**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.

**//Apply Discount for Senior Citizens**

BEGIN

FOR cust IN (SELECT customer\_id, age, interest\_rate FROM customers) LOOP

IF cust.age > 60 THEN

UPDATE customers

SET interest\_rate = interest\_rate - 1

WHERE customer\_id = cust.customer\_id;

END IF;

END LOOP;

COMMIT;

END;

**//Promote VIP Customers**

BEGIN

FOR cust IN (SELECT customer\_id, balance FROM customers) LOOP

IF cust.balance > 10000 THEN

UPDATE customers

SET isVIP = 'TRUE'

WHERE customer\_id = cust.customer\_id;

END IF;

END LOOP;

COMMIT;

END;

**//Reminders for Loan Dues**

BEGIN

FOR loan IN (

SELECT customer\_id, loan\_id, due\_date

FROM loans

WHERE due\_date BETWEEN SYSDATE AND SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || loan.loan\_id ||

' for Customer ' || loan.customer\_id ||

' is due on ' || loan.due\_date);

END LOOP;

END;

**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.

**//Process Monthly Interest**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

FOR acc IN (SELECT account\_id, balance FROM savings\_accounts) LOOP

UPDATE savings\_accounts

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

WHERE account\_id = acc.account\_id;

END LOOP;

COMMIT;

END;

**//Employee Bonus by Department**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept\_id IN NUMBER,

bonus\_percent IN NUMBER

) AS

BEGIN

UPDATE employees

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

WHERE department\_id = dept\_id;

COMMIT;

END;

**//Transfer Funds**

CREATE OR REPLACE PROCEDURE TransferFunds (

source\_acc\_id IN NUMBER,

dest\_acc\_id IN NUMBER,

amount IN NUMBER

) AS

src\_balance NUMBER;

BEGIN

SELECT balance INTO src\_balance FROM accounts WHERE account\_id = source\_acc\_id FOR UPDATE;

IF src\_balance < amount THEN

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

ELSE

UPDATE accounts

SET balance = balance - amount

WHERE account\_id = source\_acc\_id;

UPDATE accounts

SET balance = balance + amount

WHERE account\_id = dest\_acc\_id;

COMMIT;

END IF;

END;