**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