**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

**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.

**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.

**Scenario 1:**

CREATE OR REPLACE FUNCTION GetCustomerAge(

birth\_date IN DATE

) RETURN NUMBER

IS

v\_age NUMBER;

BEGIN

v\_age := FLOOR(MONTHS\_BETWEEN(CURRENT\_DATE, birth\_date) / 12);

RETURN v\_age;

END GetCustomerAge;

-- Example usage

SELECT GetCustomerAge(

(SELECT DOB FROM Customers WHERE CustomerID = 1)

) AS Customer\_Age

FROM dual;

**Scenario 2:**

CREATE OR REPLACE FUNCTION ComputeMonthlyInstallment(

principal\_amount IN NUMBER,

yearly\_interest\_rate IN NUMBER,

loan\_period\_years IN NUMBER

) RETURN NUMBER

IS

monthly\_rate NUMBER;

monthly\_payment NUMBER;

BEGIN

monthly\_rate := yearly\_interest\_rate / (12 \* 100);

monthly\_payment := (principal\_amount \* monthly\_rate \* POWER((1 + monthly\_rate), loan\_period\_years \* 12)) /

(POWER((1 + monthly\_rate), loan\_period\_years \* 12) - 1);

RETURN monthly\_payment;

END ComputeMonthlyInstallment;

-- Example usage

SELECT ROUND(

ComputeMonthlyInstallment(

(SELECT LoanAmount FROM Loans WHERE LoanID = 1),

(SELECT InterestRate FROM Loans WHERE LoanID = 1),

FLOOR(MONTHS\_BETWEEN((SELECT EndDate FROM Loans WHERE LoanID = 1), (SELECT StartDate FROM Loans WHERE LoanID = 1)) / 12)

), 2

) AS Monthly\_Installment

FROM dual;

**Scenario 3:**

CREATE OR REPLACE FUNCTION IsBalanceSufficient(

account\_id IN NUMBER,

required\_amount IN NUMBER

) RETURN BOOLEAN

IS

account\_balance NUMBER;

BEGIN

SELECT Balance INTO account\_balance

FROM Accounts

WHERE AccountID = account\_id;

IF account\_balance >= required\_amount THEN

RETURN TRUE;

ELSE

RETURN FALSE;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN FALSE;

END IsBalanceSufficient;

-- Example usage

DECLARE

sufficient\_balance BOOLEAN;

BEGIN

sufficient\_balance := IsBalanceSufficient(1, 500);

DBMS\_OUTPUT.PUT\_LINE('Sufficient balance available: ' || CASE

WHEN sufficient\_balance THEN 'TRUE'

ELSE 'FALSE'

END);

END;