**HANDSON EXERCISES - WEEK 2**

**Skill : PL/SQL**

**Exercise 1: Control Structures**

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

**Question 1 :** 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.

**CODE :**

BEGIN

FOR cust IN (SELECT CustomerID, DOB FROM Customers) LOOP

DECLARE

v\_age NUMBER;

BEGIN

SELECT FLOOR(MONTHS\_BETWEEN(SYSDATE, cust.DOB) / 12)

INTO v\_age

FROM dual;

IF v\_age > 60 THEN

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = cust.CustomerID;

DBMS\_OUTPUT.PUT\_LINE('Discount applied for Customer ID: ' || cust.CustomerID || ', Age: ' || v\_age);

END IF;

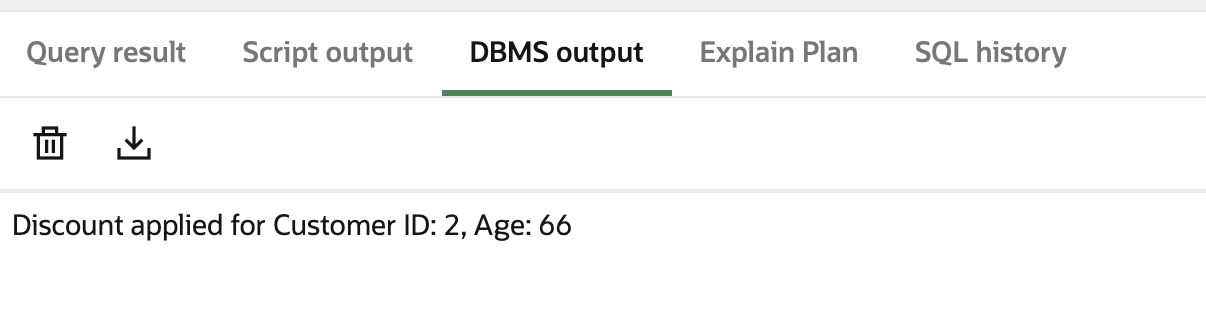
END;

END LOOP;

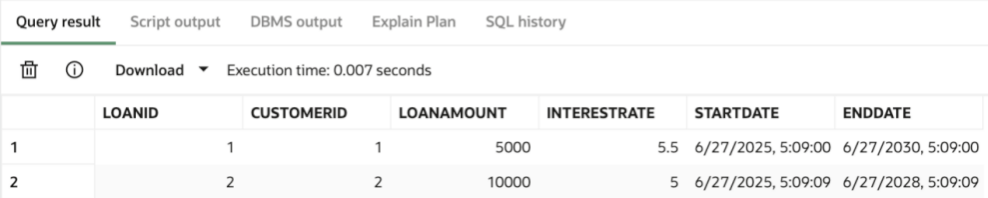
COMMIT;

END;

/

**OUTPUT :**  


SELECT \* FROM Loans;



**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.

**CODE :**

BEGIN

FOR cust IN (SELECT CustomerID, Name, Balance FROM Customers) LOOP

IF cust.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = cust.CustomerID;

DBMS\_OUTPUT.PUT\_LINE('Customer ' || cust.Name || ' (ID: ' || cust.CustomerID || ') promoted to VIP.');

ELSE

UPDATE Customers

SET IsVIP = 'FALSE'

WHERE CustomerID = cust.CustomerID;

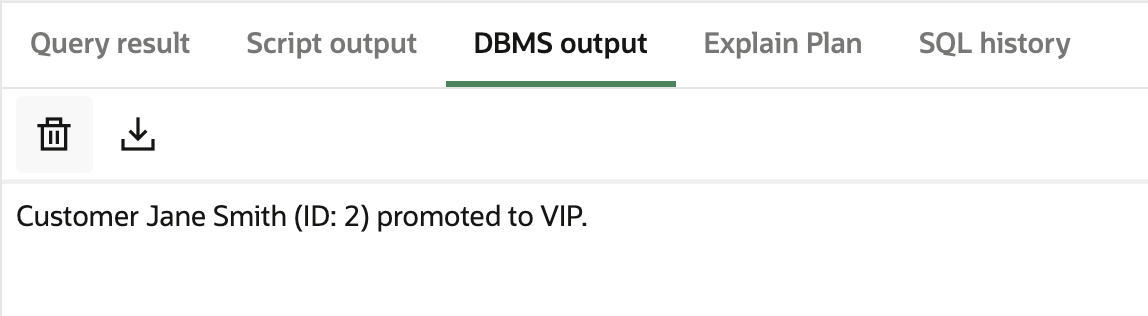
END IF;

END LOOP;

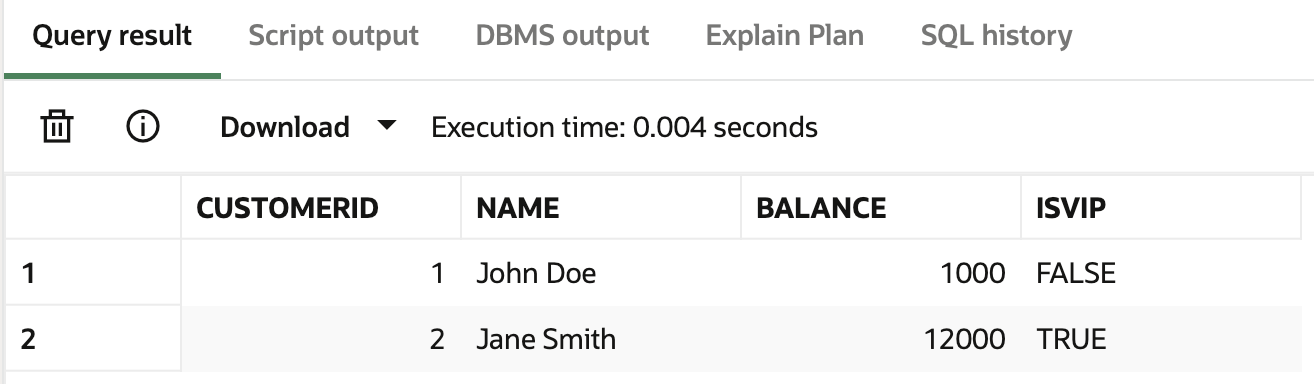
COMMIT;

END;

/

**OUTPUT :**

SELECT CustomerID, Name, Balance, IsVIP FROM Customers;



**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.

**CODE :**

UPDATE Loans SET EndDate = SYSDATE + 10 WHERE LoanID = 2;

**After updating ,**BEGIN

FOR loan\_rec IN (

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

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Reminder: Loan ID ' || loan\_rec.LoanID || ' for customer ' || loan\_rec.Name ||

' is due on ' || TO\_CHAR(loan\_rec.EndDate, 'DD-MON-YYYY')

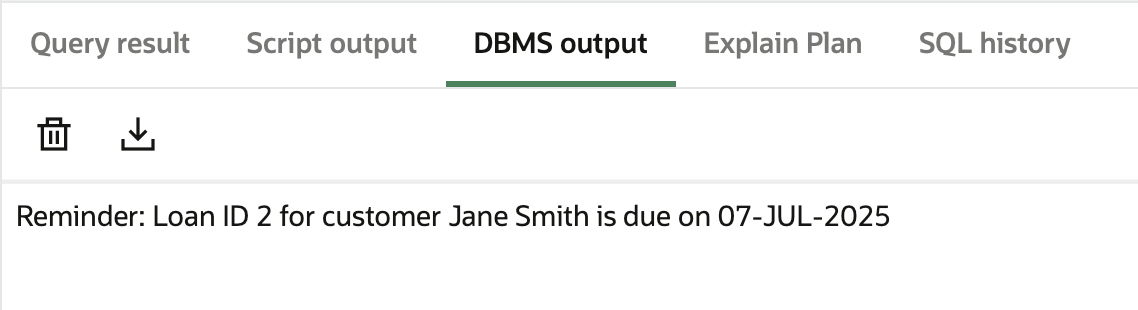
);

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.

**CODE :**

**Creating a Procedure**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR acc IN (

SELECT AccountID, Balance

FROM Accounts

WHERE AccountType = 'Savings'

) LOOP

UPDATE Accounts

SET Balance = Balance + (acc.Balance \* 0.01)

WHERE AccountID = acc.AccountID;

DBMS\_OUTPUT.PUT\_LINE(

'1% interest added to Account ID: ' || acc.AccountID ||

', New Balance: ' || TO\_CHAR(acc.Balance \* 1.01, '9999.99')

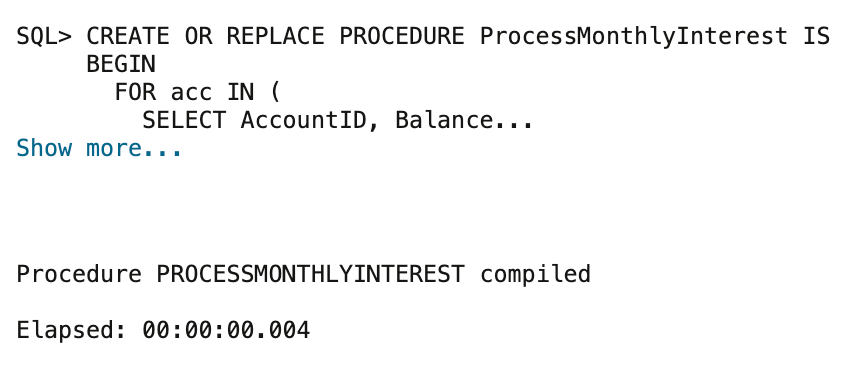
);

END LOOP;

COMMIT;

END;

/

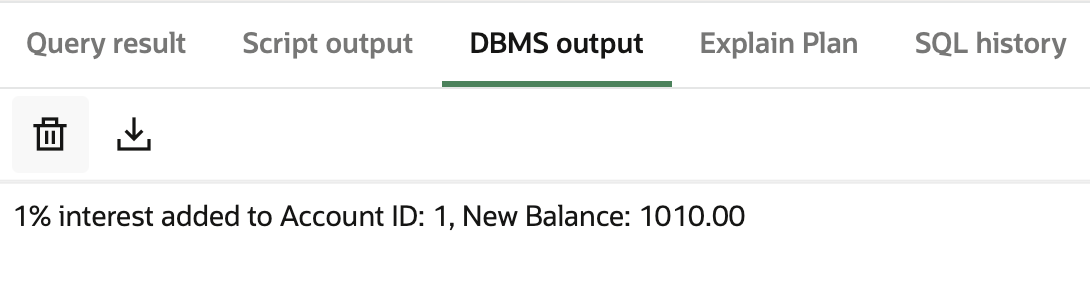


**Running the Procedure :**BEGIN

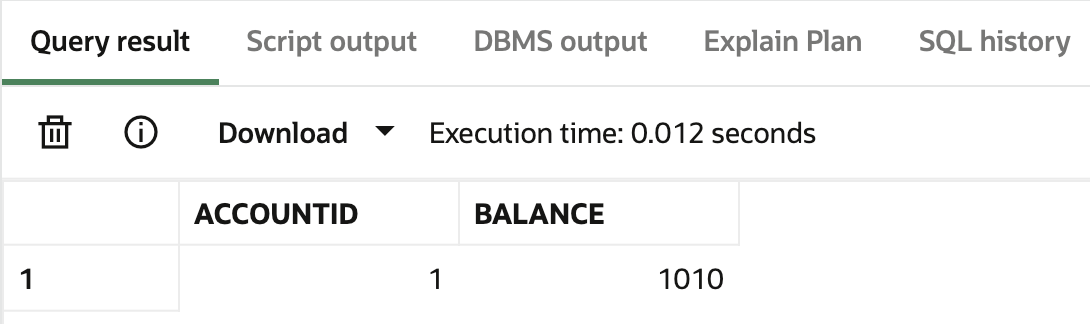
ProcessMonthlyInterest;

END;

/



SELECT AccountID, Balance FROM Accounts WHERE AccountID = 1;



**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.

**CODE :**

**Creating a Procedure**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_dept IN VARCHAR2,

p\_bonus\_percent IN NUMBER

) IS

BEGIN

FOR emp IN (

SELECT EmployeeID, Name, Salary

FROM Employees

WHERE Department = p\_dept

) LOOP

UPDATE Employees

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

WHERE EmployeeID = emp.EmployeeID;

DBMS\_OUTPUT.PUT\_LINE(

'Bonus of ' || p\_bonus\_percent || '% applied to ' || emp.Name ||

' (ID: ' || emp.EmployeeID || '), New Salary: ' ||

TO\_CHAR(emp.Salary \* (1 + p\_bonus\_percent / 100), '999999.99')

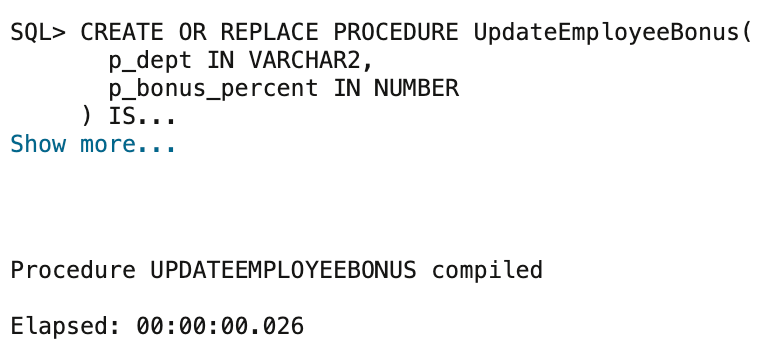
);

END LOOP;

COMMIT;

END;

/



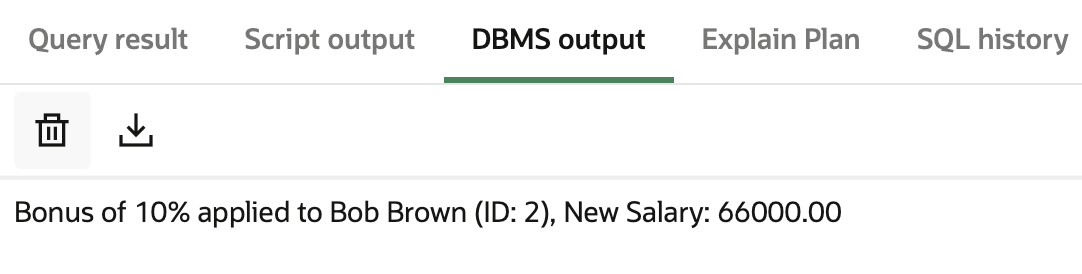
**Running the Procedure :**

BEGIN

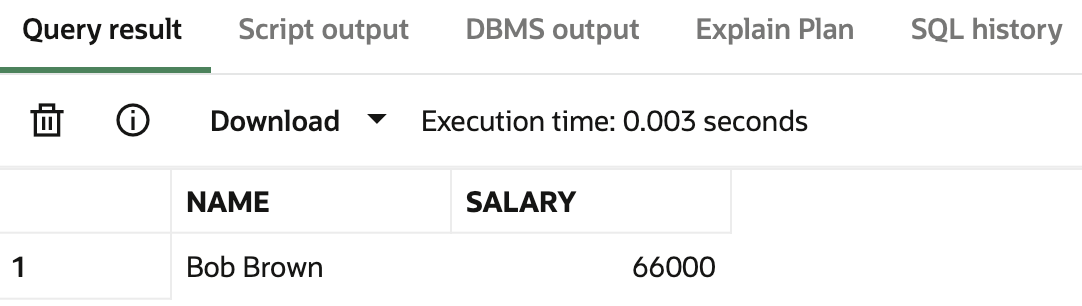
UpdateEmployeeBonus('IT', 10);

END;

/



SELECT Name, Salary FROM Employees WHERE EmployeeID = 2;



**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.

**CODE :**

**Creating a Procedure**

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_from\_account\_id IN NUMBER,

p\_to\_account\_id IN NUMBER,

p\_amount IN NUMBER

) IS

v\_from\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_from\_balance

FROM Accounts

WHERE AccountID = p\_from\_account\_id;

IF v\_from\_balance >= p\_amount THEN

UPDATE Accounts

SET Balance = Balance - p\_amount

WHERE AccountID = p\_from\_account\_id;

UPDATE Accounts

SET Balance = Balance + p\_amount

WHERE AccountID = p\_to\_account\_id;

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

ELSE

DBMS\_OUTPUT.PUT\_LINE('Insufficient funds in Account ' || p\_from\_account\_id);

END IF;

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

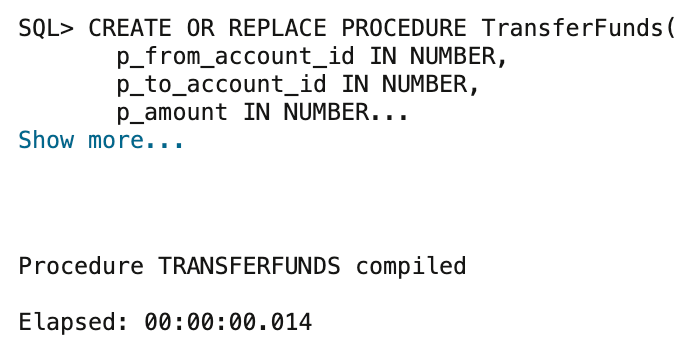
DBMS\_OUTPUT.PUT\_LINE('One of the accounts does not exist.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An unexpected error occurred: ' || SQLERRM);

END;

/



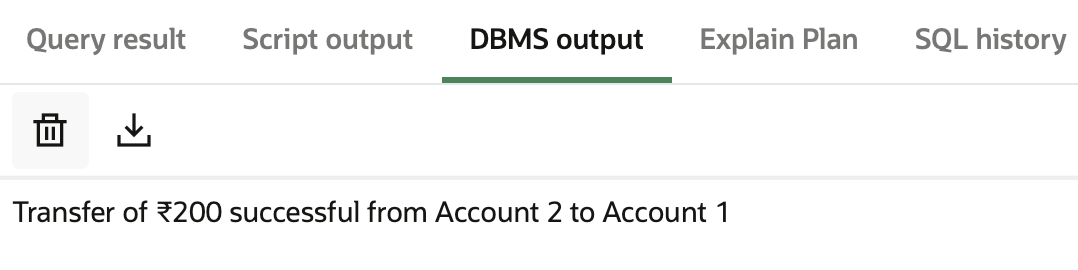
**Running the Procedure :**

BEGIN

TransferFunds(2, 1, 200);

END;

/



SELECT AccountID, Balance FROM Accounts;

