## Exercise 1: Control Structures

```
CREATE TABLE IF NOT EXISTS customers (
    customerID NUMBER PRIMARY KEY,
    Cname VARCHAR2(30),
    age NUMBER,
    balance NUMBER,
    interest NUMBER,
    vip VARCHAR2(5) DEFAULT 'false'
);
SELECT * FROM customers;
INSERT INTO customers (customerID, Cname, age, balance, interest, vip)
VALUES
    (101, 'John Doe', 65, 10000, 5.5, 'No'),
    (102, 'Jane Smith', 45, 15000, 6.0, 'No'),
    (103, 'Bob Lee', 70, 20000, 5.0, 'Yes');

SELECT * FROM customers;
```

### -- Scenario 1

```
BEGIN
    FOR record IN (SELECT customerID, interest
                   FROM customers
                   WHERE age > 60)
    LOOP
        UPDATE customers
        SET interest = interest - 1
        WHERE customerID = record.customerID;
    END LOOP;
    COMMIT;
END;
/

SELECT * FROM customers;
```

**-- Scenario 2**

```
BEGIN
    FOR record IN (SELECT customerID FROM customers WHERE balance > 10000)
    LOOP
        UPDATE customers
        SET vip = 'true'
        WHERE customerID = record.customerID;
    END LOOP;
    COMMIT;
END;
/

SELECT * FROM customers;
```

**-- Scenario 3**

```
CREATE TABLE loans (
    loanid NUMBER PRIMARY KEY,
    customerid NUMBER,
    duedate DATE,
    FOREIGN KEY (customerid) REFERENCES customers(customerid)
);
```

```
SELECT * FROM loans;

INSERT INTO loans (loanid, customerid, duedate)
VALUES
   (1, 101, SYSDATE + 10),
   (2, 102, SYSDATE + 35),
   (3, 103, SYSDATE + 10);

SELECT * FROM loans;

SET SERVEROUTPUT ON;

BEGIN
   FOR record IN (SELECT customerid, loanid, duedate
                  FROM loans
                  WHERE duedate <= SYSDATE + 30)
   LOOP
      DBMS_OUTPUT.PUT_LINE('Reminder: Loan ID ' || record.loanid ||
                           ' for customer ' || record.customerid ||
                           ' is due on ' || TO_CHAR(record.duedate, 'DD-MM-YYYY'));
   END LOOP;
END;
/

SELECT * FROM loans;
```

## Exercise 3: Stored Procedures

```
create table accounts (
    AccountId int primary key,
    CustomerName varchar(30),
    Balance decimal(12, 2),
    AccountType varchar(20)
);

create table employees (
    EmployeeId int primary key,
    Ename varchar(30),
    salary decimal(12, 2),
    DepartmentId int
);
```

**-- Scenario 1**

```
CREATE PROCEDURE ProcessMntInterest as
BEGIN
    UPDATE accounts
    SET balance = balance + (balance * 0.01)
    WHERE AccountType = 'Savings';
end;
/

drop PROCEDURE ProcessMntInterest;
```

**-- Scenario 2**

```
create or replace procedure UpdateEmployeeBonus (
    dept_id in number,
    bonus_percent in number
) as
BEGIN
    UPDATE EMPLOYEES
    SET SALARY = SALARY + (SALARY * bonus_percent / 100)
    where DEPARTMENTID = dept_id;
end;
/
```

## -- Scenario 3

```
CREATE OR REPLACE PROCEDURE TransferFunds (
    from_acc IN NUMBER,
    to_acc   IN NUMBER,
    amount   IN NUMBER
) AS
    from_balance NUMBER;
BEGIN
    SELECT Balance
    INTO from_balance
    FROM Accounts
    WHERE AccountID = from_acc
    FOR UPDATE;

    IF from_balance >= amount THEN
        UPDATE Accounts
        SET Balance = Balance - amount
        WHERE AccountID = from_acc;
        UPDATE Accounts
        SET Balance = Balance + amount
        WHERE AccountID = to_acc;

    ELSE
        RAISE_APPLICATION_ERROR(-20001, 'Insufficient funds in source account.');
    END IF;
END;
/
```