Exercise 1: Control Structures

Scenario 1:

```
DECLARE
   v_age NUMBER;
   CURSOR cur_customers IS
      SELECT CustomerID, DOB FROM Customers;
BEGIN
   FOR cust IN cur_customers LOOP

      v_age := TRUNC(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:

```
ALTER TABLE Customers ADD IsVIP CHAR(1) DEFAULT 'N';
BEGIN
   FOR cust IN (SELECT CustomerID, Balance FROM Customers) LOOP
      IF cust.Balance > 10000 THEN
         UPDATE Customers
```

```
        SET IsVIP = 'Y'
        WHERE CustomerID = cust.CustomerID;
    END IF;
  END LOOP;

  COMMIT;
END;
```

Scenario 3:

```
DECLARE
  CURSOR cur_due_loans IS
    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;
BEGIN
  FOR rec IN cur_due_loans LOOP
    DBMS_OUTPUT.PUT_LINE('Reminder: Loan ID ' || rec.LoanID ||
                         ' for customer ' || rec.Name ||
                         ' is due on ' || TO_CHAR(rec.EndDate, 'YYYY-MM-DD'));
  END LOOP;
END;
/
```

Exercise 3: Stored Procedures

Scenario 1:

```
CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS
BEGIN
   UPDATE Accounts
   SET Balance = Balance + (Balance * 0.01),
      LastModified = SYSDATE
   WHERE AccountType = 'Savings';

   COMMIT;
END;
/
```

Scenario 2:

```
CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (
   p_department IN VARCHAR2,
   p_bonus_pct IN NUMBER  -- e.g., 10 for 10%
) AS
BEGIN
   UPDATE Employees
   SET Salary = Salary + (Salary * p_bonus_pct / 100)
   WHERE Department = p_department;

   COMMIT;
END;
/
```

Scenario 3:

```
CREATE OR REPLACE PROCEDURE TransferFunds (
```

```
    p_from_account IN NUMBER,
    p_to_account IN NUMBER,
    p_amount IN NUMBER
) AS
    v_balance NUMBER;
BEGIN
    -- Check if source account has enough balance
    SELECT Balance INTO v_balance
    FROM Accounts
    WHERE AccountID = p_from_account
    FOR UPDATE;

    IF v_balance < p_amount THEN
        RAISE_APPLICATION_ERROR(-20001, 'Insufficient balance in source account.');
    END IF;

    -- Deduct from source
    UPDATE Accounts
    SET Balance = Balance - p_amount,
        LastModified = SYSDATE
    WHERE AccountID = p_from_account;

    -- Add to destination
    UPDATE Accounts
    SET Balance = Balance + p_amount,
        LastModified = SYSDATE
    WHERE AccountID = p_to_account;
```

```
    COMMIT;
END;
/
```