**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: CalculateAge**

CREATE DEFINER=`root`@`localhost` FUNCTION `CalculateAge`(dob DATE) RETURNS int

DETERMINISTIC

BEGIN

DECLARE age INT;

SET age = TIMESTAMPDIFF(YEAR,dob,CURDATE());

RETURN age;

RETURN 1;

END

**Scenario 2: CalculateMonthlyInstallment**

CREATE DEFINER=`root`@`localhost` FUNCTION `CalculateMonthlyInstallment`(loan\_amount DECIMAL(10,2),interest\_rate DECIMAL(5,2), loan\_duration INT) RETURNS decimal(10,2)

DETERMINISTIC

BEGIN

DECLARE monthly\_rate DECIMAL(10,7);

DECLARE number\_of\_payments INT;

DECLARE installment DECIMAL(10,2);

SET monthly\_rate = interest\_rate / 12 / 100;

SET number\_of\_payments = loan\_duration \* 12;

set installment = loan\_amount \*( monthly\_rate \* POWER(1+monthly\_rate, number\_of\_payments)) / (POWER(1+monthly\_rate, number\_of\_payments)-1);

RETURN installment;

END

**Scenario 3: HasSufficientBalance**  
CREATE DEFINER=`root`@`localhost` FUNCTION `HasSufficientBalance`(account\_id INT,amount DECIMAL(10,2)) RETURNS int

DETERMINISTIC

BEGIN

DECLARE account\_balance DECIMAL(10,2);

select Balance INTO account\_balance

FROM Accounts Where ACCOUNTID = account\_id;

RETURN account\_balance >= amount;

END