**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.**

CREATE OR REPLACE FUNCTION CalculateAge(p\_dob IN DATE)

RETURN NUMBER

AS

v\_age NUMBER;

BEGIN

IF p\_dob IS NULL THEN

RETURN NULL; -- Return NULL if input date is NULL

END IF;

SELECT FLOOR(MONTHS\_BETWEEN(SYSDATE, p\_dob) / 12) INTO v\_age FROM dual;

RETURN v\_age;

EXCEPTION

WHEN OTHERS THEN

RETURN NULL;

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.**

CREATE OR REPLACE FUNCTION CalculateMonthlyInstallment(

p\_loan\_amount NUMBER,

p\_annual\_interest\_rate NUMBER,

p\_loan\_duration\_years NUMBER

) RETURN NUMBER

IS

v\_monthly\_interest\_rate NUMBER;

v\_number\_of\_months NUMBER;

v\_monthly\_installment NUMBER;

BEGIN

IF p\_loan\_amount <= 0 OR p\_loan\_duration\_years <= 0 THEN

RETURN NULL; -- Invalid loan amount or duration

END IF;

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

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

IF v\_monthly\_interest\_rate > 0 THEN

v\_monthly\_installment := (p\_loan\_amount \* v\_monthly\_interest\_rate) /

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

ELSE

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

END IF;

RETURN v\_monthly\_installment;

EXCEPTION

WHEN OTHERS THEN

RETURN NULL;

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.**

CREATE OR REPLACE FUNCTION HasSufficientBalance(

p\_account\_id NUMBER,

p\_amount NUMBER

) RETURN BOOLEAN

IS

v\_balance NUMBER;

BEGIN

IF p\_amount < 0 THEN

RETURN FALSE; -- Amount cannot be negative

END IF;

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;

WHEN OTHERS THEN

RETURN FALSE;

END;

END HasSufficientBalance;