**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.

**CODE:**

CREATE OR REPLACE FUNCTION CalculateAge(p\_DOB DATE) RETURN NUMBER IS

BEGIN

RETURN FLOOR(MONTHS\_BETWEEN(SYSDATE, p\_DOB)/12);

END;

/

SELECT CalculateAge(TO\_DATE('15-MAY-1990', 'DD-MON-YYYY')) AS Age FROM dual;

**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.

**CODE:**

CREATE OR REPLACE FUNCTION CalculateMonthlyInstallment(p\_LoanAmount NUMBER, p\_Rate NUMBER, p\_Years NUMBER) RETURN NUMBER IS

v\_MonthlyRate NUMBER := p\_Rate / 1200;

v\_NumPayments NUMBER := p\_Years \* 12;

BEGIN

RETURN p\_LoanAmount \* v\_MonthlyRate / (1 - POWER(1 + v\_MonthlyRate, -v\_NumPayments));

END;

/

CREATE OR REPLACE FUNCTION CalculateMonthlyInstallment(p\_LoanAmount NUMBER, p\_Rate NUMBER, p\_Years NUMBER) RETURN NUMBER IS

v\_MonthlyRate NUMBER := p\_Rate / 1200;

v\_NumPayments NUMBER := p\_Years \* 12;

BEGIN

RETURN p\_LoanAmount \* v\_MonthlyRate / (1 - POWER(1 + v\_MonthlyRate, -v\_NumPayments));

END;

/

-- Now, execute the function

SELECT CalculateMonthlyInstallment(300000, 4.5, 30) AS Monthly\_Payment FROM dual;

SELECT CalculateMonthlyInstallment(200000, 3.75, 15) AS Monthly\_Payment FROM dual;

SELECT CalculateMonthlyInstallment(10000, 7.5, 5) AS Monthly\_Payment FROM dual;

SELECT CalculateMonthlyInstallment(5000, 10, 1) AS Monthly\_Payment 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.

**CODE:**

CREATE OR REPLACE FUNCTION HasSufficientBalance(p\_AccountID NUMBER, p\_Amount NUMBER) RETURN NUMBER IS

v\_Balance NUMBER;

BEGIN

SELECT Balance INTO v\_Balance FROM Accounts WHERE AccountID = p\_AccountID;

-- Return 1 for TRUE, 0 for FALSE

IF v\_Balance >= p\_Amount THEN

RETURN 1; -- Sufficient balance

ELSE

RETURN 0; -- Insufficient balance

END IF;

END;

/

SELECT HasSufficientBalance(6, 1000) AS Sufficient\_Balance

FROM dual;