**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.**

**Query:**

**-- Customers table**

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE

);

**-- Loans table**

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

LoanAmount NUMBER,

InterestRate NUMBER,

StartDate DATE,

EndDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

INSERT INTO Customers VALUES (1, 'John Doe', TO\_DATE('1985-05-15', 'YYYY-MM-DD'), 1000, SYSDATE);

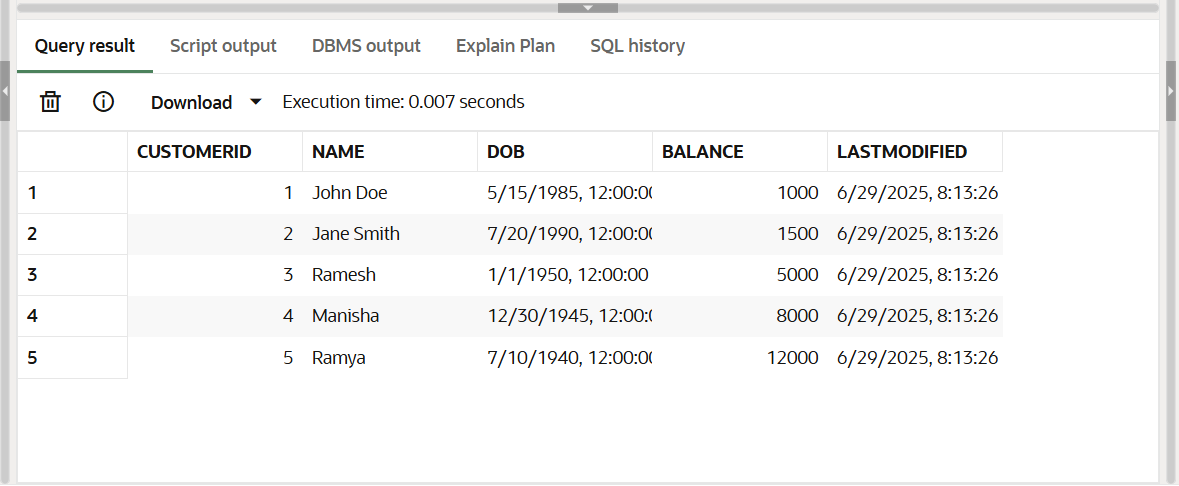
INSERT INTO Customers VALUES (2, 'Jane Smith', TO\_DATE('1990-07-20', 'YYYY-MM-DD'), 1500, SYSDATE);

INSERT INTO Customers VALUES (3, 'Ramesh', TO\_DATE('1950-01-01', 'YYYY-MM-DD'), 5000, SYSDATE);

INSERT INTO Customers VALUES (4, 'Manisha', TO\_DATE('1945-12-30', 'YYYY-MM-DD'), 8000, SYSDATE);

INSERT INTO Customers VALUES (5, 'Ramya', TO\_DATE('1940-07-10', 'YYYY-MM-DD'), 12000, SYSDATE);

SELECT \* FROM Customers;



-- Loans for each customer

INSERT INTO Loans VALUES (101, 1, 8000, 10, SYSDATE, SYSDATE + 365); -- John Doe

INSERT INTO Loans VALUES (102, 2, 9000, 10, SYSDATE, SYSDATE + 365); -- Jane Smith

INSERT INTO Loans VALUES (103, 3, 10000, 10, SYSDATE, SYSDATE + 365); -- Senior User

INSERT INTO Loans VALUES (104, 4, 11000, 10, SYSDATE, SYSDATE + 365); -- Old Customer A

INSERT INTO Loans VALUES (105, 5, 12000, 10, SYSDATE, SYSDATE + 365); -- Old Customer B

**PL/SQL:**

BEGIN

FOR rec IN (

SELECT CustomerID

FROM Customers

WHERE (SYSDATE - DOB)/365 > 60

) LOOP

UPDATE Loans

SET InterestRate = InterestRate - 1

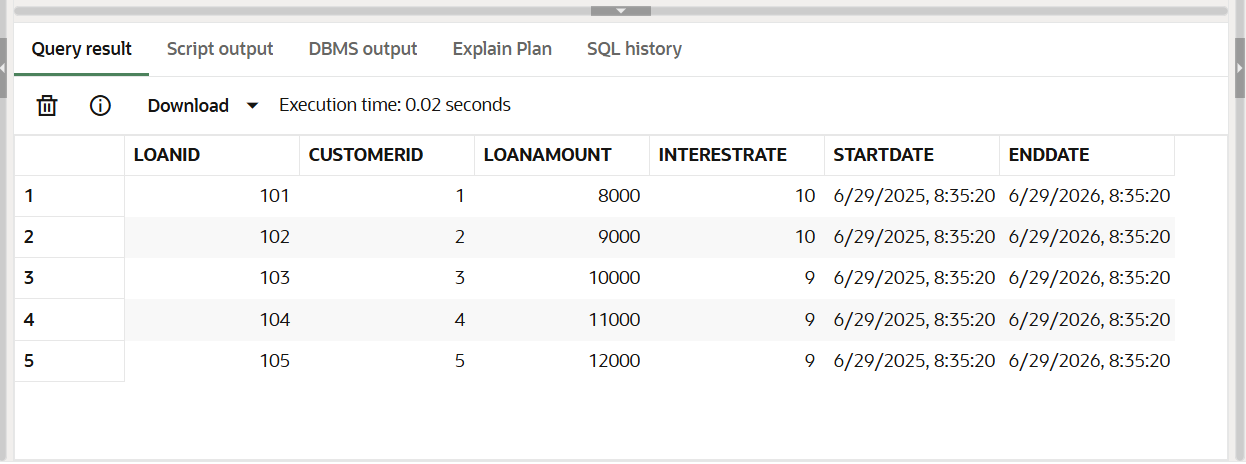
WHERE CustomerID = rec.CustomerID;

END LOOP;

END;

/

SELECT \* FROM Loans;

**OUTPUT:**

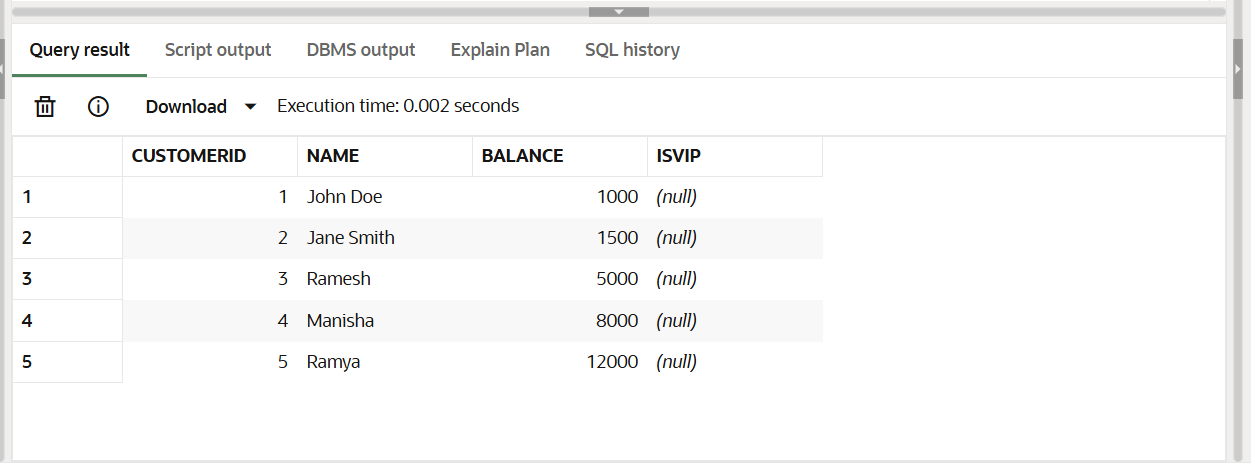
**Scenario 2: A customer can be promoted to VIP status based on their balance.**

**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.**

**Query:**

ALTER TABLE Customers ADD IsVIP VARCHAR2(5);

SELECT CustomerID, Name, Balance, IsVIP FROM Customers;

-- Set VIPs

**PL/SQL:**

BEGIN

FOR rec IN (

SELECT CustomerID

FROM Customers

WHERE Balance > 10000

) LOOP

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = rec.CustomerID;

END LOOP;

END;

/

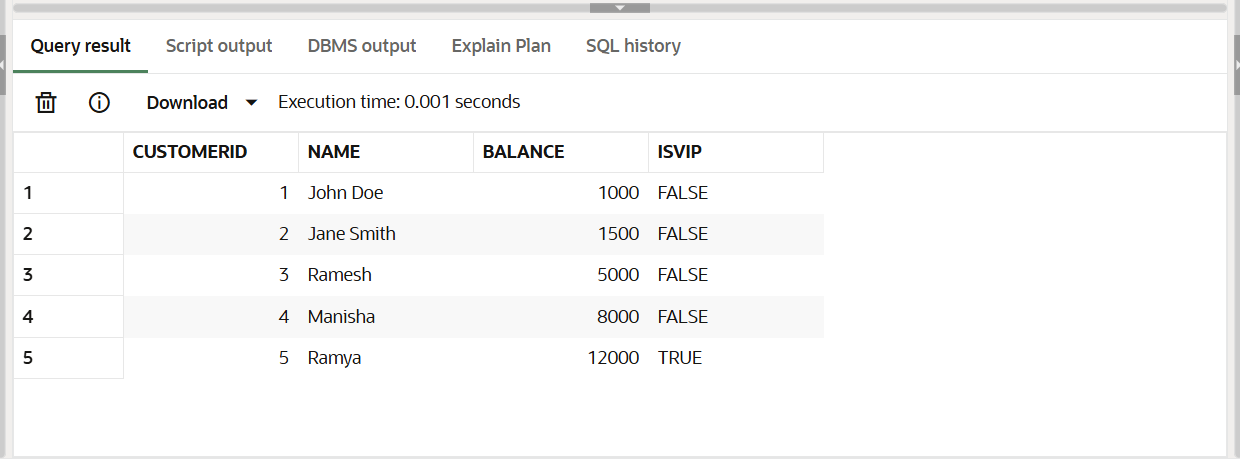
-- Set others as non-VIPs

UPDATE Customers

SET IsVIP = 'FALSE'

WHERE IsVIP IS NULL;

SELECT CustomerID, Name, Balance, IsVIP FROM Customers;

**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.**

**Query:**

SELECT CustomerID, Name FROM Customers WHERE CustomerID IN (101, 103);

-- Loan for John Doe

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

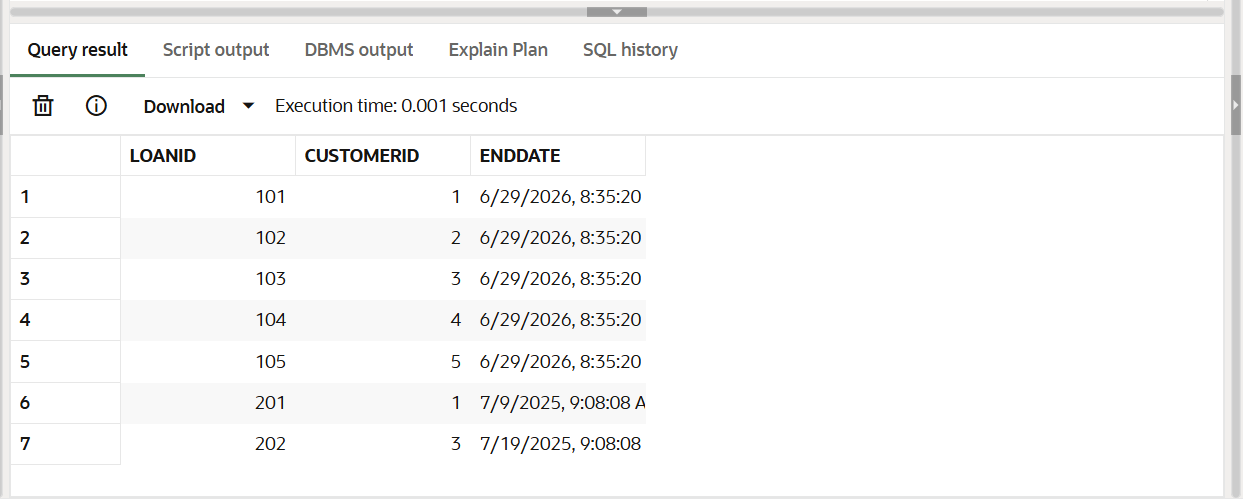
VALUES (201, 101, 6000, 10, SYSDATE, SYSDATE + 10);

-- Loan for Senior User

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (202, 103, 7000, 10, SYSDATE, SYSDATE + 20);

SELECT LoanID, CustomerID, EndDate FROM Loans WHERE LoanID IN (201, 202);



**PL/SQL:**

SET SERVEROUTPUT ON;

BEGIN

FOR rec IN (

SELECT l.LoanID, l.CustomerID, 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 ' || rec.LoanID ||

' for Customer ' || rec.Name ||

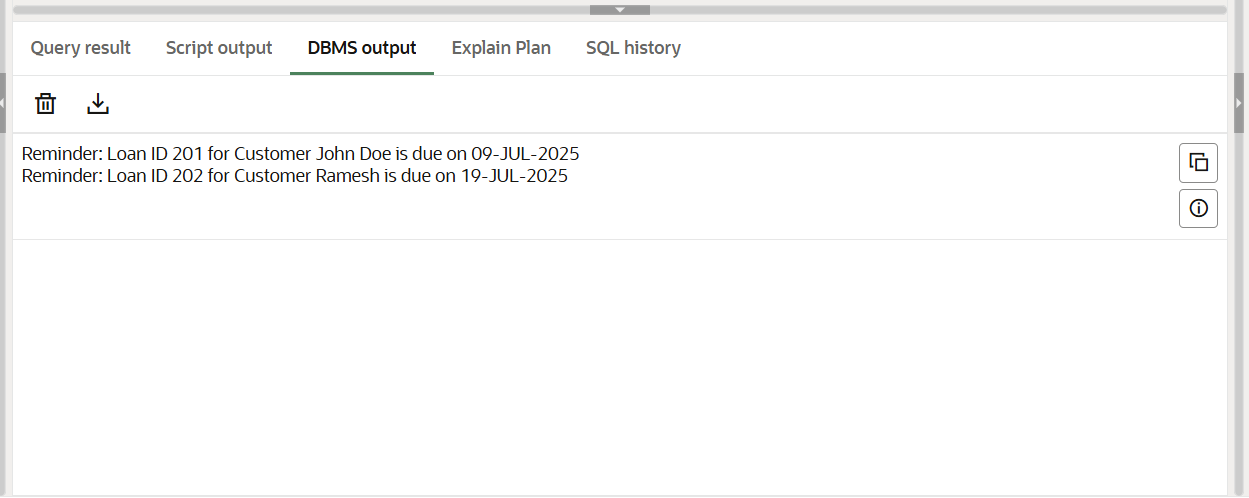
' is due on ' || TO\_CHAR(rec.EndDate, 'DD-MON-YYYY'));

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.**

**-- Customers table**

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE

);

**-- Accounts table**

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

AccountType VARCHAR2(20),

Balance NUMBER,

LastModified DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

INSERT INTO Customers VALUES (101, 'John Doe', TO\_DATE('1985-05-15', 'YYYY-MM-DD'), 1000, SYSDATE);

INSERT INTO Customers VALUES (102, 'Jane Smith', TO\_DATE('1990-07-20', 'YYYY-MM-DD'), 1500, SYSDATE);

INSERT INTO Customers VALUES (103, 'Senior User', TO\_DATE('1950-01-01', 'YYYY-MM-DD'), 12000, SYSDATE);

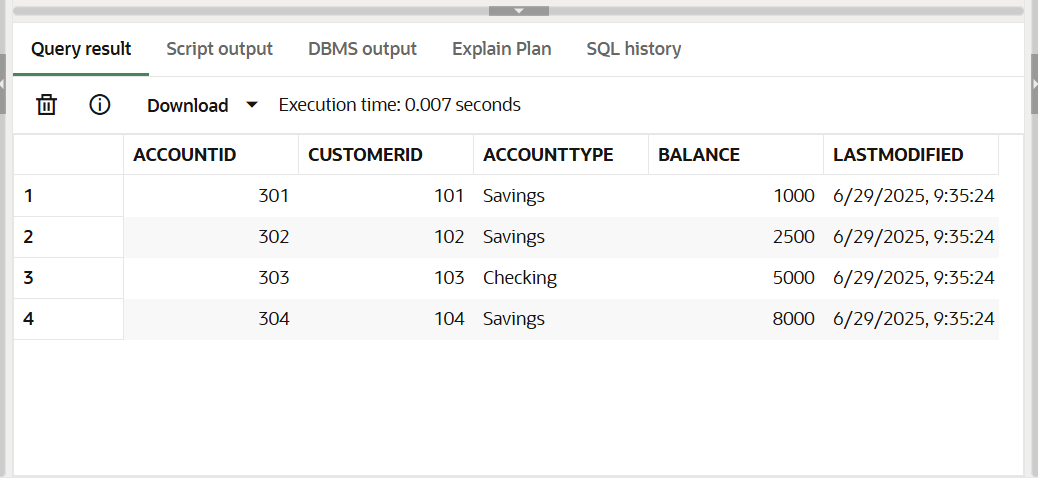
INSERT INTO Customers VALUES (104, 'Ram Kumar', TO\_DATE('1980-03-12', 'YYYY-MM-DD'), 9000, SYSDATE);

INSERT INTO Accounts VALUES (301, 101, 'Savings', 1000, SYSDATE);

INSERT INTO Accounts VALUES (302, 102, 'Savings', 2500, SYSDATE);

INSERT INTO Accounts VALUES (303, 103, 'Checking', 5000, SYSDATE);

INSERT INTO Accounts VALUES (304, 104, 'Savings', 8000, SYSDATE);

****

**PL/SQL:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE Accounts

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

LastModified = SYSDATE

WHERE AccountType = 'Savings';

END;

/

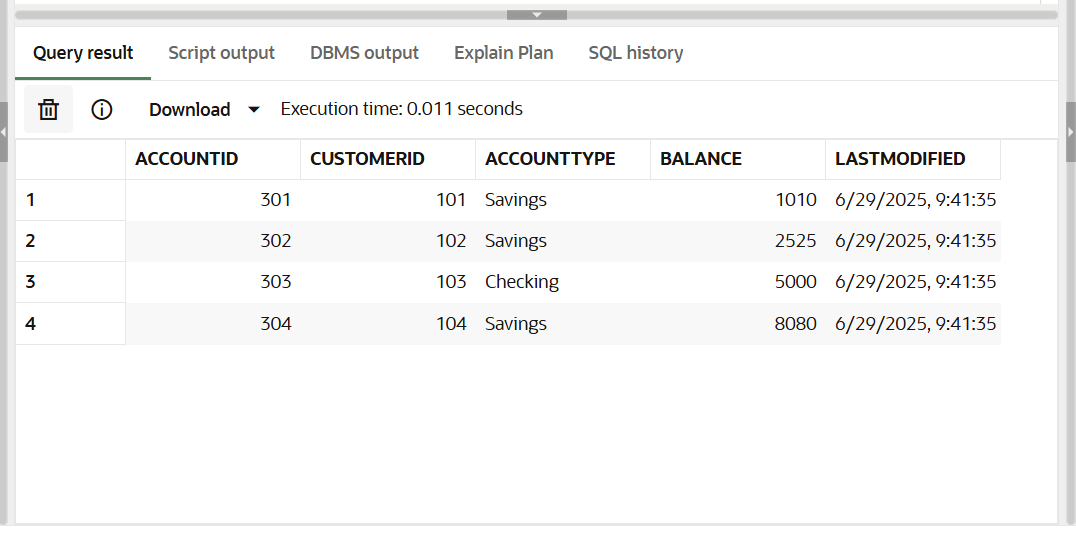
BEGIN

ProcessMonthlyInterest;

END;

/

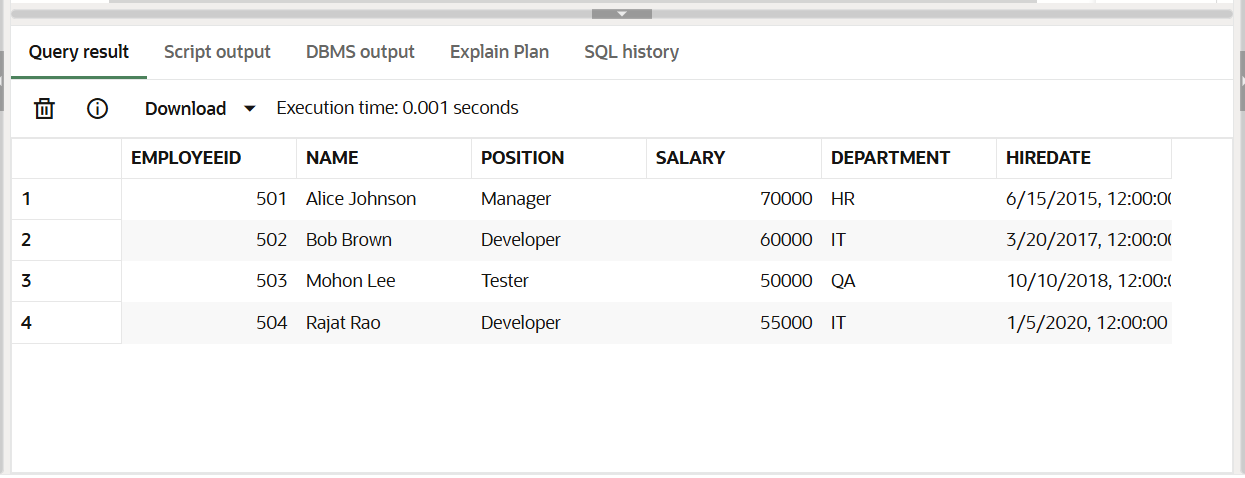
SELECT \* FROM Accounts;

****

**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.**

SELECT \* FROM Employees;



**PL/SQL:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

dept IN VARCHAR2,

bonus\_percent IN NUMBER

) IS

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* bonus\_percent / 100)

WHERE Department = dept;

END;

/

BEGIN

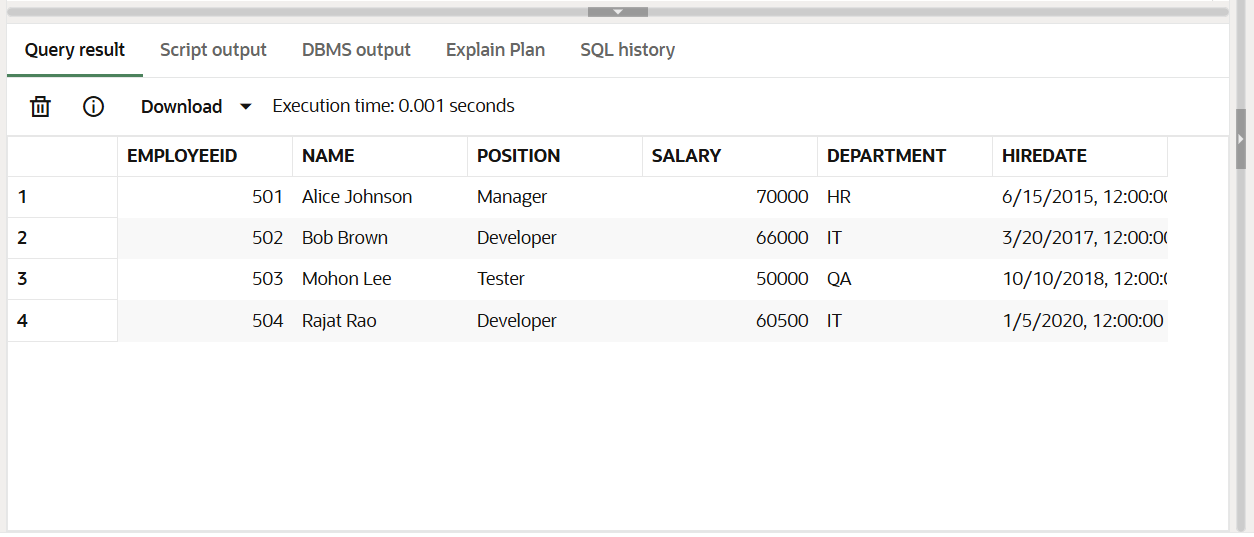
UpdateEmployeeBonus('IT', 10);

END;

/

SELECT \* FROM Employees;

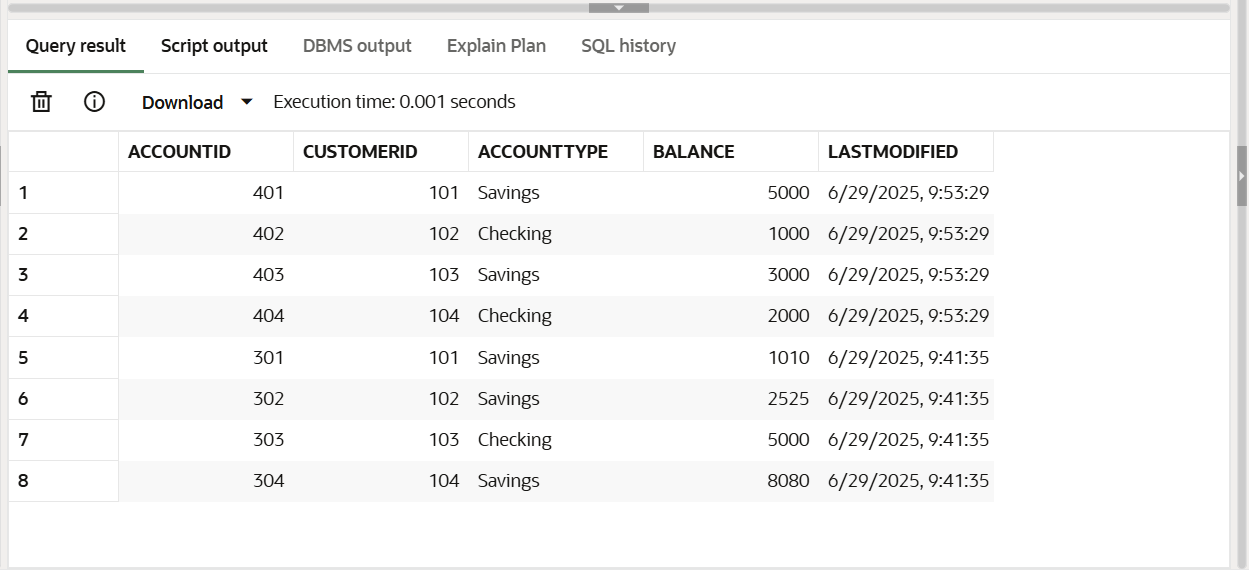
**Output:**



**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.**

SELECT \* FROM Accounts;



**PL/SQL:**

CREATE OR REPLACE PROCEDURE TransferFunds(

from\_acc IN NUMBER,

to\_acc IN NUMBER,

amount IN NUMBER

) IS

source\_balance NUMBER;

BEGIN

-- Get balance of source account

SELECT Balance INTO source\_balance FROM Accounts WHERE AccountID = from\_acc;

IF source\_balance >= amount THEN

-- Deduct from source

UPDATE Accounts

SET Balance = Balance - amount,

LastModified = SYSDATE

WHERE AccountID = from\_acc;

-- Add to destination

UPDATE Accounts

SET Balance = Balance + amount,

LastModified = SYSDATE

WHERE AccountID = to\_acc;

DBMS\_OUTPUT.PUT\_LINE('Transfer successful.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: insufficient balance.');

END IF;

END;

/

SET SERVEROUTPUT ON;

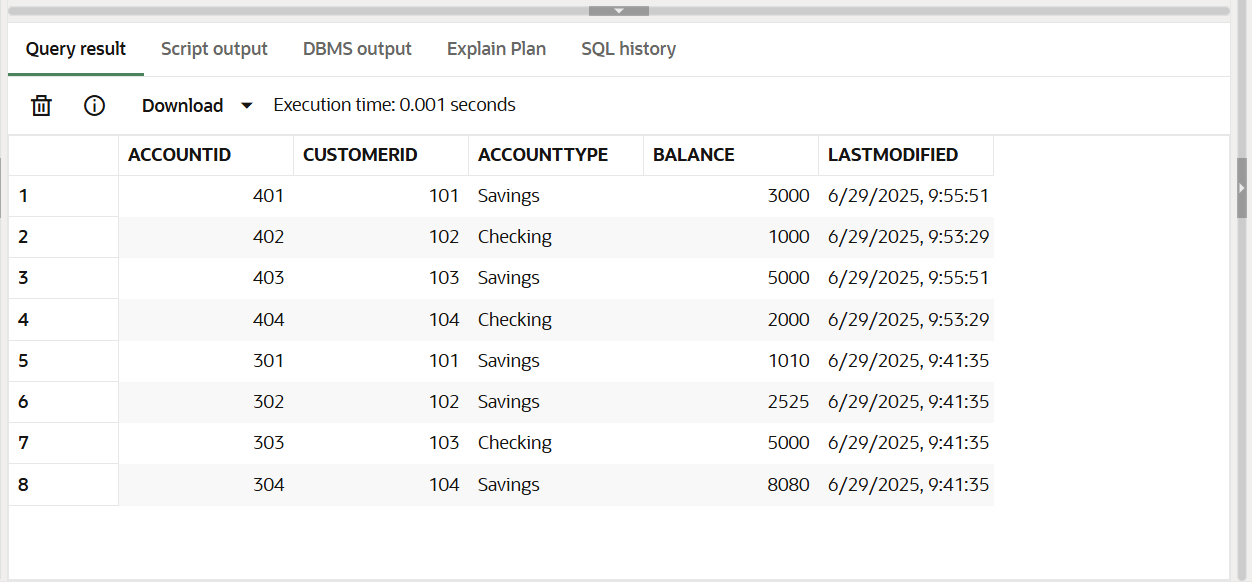
BEGIN

TransferFunds(401, 403, 2000);

END;

/

SELECT \* FROM Accounts;

**OUTPUT:**