**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.

set serveroutput on;

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;

END CalculateAge;

/

DECLARE

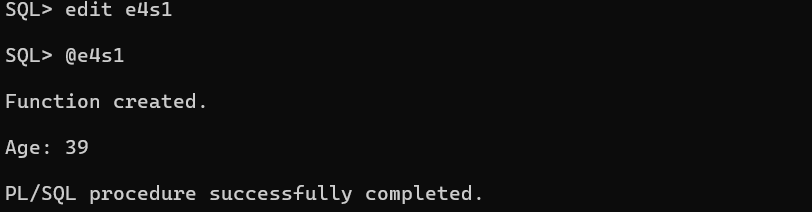
v\_age NUMBER;

BEGIN

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

DBMS\_OUTPUT.PUT\_LINE('Age: ' || v\_age);

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.

set serveroutput on;

CREATE OR REPLACE FUNCTION CalculateMonthlyInstallment(p\_loan\_amount NUMBER, p\_interest\_rate NUMBER, p\_duration\_years NUMBER) RETURN NUMBER IS

v\_monthly\_rate NUMBER;

v\_num\_payments NUMBER;

v\_monthly\_installment NUMBER;

BEGIN

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

v\_num\_payments := p\_duration\_years \* 12;

v\_monthly\_installment := p\_loan\_amount \* v\_monthly\_rate / (1 - POWER(1 + v\_monthly\_rate, -v\_num\_payments));

RETURN v\_monthly\_installment;

END CalculateMonthlyInstallment;

/

DECLARE

v\_installment NUMBER;

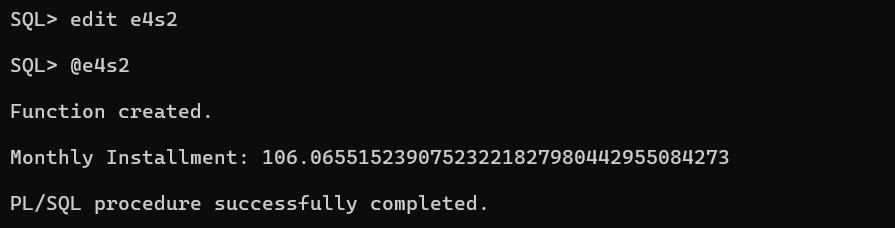
BEGIN

v\_installment := CalculateMonthlyInstallment(10000, 5, 10);

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

END;

/



**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.

set serveroutput on;

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;

END HasSufficientBalance;

/

DECLARE

v\_sufficient BOOLEAN;

BEGIN

v\_sufficient := HasSufficientBalance(1, 1500);

IF v\_sufficient THEN

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

ELSE

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

END IF;

END;

/

****