**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.

**Procedure:**

CREATE OR REPLACE FUNCTION CalculateAge(p\_dob DATE)

RETURN NUMBER

IS

v\_age NUMBER;

BEGIN

SELECT FLOOR(MONTHS\_BETWEEN(SYSDATE, p\_dob) / 12) INTO v\_age FROM dual;

RETURN v\_age;

EXCEPTION

WHEN OTHERS THEN

RETURN NULL;

END;

DECLARE

v\_age NUMBER;

BEGIN

v\_age := CalculateAge(TO\_DATE('1985-05-15', 'YYYY-MM-DD'));

DBMS\_OUTPUT.PUT\_LINE('The calculated age is: ' || v\_age);

END;

/

**Output:**

Function CALCULATEAGE compiled

The calculated age is: 39

PL/SQL procedure successfully completed.

**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.

**Procedure:**

CREATE OR REPLACE FUNCTION CalculateMonthlyInstallment(

p\_loan\_amount NUMBER,

p\_annual\_interest\_rate NUMBER,

p\_loan\_duration\_years NUMBER

)

RETURN NUMBER

IS

v\_monthly\_interest\_rate NUMBER;

v\_number\_of\_months NUMBER;

v\_monthly\_installment NUMBER;

BEGIN

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

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

IF v\_monthly\_interest\_rate> 0 THEN

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

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

ELSE

v\_monthly\_installment := p\_loan\_amount / v\_number\_of\_months;

END IF;

RETURN v\_monthly\_installment;

EXCEPTION

WHEN OTHERS THEN

RETURN NULL;

END;

DECLARE

v\_installment NUMBER;

BEGIN

v\_installment := CalculateMonthlyInstallment(1000, 7, 5);

DBMS\_OUTPUT.PUT\_LINE('Monthly Installment: ' || v\_installment);

END;

/

**Output:**

Monthly Installment: 19.80119854034953503544073279139220348752

PL/SQL procedure successfully completed.

**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.

**Procedure:**

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;

RETURN v\_balance>= p\_amount;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN FALSE;

WHEN OTHERS THEN

RETURN FALSE;

END;

DECLARE

v\_sufficient Boolean;

BEGIN

v\_sufficient := HasSufficientBalance(1,3000);

IF v\_sufficient THEN

DBMS\_OUTPUT.PUT\_LINE('Sufficient balance.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Insufficient balance.');

END IF;

END;

/

**Output:**

Insufficient balance.

PL/SQL procedure successfully completed.