Exercise 4: Functions

Scenario 1: (Ex4-Scenario1.sql)

```
SET ECHO ON

SET SERVEROUTPUT ON SIZE UNLIMITED

SPOOL output-Ex4-Scenario1.txt

VARIABLE input VARCHAR2(30)

-- Function to calculate the age of a customer
CREATE OR REPLACE FUNCTION CalculateAge (
   p_dob DATE
) RETURN NUMBER IS
   v_age NUMBER;
BEGIN
   v_age := TRUNC(MONTHS_BETWEEN(SYSDATE, p_dob) / 12);
   RETURN v_age;
END CalculateAge;
/

-- Test the function
DECLARE
   v_age NUMBER;
BEGIN
   v_age := CalculateAge(TO_DATE('1990-01-01', 'YYYY-MM-DD'));
   DBMS_OUTPUT.PUT_LINE('Age: ' || v_age);
END;
/
```

```
SPOOL OFF
```

Scenario 2: (Ex4-Scenario2.sql)

```
SET ECHO ON

SET SERVEROUTPUT ON SIZE UNLIMITED

SPOOL output-Ex4-Scenario2.txt

VARIABLE input VARCHAR2(30)

-- Function to calculate the monthly installment for a loan
CREATE OR REPLACE FUNCTION CalculateMonthlyInstallment (
   p_loan_amount NUMBER,
   p_interest_rate NUMBER,
   p_duration_years NUMBER
) RETURN NUMBER IS
   v_monthly_installment NUMBER;
   v_monthly_rate NUMBER;
   v_total_months NUMBER;
BEGIN
   v_monthly_rate := p_interest_rate / 12 / 100;
   v_total_months := p_duration_years * 12;

   IF v_monthly_rate > 0 THEN
      v_monthly_installment := p_loan_amount * (v_monthly_rate * POWER(1 +
v_monthly_rate, v_total_months)) / (POWER(1 + v_monthly_rate, v_total_months) - 1);
   ELSE
      v_monthly_installment := p_loan_amount / v_total_months;
```

```
    END IF;

    RETURN v_monthly_installment;
END CalculateMonthlyInstallment;
/

-- Test the function
DECLARE
    v_installment NUMBER;
BEGIN
    v_installment := CalculateMonthlyInstallment(10000, 5, 10); -- Loan amount: 10000, Interest rate: 5%, Duration: 10 years
    DBMS_OUTPUT.PUT_LINE('Monthly Installment: ' || v_installment);
END;
/

SPOOL OFF
```

Scenario 3: (Ex4-Scenario3.sql)

```
@InitializeData.sql

SET ECHO ON

SET SERVEROUTPUT ON SIZE UNLIMITED

SPOOL output-Ex4-Scenario3.txt

VARIABLE input VARCHAR2(30)

-- Function to check if a customer has sufficient balance
```

```
CREATE OR REPLACE FUNCTION HasSufficientBalance (
   p_account_id NUMBER,
   p_amount NUMBER
) RETURN BOOLEAN IS
   v_balance NUMBER;
BEGIN
   SELECT Balance INTO v_balance FROM Accounts WHERE AccountID = p_account_id;

   RETURN v_balance >= p_amount;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      RETURN FALSE;
END HasSufficientBalance;
/

SELECT * FROM Accounts;

-- Test the function
DECLARE
   v_sufficient BOOLEAN;
BEGIN
   v_sufficient := HasSufficientBalance(1, 2000); -- Check if account 1 has at least 2000
   DBMS_OUTPUT.PUT_LINE('Sufficient Balance: ' || CASE WHEN v_sufficient THEN 'YES'
ELSE 'NO' END);

   v_sufficient := HasSufficientBalance(2, 500); -- Check if account 2 has at least 500
   DBMS_OUTPUT.PUT_LINE('Sufficient Balance: ' || CASE WHEN v_sufficient THEN 'YES'
ELSE 'NO' END);
END;
/
```

```
SPOOL OFF

@DropData.sql
```