**Exercise 1: Control Structures**

**Table creation:**

--Customer table creation

BEGIN

  EXECUTE IMMEDIATE 'DROP TABLE Loans CASCADE CONSTRAINTS';

EXCEPTION

  WHEN OTHERS THEN

    NULL;

END;

/

--Loans table creation

BEGIN

  EXECUTE IMMEDIATE 'DROP TABLE Customers CASCADE CONSTRAINTS';

EXCEPTION

  WHEN OTHERS THEN

    NULL;

END;

/

CREATE TABLE Customers (

    CustomerID NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

    Name VARCHAR2(50),

    Age NUMBER,

    Balance NUMBER(10,2),

    IsVIP VARCHAR2(5) DEFAULT 'FALSE',

    Email VARCHAR2(100)

);

CREATE TABLE Loans (

    LoanID NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

    CustomerID NUMBER,

    InterestRate NUMBER(5,2),

    DueDate DATE,

    CONSTRAINT fk\_customer FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

-- Insert into Customers

INSERT INTO Customers (Name, Age, Balance, Email) VALUES

('Arun', 65, 15000.00, 'arun13@gmail.com'),

('Bharathi', 59, 8000.00, 'bharathi25@gmail.com'),

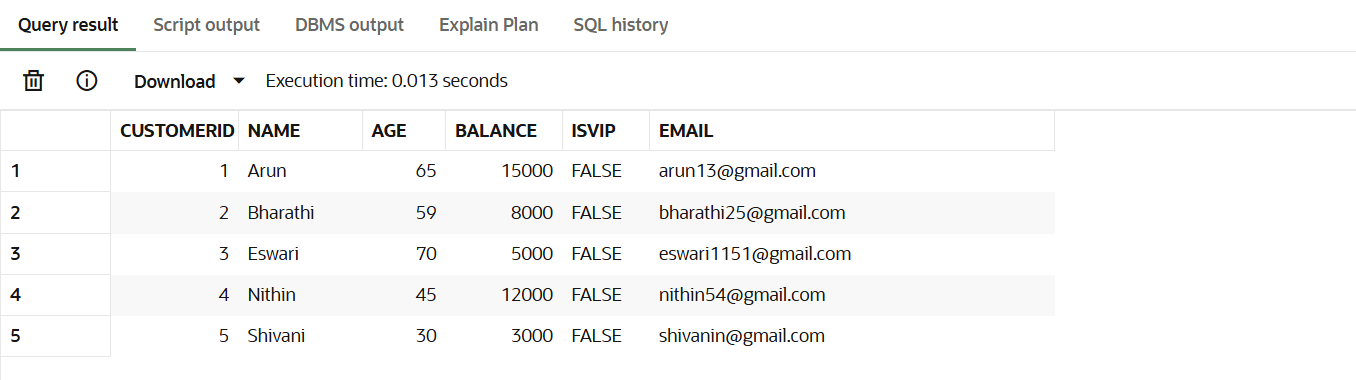
('Eswari', 70, 5000.00, 'eswari1151@gmail.com'),

('Nithin', 45, 12000.00, 'nithin54@gmail.com'),

('Shivani', 30, 3000.00, 'shivanin@gmail.com');

/

SELECT \* FROM Customers;

-- Insert into Loans

INSERT INTO Loans (CustomerID, InterestRate, DueDate) VALUES

(1, 6.5, SYSDATE + 15),

(2, 7.0, SYSDATE + 45),

(3, 8.2, SYSDATE + 10),

(4, 5.9, SYSDATE + 25),

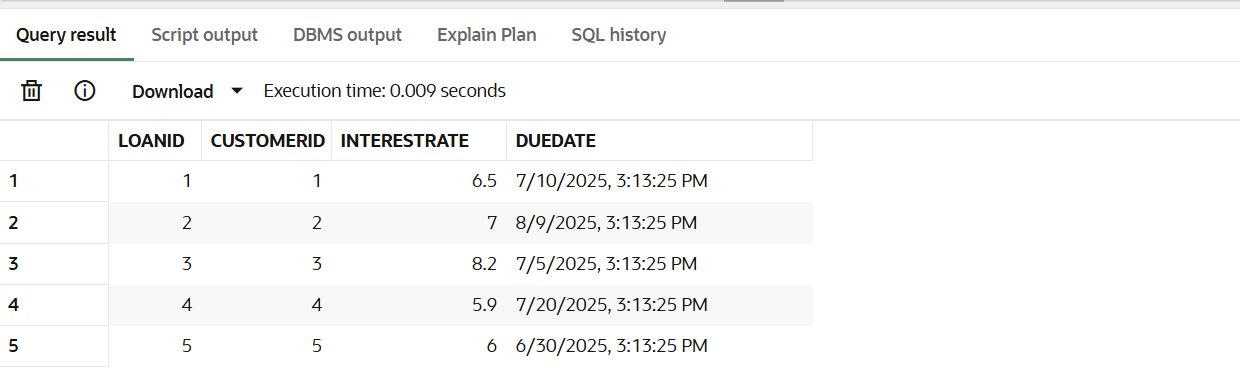
(5, 6.0, SYSDATE + 5);

/

COMMIT;

/

SELECT \* FROM Loans;



**Scenario 1:**

BEGIN

  FOR cust IN (SELECT CustomerID FROM Customers WHERE Age > 60) LOOP

    UPDATE Loans

    SET InterestRate = InterestRate - 1

    WHERE CustomerID = cust.CustomerID;

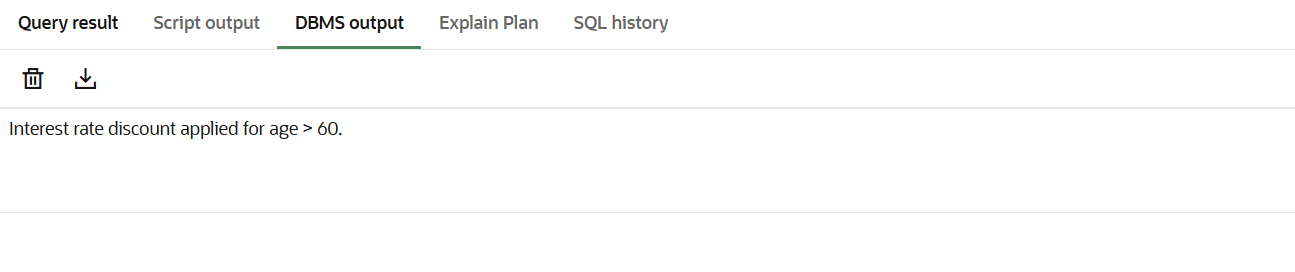
  END LOOP;

  COMMIT;

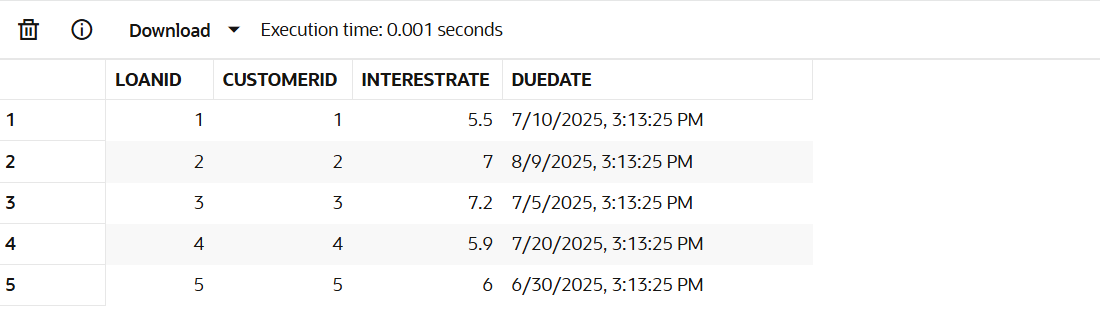
  DBMS\_OUTPUT.PUT\_LINE('Interest rate discount applied for age > 60.');

END;

/



SELECT \* FROM Loans;



**Scenario 2:**

BEGIN

  FOR cust IN (SELECT CustomerID FROM Customers WHERE Balance > 10000) LOOP

    UPDATE Customers

    SET IsVIP = 'TRUE'

    WHERE CustomerID = cust.CustomerID;

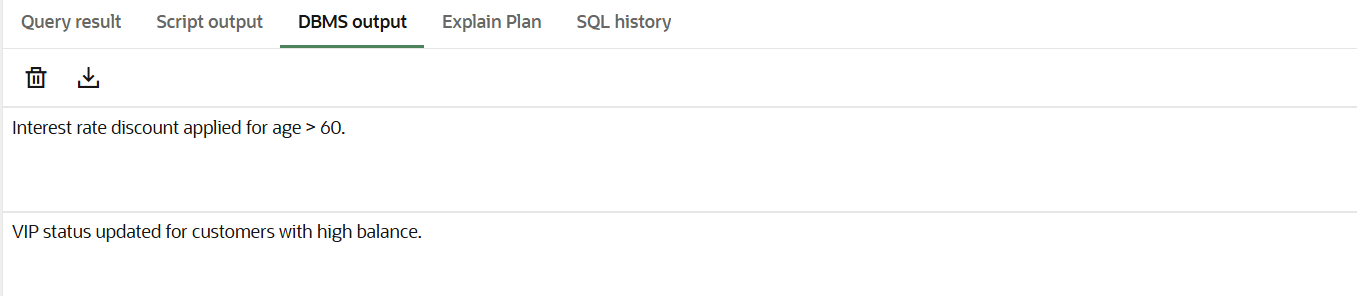
  END LOOP;

  COMMIT;

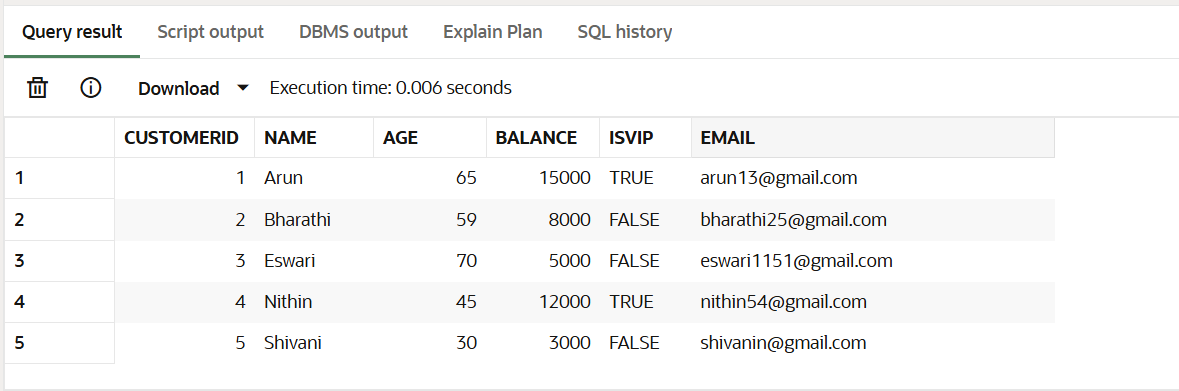
  DBMS\_OUTPUT.PUT\_LINE('VIP status updated for customers with high balance.');

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') ||

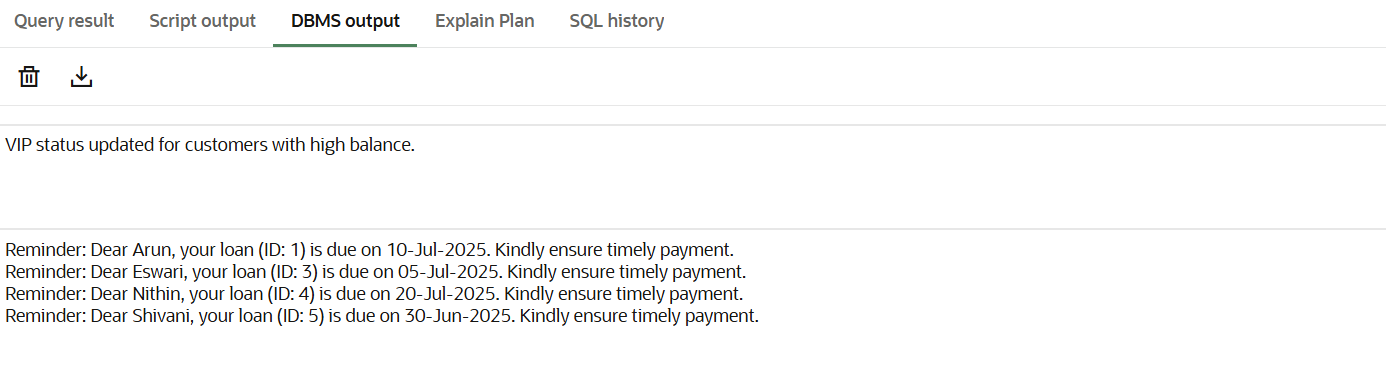
                         '. Kindly ensure timely payment.');

  END LOOP;

  CLOSE due\_loans;

END;

/



**Exercise 2: Stored Procedures**

**Table creation:**

CREATE TABLE Accounts (

  AccountID NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

  CustomerName VARCHAR2(100),

  Balance NUMBER(10, 2),

  AccountType VARCHAR2(20)  -- 'Savings' or 'Checking'

);

CREATE TABLE Employees (

  EmpID NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

  Name VARCHAR2(100),

  Department VARCHAR2(50),

  Salary NUMBER(10, 2)

);

INSERT INTO Accounts (CustomerName, Balance, AccountType) VALUES

('Arun', 10000, 'Savings'),

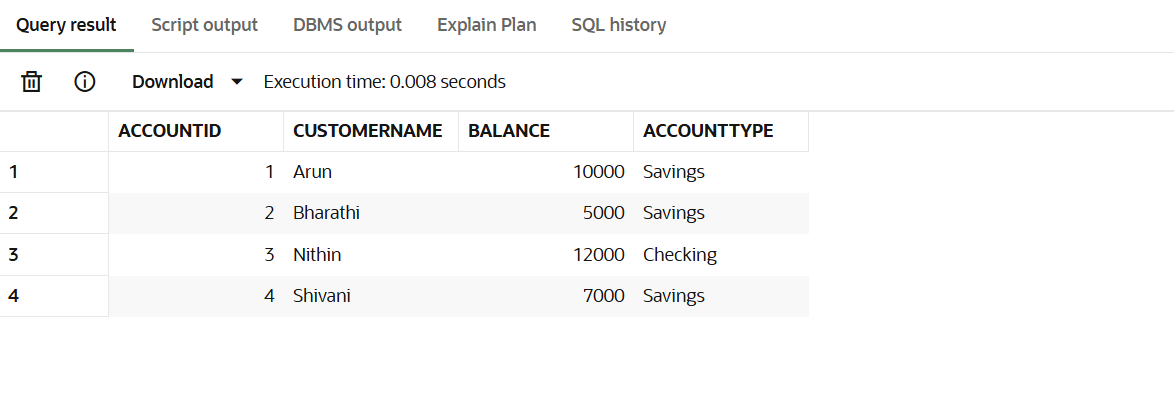
('Bharathi', 5000, 'Savings'),

('Nithin', 12000, 'Checking'),

('Shivani', 7000, 'Savings');

COMMIT;

SELECT \* FROM Accounts;



INSERT INTO Employees (Name, Department, Salary) VALUES

('Kiran', 'HR', 40000),

('Sundar', 'Finance', 45000),

('Vani', 'HR', 42000),

('Ravi', 'Tech', 60000),

('Anjali', 'Finance', 47000),

('Deepak', 'Tech', 65000),

('Priya', 'Marketing', 38000),

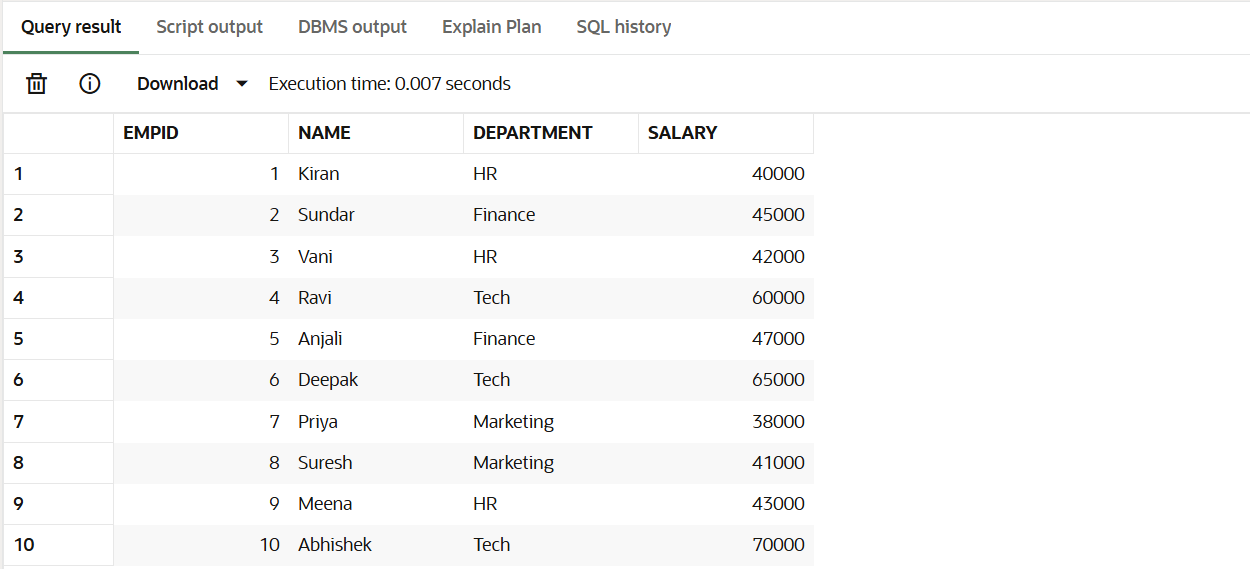
('Suresh', 'Marketing', 41000),

('Meena', 'HR', 43000),

('Abhishek', 'Tech', 70000);

COMMIT;

SELECT \* FROM Employees;



**Scenario 1:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

  UPDATE Accounts

  SET Balance = Balance + (Balance \* 0.01)

  WHERE AccountType = 'Savings';

  COMMIT;

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

END;

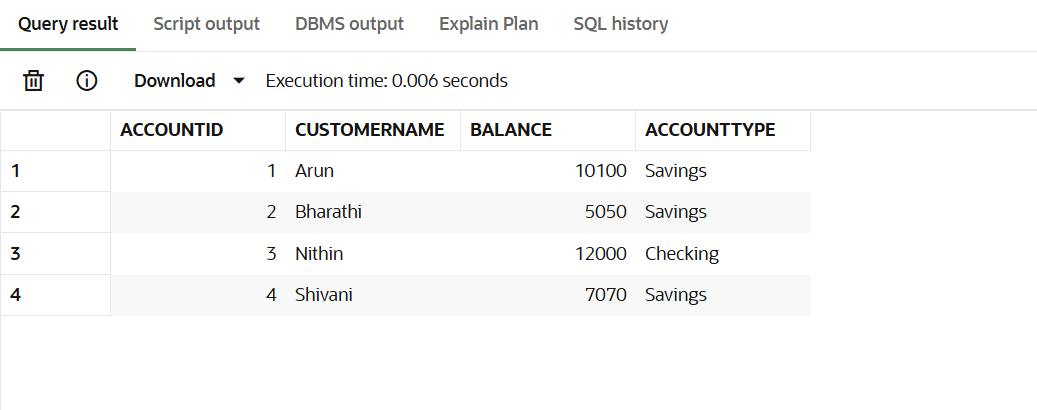
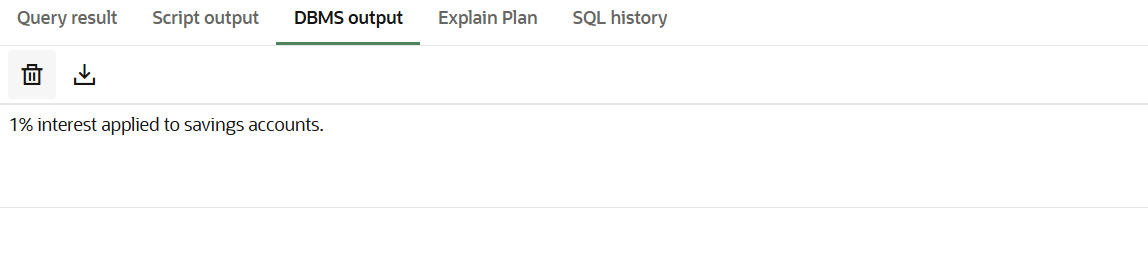
/

BEGIN

  ProcessMonthlyInterest;

END;

SELECT \* FROM Accounts;



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 || '% given to department: ' || p\_dept);

END;

/

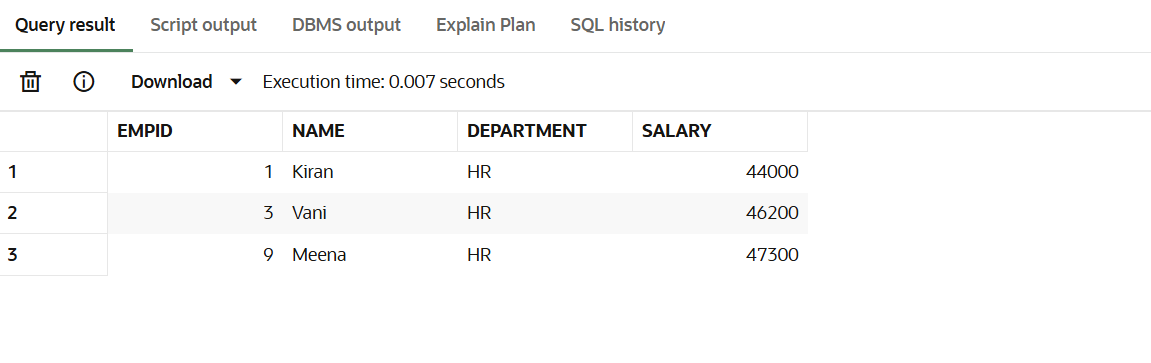
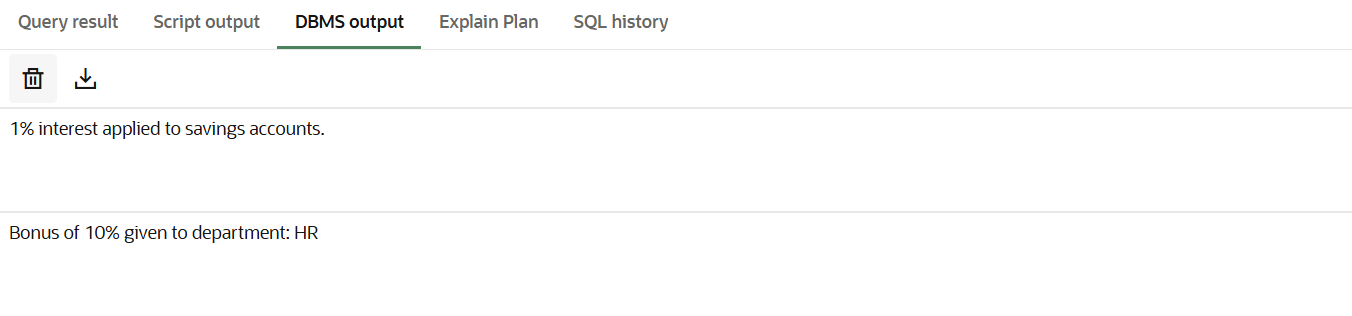
BEGIN

  UpdateEmployeeBonus('HR', 10);

END;

/

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



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

  WHERE AccountID = p\_from\_account;

  UPDATE Accounts

  SET Balance = Balance + p\_amount

  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(1, 2, 2000);

END;

/

SELECT \* FROM Accounts;

