**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

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\_loan\_amount IN NUMBER,

p\_interest\_rate IN NUMBER,

p\_duration\_years IN NUMBER

) RETURN NUMBER IS

v\_monthly\_installment NUMBER;

v\_interest\_rate\_monthly NUMBER;

v\_total\_months NUMBER;

BEGIN

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

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

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

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

RETURN v\_monthly\_installment;

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 accounts.account\_id%TYPE,

p\_amount IN NUMBER

) RETURN BOOLEAN IS

v\_balance accounts.balance%TYPE;

BEGIN

SELECT balance INTO v\_balance

FROM accounts

WHERE account\_id = p\_account\_id;

RETURN v\_balance >= p\_amount;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN FALSE;

END HasSufficientBalance;

/