**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 the CalculateAge function

CREATE OR REPLACE FUNCTION CalculateAge (

p\_DOB IN DATE

) RETURN NUMBER AS

v\_Age NUMBER;

BEGIN

-- Calculate age in years

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

RETURN v\_Age;

EXCEPTION

WHEN OTHERS THEN

-- Handle exceptions by returning a NULL value

RETURN NULL;

END CalculateAge;

/

SELECT CalculateAge(TO\_DATE('1985-05-15', 'YYYY-MM-DD')) AS Age

FROM dual;

**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 the CalculateMonthlyInstallment function

CREATE OR REPLACE FUNCTION CalculateMonthlyInstallment (

p\_LoanAmount IN NUMBER,

p\_InterestRate IN NUMBER, -- Annual interest rate as a percentage

p\_LoanDuration IN NUMBER -- Duration in years

) RETURN NUMBER AS

v\_MonthlyInterestRate NUMBER;

v\_TotalPayments NUMBER;

v\_MonthlyInstallment NUMBER;

BEGIN

-- Convert annual interest rate percentage to a monthly rate

v\_MonthlyInterestRate := p\_InterestRate / 100 / 12;

-- Calculate the total number of payments (months)

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

-- Calculate the monthly installment

IF v\_MonthlyInterestRate = 0 THEN

-- Special case for zero interest rate

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

ELSE

v\_MonthlyInstallment := (p\_LoanAmount \* v\_MonthlyInterestRate \* POWER(1 + v\_MonthlyInterestRate, v\_TotalPayments)) /

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

END IF;

RETURN v\_MonthlyInstallment;

EXCEPTION

WHEN OTHERS THEN

-- Handle exceptions by returning NULL

RETURN NULL;

END CalculateMonthlyInstallment;

/

SELECT CalculateMonthlyInstallment(10000, 5, 10) AS MonthlyInstallment

FROM dual;

**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 the HasSufficientBalance function

CREATE OR REPLACE FUNCTION HasSufficientBalance (

p\_AccountID IN Accounts.AccountID%TYPE,

p\_Amount IN NUMBER

) RETURN BOOLEAN AS

v\_Balance Accounts.Balance%TYPE;

BEGIN

-- Fetch the balance of the account

SELECT Balance INTO v\_Balance

FROM Accounts

WHERE AccountID = p\_AccountID;

-- Check if the account has sufficient balance

IF v\_Balance >= p\_Amount THEN

RETURN TRUE;

ELSE

RETURN FALSE;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

-- Handle the case where the account ID does not exist

RETURN FALSE;

WHEN OTHERS THEN

-- Handle any other exceptions

RETURN FALSE;

END HasSufficientBalance;

/

SELECT HasSufficientBalance(1, 500) AS IsSufficient

FROM dual;