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.

  Answer:

```
BEGIN

  FOR cust IN (SELECT CustomerID FROM Customers WHERE
MONTHS_BETWEEN(SYSDATE, DOB)/12 > 60) LOOP

    UPDATE Loans

    SET InterestRate = InterestRate - 1

    WHERE CustomerID = cust.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.

  Answer:

```
BEGIN

  FOR cust IN (SELECT CustomerID FROM Customers WHERE Balance >
10000) LOOP

    UPDATE Customers

    SET IsVIP = 'TRUE'

    WHERE CustomerID = cust.CustomerID;

  END LOOP;

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.

  Answer:

```
BEGIN

  FOR loan IN (

    SELECT l.LoanID, 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: Loan ID ' || loan.LoanID || ' for ' ||
loan.Name || ' is due soon.');

  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.

  Answer:

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

  UPDATE Accounts

  SET Balance = Balance + (Balance * 0.01)

  WHERE AccountType = 'Savings';
```

```
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.

  Answer:

```
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

  dept_name IN VARCHAR2,

  bonus_pct IN NUMBER

) AS

BEGIN

  UPDATE Employees

  SET Salary = Salary + (Salary * bonus_pct / 100)

  WHERE Department = dept_name;

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.

  Answer:

```
CREATE OR REPLACE PROCEDURE TransferFunds(

  from_acct IN NUMBER,

  to_acct IN NUMBER,

  amount IN NUMBER

) AS
```

```
  v_balance NUMBER;
BEGIN
  SELECT Balance INTO v_balance FROM Accounts WHERE AccountID =
from_acct;

  IF v_balance < amount THEN
    RAISE_APPLICATION_ERROR(-20001, 'Insufficient balance');
  END IF;

  UPDATE Accounts
  SET Balance = Balance - amount
  WHERE AccountID = from_acct;

  UPDATE Accounts
  SET Balance = Balance + amount
  WHERE AccountID = to_acct;

  COMMIT;
EXCEPTION
  WHEN OTHERS THEN
    ROLLBACK;
    DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
```