**Student Name:** HARINEE N J

**Registration No:** 22CSR063

**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:**

SET SERVEROUTPUT ON;

BEGIN

FOR cust IN (SELECT CustomerID FROM Customers WHERE Age > 60) LOOP

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = cust.CustomerID;

DBMS\_OUTPUT.PUT\_LINE('Applied 1% discount for Customer ID: ' || cust.CustomerID);

END LOOP;

COMMIT;

END;

/

**Scenario 2:**

SET SERVEROUTPUT ON;

BEGIN

FOR cust IN (SELECT CustomerID FROM Customers WHERE Balance > 10000) LOOP

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = cust.CustomerID;

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

END LOOP;

COMMIT;

END;

/

**Scenario 3:**

SET SERVEROUTPUT ON;

DECLARE

v\_name VARCHAR2(100);

BEGIN

FOR loan IN (

SELECT l.LoanID, l.CustomerID, 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.Name ||

' has a loan due on ' || TO\_CHAR(loan.DueDate, 'DD-MON-YYYY'));

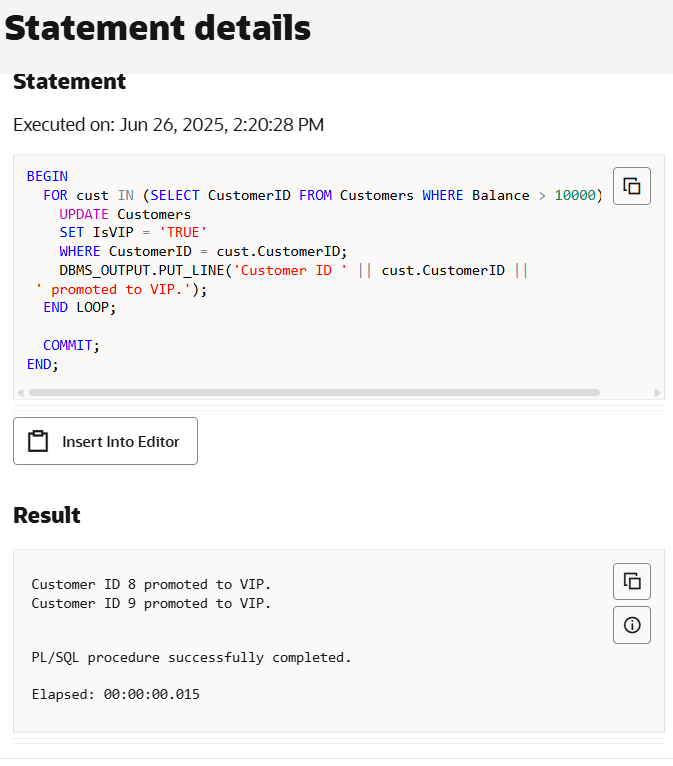
END LOOP;

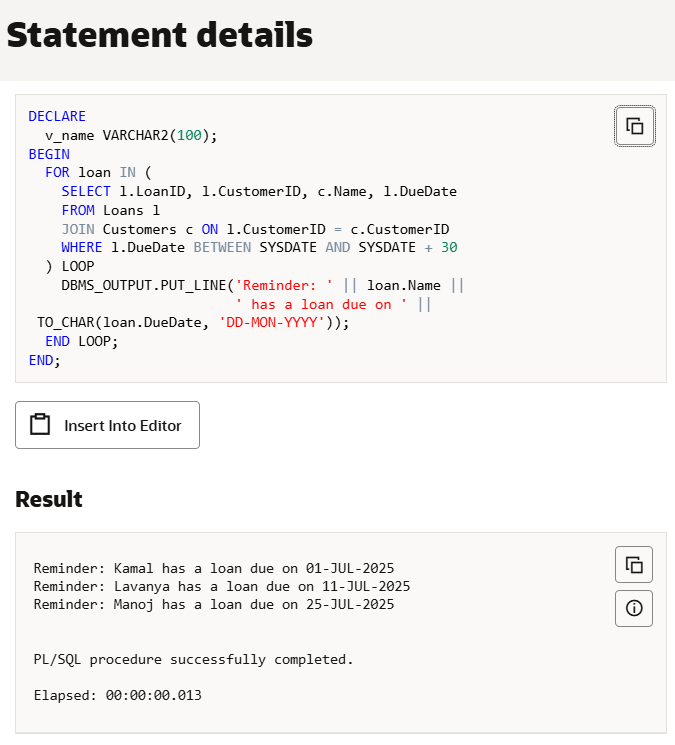
END;

/

**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 IS

v\_old\_balance NUMBER;

v\_new\_balance NUMBER;

BEGIN

FOR acc IN (SELECT AccountID, Balance FROM Accounts WHERE AccountType = 'Savings') LOOP

v\_old\_balance := acc.Balance;

v\_new\_balance := acc.Balance + (acc.Balance \* 0.01);

UPDATE Accounts

SET Balance = v\_new\_balance

WHERE AccountID = acc.AccountID;

END LOOP;

COMMIT;

END;

/

**Scenario 2:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_dept IN VARCHAR2,

p\_bonus\_percent IN NUMBER

) IS

CURSOR emp\_cur IS

SELECT EmployeeID, Salary FROM Employees WHERE Department = p\_dept;

v\_new\_salary NUMBER;

BEGIN

FOR emp IN emp\_cur LOOP

v\_new\_salary := emp.Salary + (emp.Salary \* p\_bonus\_percent / 100);

UPDATE Employees

SET Salary = v\_new\_salary

WHERE EmployeeID = emp.EmployeeID

END LOOP;

COMMIT;

END;

/

**Scenario 3:**

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_from\_acc IN NUMBER,

p\_to\_acc IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

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

IF v\_balance < p\_amount THEN

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

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

ELSE

UPDATE Accounts

SET Balance = Balance - p\_amount

WHERE AccountID = p\_from\_acc;

UPDATE Accounts

SET Balance = Balance + p\_amount

WHERE AccountID = p\_to\_acc;

END IF;

COMMIT;

END;

/

**Input:**

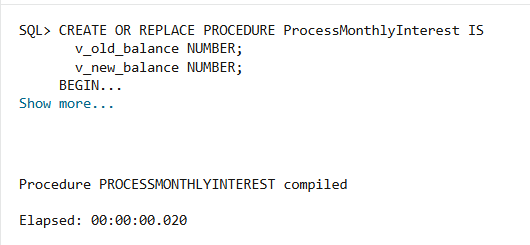
SET SERVEROUTPUT ON;

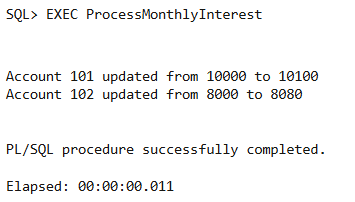
EXEC ProcessMonthlyInterest;

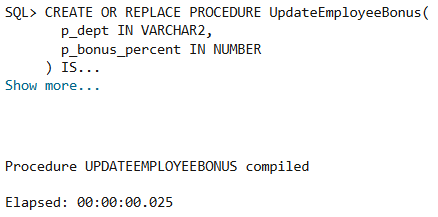
EXEC UpdateEmployeeBonus('IT', 10);

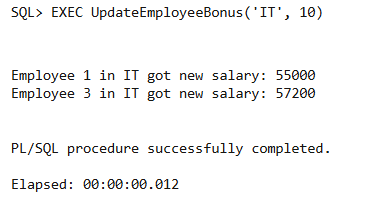
EXEC TransferFunds(101, 102, 2000);

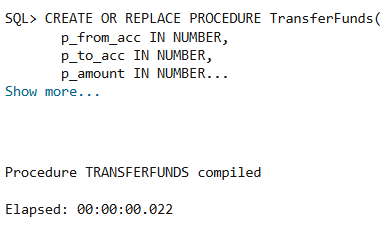
**Output:**

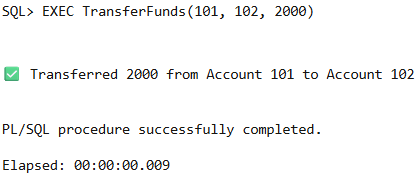
****

****

****

****

****

****

**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.