**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.

BEGIN

    FOR rec IN (

        SELECT c.CustomerID, l.LoanID, c.Name, l.interestRate

        FROM Loans l JOIN Customers c ON l.CustomerID = c.CustomerID

        WHERE TRUNC(MONTHS\_BETWEEN(SYSDATE,c.DOB)/12)>60)

    LOOP

        UPDATE Loans

        SET InterestRate = rec.InterestRate - 1

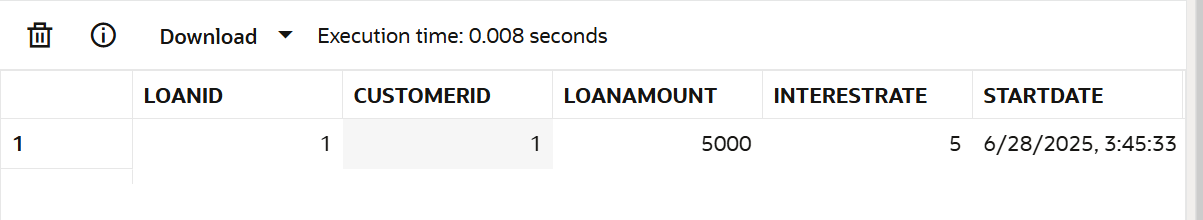
        WHERE LoanID = rec.LoanID;

    END LOOP;

    COMMIT;

END;

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.

ALTER TABLE Customers ADD IsVIP VARCHAR2(5);

BEGIN

    FOR rec IN (SELECT CustomerID, Balance FROM Customers)

    LOOP

        IF rec.Balance>10000 THEN

        UPDATE Customers

        SET IsVIP = 'TRUE'

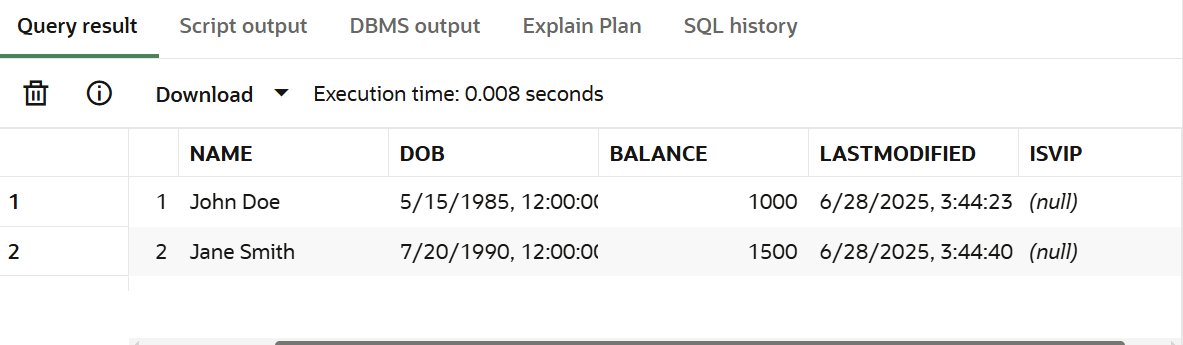
        WHERE CustomerID = rec.CustomerID;

        END IF;

    END LOOP;

END;

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.

SET SERVEROUTPUT ON;

BEGIN

    FOR rec IN (

        SELECT c.Name, c.CustomerID, l.loanID, l.EndDate FROM Customers c JOIN Loans l

        ON c.CustomerID = l.CustomerID WHERE l.EndDate <= SYSDATE + 30

    )

    LOOP

        DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || rec.LoanID ||

                     ', Customer: ' || rec.Name ||

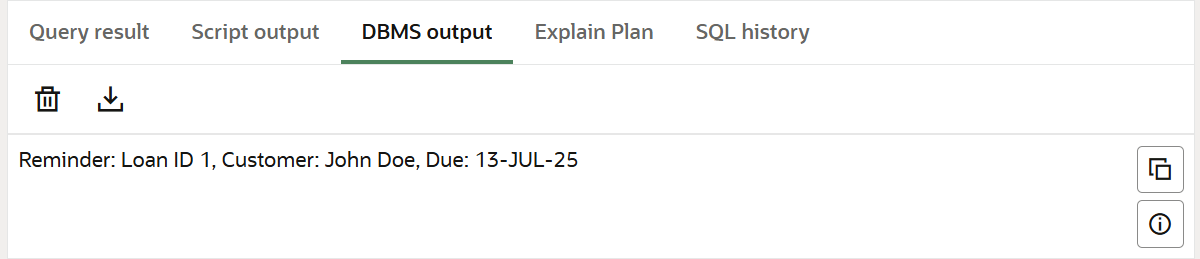
                     ', Due: ' || rec.EndDate);

    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.

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

  FOR rec IN (

    SELECT AccountID, Balance

    FROM Accounts

    WHERE AccountType = 'Savings'

  )

  LOOP

    UPDATE accounts

    SET balance = rec.balance + (rec.balance \* 0.01)

    WHERE AccountID = rec.AccountID;  -- ✅ important filter

  END LOOP;

END;

/

BEGIN

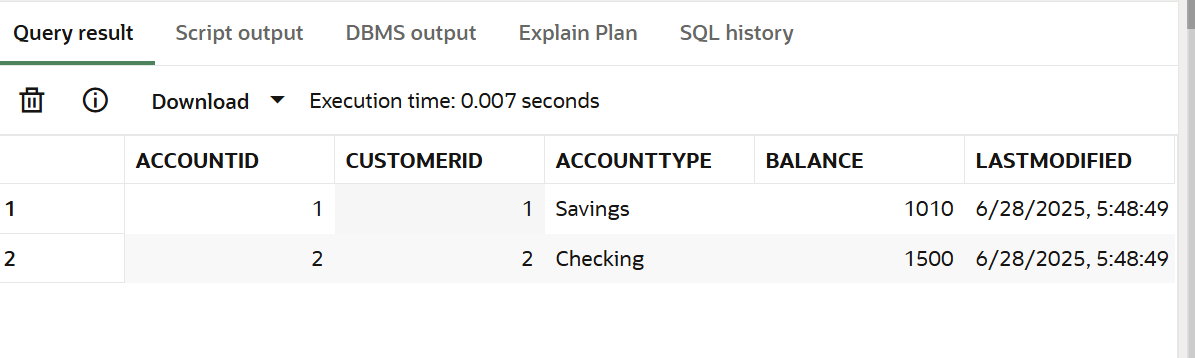
  ProcessMonthlyInterest;

END;

/

SELECT \* FROM ACCOUNTS;

**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.

create or replace procedure UpdateEmployeeBonus(

  dept in varchar2,

  bonus in number

) is

begin

  update employees

  set salary = salary+ (salary\*bonus/100)

  where department = dept;

end;

/

BEGIN

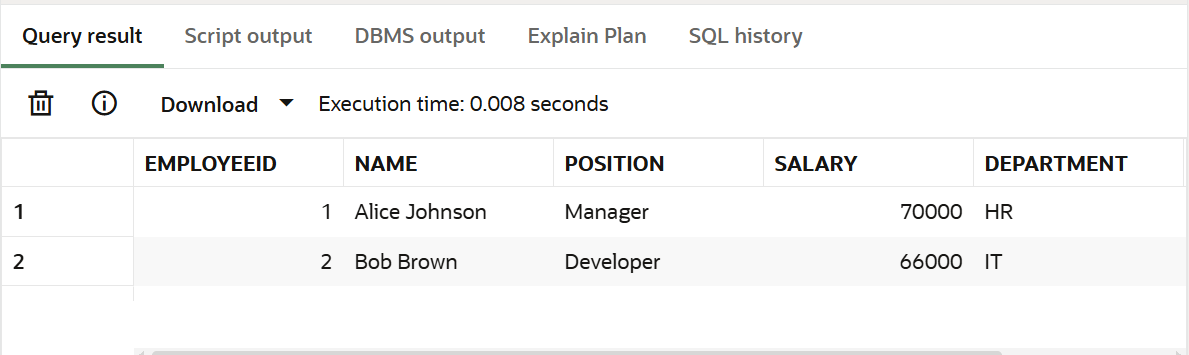
  UpdateEmployeeBonus('IT', 10);

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.

CREATE OR REPLACE PROCEDURE TransferFunds(

    from\_account IN NUMBER,

    to\_account IN NUMBER,

    amount IN NUMBER

) IS

    v\_balance NUMBER;

BEGIN

    SELECT BALANCE INTO v\_balance FROM ACCOUNTS

    WHERE ACCOUNTID = from\_account;

    IF v\_balance < amount THEN

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

    END IF;

    UPDATE ACCOUNTS

    SET BALANCE = BALANCE - amount,

        LastModified = SYSDATE

    WHERE ACCOUNTID = from\_account;

    UPDATE ACCOUNTS

    SET BALANCE = BALANCE + amount,

        LastModified = SYSDATE

    WHERE ACCOUNTID = to\_account;

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

END;

/

BEGIN

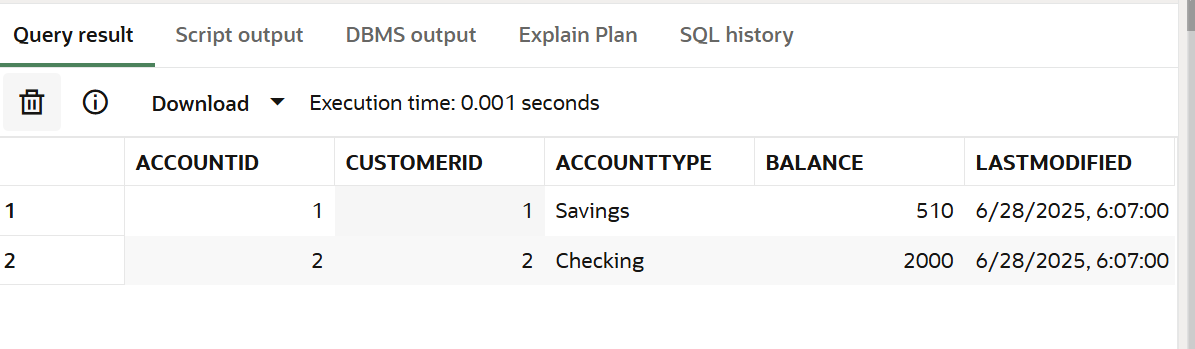
  TransferFunds(1, 2, 500);

END;

/

SELECT \* FROM ACCOUNTS;

**OUTPUT:**

****