**EXERCISE 4: FUNCTIONS**

**Scenario 1: CalculateAge - Determine Customer's Age**

CREATE OR REPLACE FUNCTION CalculateAge (

p\_DOB IN DATE

) RETURN NUMBER IS

v\_Age NUMBER;

BEGIN

-- Calculate age in years

v\_Age := TRUNC(MONTHS\_BETWEEN(SYSDATE, p\_DOB) / 12);

RETURN v\_Age;

END CalculateAge;

/

DECLARE

v\_DOB DATE := TO\_DATE('1985-05-15', 'YYYY-MM-DD');

v\_Age NUMBER;

BEGIN

v\_Age := CalculateAge(v\_DOB);

DBMS\_OUTPUT.PUT\_LINE('The age is: ' || v\_Age);

END;

/

**Scenario 2: CalculateMonthlyInstallment - Compute Loan Installment**

CREATE OR REPLACE FUNCTION CalculateMonthlyInstallment (

p\_LoanAmount IN NUMBER,

p\_InterestRate IN NUMBER,

p\_LoanDuration IN NUMBER

) RETURN NUMBER IS

v\_MonthlyInterestRate NUMBER;

v\_NumberOfPayments NUMBER;

v\_MonthlyInstallment NUMBER;

BEGIN

-- Calculate monthly interest rate (convert annual rate to monthly)

v\_MonthlyInterestRate := p\_InterestRate / (12 \* 100);

-- Calculate the number of payments (months)

v\_NumberOfPayments := p\_LoanDuration \* 12;

-- Calculate the monthly installment using the formula for annuity

IF v\_MonthlyInterestRate > 0 THEN

v\_MonthlyInstallment := p\_LoanAmount \* v\_MonthlyInterestRate /

(1 - POWER(1 + v\_MonthlyInterestRate, -v\_NumberOfPayments));

ELSE

-- If the interest rate is 0, the installment is simply the loan amount divided by the number of payments

v\_MonthlyInstallment := p\_LoanAmount / v\_NumberOfPayments;

END IF;

RETURN v\_MonthlyInstallment;

END CalculateMonthlyInstallment;

/

DECLARE

v\_LoanAmount NUMBER := 50000;

v\_InterestRate NUMBER := 5;

v\_LoanDuration NUMBER := 10;

v\_MonthlyInstallment NUMBER;

BEGIN

v\_MonthlyInstallment := CalculateMonthlyInstallment(v\_LoanAmount, v\_InterestRate, v\_LoanDuration);

DBMS\_OUTPUT.PUT\_LINE('The monthly installment is: ' || TO\_CHAR(v\_MonthlyInstallment, 'FM999,999.00'));

END;

/

**Scenario 3: HasSufficientBalance - Check Account Balance Sufficiency**

CREATE OR REPLACE FUNCTION HasSufficientBalance (

p\_AccountID IN Accounts.AccountID%TYPE,

p\_Amount IN NUMBER

) RETURN BOOLEAN IS

v\_Balance Accounts.Balance%TYPE;

BEGIN

-- Fetch the current balance of the account

SELECT Balance INTO v\_Balance

FROM Accounts

WHERE AccountID = p\_AccountID;

-- Check if the balance is sufficient

RETURN v\_Balance >= p\_Amount;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN FALSE; -- If the account does not exist, treat it as insufficient funds

WHEN OTHERS THEN

-- Log the error and return FALSE

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

RETURN FALSE;

END HasSufficientBalance;

/

DECLARE

v\_AccountID NUMBER := 1;

v\_Amount NUMBER := 200;

v\_HasBalance BOOLEAN;

BEGIN

v\_HasBalance := HasSufficientBalance(v\_AccountID, v\_Amount);

IF v\_HasBalance THEN

DBMS\_OUTPUT.PUT\_LINE('The account has sufficient balance.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('The account does not have sufficient balance.');

END IF;

END;

/