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
   v_age NUMBER;
BEGIN
   FOR cust IN (SELECT CustomerID, DOB FROM Customers) LOOP
      v_age := FLOOR(MONTHS_BETWEEN(SYSDATE, cust.DOB) / 12);
      IF v_age > 60 THEN
         UPDATE Loans
         SET InterestRate = InterestRate - 1
         WHERE CustomerID = cust.CustomerID;
      END IF;
   END LOOP;
   COMMIT;
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.

ALTER TABLE Customers ADD IsVIP CHAR(1);

```
BEGIN
   FOR cust IN (SELECT CustomerID, Balance FROM Customers) LOOP
      IF cust.Balance > 10000 THEN
         UPDATE Customers
         SET IsVIP = 'Y'
         WHERE CustomerID = cust.CustomerID;
```

```
        ELSE
            UPDATE Customers
            SET IsVIP = 'N'
            WHERE CustomerID = cust.CustomerID;
        END IF;
    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.

```
 DECLARE
    v_name VARCHAR2(100);
BEGIN
    FOR loan IN (
        SELECT L.LoanID, L.CustomerID, L.EndDate, C.Name
        FROM Loans L
        JOIN Customers C ON L.CustomerID = C.CustomerID
        WHERE L.EndDate BETWEEN SYSDATE AND SYSDATE + 30
    ) LOOP
        DBMS_OUTPUT.PUT_LINE(
            'Reminder: Dear ' || loan.Name ||
            ', your loan (Loan ID: ' || loan.LoanID ||
            ') is due on ' || TO_CHAR(loan.EndDate, '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 OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

   UPDATE Accounts

   SET Balance = Balance + (Balance * 0.01),

      LastModified = SYSDATE

   WHERE AccountType = 'Savings';

   COMMIT;

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(
   p_Department IN VARCHAR2,
   p_BonusPercent IN NUMBER
) IS
BEGIN
   UPDATE Employees
   SET Salary = Salary + (Salary * p_BonusPercent / 100)
   WHERE Department = p_Department;

   COMMIT;
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.

```
CREATE OR REPLACE PROCEDURE TransferFunds(
    p_SourceAccountID IN NUMBER,
    p_TargetAccountID IN NUMBER,
    p_Amount IN NUMBER
) IS
    v_SourceBalance NUMBER;
BEGIN
    SELECT Balance INTO v_SourceBalance
    FROM Accounts
    WHERE AccountID = p_SourceAccountID
    FOR UPDATE;

    IF v_SourceBalance < p_Amount THEN
        RAISE_APPLICATION_ERROR(-20001, 'Insufficient funds in source account.');
    END IF;

    UPDATE Accounts
    SET Balance = Balance - p_Amount,
        LastModified = SYSDATE
    WHERE AccountID = p_SourceAccountID;

    UPDATE Accounts
    SET Balance = Balance + p_Amount,
        LastModified = SYSDATE
    WHERE AccountID = p_TargetAccountID;

    INSERT INTO Transactions (
        TransactionID, AccountID, TransactionDate, Amount, TransactionType
    )
    VALUES (Transactions_seq.NEXTVAL, p_SourceAccountID, SYSDATE, p_Amount,
'Transfer');

    INSERT INTO Transactions (
        TransactionID, AccountID, TransactionDate, Amount, TransactionType
    )
    VALUES (Transactions_seq.NEXTVAL, p_TargetAccountID, SYSDATE, p_Amount,
'Transfer');

    COMMIT;
END;
/
```