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.

CODE

BEGIN

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

UPDATE Customers

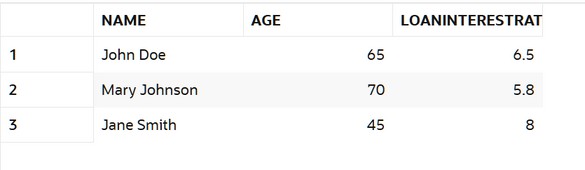
SET LoanInterestRate = LoanInterestRate - 1 WHERE CustomerID = cust.CustomerID;

END LOOP;

COMMIT;

END;

SELECT Name, Age, LoanInterestRate FROM Customers; output



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.

BEGIN

FOR cust IN (SELECT \* FROM Customers WHERE Balance > 10000) LOOP UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = cust.CustomerID; END LOOP;

COMMIT;

END;

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.

BEGIN

FOR loan IN (

SELECT L.LoanID, L.DueDate, C.Name FROM Loans L

JOIN Customers C ON L.CustomerID = C.CustomerID WHERE L.DueDate <= SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ' || loan.LoanID || ' for customer ' || loan.Name || ' is due on ' || TO\_CHAR(loan.DueDate, '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.

CREATE TABLE SavingsAccounts ( AccountID INT PRIMARY KEY,

CustomerID INT, Balance DECIMAL(10,2)

);

INSERT INTO SavingsAccounts (AccountID, CustomerID, Balance) VALUES (101, 1001, 5000.00),

(102, 1002, 12000.00),

(103, 1003, 7500.00);

CREATE PROCEDURE ProcessMonthlyInterest() BEGIN

UPDATE SavingsAccounts

SET Balance = Balance + (Balance \* 0.01); DELIMITER ;

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.

CREATE TABLE Employees ( EmployeeID INT PRIMARY KEY, Name VARCHAR(100),

DepartmentID INT, Salary DECIMAL(10,2)

);

INSERT INTO Employees (EmployeeID, Name, DepartmentID, Salary) VALUES (1, 'Alice', 10, 40000.00),

(2, 'Bob', 10, 42000.00),

(3, 'Charlie', 20, 38000.00),

(4, 'David', 20, 45000.00);

CREATE PROCEDURE UpdateEmployeeBonus( IN dept\_id INT,

IN bonus\_percent DECIMAL(5,2)

)

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* (bonus\_percent / 100)) WHERE DepartmentID = dept\_id;

DELIMITER ;

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.

CREATE TABLE BankAccounts ( AccountID INT PRIMARY KEY,

CustomerID INT, Balance DECIMAL(10,2)

);

INSERT INTO BankAccounts (AccountID, CustomerID, Balance) VALUES (101, 1001, 10000.00),

(102, 1002, 15000.00),

(201, 1004, 8000.00),

(202, 1005, 3000.00);

CREATE PROCEDURE TransferFunds( IN from\_account INT,

IN to\_account INT,

IN amount DECIMAL(10,2)

)

BEGIN

DECLARE from\_balance DECIMAL(10,2); SELECT Balance INTO from\_balance FROM BankAccounts

WHERE AccountID = from\_account;

IF from\_balance >= amount THEN UPDATE BankAccounts

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

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

ELSE

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Insufficient funds in source account.'; END IF;

DELIMITER ;