**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 DATE) RETURN NUMBER

IS

    v\_age NUMBER;

BEGIN

    v\_age := TRUNC(MONTHS\_BETWEEN(SYSDATE, p\_dob) / 12);

    RETURN v\_age;

END CalculateAge;

BEGIN

    DBMS\_OUTPUT.PUT\_LINE('Age is ' || CalculateAge('20-JUL-1990'));

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\_loanAmt NUMBER,p\_iRate NUMBER,p\_durInYears NUMBER)

RETURN NUMBER

IS

    v\_r NUMBER;

    v\_n NUMBER;

    v\_monthlyInstallment NUMBER;

BEGIN

    v\_r := p\_iRate / 12 / 100;  -- rate per month

    v\_n := p\_durInYears \* 12;

    v\_monthlyInstallment := p\_loanAmt \* v\_r \* ((1 + v\_r) \*\* v\_n) / ((1 + v\_r) \*\* v\_n - 1);

    RETURN v\_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.

CREATE OR REPLACE FUNCTION HasSufficientBalance(p\_accID NUMBER,p\_amt NUMBER)

RETURN BOOLEAN

IS

    v\_balance NUMBER;

BEGIN

  SELECT balance INTO v\_balance FROM Accounts WHERE accountid = p\_accID;

  RETURN v\_balance >= p\_amt;

EXCEPTION

  WHEN no\_data\_found THEN

    RETURN FALSE;

  WHEN others THEN

    RETURN FALSE;

END HasSufficientBalance;

/