**Exercise 4: Functions**

**Scenario 1:** Calculate the age of customers for eligibility checks.

* + **Question:** Write a function CalculateAge that takes a customer's date of birth as input and returns their age in years.

**Code :**

CREATE OR REPLACE FUNCTION CalculateAge (

p\_date\_of\_birth IN DATE

) RETURN NUMBER IS

v\_age NUMBER;

BEGIN

-- Calculate age based on the current date and the date of birth

v\_age := TRUNC(MONTHS\_BETWEEN(SYSDATE, p\_date\_of\_birth) / 12);

RETURN v\_age;

EXCEPTION

WHEN OTHERS THEN

-- Handle any unexpected errors

RAISE\_APPLICATION\_ERROR(-20001, 'Error calculating age: ' || SQLERRM);

END CalculateAge;

**Scenario 2:** The bank needs to compute the monthly installment for a loan.

* + **Question:** Write a function **CalculateMonthlyInstallment** that takes the loan amount, interest rate, and loan duration in years as input and returns the monthly installment amount.

**CODE :**

CREATE OR REPLACE FUNCTION CalculateMonthlyInstallment (

p\_loan\_amount IN NUMBER,

p\_annual\_interest\_rate IN NUMBER,

p\_loan\_duration\_years IN NUMBER

) RETURN NUMBER IS

v\_monthly\_installment NUMBER;

v\_monthly\_interest\_rate NUMBER;

v\_number\_of\_payments NUMBER;

BEGIN

-- Convert annual interest rate to monthly and calculate number of payments

v\_monthly\_interest\_rate := p\_annual\_interest\_rate / 12 / 100;

v\_number\_of\_payments := p\_loan\_duration\_years \* 12;

-- Calculate the monthly installment using the formula

IF v\_monthly\_interest\_rate > 0 THEN

v\_monthly\_installment := p\_loan\_amount \*

(v\_monthly\_interest\_rate \* POWER(1 + v\_monthly\_interest\_rate, v\_number\_of\_payments)) /

(POWER(1 + v\_monthly\_interest\_rate, v\_number\_of\_payments) - 1);

ELSE

-- If the interest rate is 0%, simple division

v\_monthly\_installment := p\_loan\_amount / v\_number\_of\_payments;

END IF;

RETURN v\_monthly\_installment;

EXCEPTION

WHEN OTHERS THEN

-- Handle any unexpected errors

RAISE\_APPLICATION\_ERROR(-20001, 'Error calculating monthly installment: ' || SQLERRM);

END CalculateMonthlyInstallment;

**Scenario 3:** Check if a customer has sufficient balance before making a transaction.

* + **Question:** Write a function **HasSufficientBalance** that takes an account ID and an amount as input and returns a boolean indicating whether the account has at least the specified amount.

**CODE :**

CREATE OR REPLACE FUNCTION HasSufficientBalance (

p\_account\_id IN ACCOUNTS.ACCOUNT\_ID%TYPE,

p\_amount IN NUMBER

) RETURN BOOLEAN IS

v\_balance ACCOUNTS.BALANCE%TYPE;

BEGIN

-- Select the current balance of the account

SELECT BALANCE INTO v\_balance

FROM ACCOUNTS

WHERE ACCOUNT\_ID = p\_account\_id;

-- Check if the balance is sufficient

IF v\_balance >= p\_amount THEN

RETURN TRUE;

ELSE

RETURN FALSE;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

-- Handle cases where the account ID does not exist

RETURN FALSE;

WHEN OTHERS THEN

-- Handle any unexpected errors

RAISE\_APPLICATION\_ERROR(-20001, 'Error checking balance: ' || SQLERRM);

END HasSufficientBalance;