**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

IS

v\_age NUMBER;

BEGIN

-- Calculate the age based on the date of birth

v\_age := FLOOR(MONTHS\_BETWEEN(SYSDATE, p\_dob) / 12);

RETURN v\_age;

EXCEPTION

WHEN OTHERS THEN

-- Handle any unexpected errors

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

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 IN NUMBER,

p\_annual\_interest\_rate IN NUMBER,

p\_duration\_years IN NUMBER

)

RETURN NUMBER

IS

v\_monthly\_interest\_rate NUMBER;

v\_total\_payments NUMBER;

v\_monthly\_installment NUMBER;

BEGIN

-- Calculate the monthly interest rate

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

-- Calculate the total number of payments

v\_total\_payments := p\_duration\_years \* 12;

-- Calculate the monthly installment using the formula

IF v\_monthly\_interest\_rate = 0 THEN

-- Handle the case where the interest rate is 0

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

ELSE

v\_monthly\_installment := (p\_loan\_amount \* v\_monthly\_interest\_rate \* POWER(1 + v\_monthly\_interest\_rate, v\_total\_payments)) /

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

END IF;

RETURN v\_monthly\_installment;

EXCEPTION

WHEN OTHERS THEN

-- Handle any unexpected errors

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

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 IN NUMBER,

p\_amount IN NUMBER

)

RETURN CHAR

IS

v\_balance NUMBER;

BEGIN

-- Retrieve the balance of the specified account

SELECT Balance INTO v\_balance

FROM Accounts

WHERE AccountID = p\_account\_id;

-- Check if the balance is sufficient

IF v\_balance >= p\_amount THEN

RETURN 'Y'; -- Sufficient balance

ELSE

RETURN 'N'; -- Insufficient balance

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

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

RETURN 'N'; -- Account not found

WHEN OTHERS THEN

-- Handle any unexpected errors

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

RETURN 'N'; -- Return 'N' in case of error

END HasSufficientBalance;

/