**Student Name:** KAMALESHWARAN A

**Registration No:** 22CSR087

**Course/Batch:** KONGU ENGINEERING COLLEGE (B.E COMPUTER SCIENCE AND ENGINEERING)

**EXERCISE 1: CONTROL STRUCTURES**

**Introduction:**

This PL/SQL script automates customer-based operations, including applying loan interest discounts, promoting customers to VIP status, and generating upcoming loan due reminders.

**Objective:**

* **Apply Interest Discounts:** Reduce the interest rate by 1% for customers above 60 years of age.
* **Promote VIP Status:** Identify customers with balances over ₹10,000 and update their status to VIP.
* **Send Loan Reminders:** Generate reminders for customers with loans due within the next 30 days.

**Implementation Breakdown:**

**Scenario 1:**

BEGIN

FOR c IN (

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

FROM Customers c

JOIN Loans l ON c.CustomerID = l.CustomerID

) LOOP

IF c.Age > 60 THEN

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE LoanID = c.LoanID;

END IF;

END LOOP;

COMMIT;

END;

**Scenario 2:**

BEGIN

FOR c IN (

SELECT CustomerID, Name, Balance, IsVIP

FROM Customers

) LOOP

IF c.Balance > 10000 AND c.IsVIP = 'N' THEN

UPDATE Customers

SET IsVIP = 'Y'

WHERE CustomerID = c.CustomerID;

END IF;

END LOOP;

COMMIT;

END;

**Scenario 3:**

BEGIN

FOR l IN (

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

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

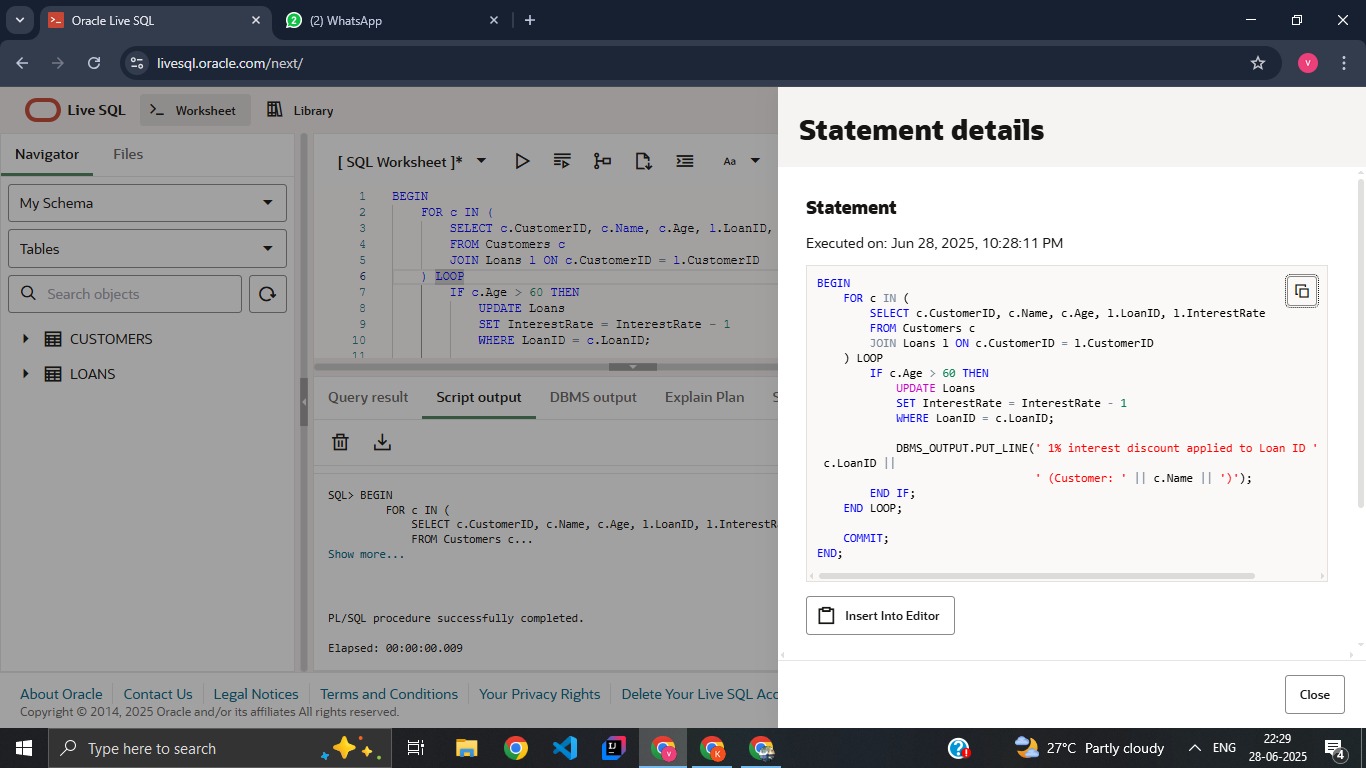
WHERE l.DueDate BETWEEN SYSDATE AND SYSDATE + 30

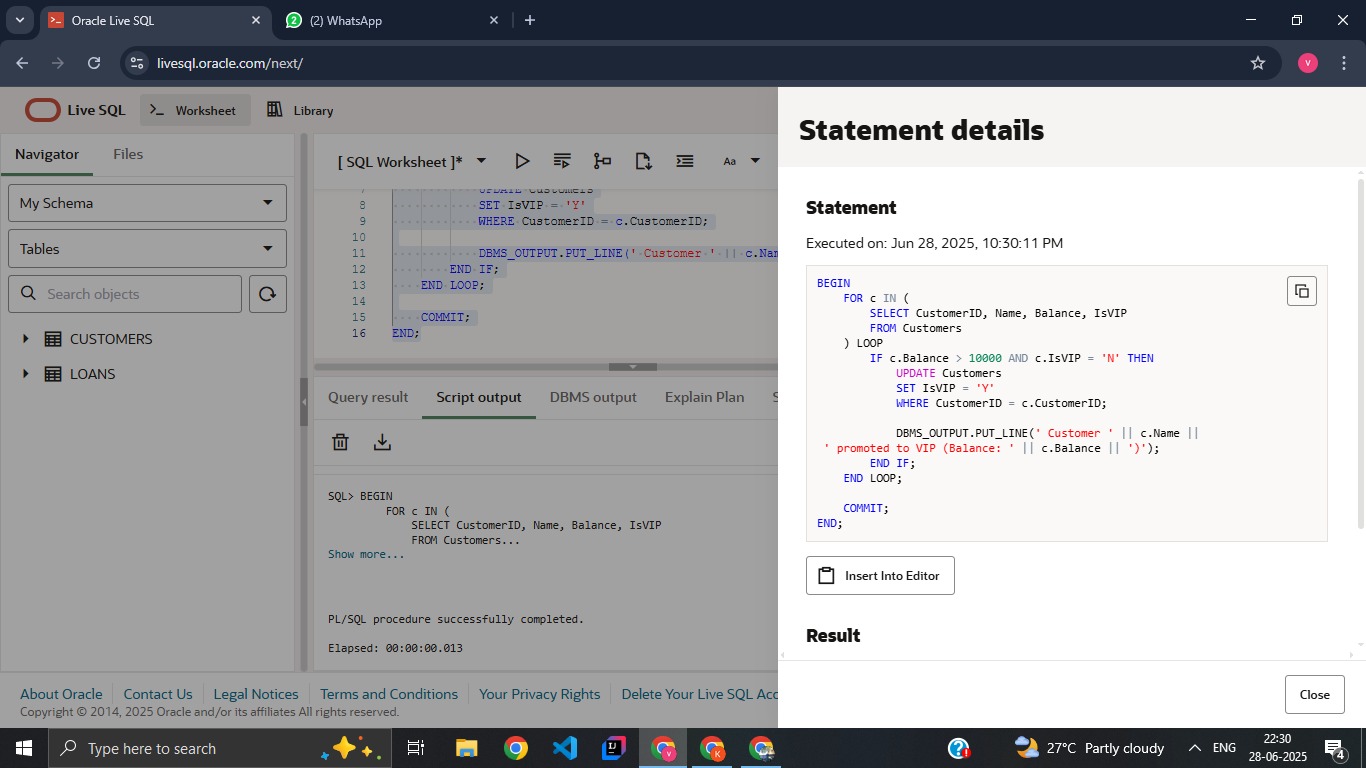
) LOOP

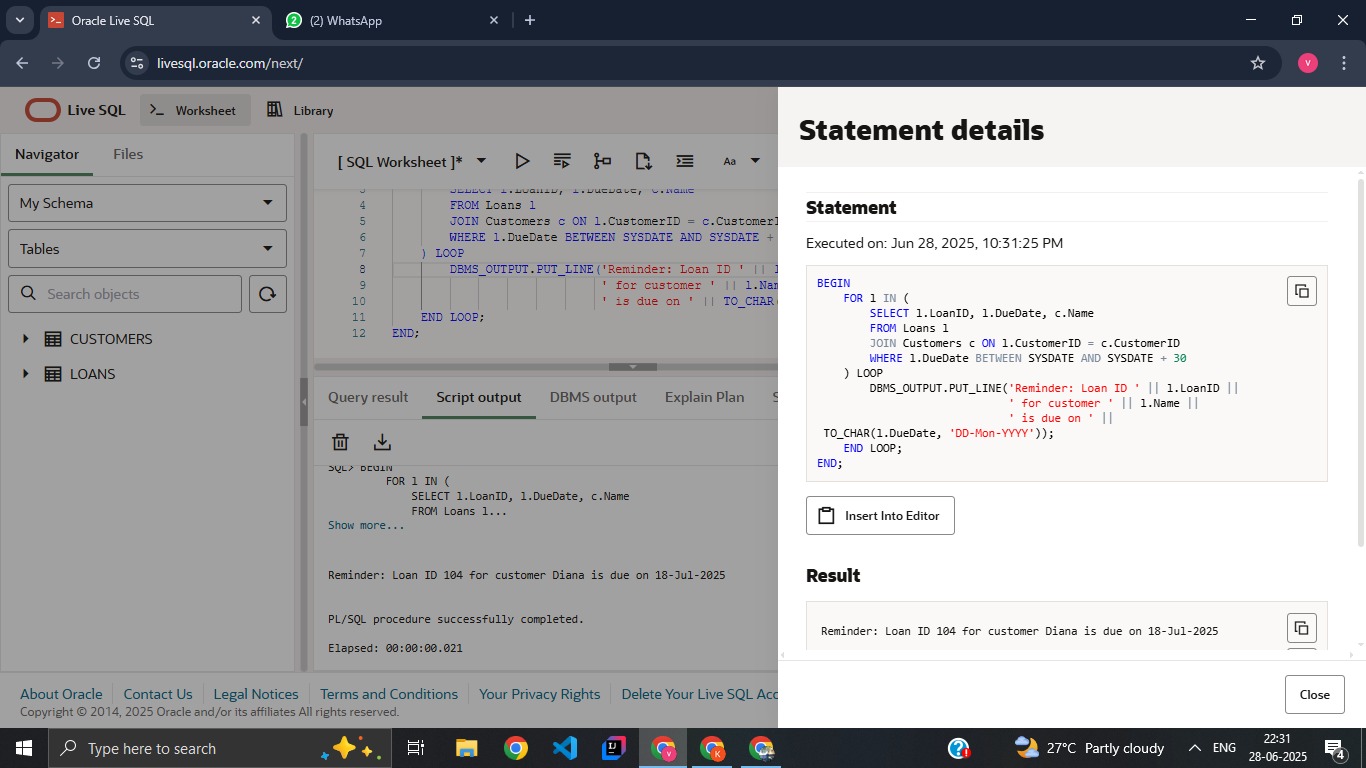
END LOOP;

END;

**Output:**

****

****

****

**Conclusion:**

This PL/SQL block successfully automates the application of a 1% interest rate discount for customers above 60 years of age. It enhances efficiency by eliminating the need for manual updates and ensures that only eligible customers are affected. Logging with DBMS\_OUTPUT also improves traceability, making this a simple yet effective real-world database solution.

**EXERCISE 3: STORED STRUCTURES**

**Introduction:**

This PL/SQL program defines procedures to automate key banking operations such as applying monthly interest to savings accounts, updating employee bonuses by department, and securely transferring funds between accounts.

**Objective:**

* **Apply Monthly Interest:** Calculate and update a 1% interest on all savings account balances.
* **Update Employee Bonuses:** Increase salaries of employees in a specified department based on a given bonus percentage.
* **Transfer Funds Securely:** Ensure safe transfer of funds between two accounts with sufficient balance checks.

**Implementation Breakdown:**

**Scenario 1:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

FOR acc IN (

SELECT AccountID, Balance

FROM Accounts

WHERE AccountType = 'Savings'

)

LOOP

UPDATE Accounts

SET Balance = acc.Balance \* 1.01

WHERE AccountID = acc.AccountID;

DBMS\_OUTPUT.PUT\_LINE('Interest applied to Account ' || acc.AccountID ||

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

END LOOP;

END;

**Scenario 2:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_department IN VARCHAR2,

p\_bonus\_pct IN NUMBER

) AS

BEGIN

FOR emp IN (

SELECT EmpID, Name, Salary FROM Employees WHERE Department = p\_department

)

LOOP

UPDATE Employees

SET Salary = Salary + (Salary \* p\_bonus\_pct / 100)

WHERE EmpID = emp.EmpID;

END LOOP;

END;

**Scenario 3:**

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_from\_acc IN NUMBER,

p\_to\_acc IN NUMBER,

p\_amount IN NUMBER

) AS

v\_balance NUMBER;

BEGIN

-- Get balance of source account

SELECT Balance INTO v\_balance FROM Accounts WHERE AccountID = p\_from\_acc FOR UPDATE;

IF v\_balance >= p\_amount THEN

-- Deduct from source

UPDATE Accounts

SET Balance = Balance - p\_amount

WHERE AccountID = p\_from\_acc;

-- Add to destination

UPDATE Accounts

SET Balance = Balance + p\_amount

WHERE AccountID = p\_to\_acc;

DBMS\_OUTPUT.PUT\_LINE('Transfer of ₹' || p\_amount ||

' from Account ' || p\_from\_acc ||

' to Account ' || p\_to\_acc || ' successful.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Transfer failed. Insufficient balance in Account ' || p\_from\_acc);

END IF;

END;

**Input:**

SET SERVEROUTPUT ON;

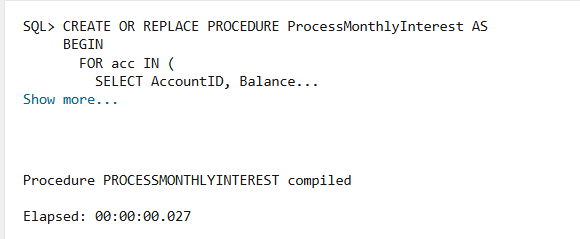
EXEC ProcessMonthlyInterest;

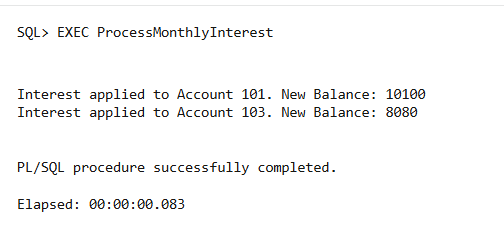
EXEC UpdateEmployeeBonus('IT', 10);

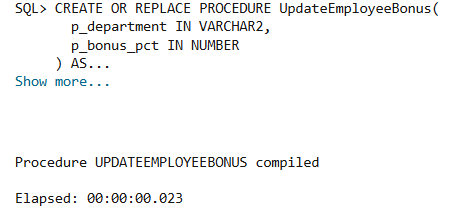
EXEC TransferFunds(101, 102, 2000);

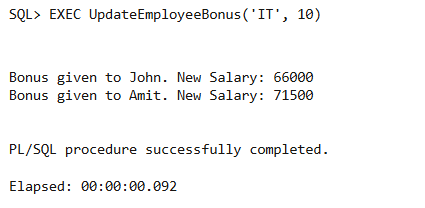
EXEC TransferFunds(104, 103, 5000);

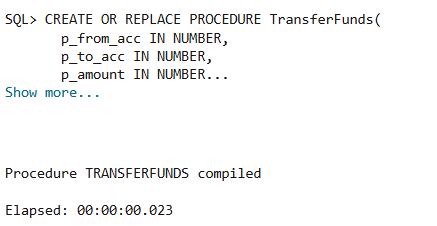
**Output:**

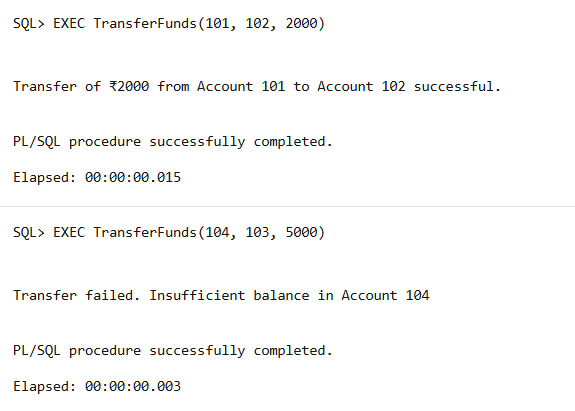
****

****

****

****

****

****

**Conclusion:**

The procedures execute essential financial tasks efficiently, ensuring accurate balance updates, fair employee compensation, and secure fund transfers, thereby enhancing overall system reliability and user satisfaction.