**Exercise 1: Control Structures**

**Table creation:**

CREATE TABLE Customers (

    CustomerID NUMBER PRIMARY KEY,

    Name VARCHAR2(100),

    DOB DATE,

    Balance NUMBER,

    LastModified DATE

);

ALTER TABLE Customers ADD IsVIP VARCHAR2(5) DEFAULT 'FALSE';

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified, IsVIP) VALUES

(1, 'Arun',    TO\_DATE('1958-06-10', 'YYYY-MM-DD'), 15000, SYSDATE, 'FALSE'),

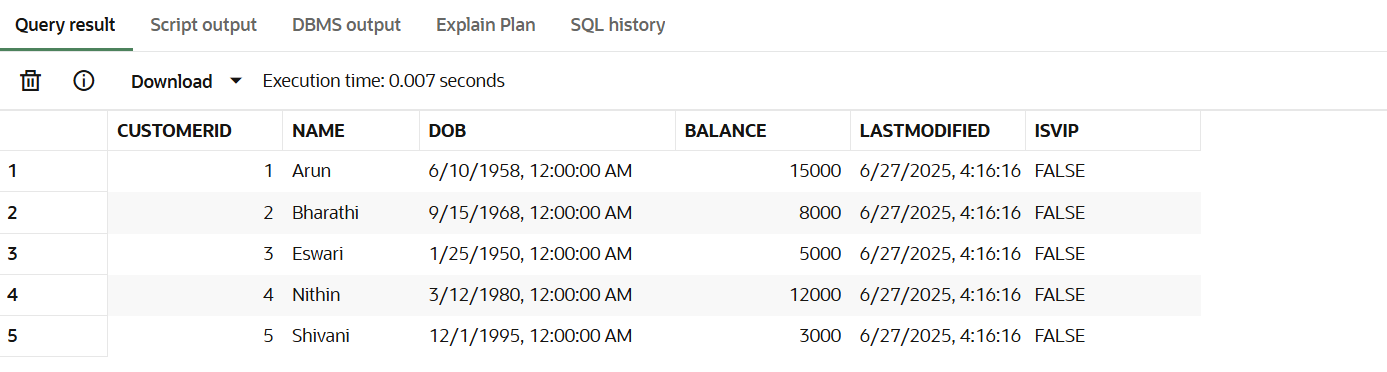
(2, 'Bharathi',TO\_DATE('1968-09-15', 'YYYY-MM-DD'), 8000,  SYSDATE, 'FALSE'),

(3, 'Eswari',  TO\_DATE('1950-01-25', 'YYYY-MM-DD'), 5000,  SYSDATE, 'FALSE'),

(4, 'Nithin',  TO\_DATE('1980-03-12', 'YYYY-MM-DD'), 12000, SYSDATE, 'FALSE'),

(5, 'Shivani', TO\_DATE('1995-12-01', 'YYYY-MM-DD'), 3000,  SYSDATE, 'FALSE');

SELECT \* FROM Customers;



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 Loans

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

(201, 1, 100000, 6.5, SYSDATE - 5, SYSDATE + 15),  -- Arun (66, due soon)

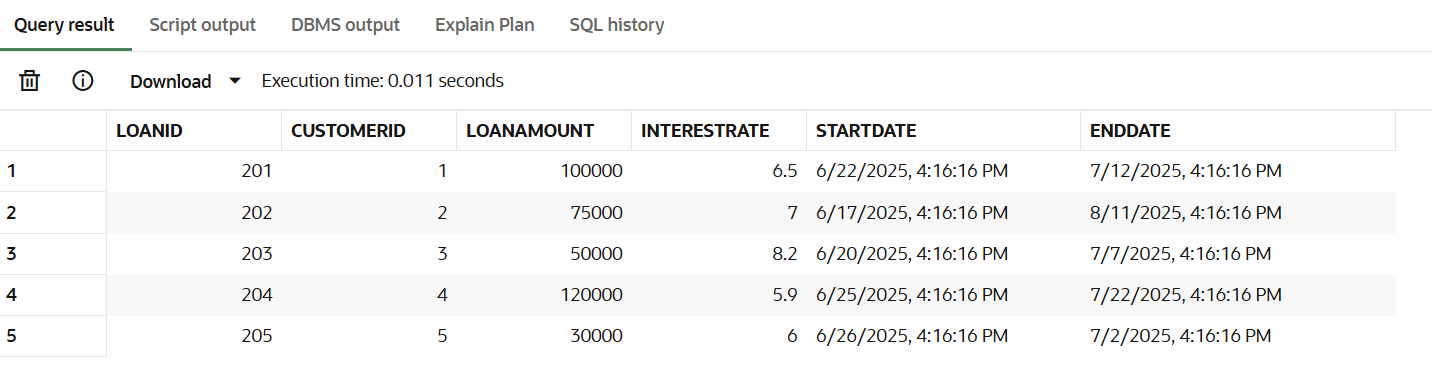
(202, 2, 75000,  7.0, SYSDATE - 10, SYSDATE + 45), -- Bharathi (not senior)

(203, 3, 50000,  8.2, SYSDATE - 7, SYSDATE + 10),  -- Eswari (75, due soon)

(204, 4, 120000, 5.9, SYSDATE - 2, SYSDATE + 25),  -- Nithin (VIP, due soon)

(205, 5, 30000,  6.0, SYSDATE - 1, SYSDATE + 5);   -- Shivani (young, due soon)

SELECT \* FROM Loans;



CREATE TABLE Accounts (

    AccountID NUMBER PRIMARY KEY,

    CustomerID NUMBER,

    AccountType VARCHAR2(20),

    Balance NUMBER,

    LastModified DATE,

    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified) VALUES

(301, 1, 'Savings', 15000, SYSDATE),

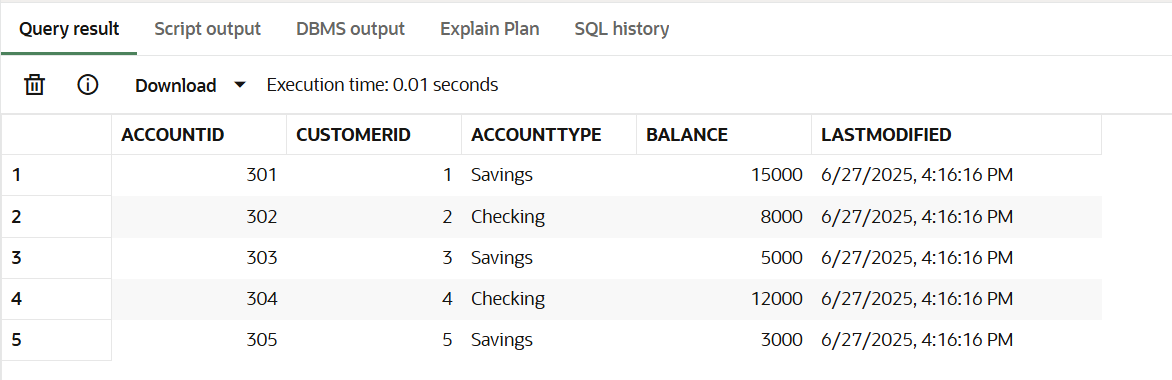
(302, 2, 'Checking', 8000, SYSDATE),

(303, 3, 'Savings', 5000, SYSDATE),

(304, 4, 'Checking', 12000, SYSDATE),

(305, 5, 'Savings', 3000, SYSDATE);

SELECT \* FROM Accounts;



CREATE TABLE Employees (

    EmployeeID NUMBER PRIMARY KEY,

    Name VARCHAR2(100),

    Position VARCHAR2(50),

    Salary NUMBER,

    Department VARCHAR2(50),

    HireDate DATE

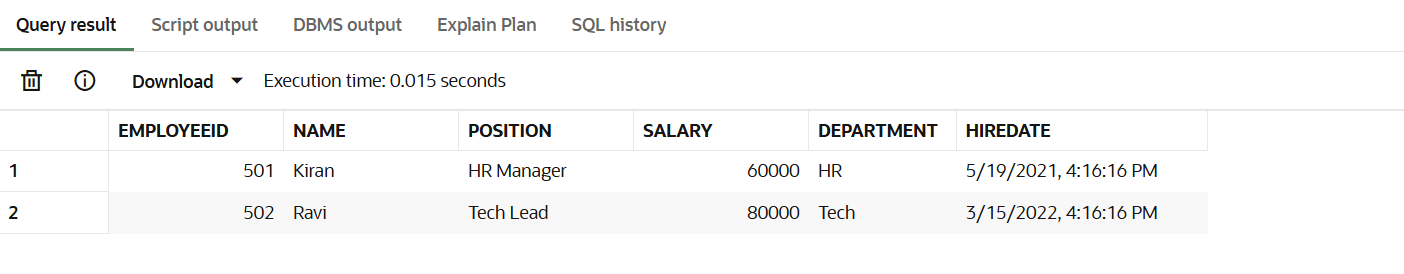
);

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate) VALUES

(501, 'Kiran', 'HR Manager', 60000, 'HR', SYSDATE - 1500),

(502, 'Ravi', 'Tech Lead', 80000, 'Tech', SYSDATE - 1200);

SELECT \* FROM Employees;



CREATE TABLE Transactions (

    TransactionID NUMBER PRIMARY KEY,

    AccountID NUMBER,

    TransactionDate DATE,

    Amount NUMBER,

    TransactionType VARCHAR2(10),

    FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)

);

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified) VALUES

(301, 1, 'Savings', 15000, SYSDATE),

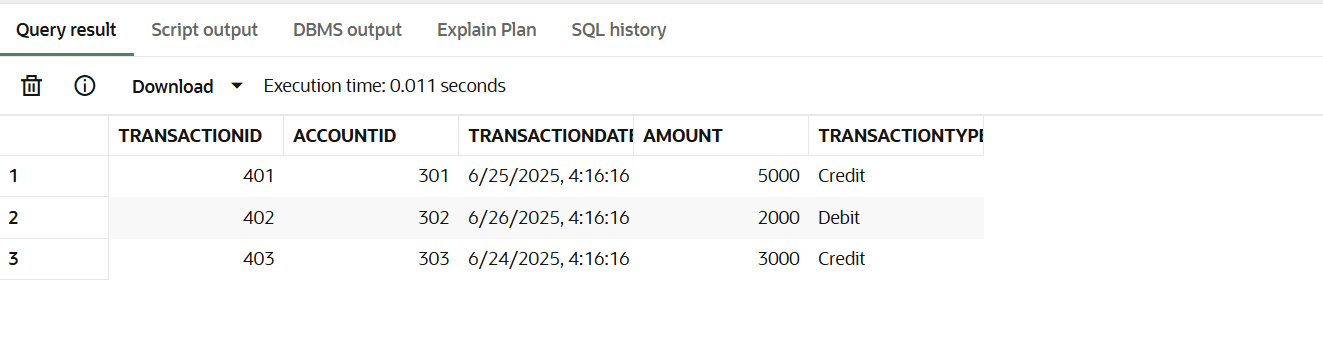
(302, 2, 'Checking', 8000, SYSDATE),

(303, 3, 'Savings', 5000, SYSDATE),

(304, 4, 'Checking', 12000, SYSDATE),

(305, 5, 'Savings', 3000, SYSDATE);

SELECT \* FROM Transactions;



**Scenario 1:**

BEGIN

  FOR cust\_rec IN (

    SELECT CustomerID

    FROM Customers

    WHERE FLOOR(MONTHS\_BETWEEN(SYSDATE, DOB) / 12) > 60

  ) LOOP

    UPDATE Loans

    SET InterestRate = InterestRate - 1

    WHERE CustomerID = cust\_rec.CustomerID;

    UPDATE Customers

    SET LastModified = SYSDATE

    WHERE CustomerID = cust\_rec.CustomerID;

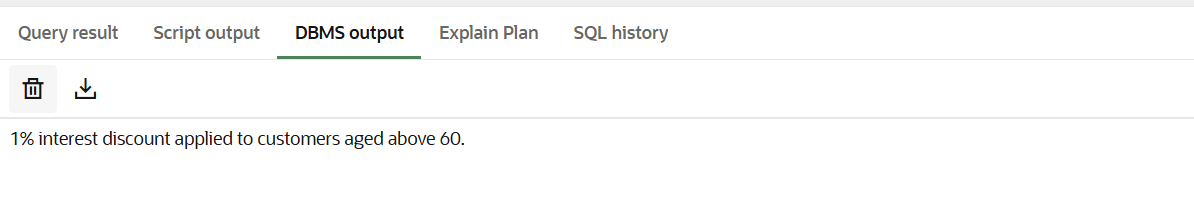
  END LOOP;

  COMMIT;

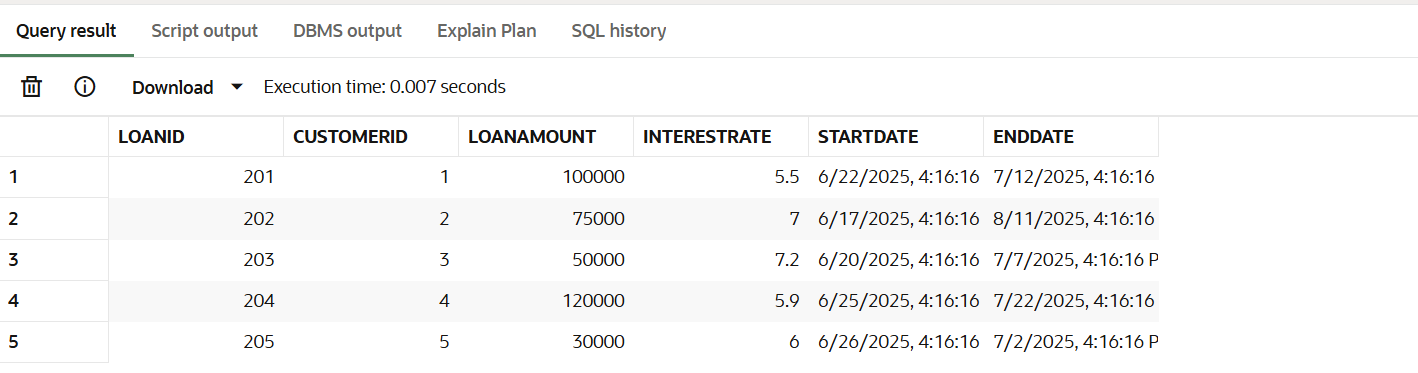
  DBMS\_OUTPUT.PUT\_LINE('1% interest discount applied to customers aged above 60.');

END;

/



SELECT \* FROM Loans;



**Scenario 2:**

BEGIN

  FOR cust\_rec IN (

    SELECT CustomerID

    FROM Customers

    WHERE Balance > 10000

  ) LOOP

    UPDATE Customers

    SET IsVIP = 'TRUE',

        LastModified = SYSDATE

    WHERE CustomerID = cust\_rec.CustomerID;

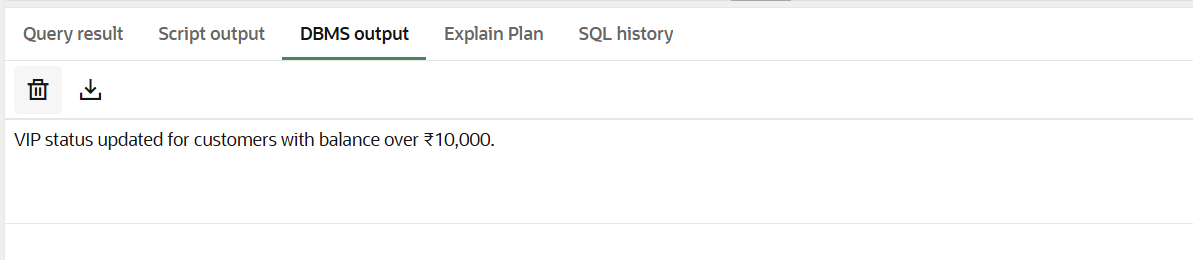
  END LOOP;

  COMMIT;

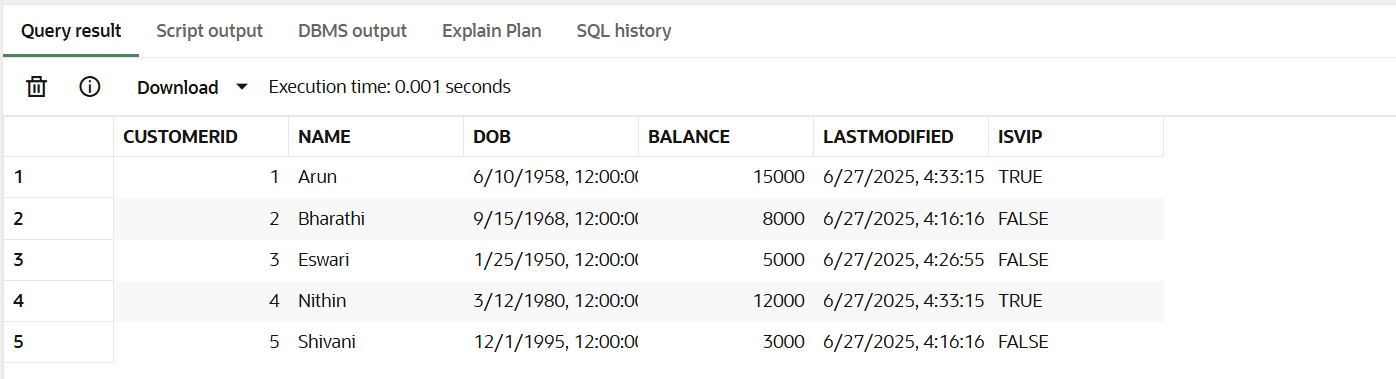
  DBMS\_OUTPUT.PUT\_LINE('VIP status updated for customers with balance over ₹10,000.');

END;

/



SELECT \* FROM Customers;



**Scenario 3:**

DECLARE

  CURSOR due\_loans IS

    SELECT L.LoanID, C.Name, C.Email, L.DueDate

    FROM Loans L

    JOIN Customers C ON L.CustomerID = C.CustomerID

    WHERE L.DueDate BETWEEN SYSDATE AND SYSDATE + 30;

  v\_loan\_id Loans.LoanID%TYPE;

  v\_name Customers.Name%TYPE;

  v\_email Customers.Email%TYPE;

  v\_due\_date Loans.DueDate%TYPE;

BEGIN

  OPEN due\_loans;

  LOOP

    FETCH due\_loans INTO v\_loan\_id, v\_name, v\_email, v\_due\_date;

    EXIT WHEN due\_loans%NOTFOUND;

    DBMS\_OUTPUT.PUT\_LINE('Reminder: Dear ' || v\_name ||

                         ', your loan (ID: ' || v\_loan\_id ||

                         ') is due on ' || TO\_CHAR(v\_due\_date, 'DD-Mon-YYYY') ||

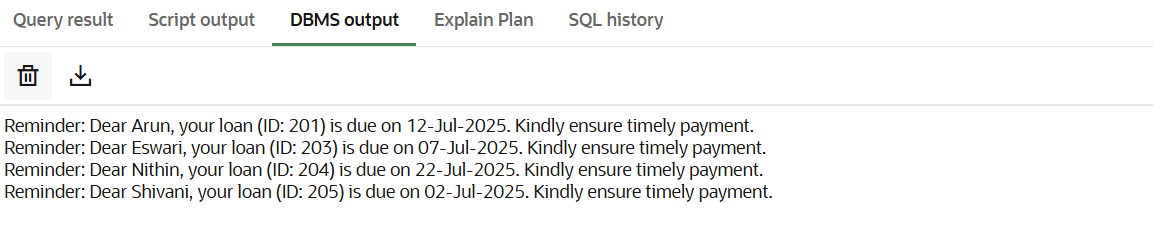
                         '. Kindly ensure timely payment.');

  END LOOP;

  CLOSE due\_loans;

END;

/

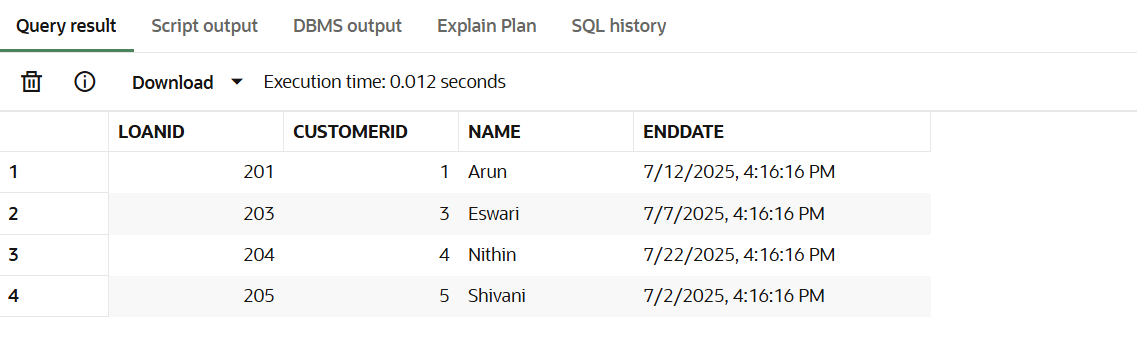


SELECT L.LoanID, L.CustomerID, C.Name, L.EndDate

FROM Loans L

JOIN Customers C ON C.CustomerID = L.CustomerID

WHERE L.EndDate BETWEEN SYSDATE AND SYSDATE + 30;



**Exercise 2: Stored Procedures**

**Scenario 1:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

  UPDATE Accounts

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

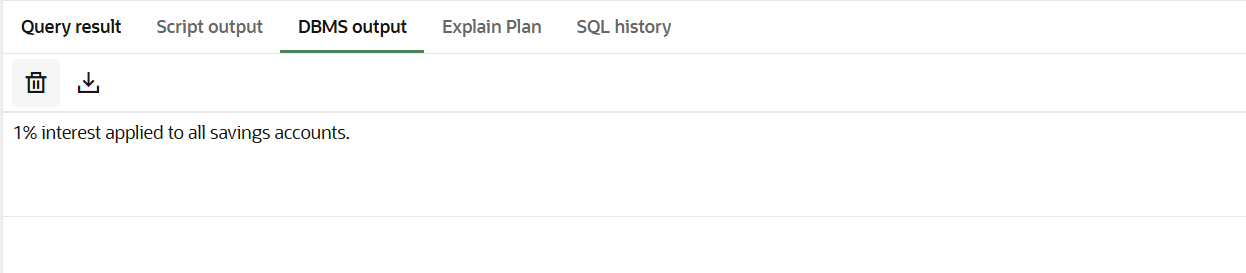
      LastModified = SYSDATE

  WHERE AccountType = 'Savings';

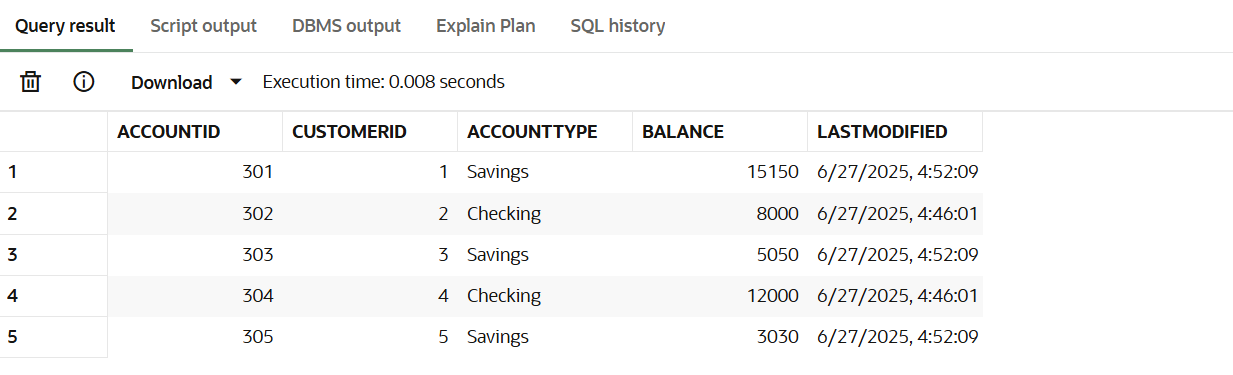
  COMMIT;

  DBMS\_OUTPUT.PUT\_LINE('1% interest applied to all savings accounts.');

END;



SELECT \* FROM Accounts;



**Scenario 2:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

  p\_dept IN VARCHAR2,

  p\_bonus\_percent IN NUMBER

) IS

BEGIN

  UPDATE Employees

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

  WHERE Department = p\_dept;

  COMMIT;

  DBMS\_OUTPUT.PUT\_LINE('Bonus of ' || p\_bonus\_percent || '% applied to ' || p\_dept || ' department.');

END;

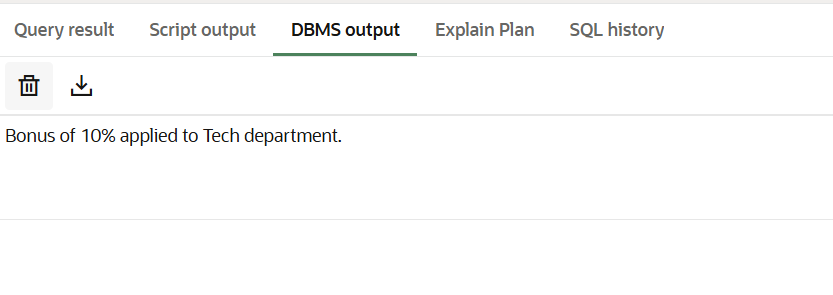
/

BEGIN

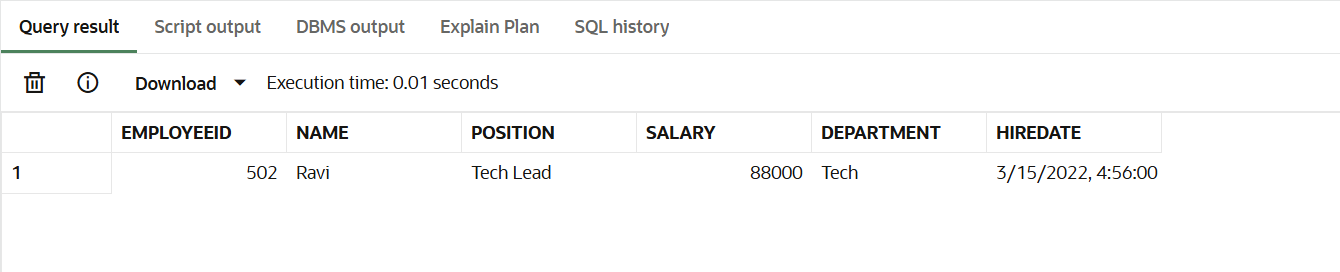
  UpdateEmployeeBonus('Tech', 10);

END;

/



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



**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

  SELECT Balance INTO v\_balance

  FROM Accounts

  WHERE AccountID = p\_from\_account;

  IF v\_balance < p\_amount THEN

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

  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;

  COMMIT;

  DBMS\_OUTPUT.PUT\_LINE('₹' || p\_amount || ' transferred from Account ' || p\_from\_account || ' to Account ' || p\_to\_account);

END;

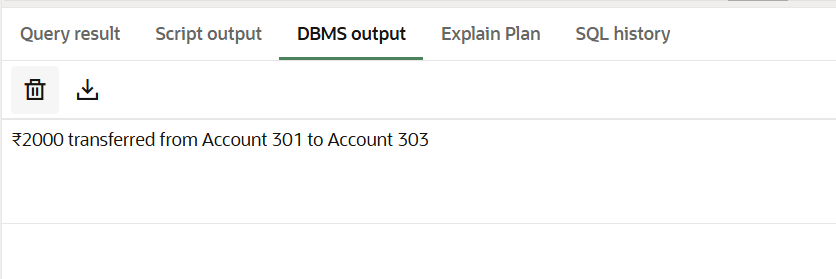
/

BEGIN

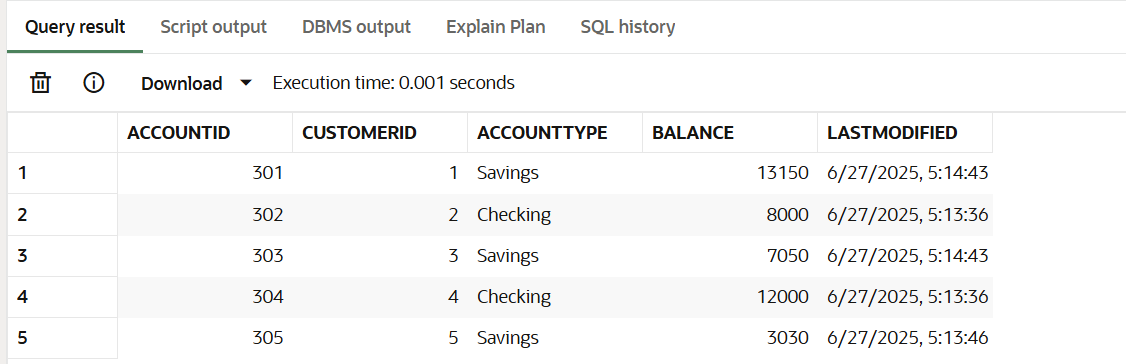
  TransferFunds(301, 303, 2000);

END;

/



SELECT \* FROM Accounts;

****