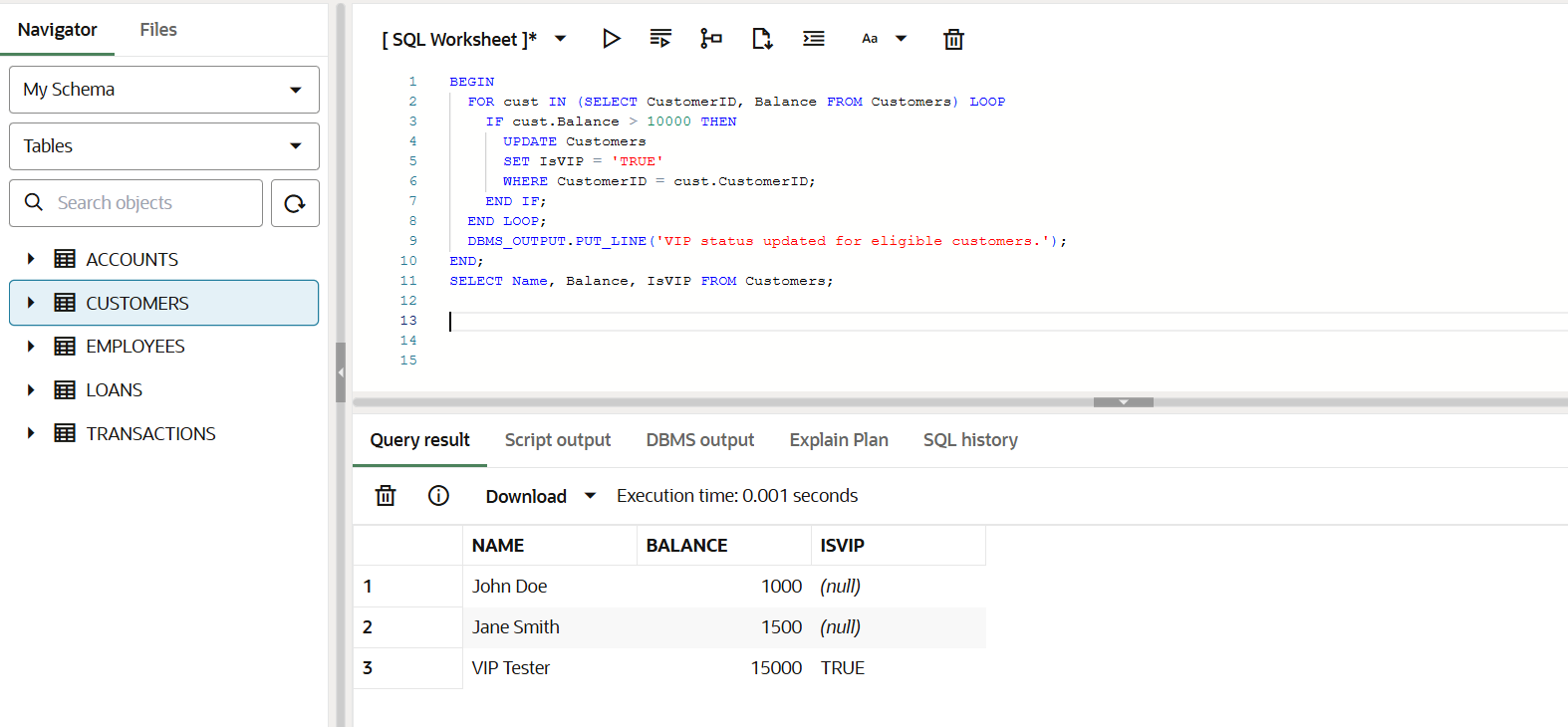
  END LOOP;

  DBMS\_OUTPUT.PUT\_LINE('VIP status updated for eligible customers.');

END;

SELECT Name, Balance, IsVIP FROM Customers;

**THIS IS MY 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.

**THIS IS MY CODE :**

BEGIN

  FOR l IN (

    SELECT l.LoanID, c.Name, l.EndDate

    FROM Loans l

    JOIN Customers c ON l.CustomerID = c.CustomerID

    WHERE l.EndDate <= SYSDATE + 30

  ) LOOP

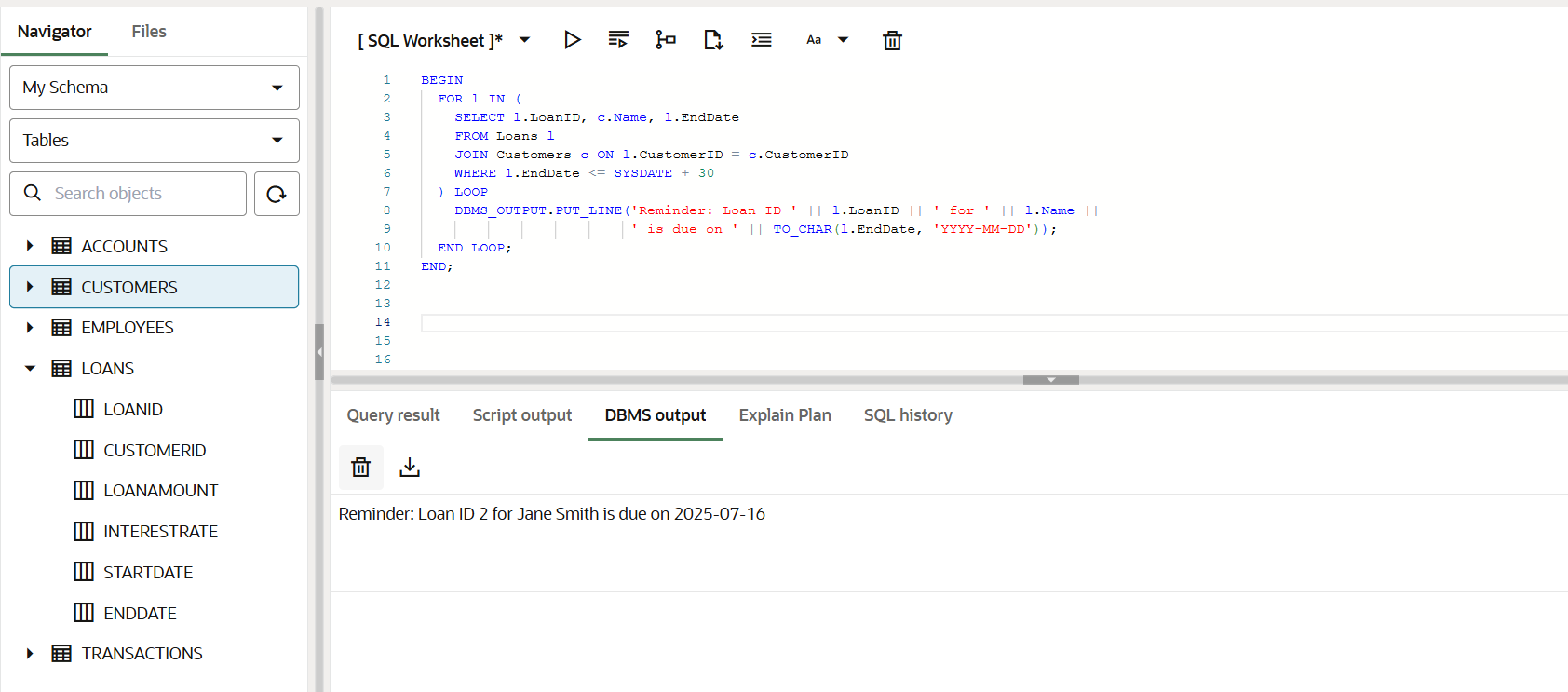
    DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || l.LoanID || ' for ' || l.Name ||

                         ' is due on ' || TO\_CHAR(l.EndDate, 'YYYY-MM-DD'));

  END LOOP;

END;

**THIS IS MY OUTPUT:**

****

**TASK 2: 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.

**THIS IS MY CODE :**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

  FOR acc IN (

    SELECT AccountID, Balance

    FROM Accounts

    WHERE AccountType = 'Savings'

  ) LOOP

    UPDATE Accounts

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

        LastModified = SYSDATE

    WHERE AccountID = acc.AccountID;

  END LOOP;

  DBMS\_OUTPUT.PUT\_LINE('Monthly interest processed for all savings accounts.');

END;

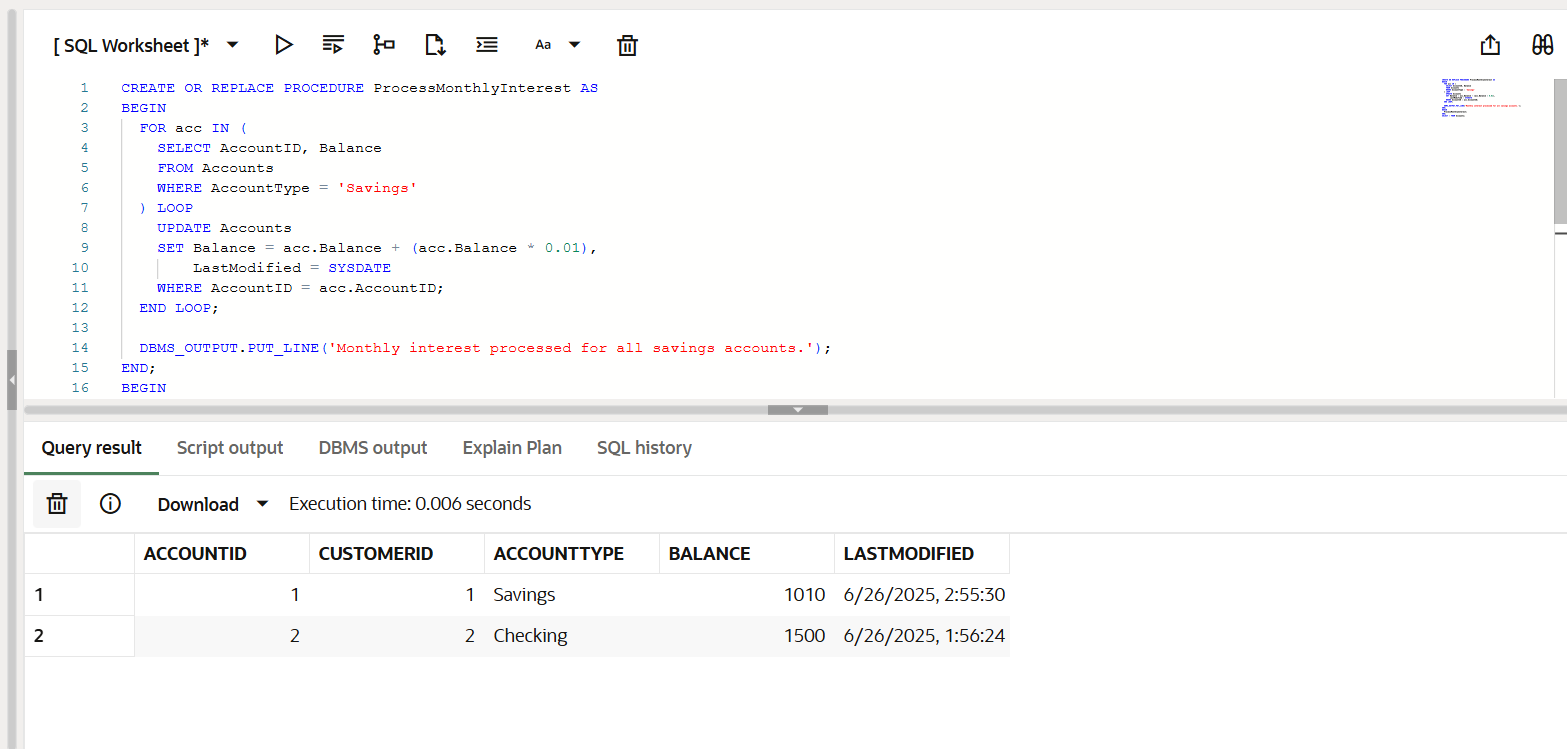
BEGIN

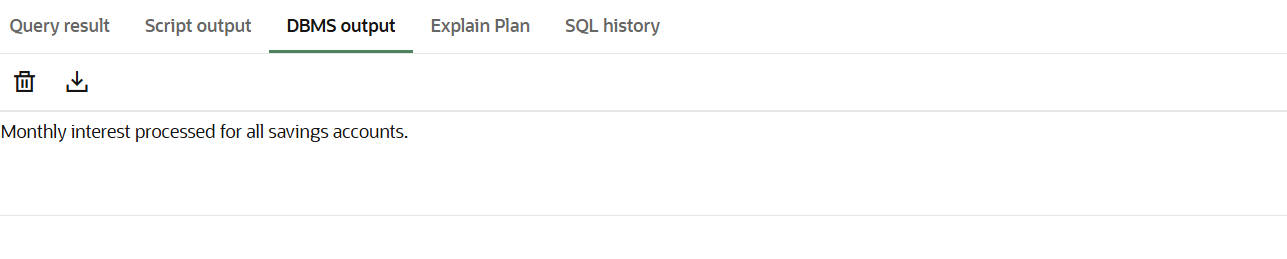
  ProcessMonthlyInterest;

END;

SELECT \* FROM Accounts;

**THIS IS MY 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.

**THIS IS MY CODE :**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

  deptName IN VARCHAR2,

  bonusPercent IN NUMBER

) AS

BEGIN

  FOR emp IN (

    SELECT EmployeeID, Salary FROM Employees

    WHERE Department = deptName

  ) LOOP

    UPDATE Employees

    SET Salary = emp.Salary + (emp.Salary \* bonusPercent / 100)

    WHERE EmployeeID = emp.EmployeeID;

  END LOOP;

  DBMS\_OUTPUT.PUT\_LINE('Bonus applied to all employees in department: ' || deptName);

END;

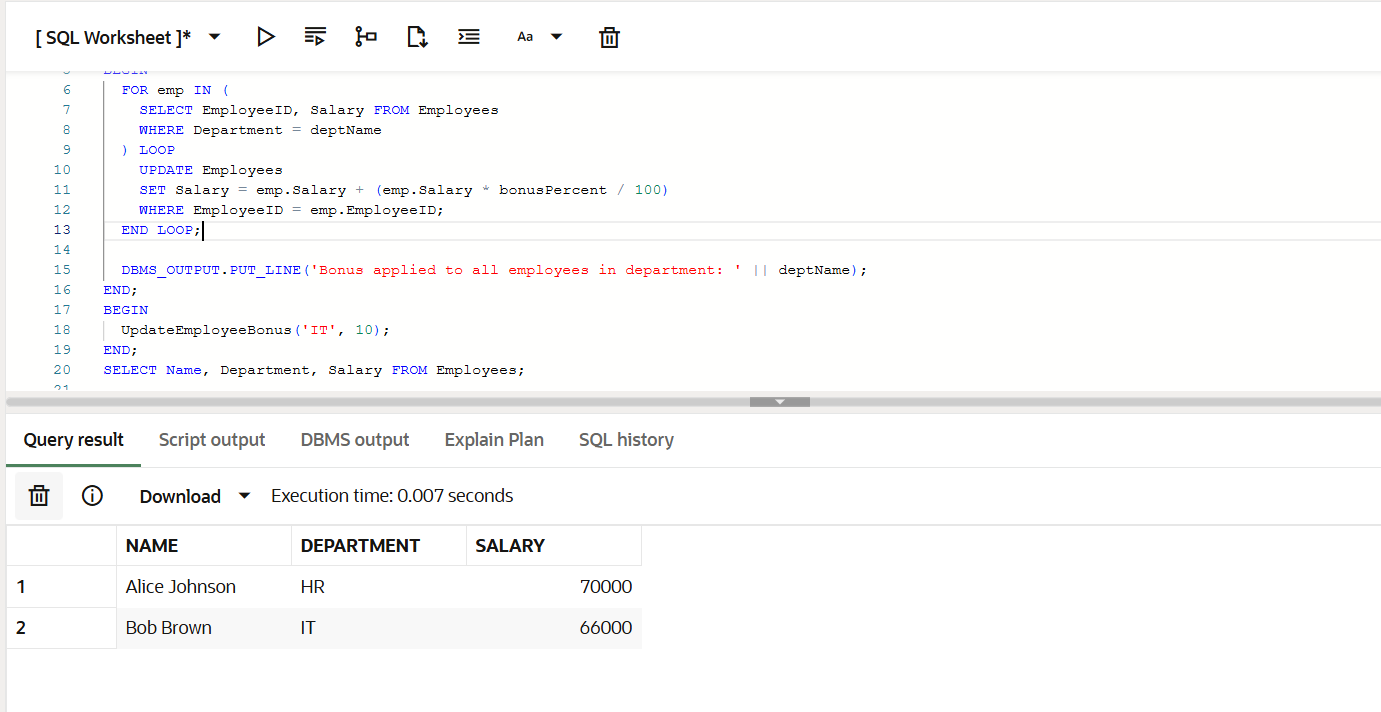
BEGIN

  UpdateEmployeeBonus('IT', 10);

END;

SELECT Name, Department, Salary FROM Employees;

**THIS IS MY 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.

**THIS IS MY CODE :**

CREATE OR REPLACE PROCEDURE TransferFunds (

  FromAccountID IN NUMBER,

  ToAccountID IN NUMBER,

  Amount IN NUMBER

) AS

  fromBalance NUMBER;

BEGIN

  SELECT Balance INTO fromBalance FROM Accounts WHERE AccountID = FromAccountID;

  IF fromBalance >= Amount THEN

    UPDATE Accounts

    SET Balance = Balance - Amount,

        LastModified = SYSDATE

    WHERE AccountID = FromAccountID;

    UPDATE Accounts

    SET Balance = Balance + Amount,

        LastModified = SYSDATE

    WHERE AccountID = ToAccountID;

    DBMS\_OUTPUT.PUT\_LINE('Transfer of ₹' || Amount || ' successful.');

  ELSE

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

  END IF;

END;

BEGIN

  TransferFunds(1, 2, 200);

END;

SELECT AccountID, Balance FROM Accounts;

**THIS IS MY OUTPUT:**

