**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 IN DATE)

RETURN NUMBER IS

v\_age NUMBER;

BEGIN

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

RETURN v\_age;

END;

/

SELECT Name, CalculateAge(DOB) AS Age FROM Customers;

**Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**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\_loan\_amount IN NUMBER,

p\_interest\_rate IN NUMBER,

p\_years IN NUMBER

)

RETURN NUMBER IS

v\_monthly\_rate NUMBER;

v\_months NUMBER;

v\_emi NUMBER;

BEGIN

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

v\_months := p\_years \* 12;

IF v\_monthly\_rate = 0 THEN

v\_emi := p\_loan\_amount / v\_months;

ELSE

v\_emi := p\_loan\_amount \* v\_monthly\_rate \*

POWER(1 + v\_monthly\_rate, v\_months) /

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

END IF;

RETURN ROUND(v\_emi, 2);

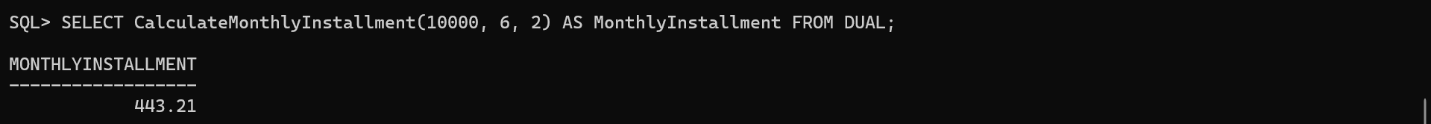
END;

/

SELECT CalculateMonthlyInstallment(10000, 6, 2) AS MonthlyInstallment FROM DUAL;

**Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**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\_account\_id IN NUMBER,

p\_amount IN 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;

END;

/

DECLARE

v\_result BOOLEAN;

BEGIN

v\_result := HasSufficientBalance(1, 500);

IF v\_result THEN

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

ELSE

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

END IF;

END;

/

**Output:**

**A black screen with white text

AI-generated content may be incorrect.A computer screen with white text

AI-generated content may be incorrect.**