**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 FUNCTION CalculateAge (

p\_DOB IN DATE

) RETURN NUMBER

IS

v\_Age NUMBER;

BEGIN

*-- Calculate the age*

SELECT FLOOR(MONTHS\_BETWEEN(SYSDATE, p\_DOB) / 12) INTO v\_Age

FROM dual;

RETURN v\_Age;

END CalculateAge;

/

**Example usage:**

SELECT CalculateAge(DOB) AS Age

FROM Customers

WHERE CustomerID = 1;

**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 FUNCTION CalculateMonthlyInstallment (

p\_LoanAmount IN NUMBER,

p\_InterestRate IN NUMBER, *-- Annual interest rate as a percentage*

p\_LoanDuration IN NUMBER *-- Duration in years*

) RETURN NUMBER

IS

v\_MonthlyRate NUMBER;

v\_TotalPayments NUMBER;

v\_MonthlyInstallment NUMBER;

BEGIN

*-- Convert annual interest rate percentage to monthly rate as a decimal*

v\_MonthlyRate := (p\_InterestRate / 100) / 12;

*-- Calculate total number of payments (months)*

v\_TotalPayments := p\_LoanDuration \* 12;

*-- Calculate the monthly installment using the formula*

IF v\_MonthlyRate = 0 THEN

*-- If interest rate is zero, simply divide the loan amount by total payments*

v\_MonthlyInstallment := p\_LoanAmount / v\_TotalPayments;

ELSE

v\_MonthlyInstallment := (p\_LoanAmount \* v\_MonthlyRate \* POWER(1 + v\_MonthlyRate, v\_TotalPayments)) / (POWER(1 + v\_MonthlyRate, v\_TotalPayments) - 1);

END IF;

RETURN v\_MonthlyInstallment;

END CalculateMonthlyInstallment;

/

**Example usage:**

SELECT CalculateMonthlyInstallment(100000, 5, 10) AS MonthlyInstallment

FROM dual;

This will calculate and return the monthly installment for a loan of $100,000 with a 5% annual interest rate and a duration of 10 years.

**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 FUNCTION HasSufficientBalance (

p\_AccountID IN Accounts.AccountID%TYPE,

p\_Amount IN NUMBER

) RETURN BOOLEAN

IS

v\_Balance NUMBER;

BEGIN

*-- Retrieve the balance for the given account ID*

SELECT Balance INTO v\_Balance

FROM Accounts

WHERE AccountID = p\_AccountID;

*-- Check if the balance is sufficient*

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 HasSufficientBalance;

/

**Example usage:**

SELECT HasSufficientBalance(101, 500) AS IsSufficient

FROM dual;

This will return TRUE if the account with ID 101 has at least 500 rs; otherwise, it will return FALSE.