**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 := TRUNC((SYSDATE - p\_dob) / 365.25);

RETURN v\_age;

END;

**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 IN NUMBER, p\_interestRate IN NUMBER, p\_durationYears IN NUMBER)

RETURN NUMBER

IS

v\_monthlyInstallment NUMBER;

BEGIN

v\_monthlyInstallment := p\_loanAmount \* (p\_interestRate / 1200) / (1 - POWER(1 + p\_interestRate / 1200, -p\_durationYears \* 12));

RETURN v\_monthlyInstallment;

END;

**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 IN NUMBER, p\_amount IN NUMBER)

RETURN BOOLEAN

IS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance FROM Accounts WHERE AccountID = p\_accountID;

RETURN v\_balance >= p\_amount;

END;