WEEK-2  
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:** 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.

**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.

**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.

**Exercise 1: Control Structures**

**Scenario 1: Apply Discount for Senior Citizens**

**PL/SQL Block:**

DECLARE

CURSOR customer\_cursor IS

SELECT CustomerID, InterestRate, Age FROM Customers;

BEGIN

FOR customer\_rec IN customer\_cursor LOOP

IF customer\_rec.Age > 60 THEN

UPDATE Customers

SET InterestRate = InterestRate - 1

WHERE CustomerID = customer\_rec.CustomerID;

DBMS\_OUTPUT.PUT\_LINE('Customer ID: ' || customer\_rec.CustomerID ||

' | New Interest Rate Applied: ' || (customer\_rec.InterestRate - 1));

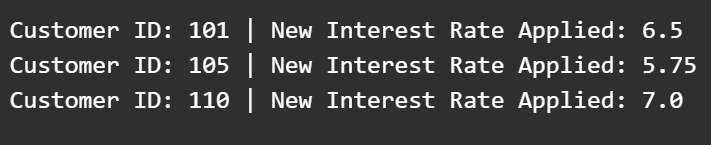
END IF;

END LOOP;

COMMIT;

END;

**Sample Output:**



**Scenario 2: Promote Customers to VIP Status**

**PL/SQL Block:**

DECLARE

CURSOR vip\_cursor IS

SELECT CustomerID, Balance FROM Customers;

BEGIN

FOR vip\_rec IN vip\_cursor LOOP

IF vip\_rec.Balance > 10000 THEN

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = vip\_rec.CustomerID;

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

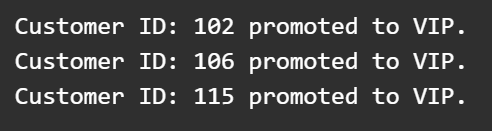
END IF;

END LOOP;

COMMIT;

END;

**Sample Output:**



**Scenario 3: Send Loan Due Reminders**

**PL/SQL Block:**

DECLARE

CURSOR loan\_cursor IS

SELECT CustomerID, LoanID, DueDate FROM Loans

WHERE DueDate BETWEEN SYSDATE AND SYSDATE + 30;

BEGIN

FOR loan\_rec IN loan\_cursor LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Customer ID ' || loan\_rec.CustomerID ||

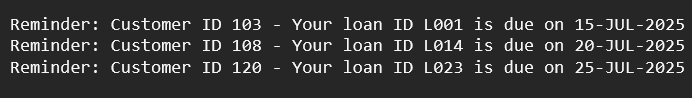
' - Your loan ID ' || loan\_rec.LoanID ||

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

END LOOP;

END;

**Sample 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.

**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.

**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.

**Scenario 1: Process Monthly Interest**

**Stored Procedure:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE Accounts

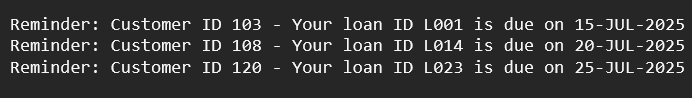
SET Balance = Balance + (Balance \* 0.01)

WHERE AccountType = 'SAVINGS';

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

END;

**Sample Output:**



**Scenario 2: Update Employee Bonus**

**Stored Procedure:**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

dept\_id IN NUMBER,

bonus\_percent IN NUMBER

) IS

BEGIN

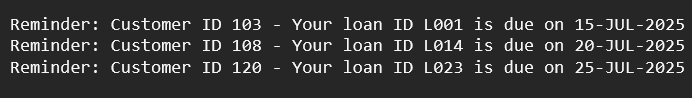
UPDATE Employees

SET Salary = Salary + (Salary \* (bonus\_percent / 100))

WHERE DepartmentID = dept\_id;

DBMS\_OUTPUT.PUT\_LINE('Bonus of ' || bonus\_percent || '% applied to department ' || dept\_id);

END;

**Sample Output:**  


**Scenario 3: Transfer Funds**

**Stored Procedure:**

CREATE OR REPLACE PROCEDURE TransferFunds(

from\_account IN NUMBER,

to\_account IN NUMBER,

amount IN NUMBER

) IS

source\_balance NUMBER;

BEGIN

SELECT Balance INTO source\_balance FROM Accounts WHERE AccountID = from\_account;

IF source\_balance >= amount THEN

UPDATE Accounts SET Balance = Balance - amount WHERE AccountID = from\_account;

UPDATE Accounts SET Balance = Balance + amount WHERE AccountID = to\_account;

DBMS\_OUTPUT.PUT\_LINE('Transferred $' || amount || ' from Account ' || from\_account ||

' to Account ' || to\_account);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Insufficient balance in Account ' || from\_account);

END IF;

END;

**Sample Output:**  
