**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.

**QUERY:**

CREATE OR REPLACE FUNCTION Calculate\_Age(

DOB IN DATE

)

RETURN NUMBER

IS

age NUMBER;

BEGIN

age := TRUNC(MONTHS\_BETWEEN(SYSDATE, DOB) / 12);

RETURN age;

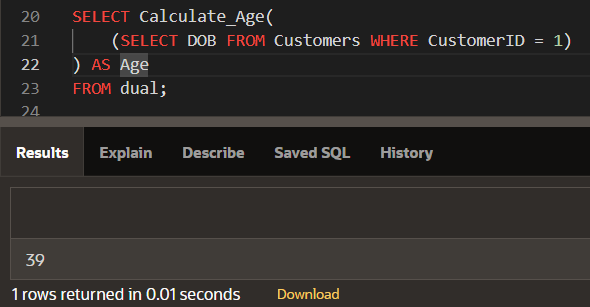
END Calculate\_Age;

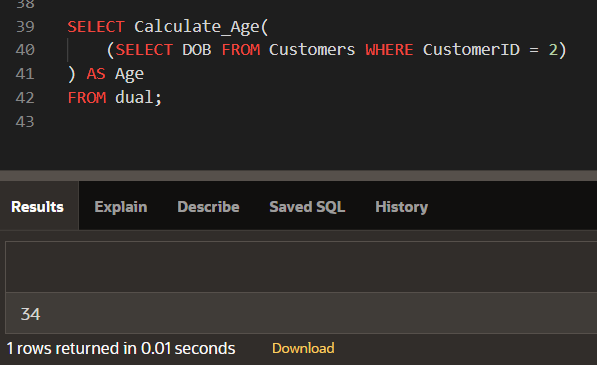
SELECT Calculate\_Age(

(SELECT DOB FROM Customers WHERE CustomerID = 1)

) 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.

**QUERY:**

CREATE OR REPLACE FUNCTION Calculate\_EMI(

loan\_amount IN NUMBER,

annual\_interest\_rate IN NUMBER,

total\_months IN NUMBER

) RETURN NUMBER

IS

monthly\_interest\_rate NUMBER;

emi NUMBER;

BEGIN

monthly\_interest\_rate := annual\_interest\_rate / (12 \* 100);

emi := (loan\_amount \* monthly\_interest\_rate \* POWER((1 + monthly\_interest\_rate), total\_months)) /

(POWER((1 + monthly\_interest\_rate), total\_months) - 1);

RETURN emi;

END Calculate\_EMI;

SELECT ROUND(

Calculate\_EMI(

(SELECT LoanAmount FROM Loans WHERE LoanID = 1),

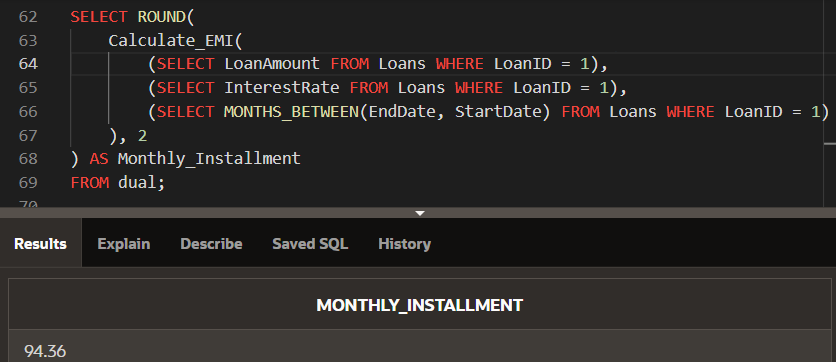
(SELECT InterestRate FROM Loans WHERE LoanID = 1),

(SELECT MONTHS\_BETWEEN(EndDate, StartDate) FROM Loans WHERE LoanID = 1)

), 2

) AS Monthly\_Installment

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.

**QUERY:**

CREATE OR REPLACE FUNCTION Check\_Account\_Balance(

p\_AccountID IN NUMBER,

p\_Amount IN NUMBER

) RETURN BOOLEAN

IS

v\_Balance NUMBER;

BEGIN

SELECT Balance INTO v\_Balance

FROM Accounts

WHERE AccountID = p\_AccountID;

IF v\_Balance >= p\_Amount THEN

RETURN TRUE;

ELSE

RETURN FALSE;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN FALSE;

END Check\_Account\_Balance;

DECLARE

v\_Result BOOLEAN;

BEGIN

v\_Result := Check\_Account\_Balance(1, 500);

DBMS\_OUTPUT.PUT\_LINE('Has sufficient balance: ' || CASE

WHEN v\_Result THEN 'TRUE'

ELSE 'FALSE'

END);

END;

