PL/SQL Solutions - Control Structures and Stored Procedures

Exercise 1: Control Structures

Scenario 1: Discount for Senior Customers (Age > 60)

sql

```
DECLARE
    CURSOR customer_cursor IS
    SELECT c.CustomerID, c.DOB, l.LoanID, l.InterestRate
    FROM Customers c
    JOIN Loans l ON c.CustomerID = l.CustomerID;

    v_age NUMBER;
v_new_rate NUMBER;
BEGIN
    FOR customer_rec IN customer_cursor LOOP        v_age :=
FLOOR(MONTHS_BETWEEN(SYSDATE, customer_rec.DOB) / 12);

        IF v_age > 60 THEN
            v_new_rate := customer_rec.InterestRate - 1;
            UPDATE Loans
            SET InterestRate = v_new_rate
            WHERE LoanID = customer_rec.LoanID;

            DBMS_OUTPUT.PUT_LINE('Applied discount to Customer ID: ' || customer_rec.CustomerID);
        END IF;
    END LOOP;

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Senior citizen discount processing completed.');
END;
/
```

Output:

```
Applied discount to Customer ID: 1
Applied discount to Customer ID: 3
Applied discount to Customer ID: 5
Senior citizen discount processing completed.
```

Scenario 2: VIP Status Based on Balance

Sql

```sql
ALTER TABLE Customers ADD (IsVIP CHAR(1) DEFAULT 'N');

DECLARE
   CURSOR customer_cursor IS
   SELECT CustomerID, Balance
   FROM Customers;
BEGIN
   FOR customer_rec IN customer_cursor LOOP
      IF customer_rec.Balance > 10000 THEN
         UPDATE Customers
         SET IsVIP = 'Y'
         WHERE CustomerID = customer_rec.CustomerID;

         DBMS_OUTPUT.PUT_LINE('Customer ID ' || customer_rec.CustomerID || ' promoted to VIP');
      ELSE
         UPDATE Customers
         SET IsVIP = 'N'
         WHERE CustomerID = customer_rec.CustomerID;
      END IF;
   END LOOP;

   COMMIT;
   DBMS_OUTPUT.PUT_LINE('VIP status update completed.');
END;
/
```

Output:

```
Customer ID 1 promoted to VIP
Customer ID 3 promoted to VIP
Customer ID 5 promoted to VIP
VIP status update completed.
```

Scenario 3: Loan Due Reminders (Next 30 Days)

Sql

```sql
DECLARE
   CURSOR loan_due_cursor IS
   SELECT l.LoanID, l.CustomerID, c.Name, l.EndDate, l.LoanAmount
   FROM Loans l
   JOIN Customers c ON l.CustomerID = c.CustomerID
   WHERE l.EndDate BETWEEN SYSDATE AND SYSDATE + 30;
```

```
    v_days_remaining NUMBER;
BEGIN
    DBMS_OUTPUT.PUT_LINE('=== LOAN DUE REMINDERS ===');
    DBMS_OUTPUT.PUT_LINE('Generated on: ' || TO_CHAR(SYSDATE, 'DD-MON-YYYY'));

    FOR loan_rec IN loan_due_cursor LOOP
        v_days_remaining := FLOOR(loan_rec.EndDate - SYSDATE);

        DBMS_OUTPUT.PUT_LINE('REMINDER: Dear ' || loan_rec.Name);
        DBMS_OUTPUT.PUT_LINE('Your loan (ID: ' || loan_rec.LoanID || ') of $' || loan_rec.LoanAmount);
        DBMS_OUTPUT.PUT_LINE('Due Date: ' || TO_CHAR(loan_rec.EndDate, 'DD-MON-YYYY'));
        DBMS_OUTPUT.PUT_LINE('Please contact us to arrange payment.');
        DBMS_OUTPUT.PUT_LINE('------------------------');
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('Reminder processing completed.');
END;
/
```

Output:

```
=== LOAN DUE REMINDERS ===
Generated on: 28-JUN-2025
REMINDER: Dear John Smith
Your loan (ID: 101) of $5000
Due Date: 15-JUL-2025
Please contact us to arrange payment.
------------------------
REMINDER: Dear Jane Doe
Your loan (ID: 102) of $7500
Due Date: 25-JUL-2025
Please contact us to arrange payment.
------------------------
Reminder processing completed.
```

Exercise 3: Stored Procedures

Scenario 1: Process Monthly Interest for Savings Accounts

Sql

```sql
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS
    CURSOR savings_cursor IS
    SELECT AccountID, Balance
    FROM Accounts
    WHERE UPPER(AccountType) = 'SAVINGS';

    v_new_balance NUMBER;
v_interest_earned NUMBER;
v_total_interest NUMBER := 0;
v_accounts_processed NUMBER := 0;
BEGIN
    DBMS_OUTPUT.PUT_LINE('=== MONTHLY INTEREST PROCESSING ===');
    DBMS_OUTPUT.PUT_LINE('Processing Date: ' || TO_CHAR(SYSDATE, 'DD-MON-YYYY'));

    FOR account_rec IN savings_cursor LOOP
v_interest_earned := account_rec.Balance * 0.01;
v_new_balance := account_rec.Balance + v_interest_earned;

        UPDATE Accounts
        SET Balance = v_new_balance,
            LastModified = SYSDATE
        WHERE AccountID = account_rec.AccountID;

        v_total_interest := v_total_interest + v_interest_earned;
v_accounts_processed := v_accounts_processed + 1;

        DBMS_OUTPUT.PUT_LINE('Account ID ' || account_rec.AccountID ||
                            ': Interest $' || ROUND(v_interest_earned, 2));
    END LOOP;

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Accounts Processed: ' || v_accounts_processed);
    DBMS_OUTPUT.PUT_LINE('Total Interest Paid: $' || ROUND(v_total_interest, 2));
    DBMS_OUTPUT.PUT_LINE('Monthly interest processing completed successfully.');

EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Error during interest processing: ' || SQLERRM);
        RAISE;
END ProcessMonthlyInterest;
/
```

Output:

```
=== MONTHLY INTEREST PROCESSING ===
Processing Date: 28-JUN-2025
Account ID 1001: Interest $25.50
Account ID 1003: Interest $175.00
Account ID 1005: Interest $89.75
Accounts Processed: 3
Total Interest Paid: $290.25
Monthly interest processing completed successfully.
```

Scenario 2: Update Employee Bonus by Department

Sql

```sql
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus
(
  p_department IN VARCHAR2,    p_bonus_percentage
IN NUMBER
) AS
  CURSOR employee_cursor IS
  SELECT EmployeeID, Name, Salary
  FROM Employees
  WHERE UPPER(Department) = UPPER(p_department);

  v_old_salary NUMBER;    v_new_salary
NUMBER;    v_bonus_amount NUMBER;
v_employees_updated NUMBER := 0;
v_total_bonus NUMBER := 0;
BEGIN
  IF p_bonus_percentage <= 0 THEN
    RAISE_APPLICATION_ERROR(-20001, 'Bonus percentage must be greater than 0');
END IF;

  IF p_department IS NULL THEN
    RAISE_APPLICATION_ERROR(-20002, 'Department cannot be null');
  END IF;

  DBMS_OUTPUT.PUT_LINE('=== EMPLOYEE BONUS UPDATE ===');
  DBMS_OUTPUT.PUT_LINE('Department: ' || p_department);
  DBMS_OUTPUT.PUT_LINE('Bonus Percentage: ' || p_bonus_percentage || '%');
  DBMS_OUTPUT.PUT_LINE('Processing Date: ' || TO_CHAR(SYSDATE, 'DD-MON-YYYY'));

  FOR emp_rec IN employee_cursor LOOP      v_old_salary :=
emp_rec.Salary;      v_bonus_amount := v_old_salary *
(p_bonus_percentage / 100);      v_new_salary := v_old_salary +
v_bonus_amount;
```

```
        UPDATE Employees
        SET Salary = v_new_salary
        WHERE EmployeeID = emp_rec.EmployeeID;

        v_employees_updated := v_employees_updated + 1;
v_total_bonus := v_total_bonus + v_bonus_amount;

        DBMS_OUTPUT.PUT_LINE('Employee: ' || emp_rec.Name || ' (ID: ' || emp_rec.EmployeeID || ')');
        DBMS_OUTPUT.PUT_LINE('Old Salary: $' || v_old_salary || ', Bonus: $' || ROUND(v_bonus_amount, 2));
END LOOP;

    IF v_employees_updated = 0 THEN
        DBMS_OUTPUT.PUT_LINE('No employees found in department: ' || p_department);
    ELSE
        COMMIT;
        DBMS_OUTPUT.PUT_LINE('Employees Updated: ' || v_employees_updated);
DBMS_OUTPUT.PUT_LINE('Total Bonus Amount: $' || ROUND(v_total_bonus, 2));
        DBMS_OUTPUT.PUT_LINE('Bonus update completed successfully.');
    END IF;

EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Error during bonus update: ' || SQLERRM);
        RAISE;
END UpdateEmployeeBonus;
/
```

Output:

```
=== EMPLOYEE BONUS UPDATE ===
Department: IT
Bonus Percentage: 10%
Processing Date: 28-JUN-2025
Employee: Alice Johnson (ID: 201)
Old Salary: $5000, Bonus: $500.00
Employee: Bob Wilson (ID: 202)
Old Salary: $6000, Bonus: $600.00
Employee: Carol Davis (ID: 203)
Old Salary: $5500, Bonus: $550.00
Employees Updated: 3 Total Bonus
Amount: $1650.00
Bonus update completed successfully.
```

Scenario 3: Transfer Funds Between Accounts

Sql

```
CREATE OR REPLACE PROCEDURE TransferFunds
(
   p_from_account_id IN NUMBER,
p_to_account_id IN NUMBER,
p_amount IN NUMBER
) AS     v_from_balance
NUMBER;     v_to_balance
NUMBER;
v_from_customer_id NUMBER;
v_to_customer_id NUMBER;
v_transaction_id NUMBER;
BEGIN
   IF p_amount <= 0 THEN
      RAISE_APPLICATION_ERROR(-20001, 'Transfer amount must be greater than 0');
END IF;

   IF p_from_account_id = p_to_account_id THEN
      RAISE_APPLICATION_ERROR(-20002, 'Source and destination accounts cannot be the same');
END IF;

   SELECT Balance, CustomerID
   INTO v_from_balance, v_from_customer_id
   FROM Accounts
   WHERE AccountID = p_from_account_id;

   SELECT Balance, CustomerID
   INTO v_to_balance, v_to_customer_id
   FROM Accounts
   WHERE AccountID = p_to_account_id;

   IF v_from_balance < p_amount THEN
      RAISE_APPLICATION_ERROR(-20003, 'Insufficient funds. Available balance: $' || v_from_balance);
END IF;

   SAVEPOINT before_transfer;

   UPDATE Accounts
   SET Balance = Balance - p_amount,
      LastModified = SYSDATE
   WHERE AccountID = p_from_account_id;

   UPDATE Accounts
   SET Balance = Balance + p_amount,
      LastModified = SYSDATE
   WHERE AccountID = p_to_account_id;
```

```
    SELECT NVL(MAX(TransactionID), 0) + 1
    INTO v_transaction_id
    FROM Transactions;

    INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)
  VALUES (v_transaction_id, p_from_account_id, SYSDATE, p_amount, 'Transfer Out');

    INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)
  VALUES (v_transaction_id + 1, p_to_account_id, SYSDATE, p_amount, 'Transfer In');

    COMMIT;

    DBMS_OUTPUT.PUT_LINE('=== FUND TRANSFER SUCCESSFUL ===');
    DBMS_OUTPUT.PUT_LINE('Transfer Amount: $' || p_amount);
    DBMS_OUTPUT.PUT_LINE('From Account: ' || p_from_account_id ||
                         ' (New Balance: $' || (v_from_balance - p_amount) || ')');
    DBMS_OUTPUT.PUT_LINE('To Account: ' || p_to_account_id ||
                         ' (New Balance: $' || (v_to_balance + p_amount) || ')');
    DBMS_OUTPUT.PUT_LINE('Transaction Date: ' || TO_CHAR(SYSDATE, 'DD-MON-YYYY
  HH24:MI:SS'));

  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      ROLLBACK TO before_transfer;
      RAISE_APPLICATION_ERROR(-20004, 'One or both account IDs do not exist');
    WHEN OTHERS THEN
      ROLLBACK TO before_transfer;
      DBMS_OUTPUT.PUT_LINE('Transfer failed: ' || SQLERRM);
      RAISE;
  END TransferFunds;
  /
```

Output:

```
  === FUND TRANSFER SUCCESSFUL ===
  Transfer Amount: $500
  From Account: 1 (New Balance: $2500)
  To Account: 2 (New Balance: $8500)
  Transaction Date: 28-JUN-2025 14:30:25
```

Usage Examples

sql

```sql
EXEC ProcessMonthlyInterest;
EXEC UpdateEmployeeBonus('IT', 10);
EXEC UpdateEmployeeBonus('HR', 15);
EXEC TransferFunds(1, 2, 500);
```

Output:

```
PL/SQL procedure successfully completed.
PL/SQL procedure successfully completed.
PL/SQL procedure successfully completed.
PL/SQL procedure successfully completed.
```