**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.

DECLARE

    CURSOR seniors IS

        SELECT c.CustomerID, l.LoanID, l.InterestRate

        FROM Customers c

        JOIN Loans l ON c.CustomerID = l.CustomerID

        WHERE MONTHS\_BETWEEN(SYSDATE, c.DOB) >720;

BEGIN

    FOR rec IN seniors LOOP

        UPDATE Loans

        SET InterestRate = InterestRate - 1

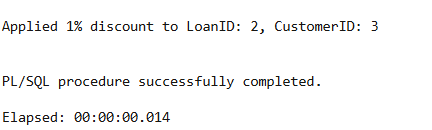
        WHERE LoanID = rec.LoanID;

        DBMS\_OUTPUT.PUT\_LINE('Applied 1% discount to LoanID: ' || rec.LoanID ||

                             ', CustomerID: ' || rec.CustomerID);

    END LOOP;

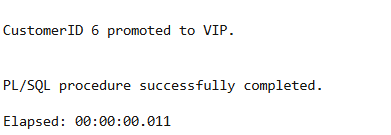
END;



**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.

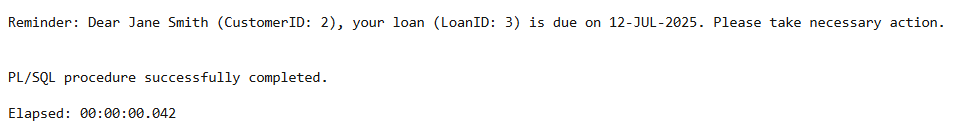
1. -- ALTER TABLE Customers ADD IsVIP VARCHAR2(5);
2. -- UPDATE customers SET isVip = 'False';
3. -- INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified, IsVip)
4. -- VALUES (6, 'John Doe', TO\_DATE('1990-07-20', 'YYYY-MM-DD'), 15000, SYSDATE, 'False');
5. BEGIN
6. FOR Rec in (SELECT CustomerId, Balance From Customers) LOOP
7. IF Rec.Balance > 10000 THEN
8. UPDATE Customers
9. SET IsVIP = 'TRUE'
10. WHERE Rec.CustomerId = CustomerId;
11. DBMS\_OUTPUT.PUT\_LINE('CustomerID ' || Rec.CustomerId || ' promoted to VIP.');
12. END IF;
13. END LOOP;
15. 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.

1. -- INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)
2. -- VALUES (3, 2, 3000, 4.5, SYSDATE, SYSDATE + 15);
3. BEGIN
4. FOR rec IN (
5. SELECT c.Name, c.CustomerID, l.LoanID, l.EndDate
6. FROM Loans l
7. JOIN Customers c ON l.CustomerID = c.CustomerID
8. WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30
9. ) LOOP
10. DBMS\_OUTPUT.PUT\_LINE(
11. 'Reminder: Dear ' || rec.Name ||
12. ' (CustomerID: ' || rec.CustomerID || '), your loan (LoanID: ' || rec.LoanID ||
13. ') is due on ' || TO\_CHAR(rec.EndDate, 'DD-MON-YYYY') || '. Please take necessary action.'
14. );
15. END LOOP;
16. END;

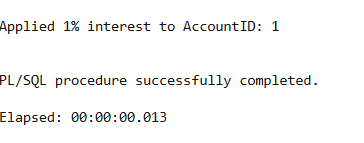


**Exercise 3: Stored Procedures**

**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.

1. CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS
2. BEGIN
3. FOR REC IN (SELECT AccountID, Balance
4. FROM Accounts
5. WHERE AccountType = 'Savings') LOOP
6. UPDATE ACCOUNTS
7. SET BALANCE = BALANCE \* 1.01, LastModified = SYSDATE
8. WHERE ACCOUNTID = REC.AccountId;
9. DBMS\_OUTPUT.PUT\_LINE('Applied 1% interest to AccountID: ' || rec.AccountID);
10. END LOOP;
11. END;
12. BEGIN
13. ProcessMonthlyInterest;
14. END;

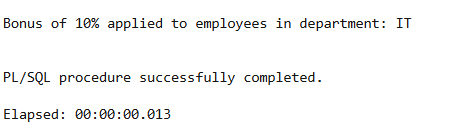


**Scenario 2:** The bank wants to implement a bonus scheme for employees based on their performance.

* + **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 OR REPLACE PROCEDURE UpdateEmployeeBonus(

1. pDepartment in VARCHAR2,
2. pBonusPercent IN NUMBER
3. ) IS
4. BEGIN
5. UPDATE Employees
6. SET Salary = Salary + (Salary \* pBonusPercent / 100)
7. WHERE Department = pDepartment;
8. DBMS\_OUTPUT.PUT\_LINE('Bonus of ' || pBonusPercent || '% applied to employees in department: ' || pDepartment);
9. COMMIT;
10. END;
11. BEGIN
12. UpdateEmployeeBonus('IT', 10);
13. END;



**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.

1. CREATE OR REPLACE PROCEDURE TransferFunds (
2. p\_source\_account\_id   IN NUMBER,
3. p\_target\_account\_id   IN NUMBER,
4. p\_amount              IN NUMBER
5. ) IS
6. v\_source\_balance  NUMBER;
7. BEGIN
8. SELECT Balance INTO v\_source\_balance
9. FROM Accounts
10. WHERE AccountID = p\_source\_account\_id;
11. IF v\_source\_balance < p\_amount THEN
12. RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient funds in the source account.');
13. END IF;
14. UPDATE Accounts
15. SET Balance = Balance - p\_amount,
16. LastModified = SYSDATE
17. WHERE AccountID = p\_source\_account\_id;
18. UPDATE Accounts
19. SET Balance = Balance + p\_amount,
20. LastModified = SYSDATE
21. WHERE AccountID = p\_target\_account\_id;
22. DBMS\_OUTPUT.PUT\_LINE('Transferred $' || p\_amount || ' from Account ' || p\_source\_account\_id ||
23. ' to Account ' || p\_target\_account\_id);
24. END;
25. BEGIN
26. TransferFunds(1, 2, 500);  -- Transfers $500 from AccountID 1 to 2
27. END;

