**PL/SQL programming**

**Exercise 1: Control Structures :**

**///Customers table///**

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Age NUMBER,

Balance NUMBER,

InterestRate NUMBER,

IsVIP VARCHAR2(5)

);

**///Loans table///**

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

DueDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

**Insertion :**

INSERT INTO Customers VALUES (1, 'Alice', 65, 12000, 8.5, 'FALSE');

INSERT INTO Customers VALUES (2, 'Bob', 45, 8000, 9.0, 'FALSE');

INSERT INTO Customers VALUES (3, 'Charlie', 70, 15000, 10.0, 'FALSE');

INSERT INTO Customers VALUES (4, 'David', 30, 5000, 7.5, 'FALSE');

INSERT INTO Customers VALUES (5, 'Eva', 62, 11000, 8.0, 'FALSE');

INSERT INTO Loans VALUES (101, 1, SYSDATE + 15);

INSERT INTO Loans VALUES (102, 2, SYSDATE + 45);

INSERT INTO Loans VALUES (103, 3, SYSDATE + 10);

INSERT INTO Loans VALUES (104, 4, SYSDATE + 5);

INSERT INTO Loans VALUES (105, 5, SYSDATE + 60);

**Scenario 1: Apply 1% discount to interest rates for age > 60**

BEGIN

FOR customer\_rec IN (

SELECT CustomerID, InterestRate

FROM Customers

WHERE Age > 60

) LOOP

UPDATE Customers

SET InterestRate = InterestRate - 1

WHERE CustomerID = customer\_rec.CustomerID;

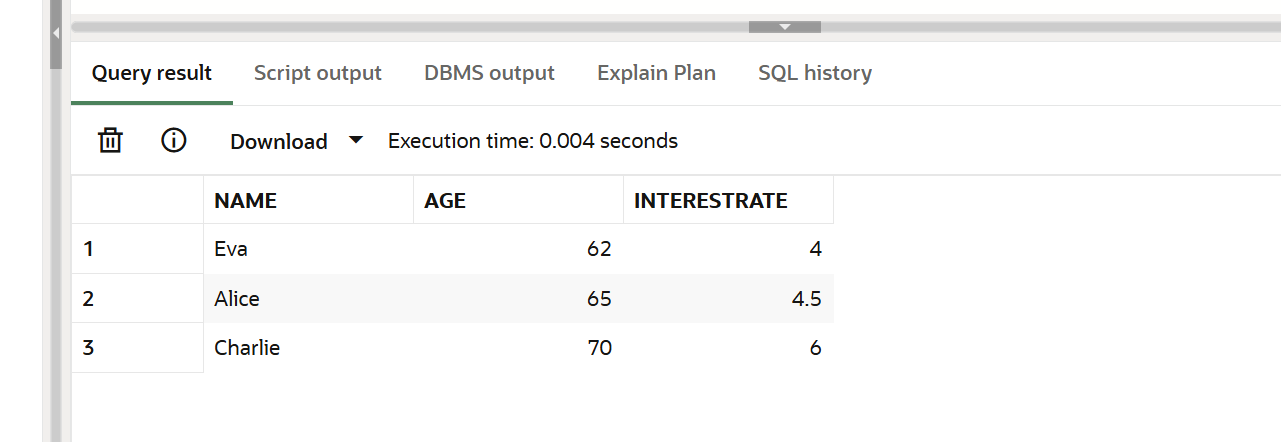
END LOOP;

COMMIT;

END;

**Displaying :**

SELECT Name, Age, InterestRate FROM Customers WHERE Age > 60;



**Scenario 2: Promote to VIP if balance > $10,000**

BEGIN

FOR cust IN (

SELECT CustomerID

FROM Customers

WHERE Balance > 10000

) LOOP

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = cust.CustomerID;

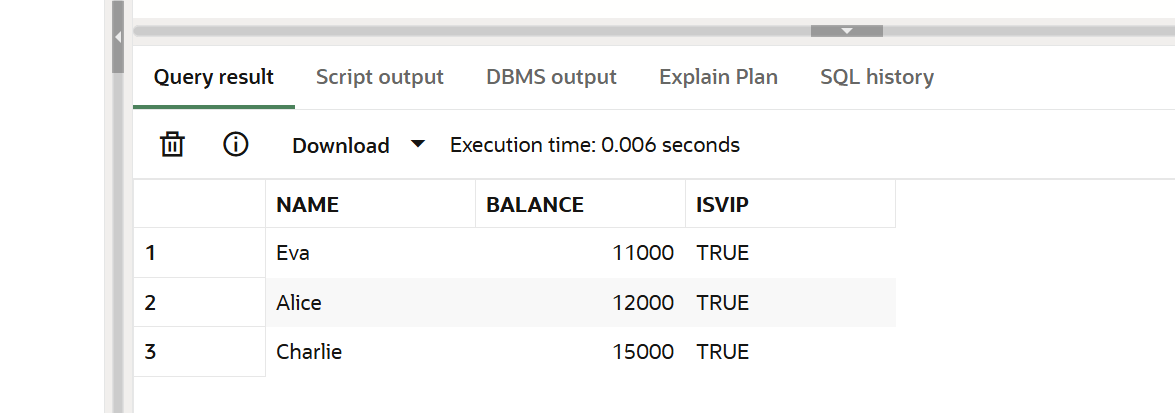
END LOOP;

COMMIT;

END;

**Displaying :**

SELECT Name, Balance, IsVIP FROM Customers WHERE IsVIP = 'TRUE';



**Scenario 3: Reminders for loans due within next 30 days**

DECLARE

CURSOR due\_loans IS

SELECT LoanID, CustomerID, DueDate

FROM Loans

WHERE DueDate <= SYSDATE + 30;

v\_name VARCHAR2(100);

BEGIN

FOR loan\_rec IN due\_loans LOOP

SELECT Name INTO v\_name

FROM Customers

WHERE CustomerID = loan\_rec.CustomerID;

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

', your loan (ID: ' || loan\_rec.LoanID ||

') is due on ' || TO\_CHAR(loan\_rec.DueDate, 'DD-Mon-YYYY'));

END LOOP;

END;

**Displaying :**

SELECT

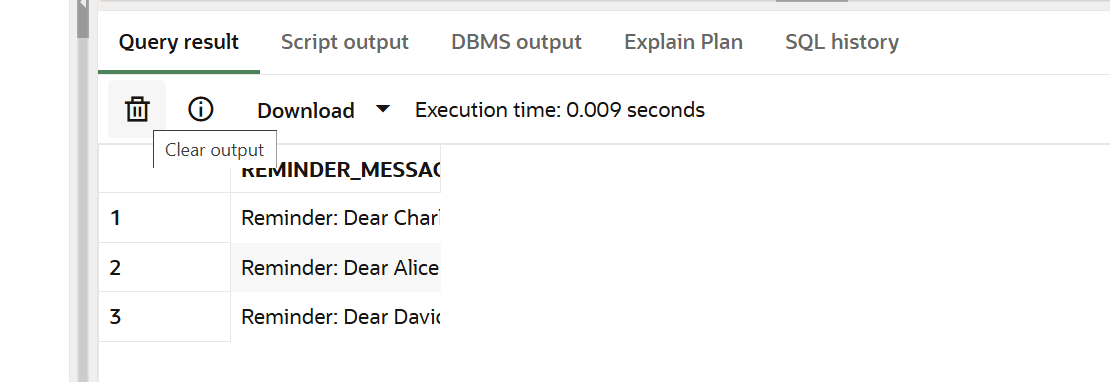
    'Reminder: Dear ' || c.Name ||

    ', your loan (ID: ' || l.LoanID ||

    ') is due on ' || TO\_CHAR(l.DueDate, 'DD-Mon-YYYY') AS Reminder\_Message

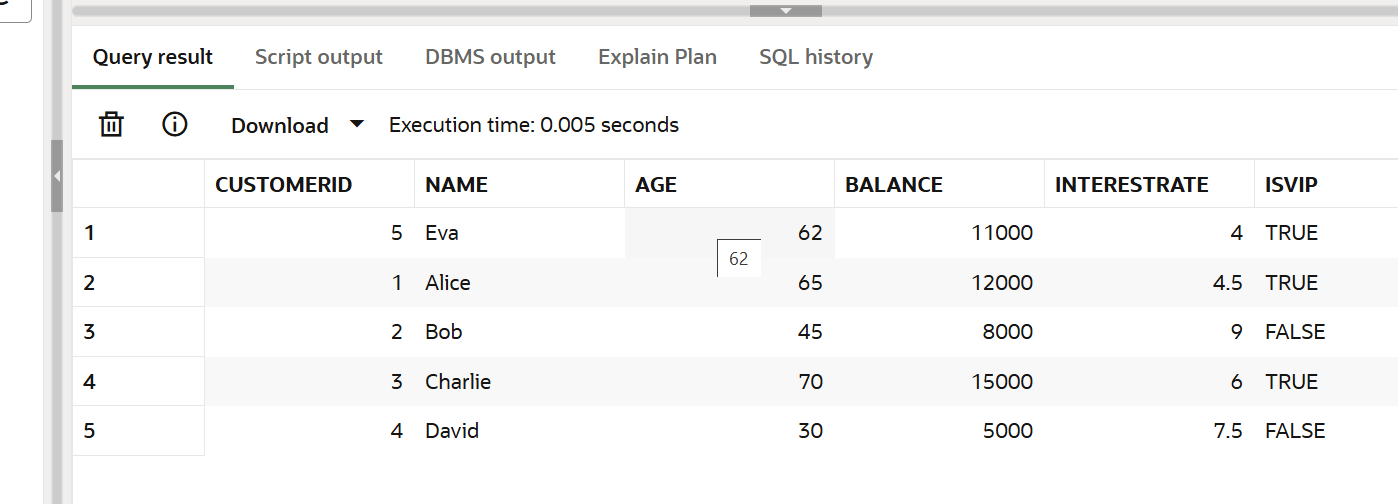
FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID WHERE l.DueDate <= SYSDATE + 30;

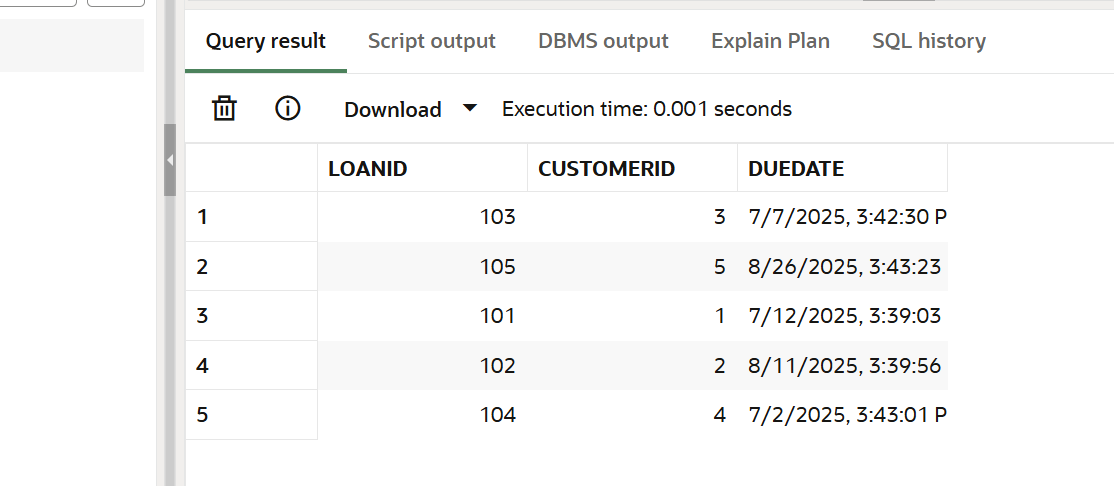


**Displaying the final results :**

SELECT \* FROM Customers;



SELECT \* FROM Loans;



**Exercise 3: Stored Procedures**

**///SavingsAccounts///**

CREATE TABLE SavingsAccounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

Balance NUMBER

);

**///Employees///**

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DepartmentID NUMBER,

Salary NUMBER

);

**///BankAccounts///**

CREATE TABLE BankAccounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

Balance NUMBER

);

**Insertion :**

**For** SavingsAccounts

INSERT INTO SavingsAccounts VALUES (201, 1, 10000);

INSERT INTO SavingsAccounts VALUES (202, 2, 5000);

INSERT INTO SavingsAccounts VALUES (203, 3, 20000);

For Employees

INSERT INTO Employees VALUES (301, 'Alice', 10, 50000);

INSERT INTO Employees VALUES (302, 'Bob', 10, 60000);

INSERT INTO Employees VALUES (303, 'Charlie', 20, 55000);

For BankAccounts

INSERT INTO BankAccounts VALUES (101, 1, 8000);

INSERT INTO BankAccounts VALUES (102, 2, 4000);

INSERT INTO BankAccounts VALUES (103, 3, 12000);

**Scenario 1: Monthly Interest for Savings Accounts**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR acc IN (SELECT AccountID, Balance FROM SavingsAccounts) LOOP

UPDATE SavingsAccounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountID = acc.AccountID;

END LOOP;

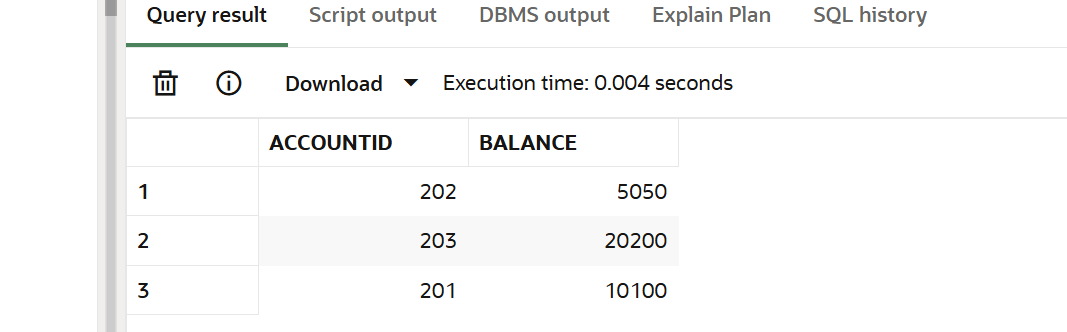
COMMIT;

END;

Procedure :

EXEC ProcessMonthlyInterest;

**Displaying:**

****

**Scenario 2: Update Employee Bonus Based on Performance**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_DepartmentID IN NUMBER,

p\_BonusPercent IN NUMBER

) IS

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* (p\_BonusPercent / 100))

WHERE DepartmentID = p\_DepartmentID;

COMMIT;

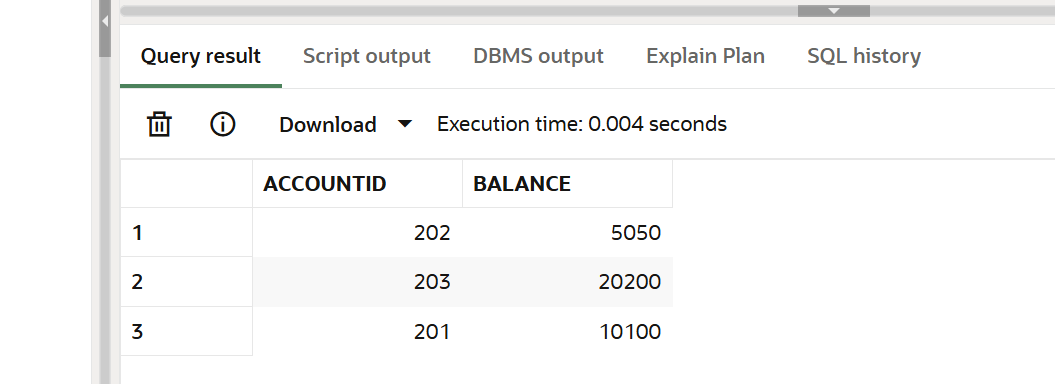
END;

Procedure :

EXEC UpdateEmployeeBonus(10, 5);

**Displaying:**

SELECT EmployeeID, Name, DepartmentID, Salary FROM Employees WHERE DepartmentID = 10;



**Scenario 3: Transfer Funds Between Accounts**

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_FromAccountID IN NUMBER,

p\_ToAccountID IN NUMBER,

p\_Amount IN NUMBER

) IS

v\_FromBalance NUMBER;

BEGIN

SELECT Balance INTO v\_FromBalance

FROM BankAccounts

WHERE AccountID = p\_FromAccountID;

IF v\_FromBalance < p\_Amount THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient balance for transfer');

END IF;

UPDATE BankAccounts

SET Balance = Balance - p\_Amount

WHERE AccountID = p\_FromAccountID;

UPDATE BankAccounts

SET Balance = Balance + p\_Amount

WHERE AccountID = p\_ToAccountID;

COMMIT;

END;

Procedure :

EXEC TransferFunds(101, 102, 500);

**Displaying:**

