**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 OR REPLACE FUNCTION CalculateAge (

p\_dob DATE

) RETURN NUMBER

IS

v\_age NUMBER;

BEGIN

-- Calculate the number of full years between current date and date of birth

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

RETURN v\_age;

END;

/

DECLARE

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

v\_age NUMBER;

BEGIN

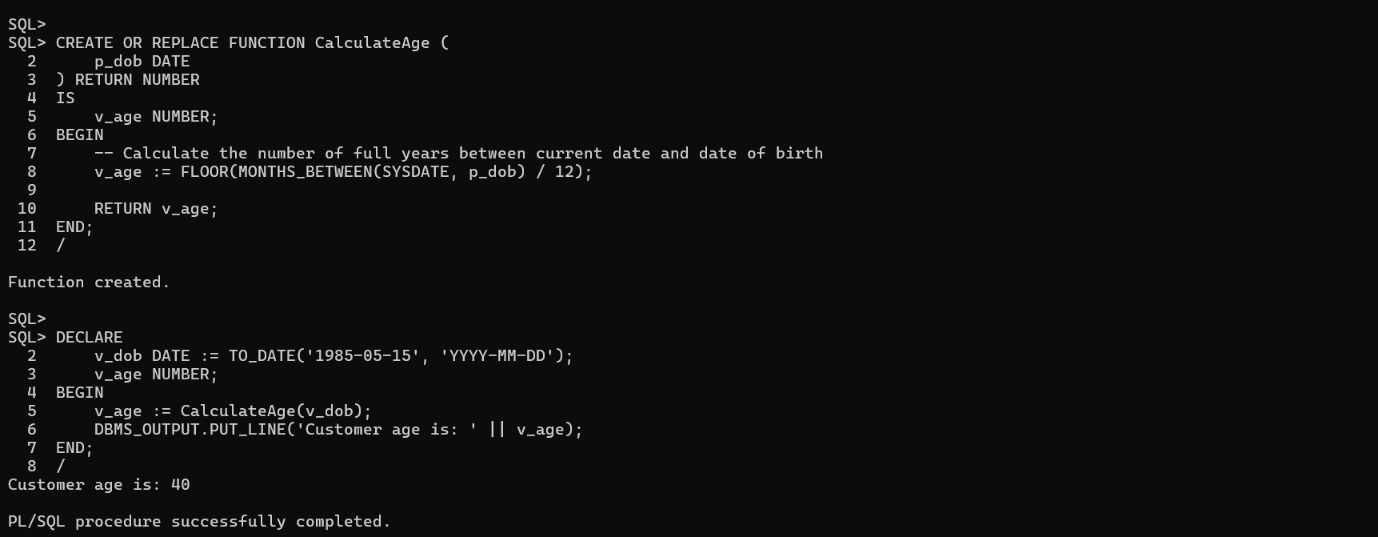
v\_age := CalculateAge(v\_dob);

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

END;

/

**Output :-**



**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 OR REPLACE FUNCTION CalculateMonthlyInstallment (

p\_loan\_amount IN NUMBER,

p\_interest\_rate IN NUMBER, -- annual rate in percent

p\_years IN NUMBER

) RETURN NUMBER

IS

v\_monthly\_rate NUMBER;

v\_months NUMBER;

v\_emi NUMBER;

BEGIN

-- Convert annual interest rate to monthly and percentage to fraction

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

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

-- EMI formula: [P \* R \* (1 + R)^N] / [(1 + R)^N - 1]

IF v\_monthly\_rate = 0 THEN

-- No interest case (simple division)

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);

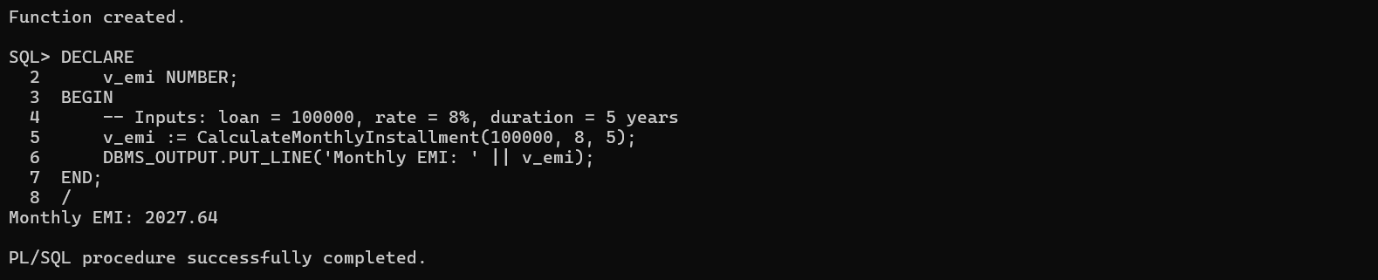
END IF;

RETURN ROUND(v\_emi, 2);

END;

/

Output :-



**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 OR REPLACE FUNCTION HasSufficientBalance (

p\_account\_id IN NUMBER,

p\_amount IN NUMBER

) RETURN BOOLEAN

IS

v\_balance NUMBER;

BEGIN

-- Fetch the current balance of the account

SELECT Balance INTO v\_balance

FROM Accounts

WHERE AccountID = p\_account\_id;

-- Return TRUE if balance is sufficient

RETURN v\_balance >= p\_amount;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Account not found.');

RETURN FALSE;

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

RETURN FALSE;

END;

/

DECLARE

v\_result BOOLEAN;

BEGIN

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

IF v\_result THEN

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

ELSE

DBMS\_OUTPUT.PUT\_LINE('Transaction denied: Insufficient balance or account not found.');

END IF;

END;

/

**Output :-**

