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.
  + **Solution:**

CREATE OR REPLACE FUNCTION CalculateAge (

p\_dob DATE

) RETURN NUMBER IS

v\_age NUMBER;

BEGIN

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

RETURN v\_age;

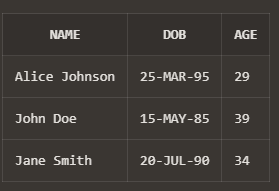
END;

/

-- Test the CalculateAge function

SELECT Name, DOB, CalculateAge(DOB) AS Age

FROM Customers;



**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.
  + **Solution:**

CREATE OR REPLACE FUNCTION CalculateMonthlyInstallment (

p\_loan\_amount NUMBER,

p\_interest\_rate NUMBER,

p\_duration\_years NUMBER

) RETURN NUMBER IS

v\_monthly\_installment NUMBER;

v\_monthly\_interest\_rate NUMBER;

v\_number\_of\_payments NUMBER;

BEGIN

-- Convert annual interest rate to monthly interest rate

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

-- Calculate the total number of monthly payments

v\_number\_of\_payments := p\_duration\_years \* 12;

-- Calculate the monthly installment using the formula for an annuity

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

(1 - POWER(1 + v\_monthly\_interest\_rate, -v\_number\_of\_payments));

RETURN v\_monthly\_installment;

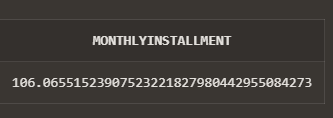
END;

/

-- Test the CalculateMonthlyInstallment function

SELECT CalculateMonthlyInstallment(10000, 5, 10) AS MonthlyInstallment

FROM DUAL;



**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.
  + **Solution:**

CREATE OR REPLACE FUNCTION HasSufficientBalance (

p\_account\_id NUMBER,

p\_amount NUMBER

) RETURN BOOLEAN IS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance

FROM Accounts

WHERE AccountID = p\_account\_id;

IF v\_balance >= p\_amount THEN

RETURN TRUE;

ELSE

RETURN FALSE;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN FALSE;

WHEN OTHERS THEN

RETURN FALSE;

END;

/