**PLSQL\_EXERCISES**

**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.

**Scenario 1: Apply 1% Discount to Loan Interest Rates for Customers Above 60**

DECLARE

CURSOR customer\_cursor IS

SELECT customer\_id, age, loan\_interest\_rate

FROM customers;

v\_customer\_id customers.customer\_id%TYPE;

v\_age customers.age%TYPE;

v\_loan\_interest\_rate customers.loan\_interest\_rate%TYPE;

BEGIN

FOR customer\_record IN customer\_cursor LOOP

v\_customer\_id := customer\_record.customer\_id;

v\_age := customer\_record.age;

v\_loan\_interest\_rate := customer\_record.loan\_interest\_rate;

IF v\_age > 60 THEN

UPDATE customers

SET loan\_interest\_rate = loan\_interest\_rate - 1

WHERE customer\_id = v\_customer\_id;

END IF;

END LOOP;

COMMIT;

END;

**Scenario 2: Promote Customers to VIP Based on Balance Over $10,000**

DECLARE

CURSOR customer\_cursor IS

SELECT customer\_id, balance

FROM customers;

v\_customer\_id customers.customer\_id%TYPE;

v\_balance customers.balance%TYPE;

BEGIN

FOR customer\_record IN customer\_cursor LOOP

v\_customer\_id := customer\_record.customer\_id;

v\_balance := customer\_record.balance;

IF v\_balance > 10000 THEN

UPDATE customers

SET IsVIP = 'TRUE'

WHERE customer\_id = v\_customer\_id;

END IF;

END LOOP;

COMMIT;

END;

**Scenario 3: Send Reminders for Loans Due in the Next 30 Days**

DECLARE

CURSOR loan\_cursor IS

SELECT customer\_id, due\_date

FROM loans

WHERE due\_date BETWEEN SYSDATE AND SYSDATE + 30;

v\_customer\_id loans.customer\_id%TYPE;

v\_due\_date loans.due\_date%TYPE;

BEGIN

FOR loan\_record IN loan\_cursor LOOP

v\_customer\_id := loan\_record.customer\_id;

v\_due\_date := loan\_record.due\_date;

DBMS\_OUTPUT.PUT\_LINE('Reminder: Customer ID ' || v\_customer\_id ||

' has a loan due on ' || TO\_CHAR(v\_due\_date, 'DD-MON-YYYY'));

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.

**Scenario 1: Process Monthly Interest for All Savings Accounts**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE accounts

SET balance = balance + (balance \* 0.01)

WHERE account\_type = 'SAVINGS';

COMMIT;

END;

**Scenario 2: Update Employee Bonus by Department**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_department\_id IN employees.department\_id%TYPE,

p\_bonus\_percent IN NUMBER

) IS

BEGIN

UPDATE employees

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

WHERE department\_id = p\_department\_id;

COMMIT;

END;

**Scenario 3: Transfer Funds Between Accounts**

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_source\_account IN accounts.account\_id%TYPE,

p\_target\_account IN accounts.account\_id%TYPE,

p\_amount IN NUMBER

) IS

v\_source\_balance accounts.balance%TYPE;

BEGIN

-- Check if source account has sufficient balance

SELECT balance INTO v\_source\_balance

FROM accounts

WHERE account\_id = p\_source\_account

FOR UPDATE;

IF v\_source\_balance < p\_amount THEN

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

ELSE

-- Deduct from source account

UPDATE accounts

SET balance = balance - p\_amount

WHERE account\_id = p\_source\_account;

-- Add to target account

UPDATE accounts

SET balance = balance + p\_amount

WHERE account\_id = p\_target\_account;

COMMIT;

END IF;

END;