Cognizant - DN 4.0 Deep Skilling Java FSE

Week 02 – PL/SQL PROGRAMMING

Superset ID: 6400600

Name: Sai Aravindh K

**Exercise 1: Control Structures**

**Scenario 1: The bank wants to apply a discount to loan interest rates for customers above 60 years old.**

BEGIN

FOR rec IN (

SELECT l.LoanID, l.InterestRate, c.Name

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE c.Age > 60

)

LOOP

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE LoanID = rec.LoanID;

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

' for customer ' || rec.Name ||

'. Old Rate: ' || rec.InterestRate ||

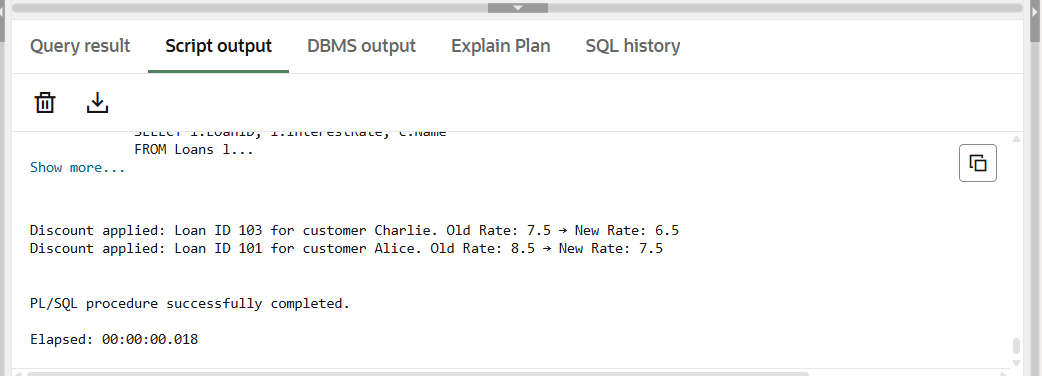
' → New Rate: ' || (rec.InterestRate - 1));

END LOOP;

END;

/

**Output:**

****

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

BEGIN

FOR rec IN (

SELECT CustomerID, Name, Balance

FROM Customers

WHERE Balance > 10000

)

LOOP

UPDATE Customers

SET IsVIP = 'Y'

WHERE CustomerID = rec.CustomerID;

-- Output message

DBMS\_OUTPUT.PUT\_LINE('Customer ' || rec.Name ||

' (ID: ' || rec.CustomerID ||

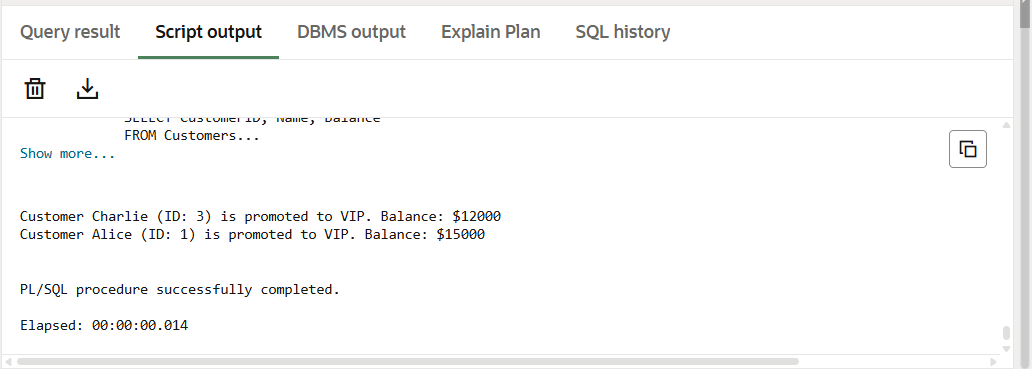
') is promoted to VIP. Balance: $' || rec.Balance);

END LOOP;

END;

/

**Output:**



**Scenario 3: The bank wants to send reminders to customers whose loans are due within the next 30 days.**

BEGIN

FOR rec IN (

SELECT l.LoanID, c.Name, l.DueDate

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.DueDate BETWEEN SYSDATE AND SYSDATE + 30

)

LOOP

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

' for customer ' || rec.Name ||

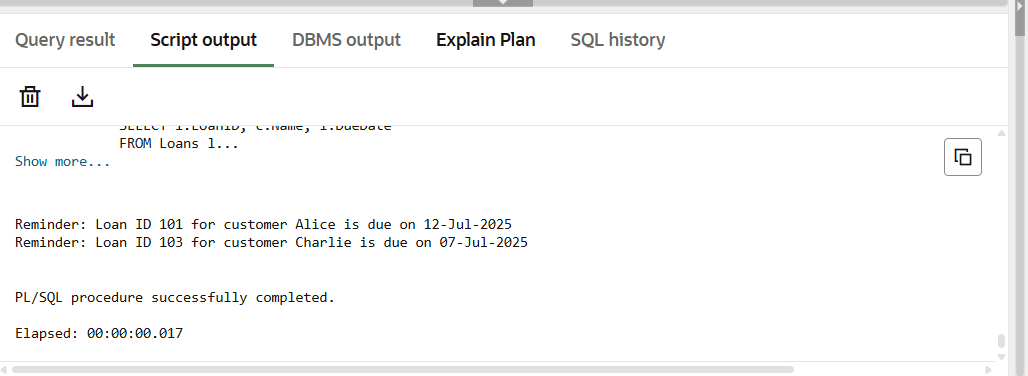
' is due on ' || TO\_CHAR(rec.DueDate, 'DD-Mon-YYYY'));

END LOOP;

END;

/

**Output:**

****

**Exercise 2: Stored Procedures**

**Scenario 1: The bank needs to process monthly interest for all savings accounts.**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountType = 'Savings';

END;

/

BEGIN

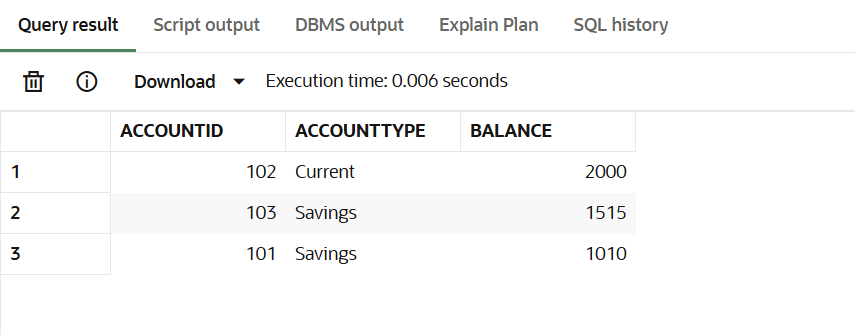
ProcessMonthlyInterest;

END;

/

SELECT \* FROM Accounts;

**Output:**

****

**Scenario 2: The bank wants to implement a bonus scheme for employees based on their 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;

END;

/

BEGIN

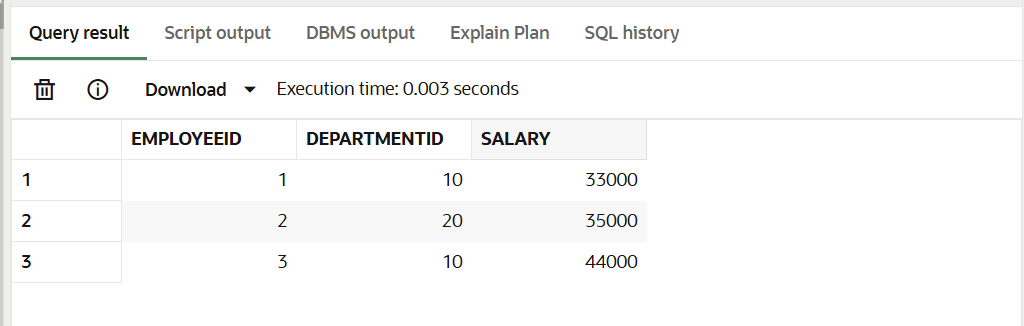
UpdateEmployeeBonus(10, 10); -- 10% bonus to dept 10

END;

/

SELECT \* FROM Employees;

**Output:**

****

**Scenario 3: Customers should be able to transfer funds between their 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 Accounts

WHERE AccountID = p\_FromAccountID;

IF v\_FromBalance >= p\_Amount THEN

UPDATE Accounts

SET Balance = Balance - p\_Amount

WHERE AccountID = p\_FromAccountID;

UPDATE Accounts

SET Balance = Balance + p\_Amount

WHERE AccountID = p\_ToAccountID;

ELSE

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

END IF;

END;

/

BEGIN

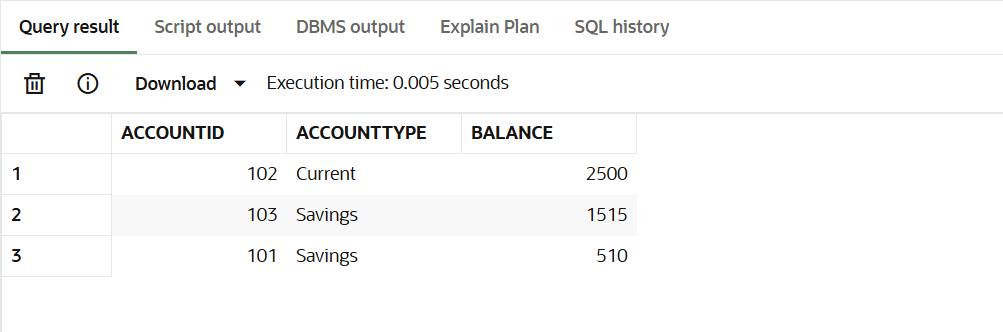
TransferFunds(101, 102, 500);

END;

/

SELECT \* FROM Accounts;

**Output:**

****