**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 DATE) RETURN NUMBER IS

v\_Age NUMBER;

BEGIN

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

RETURN v\_Age;

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\_LoanAmount NUMBER,

p\_InterestRate NUMBER,

p\_LoanDurationYears NUMBER

) RETURN NUMBER IS

v\_MonthlyInstallment NUMBER;

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

v\_TotalMonths NUMBER := p\_LoanDurationYears \* 12;

BEGIN

IF v\_MonthlyInterestRate = 0 THEN

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

ELSE

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

END IF;

RETURN v\_MonthlyInstallment;

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\_AccountID NUMBER,

p\_Amount NUMBER

) RETURN BOOLEAN IS

v\_Balance Accounts.Balance%TYPE;

BEGIN

SELECT Balance INTO v\_Balance FROM Accounts WHERE AccountID = p\_AccountID;

RETURN v\_Balance >= p\_Amount;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN FALSE;

END HasSufficientBalance;

/