**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 IsVIP 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.

**Solution:**

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified) VALUES (1, 'John Elder', TO\_DATE('1950-01-01', 'YYYY-MM-DD'), 12000, SYSDATE);

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified) VALUES (2, 'Alice Young', TO\_DATE('1990-06-15', 'YYYY-MM-DD'), 8000, SYSDATE);

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified) VALUES (3, 'Mary Senior', TO\_DATE('1955-04-20', 'YYYY-MM-DD'), 15000, SYSDATE);

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified) VALUES (4, 'Robert King', TO\_DATE('1985-09-10', 'YYYY-MM-DD'), 9800, SYSDATE);

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified) VALUES (5, 'Nancy Drew', TO\_DATE('1970-03-12', 'YYYY-MM-DD'), 11500, SYSDATE);

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified) VALUES (6, 'George Knight', TO\_DATE('1962-11-30', 'YYYY-MM-DD'), 10500, SYSDATE);

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified) VALUES (7, 'Betty Rose', TO\_DATE('1995-05-21', 'YYYY-MM-DD'), 6700, SYSDATE);

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified) VALUES (8, 'Tom Hardy', TO\_DATE('1949-08-25', 'YYYY-MM-DD'), 15500, SYSDATE);

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified) VALUES (9, 'Emma Watson', TO\_DATE('2000-01-01', 'YYYY-MM-DD'), 10200, SYSDATE);

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified) VALUES (10, 'Chris Evans', TO\_DATE('1988-12-14', 'YYYY-MM-DD'), 7200, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified) VALUES (101, 1, 'Savings', 12000, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified) VALUES (102, 2, 'Current', 8000, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified) VALUES (103, 3, 'Savings', 15000, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified) VALUES (104, 4, 'Savings', 9800, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified) VALUES (105, 5, 'Current', 11500, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified) VALUES (106, 6, 'Savings', 10500, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified) VALUES (107, 7, 'Current', 6700, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified) VALUES (108, 8, 'Savings', 15500, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified) VALUES (109, 9, 'Current', 10200, SYSDATE);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified) VALUES (110, 10, 'Savings', 7200, SYSDATE);

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate) VALUES (201, 1, 50000, 8.5, SYSDATE - 700, SYSDATE + 20);

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate) VALUES (202, 2, 40000, 9.0, SYSDATE - 400, SYSDATE + 60);

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate) VALUES (203, 3, 30000, 10.0, SYSDATE - 800, SYSDATE + 10);

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate) VALUES (204, 4, 20000, 8.0, SYSDATE - 365, SYSDATE + 90);

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate) VALUES (205, 5, 25000, 7.5, SYSDATE - 100, SYSDATE + 45);

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate) VALUES (206, 6, 15000, 9.2, SYSDATE - 200, SYSDATE + 15);

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate) VALUES (207, 7, 18000, 8.3, SYSDATE - 120, SYSDATE + 35);

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate) VALUES (208, 8, 60000, 11.0, SYSDATE - 900, SYSDATE + 5);

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate) VALUES (209, 9, 22000, 8.9, SYSDATE - 250, SYSDATE + 75);

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate) VALUES (210, 10, 16000, 7.8, SYSDATE - 180, SYSDATE + 28);

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType) VALUES (301, 101, SYSDATE - 3, 1000, 'Credit');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType) VALUES (302, 102, SYSDATE - 2, 500, 'Debit');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType) VALUES (303, 103, SYSDATE - 10, 1200, 'Credit');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType) VALUES (304, 104, SYSDATE - 4, 700, 'Debit');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType) VALUES (305, 105, SYSDATE - 1, 300, 'Credit');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType) VALUES (306, 106, SYSDATE - 8, 1100, 'Debit');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType) VALUES (307, 107, SYSDATE - 6, 400, 'Credit');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType) VALUES (308, 108, SYSDATE - 12, 900, 'Debit');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType) VALUES (309, 109, SYSDATE - 5, 600, 'Credit');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType) VALUES (310, 110, SYSDATE - 9, 1500, 'Debit');

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate) VALUES (401, 'Linda Scott', 'Manager', 80000, 'Loans', TO\_DATE('2010-05-01', 'YYYY-MM-DD'));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate) VALUES (402, 'David Lee', 'Clerk', 40000, 'Accounts', TO\_DATE('2015-07-12', 'YYYY-MM-DD'));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate) VALUES (403, 'Sophia Turner', 'Analyst', 60000, 'IT', TO\_DATE('2017-09-01', 'YYYY-MM-DD'));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate) VALUES (404, 'James Bond', 'Security Officer', 50000, 'Security', TO\_DATE('2012-01-20', 'YYYY-MM-DD'));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate) VALUES (405, 'Rachel Green', 'HR Executive', 55000, 'HR', TO\_DATE('2018-03-15', 'YYYY-MM-DD'));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate) VALUES (406, 'Ross Geller', 'Loan Officer', 62000, 'Loans', TO\_DATE('2014-11-01', 'YYYY-MM-DD'));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate) VALUES (407, 'Monica Geller', 'Branch Manager', 90000, 'Management', TO\_DATE('2009-06-25', 'YYYY-MM-DD'));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate) VALUES (408, 'Chandler Bing', 'IT Support', 58000, 'IT', TO\_DATE('2016-02-18', 'YYYY-MM-DD'));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate) VALUES (409, 'Joey Tribbiani', 'Customer Rep', 45000, 'Customer Service', TO\_DATE('2020-04-10', 'YYYY-MM-DD'));

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate) VALUES (410, 'Phoebe Buffay', 'Auditor', 67000, 'Audit', TO\_DATE('2013-08-19', 'YYYY-MM-DD'));

COMMIT;

**Scenario 1:**

DECLARE

    v\_age NUMBER;

BEGIN

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

        v\_age := FLOOR(MONTHS\_BETWEEN(SYSDATE, cust.DOB) / 12);

        IF v\_age > 60 THEN

            UPDATE Loans

            SET InterestRate = InterestRate - 1

            WHERE CustomerID = cust.CustomerID;

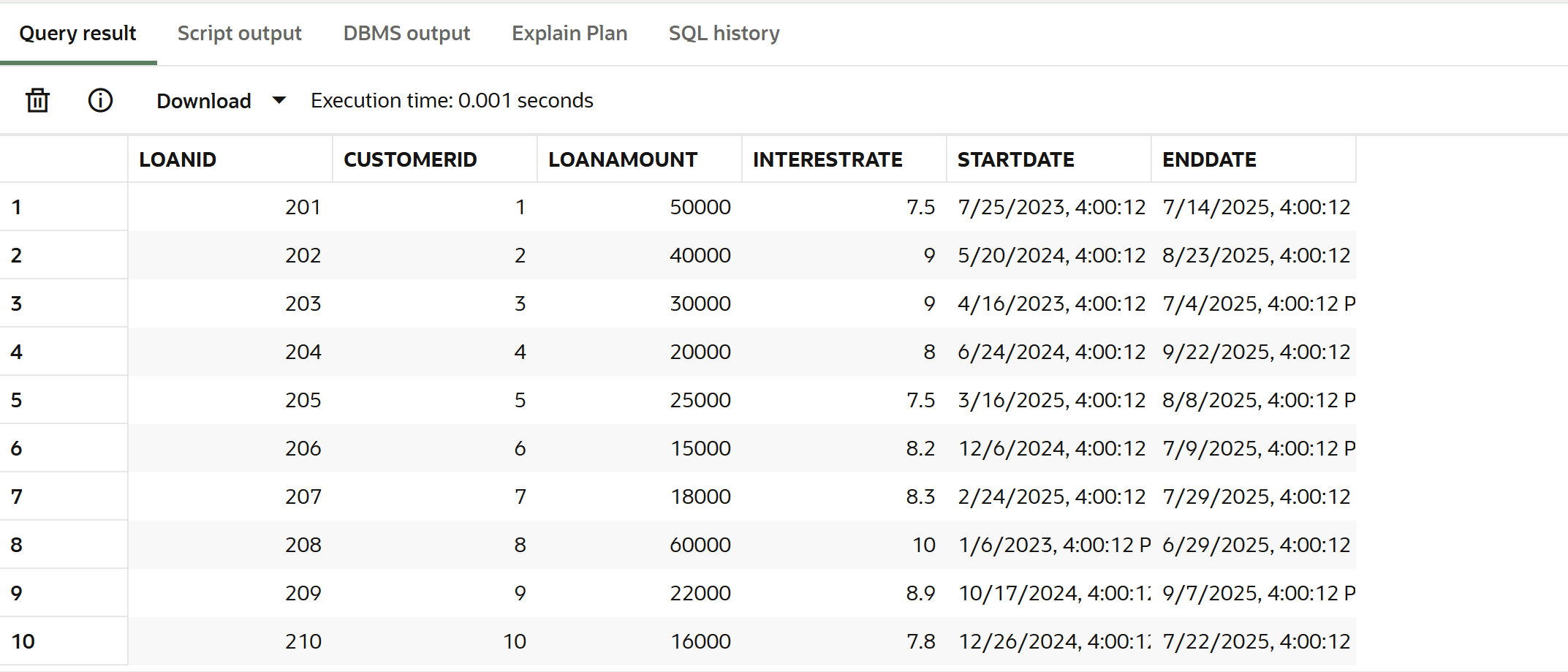
        END IF;

    END LOOP;

    COMMIT;

END;

SELECT \* FROM Loans;



**Scenario 2:**

BEGIN

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

        IF cust.Balance > 10000 THEN

            UPDATE Customers

            SET IsVIP = 'TRUE'

            WHERE CustomerID = cust.CustomerID;

        ELSE

            UPDATE Customers

            SET IsVIP = 'FALSE'

            WHERE CustomerID = cust.CustomerID;

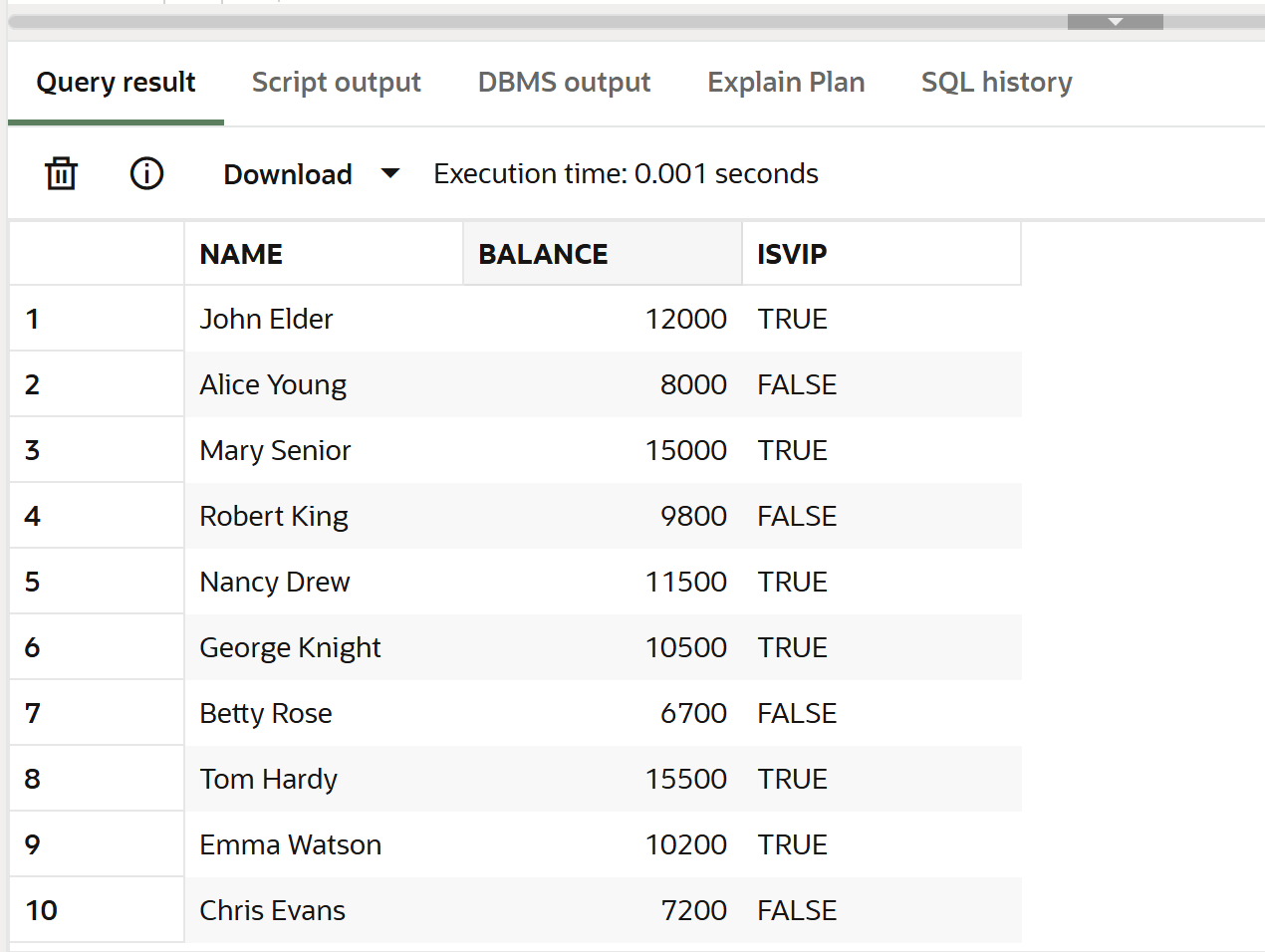
        END IF;

    END LOOP;

    COMMIT;

END;

SELECT Name, Balance, IsVIP FROM Customers;



**Scenario 3:**

DECLARE

    v\_name Customers.Name%TYPE;

BEGIN

    FOR loan IN (

        SELECT CustomerID, EndDate

        FROM Loans

        WHERE EndDate BETWEEN SYSDATE AND SYSDATE + 30

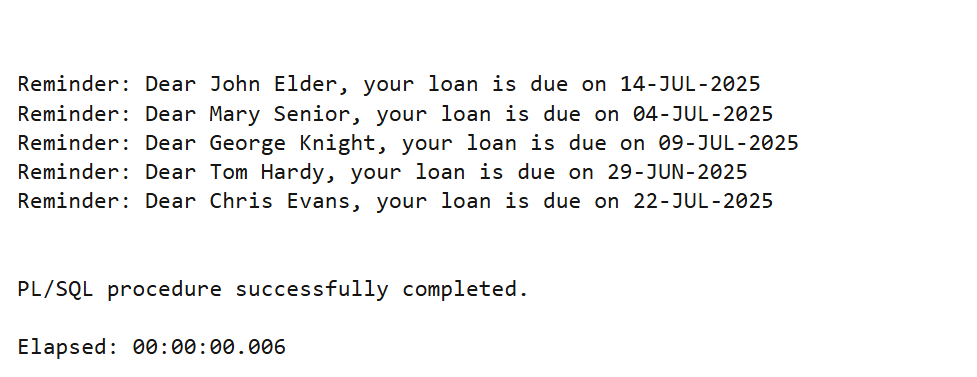
    ) LOOP

        SELECT Name INTO v\_name FROM Customers WHERE CustomerID = loan.CustomerID;

        DBMS\_OUTPUT.PUT\_LINE('Reminder: Dear ' || v\_name || ', your loan is due on ' || TO\_CHAR(loan.EndDate, 'DD-MON-YYYY'));

    END LOOP;

END;



**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.

**Solutions:**

**Scenario 1:**CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE Accounts

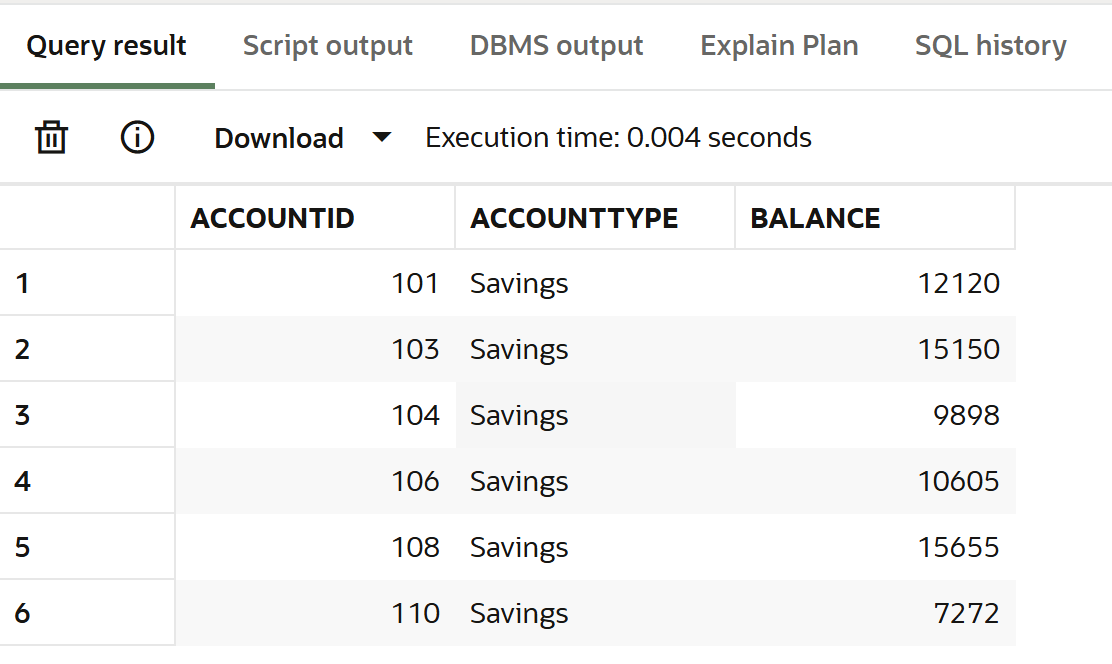
SET Balance = Balance + (Balance \* 0.01)

WHERE AccountType = 'Savings';

COMMIT;

END;

/  
EXEC ProcessMonthlyInterest;

SELECT AccountID, AccountType, Balance FROM Accounts WHERE AccountType = 'Savings';  
  


**Scenario 2:**CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_department IN VARCHAR2,

p\_bonus\_percent IN NUMBER

) IS

BEGIN

UPDATE Employees

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

WHERE Department = p\_department;

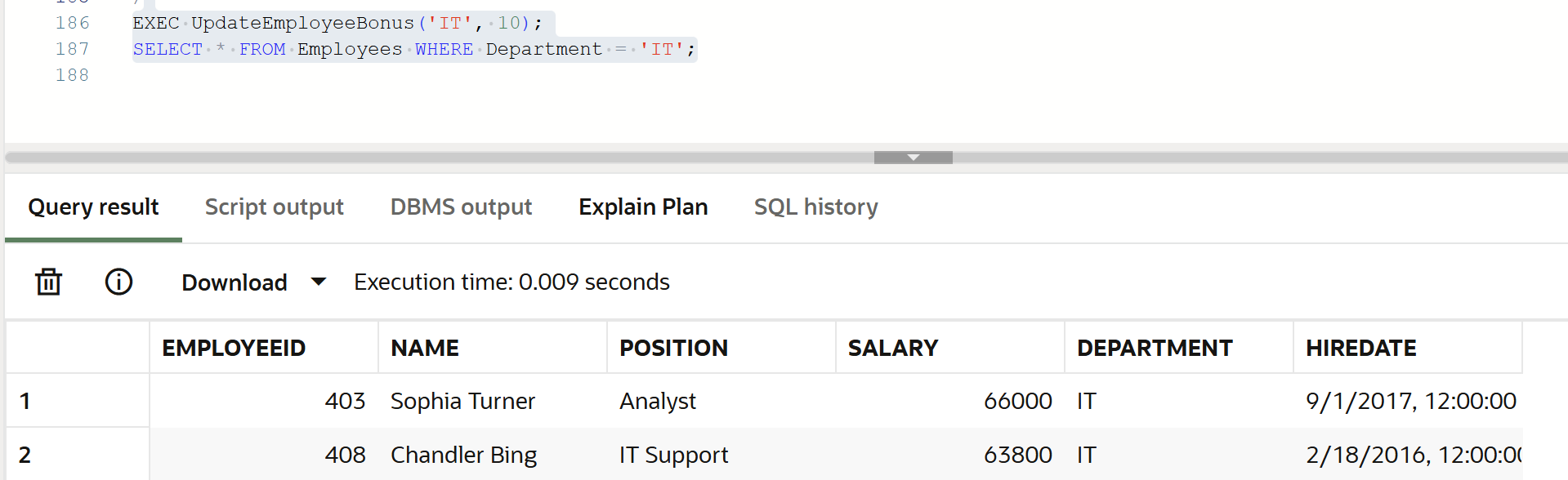
COMMIT;

END;

/

EXEC UpdateEmployeeBonus('IT', 10);

SELECT \* FROM Employees WHERE Department = 'IT';



**Scenario 3:**CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

-- Check if sufficient balance exists

SELECT Balance INTO v\_balance FROM Accounts WHERE AccountID = p\_from\_account FOR UPDATE;

IF v\_balance < p\_amount THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient balance in source account.');

END IF;

-- Deduct from source

UPDATE Accounts

SET Balance = Balance - p\_amount

WHERE AccountID = p\_from\_account;

-- Add to destination

UPDATE Accounts

SET Balance = Balance + p\_amount

WHERE AccountID = p\_to\_account;

COMMIT;

END;

/

EXEC TransferFunds(101, 102, 500);

SELECT AccountID, Balance FROM Accounts WHERE AccountID IN (101, 102);  
  
  
  
  
  
  
  
  
