**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.

**Ans:**

CREATE OR REPLACE FUNCTION CalculateAge (

dob IN DATE

) RETURN NUMBER AS

age NUMBER;

BEGIN

SELECT FLOOR(MONTHS\_BETWEEN(SYSDATE, dob) / 12) INTO age FROM dual;

RETURN 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.

**Ans:**

CREATE OR REPLACE FUNCTION CalculateMonthlyInstallment (

la IN NUMBER,

annualInterestRate IN NUMBER,

durationYears IN NUMBER

) RETURN NUMBER AS

monthlyInterestRate NUMBER;

totalPayments NUMBER;

monthlyInstallment NUMBER;

BEGIN

monthlyInterestRate := annualInterestRate / 1200;

totalPayments := durationYears \* 12;

IF monthlyInterestRate > 0 THEN

monthlyInstallment := (la \* monthlyInterestRate \* POWER(1 + monthlyInterestRate, totalPayments)) /

(POWER(1 + monthlyInterestRate, totalPayments) - 1);

ELSE

monthlyInstallment := la / totalPayments;

END IF;

RETURN 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.

**Ans:**

CREATE OR REPLACE FUNCTION HasSufficientBalance (

accID IN NUMBER,

amt IN NUMBER

) RETURN BOOLEAN AS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance

FROM Accounts

WHERE AccountID = accID;

RETURN v\_balance >= amt;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN FALSE;

END HasSufficientBalance;

/