**WEEK-2: PL/SQL**

## **Exercise-1 Control Structures**

**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

Solution:

DECLARE

v\_customer\_id Customers.CustomerID%TYPE;

v\_dob Customers.DOB%TYPE;

v\_age NUMBER;

BEGIN

FOR customer\_rec IN (SELECT CustomerID, DOB FROM Customers) LOOP

v\_age := TRUNC(MONTHS\_BETWEEN(SYSDATE, customer\_rec.DOB) / 12);

IF v\_age > 60 THEN

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = customer\_rec.CustomerID;

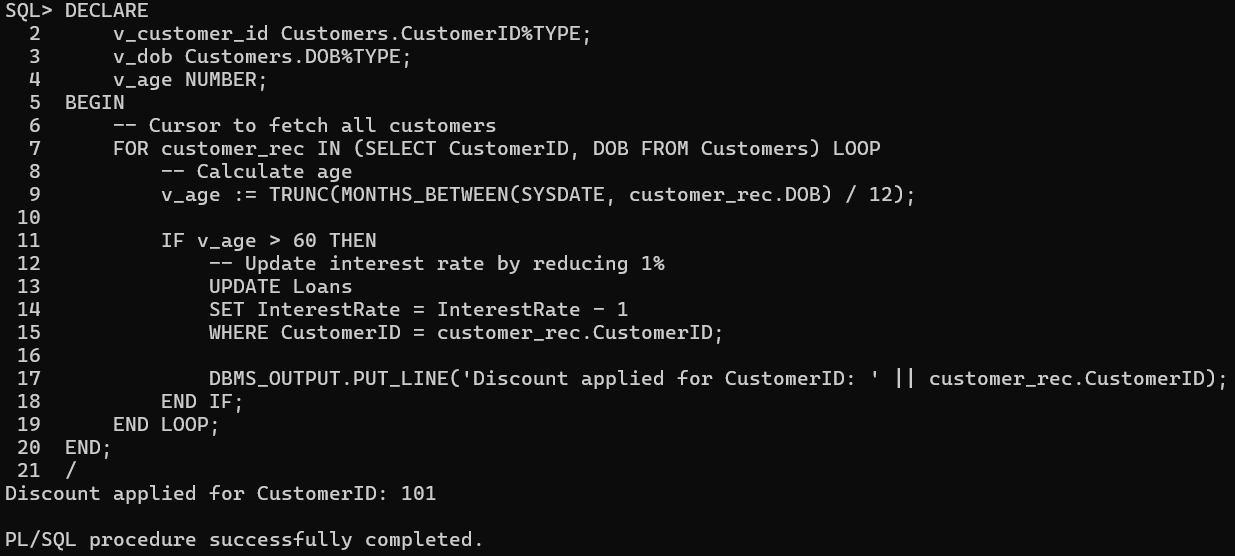
DBMS\_OUTPUT.PUT\_LINE('Discount applied for CustomerID: ' || customer\_rec.CustomerID);

END IF;

END LOOP;

END;

/

**Output:**

**Question 2:** 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.

**Solution:**

//First we have to add a column isvip to customers table//

ALTER TABLE Customers ADD IsVIP CHAR(1) DEFAULT 'N' CHECK (IsVIP IN ('TRUE', 'FALSE'));

**Actual Code:**

BEGIN

FOR customer\_rec IN (SELECT CustomerID, Balance FROM Customers) LOOP

IF customer\_rec.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'True'

WHERE CustomerID = customer\_rec.CustomerID;

ELSE

UPDATE Customers

SET IsVIP = 'False'

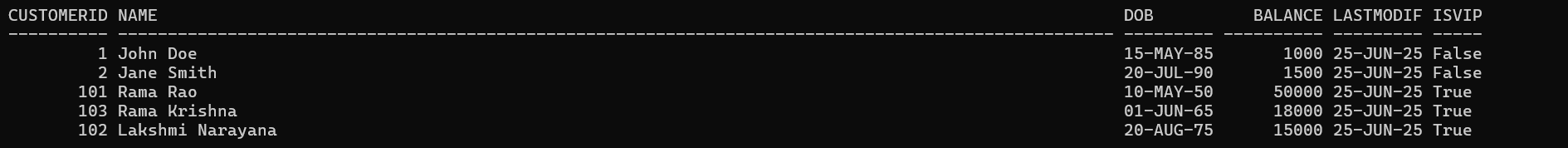
WHERE CustomerID = customer\_rec.CustomerID;

END IF;

END LOOP;

END;

/

**Output**:



**Question 3:** Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

**Solution:**

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')

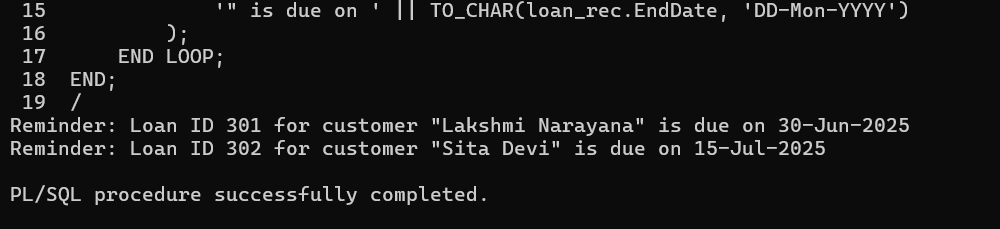
);

END LOOP;

END;

/

**Output:**



## **Exercise-3 Stored Procedures**

**Question-1:** 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.

**Solution:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountType = 'Savings';

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Monthly interest applied to all savings accounts.');

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END;

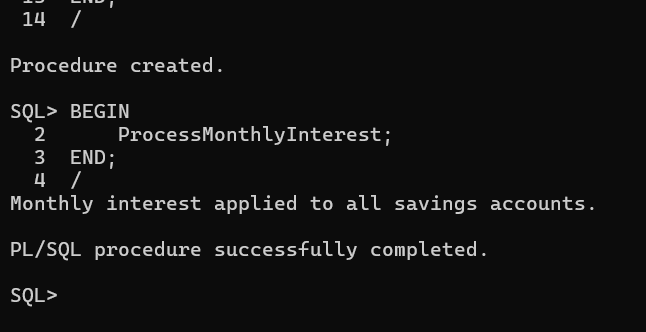
/

BEGIN

ProcessMonthlyInterest;

END;

/

**Output:**

**Question-2:** Write a stored procedure **UpdateEmployeeBonus** that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

**Solution:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_department IN VARCHAR2,

p\_bonus\_percent IN NUMBER

) IS

BEGIN

UPDATE Employees

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

WHERE Department = p\_department;

IF SQL%ROWCOUNT = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('No employees found in department ' || p\_department);

ELSE

DBMS\_OUTPUT.PUT\_LINE(SQL%ROWCOUNT || ' employee(s) received a bonus.');

END IF;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END;

/

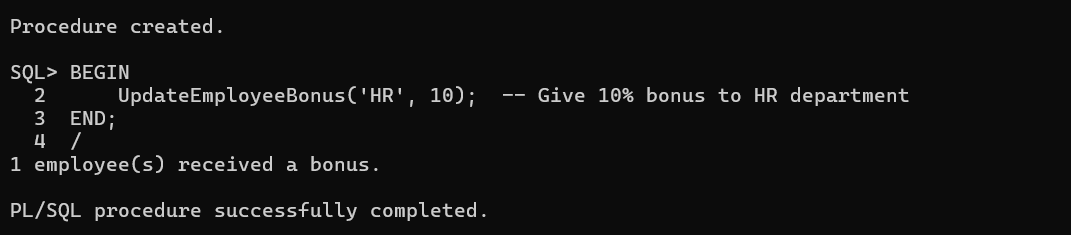
BEGIN

UpdateEmployeeBonus('HR', 10);

END;

/

**Output:**



**Question-3:** 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.

**Solution:**

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_account\_id IN NUMBER,

p\_to\_account\_id IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance

FROM Accounts

WHERE AccountID = p\_from\_account\_id;

IF v\_balance < p\_amount THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient balance in source account.');

END IF;

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;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Funds transferred successfully.');

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: ' || SQLERRM);

END;

/

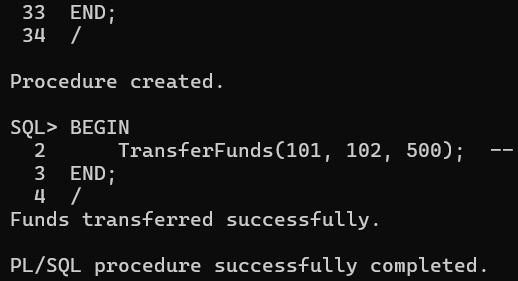
BEGIN

TransferFunds(101, 102, 500);

END;

/

**Output:**

****