|  |  |
| --- | --- |
| نام و نام خانوادگی | **پاشا براهیمی** |
| شماره دانشجویی | **٨١٠١٩٩٣٨٥** |
| تاریخ ارسال گزارش | **٢٩/٠٩/١٤٠٢** |

|  |  |  |
| --- | --- | --- |
|  | **به نام خدا**  **دانشگاه تهران**  **دانشکده‌ مهندسی برق و کامپیوتر** |  |
| **درس آزمایشگاه پایگاه‌داده**  **پیش‌گزارش هفتم** | | |

فهرست

[پاسخ ١. معرفی توابع پنجره‌ای 2](#_Toc153744053)

[۱-۱. نمونه ساده استفاده از OVER() 2](#_Toc153744054)

[١-٢. دستور Partition By 3](#_Toc153744055)

[١-٣. تابع ROW\_NUMBER() و استفاده از Order By 4](#_Toc153744056)

[١-٤. تابع RANK() 5](#_Toc153744057)

[١-٥. تابع DENSE\_RANK() 6](#_Toc153744058)

[١-٦. تابع LAG 7](#_Toc153744059)

[١-٧. Frame Clause 8](#_Toc153744060)

[پاسخ ٢. معرفی تریگرها 9](#_Toc153744061)

[٢-١. ایجاد تریگر 9](#_Toc153744062)

[٢-٢. حذف تریگر 11](#_Toc153744063)

# پاسخ ١. معرفی توابع پنجره‌ای

۱-۱. نمونه ساده استفاده از OVER()

در این مثال، می‌خواهیم در کنار تمام سطرها، بالاترین و پایین‌ترین نمره را نیز با نام‌های maximum\_score و minimum\_score نمایش دهیم.

SELECT

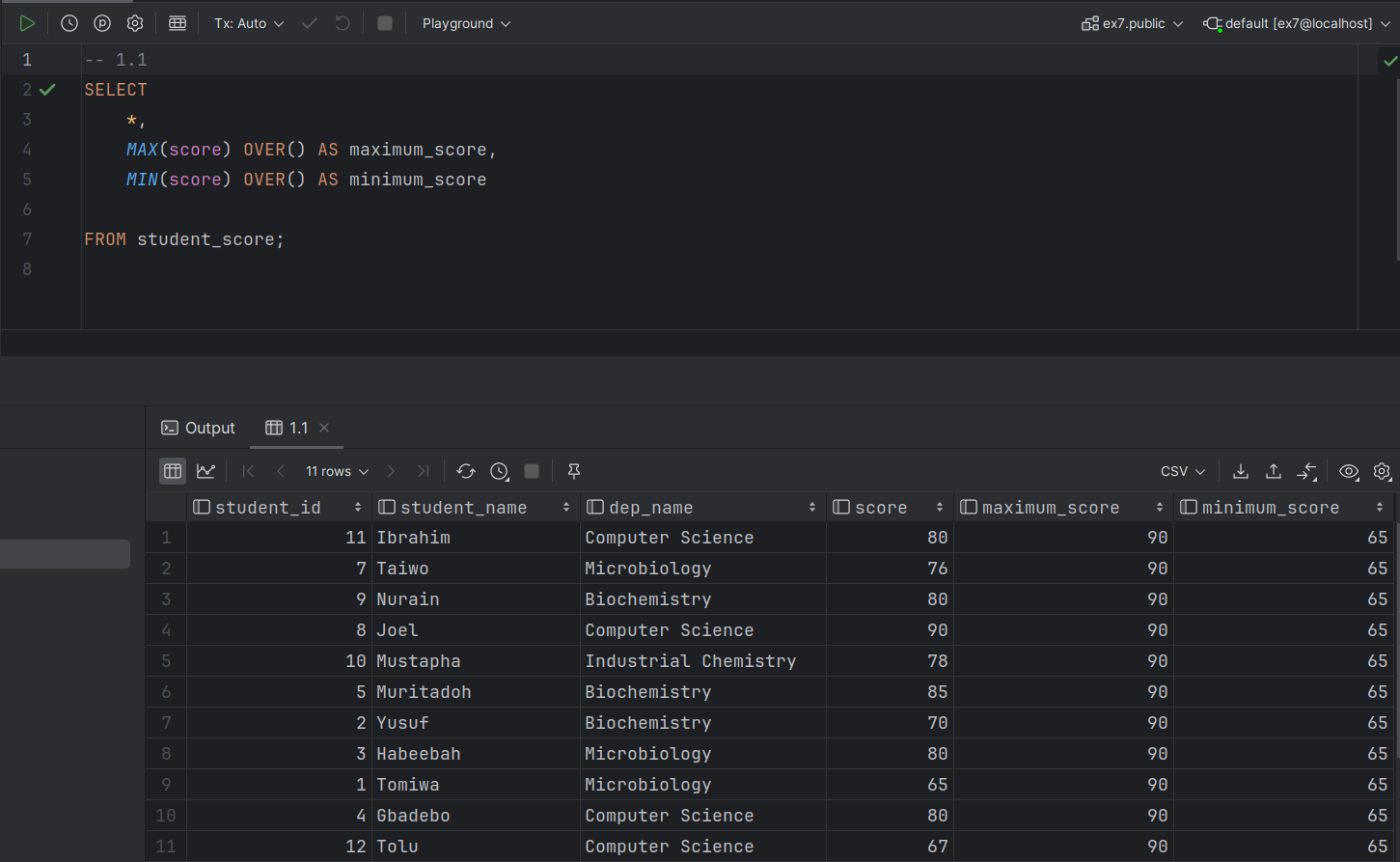
    \*,

    MAX(score) OVER() AS maximum\_score,

    MIN(score) OVER() AS minimum\_score

FROM student\_score;

خروجی به صورت زیر است:



همانطور که مشاهده می‌شود، در هر سطر دو ستون maximum\_score با مقدار 90 و minimum\_score با مقدار 65 اضافه شده است. این دستور معادل با دستور زیر (بدون استفاده از توابع پنجره‌ای) است:

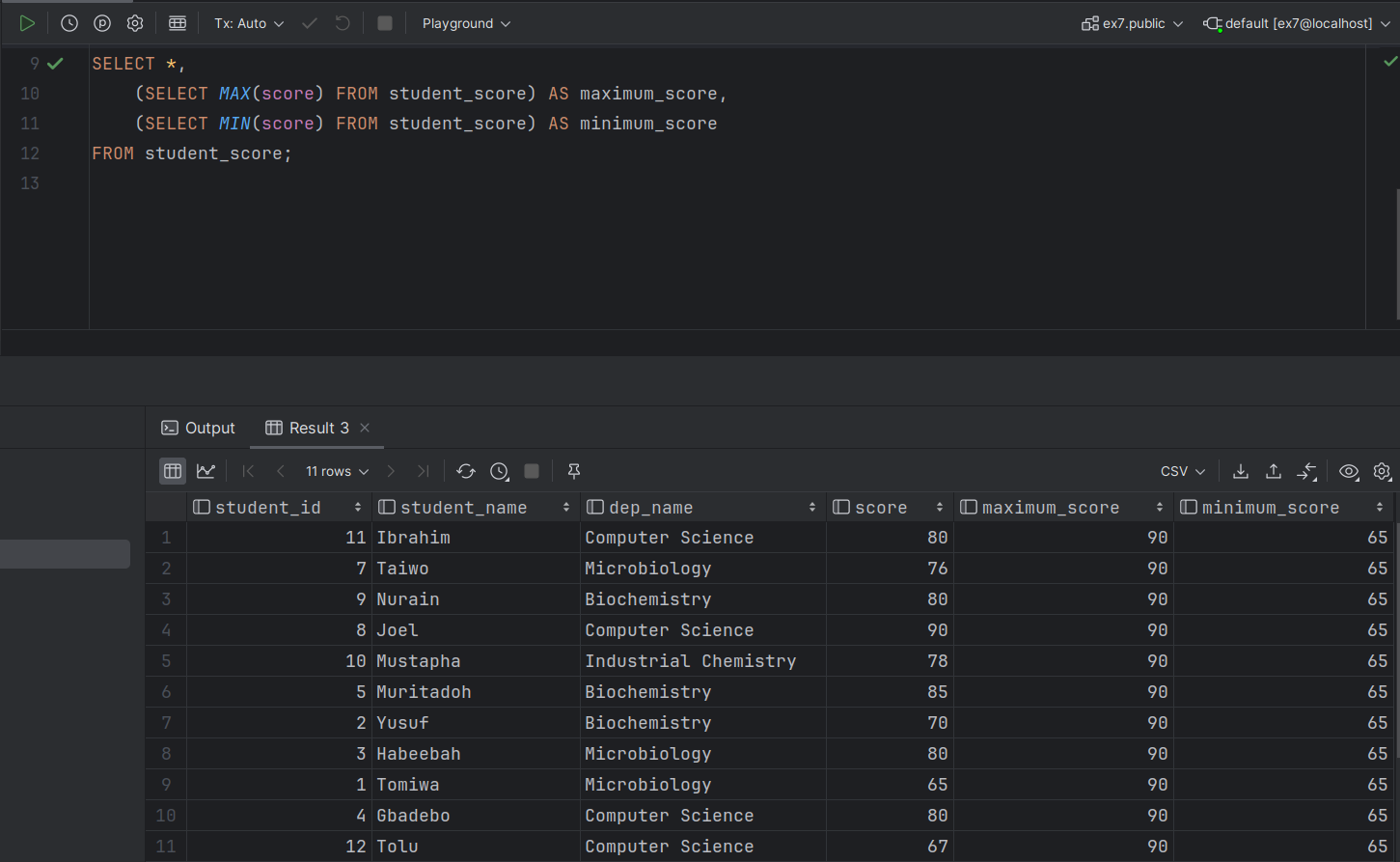
SELECT \*,

    (SELECT MAX(score) FROM student\_score) AS maximum\_score,

    (SELECT MIN(score) FROM student\_score) AS minimum\_score

FROM student\_score;

خروجی این دستور نیز مشابه با دستور قبلی و به صورت زیر است:



## ١-٢. دستور Partition By

در دستور زیر، به ازای هر دپارتمان، ماکسیمم و میانگین score را بدست آورده و در دو ستون جدید در هر سطر نمایش می‌دهیم. همانطور که در نتیجه می‌شود دید، این مقادیر در یک دپارتمان ثابت‌اند.

SELECT

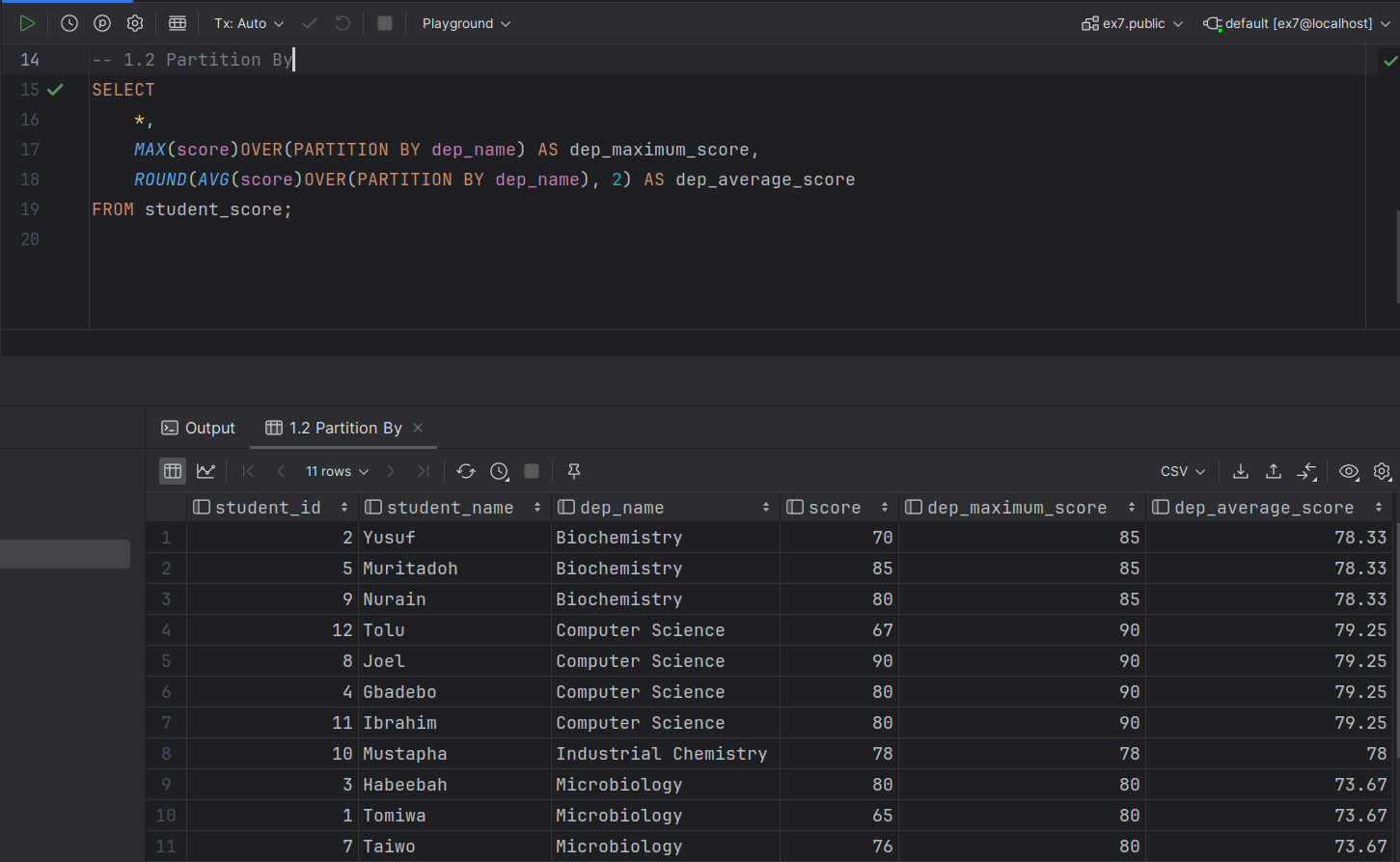
    \*,

    MAX(score)OVER(PARTITION BY dep\_name) AS dep\_maximum\_score,

    ROUND(AVG(score)OVER(PARTITION BY dep\_name), 2) AS dep\_average\_score

FROM student\_score;

نتیجه به صورت زیر است:



## ١-٣. تابع ROW\_NUMBER() و استفاده از Order By

در دستور زیر، ابتدا بر حسب نام دانشجوها، جدول را مرتب کرده و سپس به هر یک از سطرها یک شماره یکتا اختصاص داده می‌شود که شماره آن سطر را مشخص می‌کند. برای مثال سطر اول شماره 1 و سطر دوم شماره 2 خواهد گرفت.

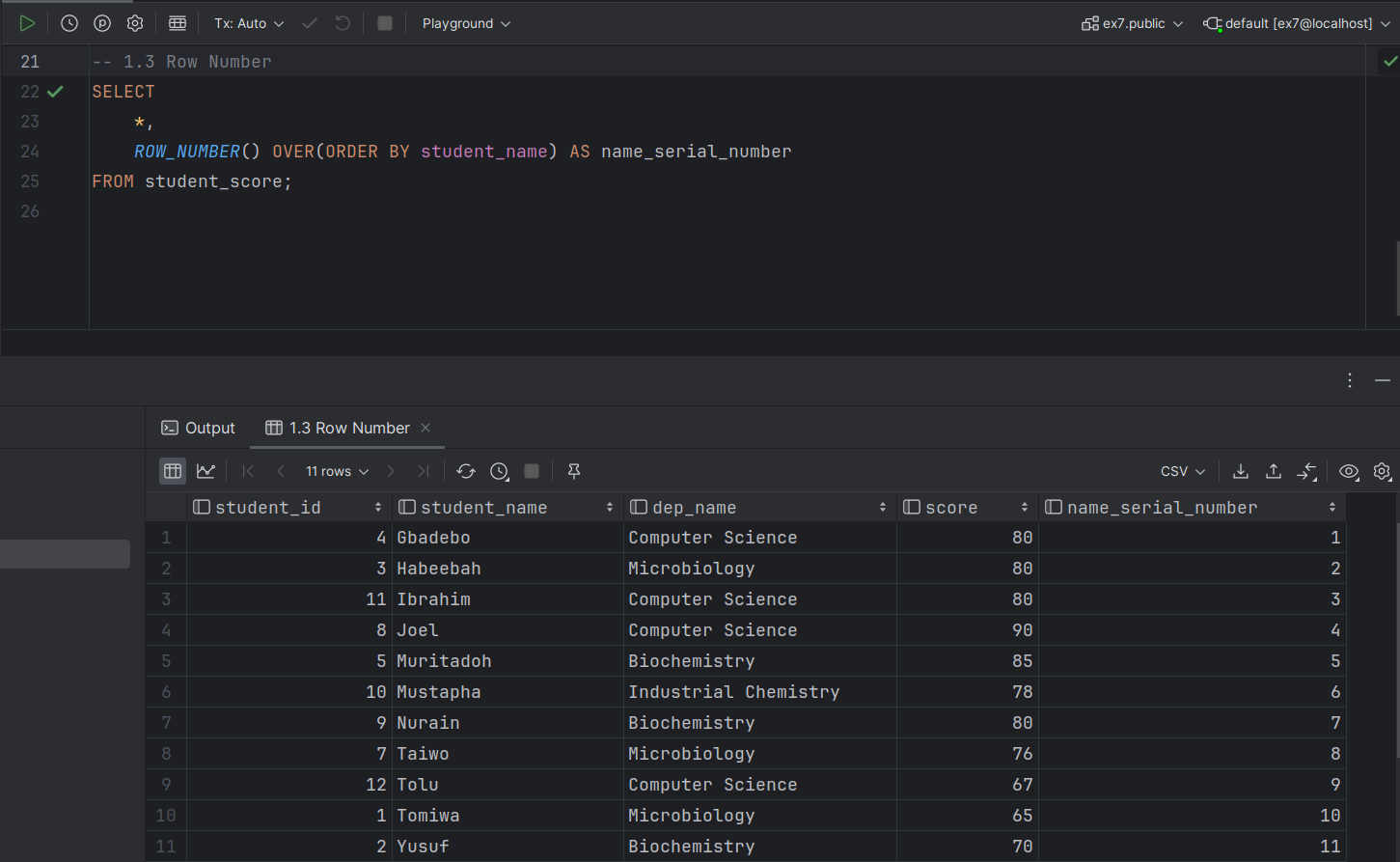
SELECT

    \*,

    ROW\_NUMBER() OVER(ORDER BY student\_name) AS name\_serial\_number

FROM student\_score;

خروجی به صورت زیر است:



همانطور که دیده می‌شود، شماره هر سطر در ستون name\_serial\_number داده شده است.

## ١-٤. تابع RANK()

در کوئری زیر، برای هر دپارتمان، ابتدا بر اساس score و به صورت نزولی مرتب کرده و بر اساس همان score، یک rank به هر سطر اختصاص می‌دهد. بدیهی‌ست که دو سطر که دپارتمان و score مشابه دارند، rank یکسان هم خواهد داشت. همچنین لازم به ذکر است که این تابع، زمانی که rank یکسان برای دو سطر در نظر بگیرد، یک gap پس از آن قرار می‌دهد. برای مثال، اگر در یک دپارتمان، سه نفر score برابر با 80 داشته باشند، هر سه rank یکسان (برای مثال در اینجا برابر با 2) دارند و rank بعدی به جای 3 برابر با 5 خواهد بود.

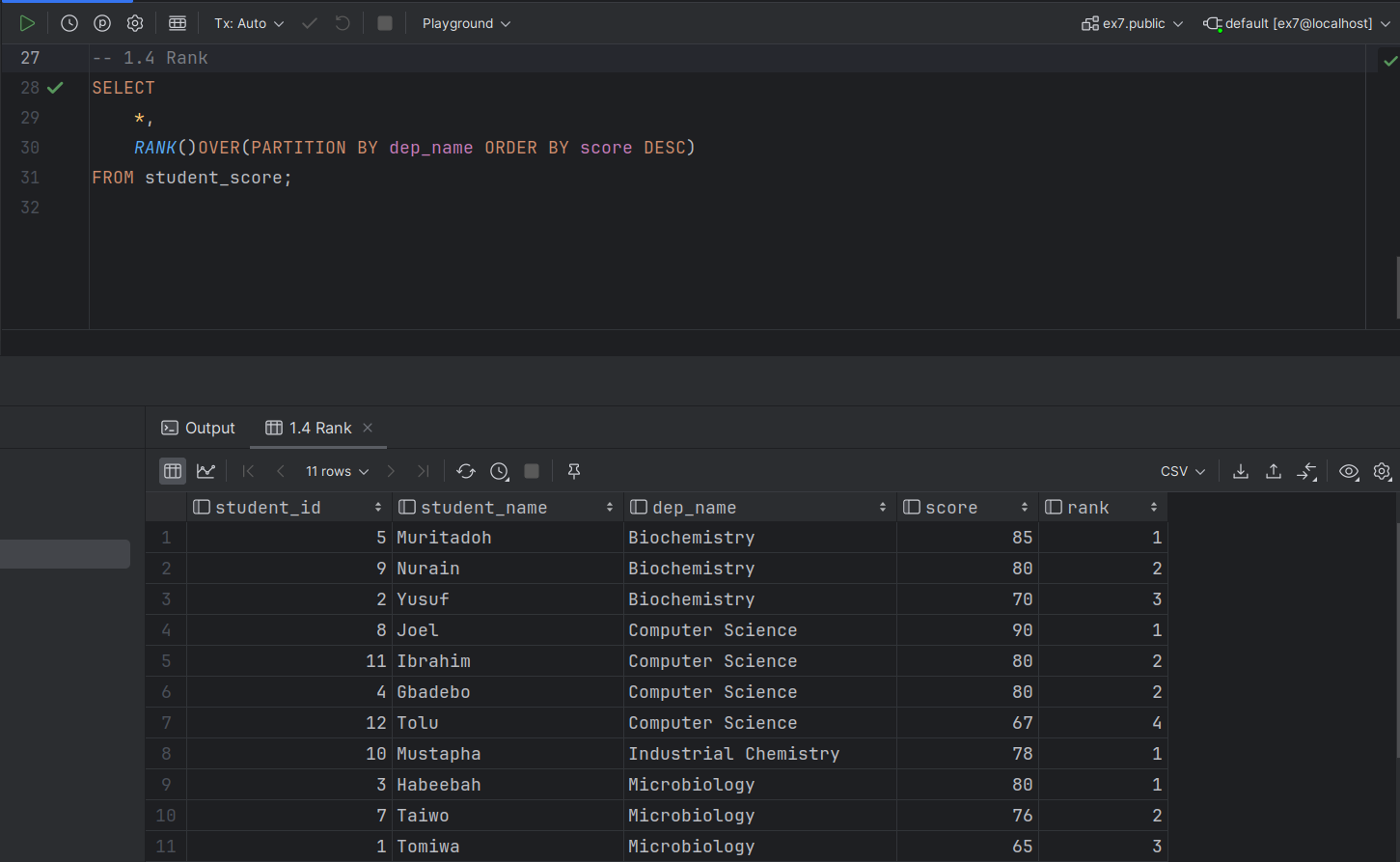
SELECT

    \*,

    RANK()OVER(PARTITION BY dep\_name ORDER BY score DESC)

FROM student\_score;

خروجی به صورت زیر است:



همانطور که دیده می‌شود، به ازای هر دپارتمان، دانشجویان بر حسب نمره‌شان یک rank گرفته‌اند و دانشجویان با score برابر، rank یکسان هم دارند.

## ١-٥. تابع DENSE\_RANK()

این تابع همانند تابع RANK است با این تفاوت که در صورت برابری rank چند سطر، gap بعد از آن را نخواهیم داشت و rank بعدی دقیقا یک واحد بیشتر از rank قبلی خواهد بود.

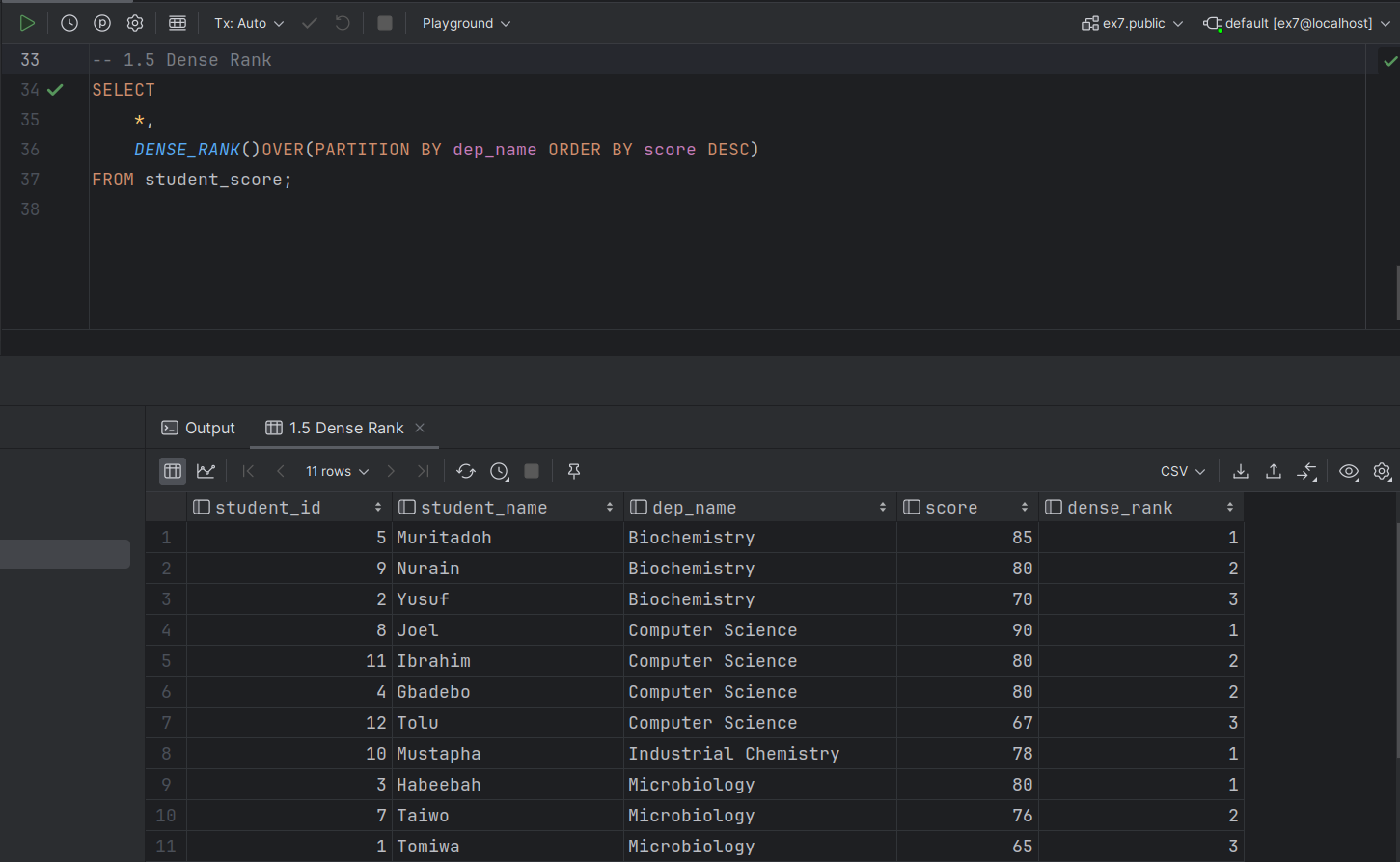
SELECT

    \*,

    DENSE\_RANK()OVER(PARTITION BY dep\_name ORDER BY score DESC)

FROM student\_score;

خروجی به صورت زیر است:



## ١-٦. تابع LAG

در این تابع، می‌خواهیم مقداری از رکوردهای قبلی را در رکورد فعلی نمایش دهیم. این تابع به صورت دیفالت مقدار سطر قبلی را بازمی‌گرداند. برای مثال در تابع زیر، در کنار score هر دانشجو، score سطر قبلی را نیز قرار می‌دهیم. با توجه به اینکه از Partition By استفاده کردیم، این مورد در خصوص هر دپارتمان به صورت جداگانه عمل می‌کند.

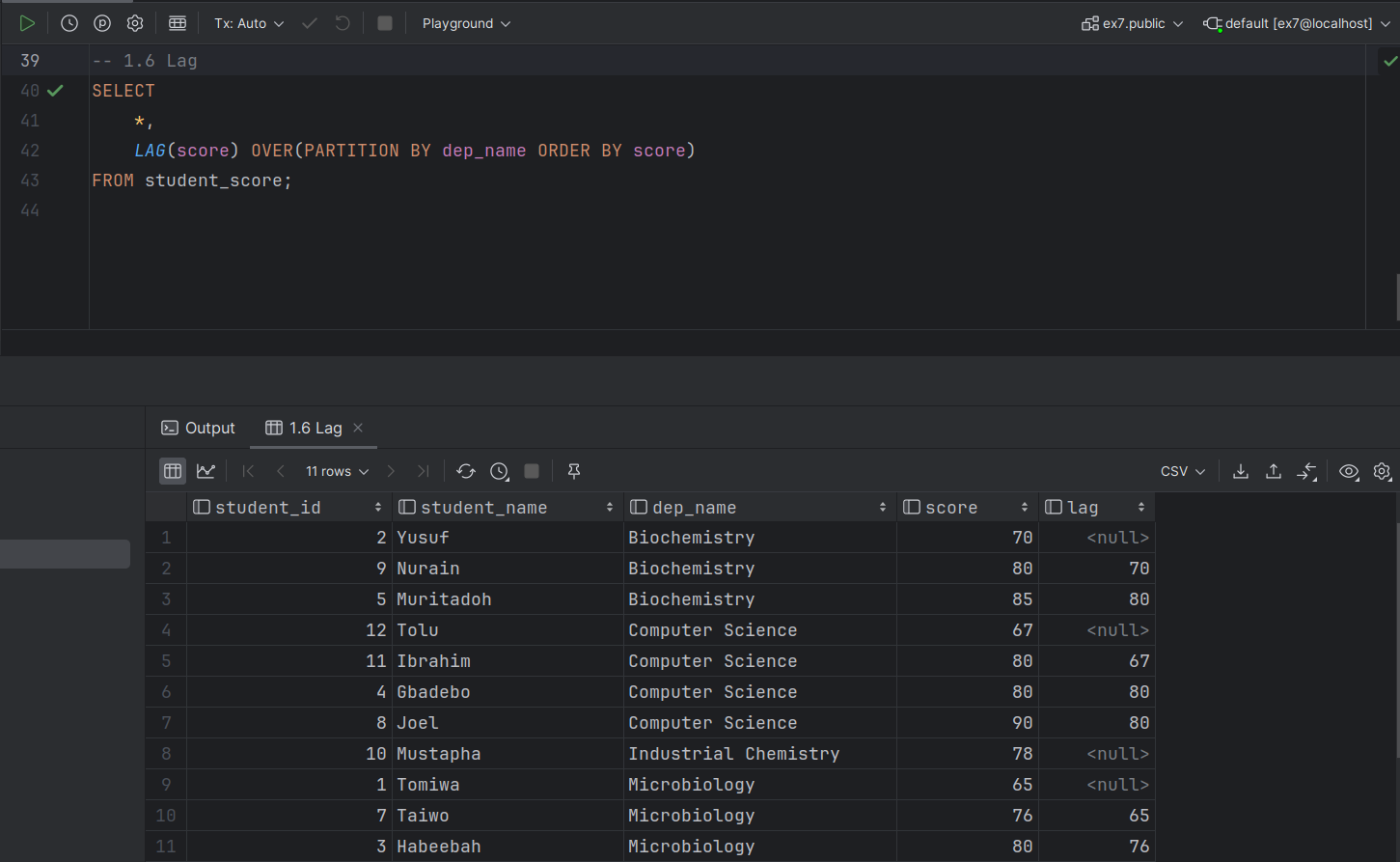
SELECT

    \*,

    LAG(score) OVER(PARTITION BY dep\_name ORDER BY score)

FROM student\_score;

نتیجه به صورت زیر است:



همانطور که دیده می‌شود، اولین سطر مربوط به هر دپارتمان (چون سطر قبلی ندارد)، مقدار lag برابر با null است.

## ١-٧. Frame Clause

در این تابع، می‌توانیم داده‌هایی از سطرهای قبل یا بعد از سطر فعلی را بدست آوریم. برای مثال در این کوئری می‌خواهیم به ازای هر دپارتمان، cumulative sum را در نمرات بدست آوریم. در واقع یک ستون در هر سطر داشته باشیم که مجموع نمره تمام سطرهای قبلی را به همراه سطر فعلی داشته باشد. در این بخش UNBOUND PRECDING به معنای سطرهای قبلی و CURRENT ROW به معنای سطر فعلی است.

-- 1.7 Frame Clause

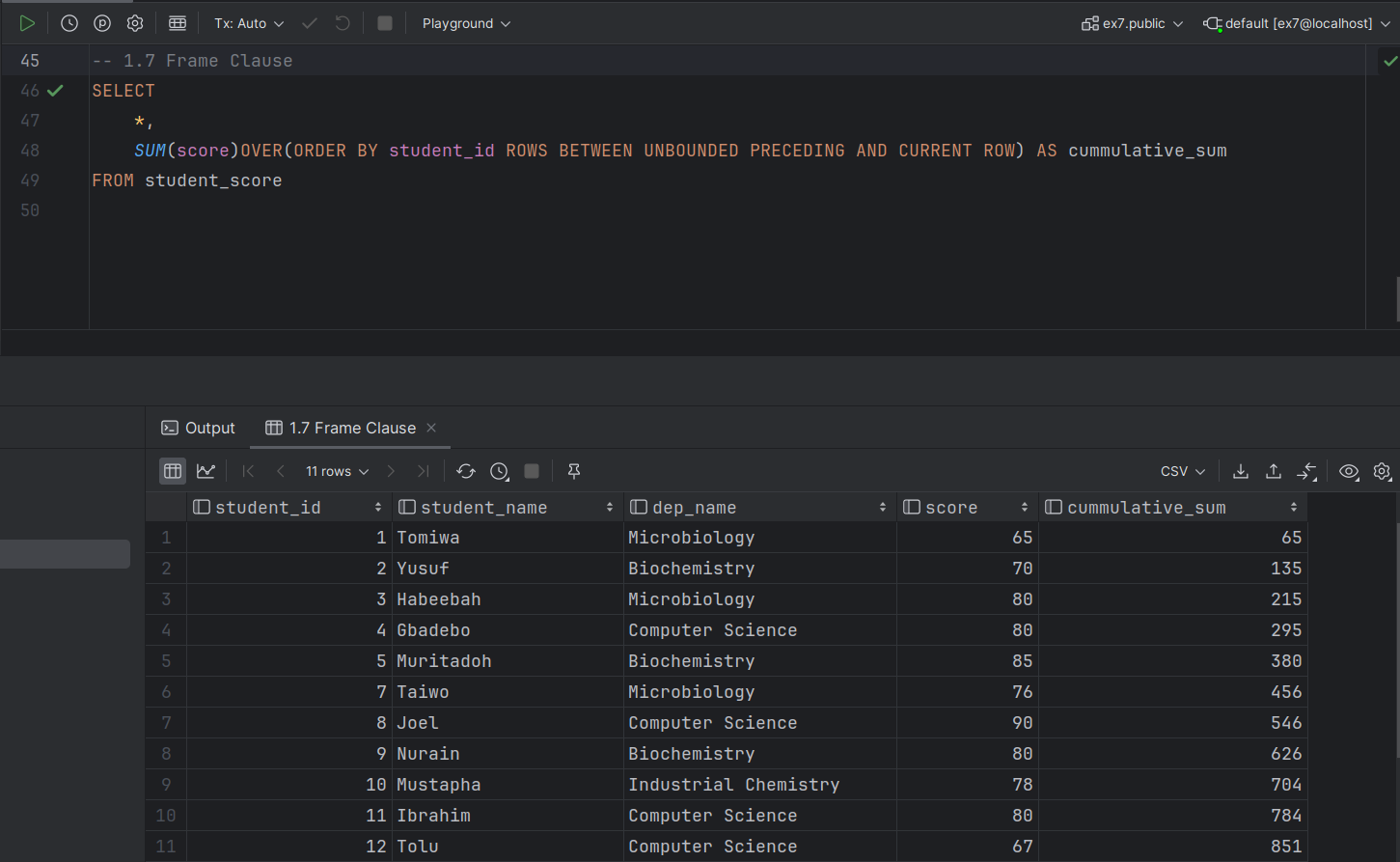
SELECT

    \*,

    SUM(score)OVER(ORDER BY student\_id ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS cummulative\_sum

FROM student\_score;

نتیجه به صورت زیر است:



# پاسخ ٢. معرفی تریگرها

## ٢-١. ایجاد تریگر

در این بخش چون سیتکس قرار داده شده در سایت برای MySQL است، آن را طوری تغییر دادم که در postgres اجرا شود. در این حالت، ابتدا یک تابع hash\_password داریم که در تریگر، به ازای هر یوزر اضافه شده، این تابع کال می‌شود. ابتدا جدول جدید را می‌سازیم:

CREATE TABLE

    users (

        fullname VARCHAR(120),

        email VARCHAR(120),

        username VARCHAR(30),

        password VARCHAR(60)

    );

سپس تریگر را به صورت ذکر شده ایجاد می‌کنیم:

CREATE OR REPLACE FUNCTION hash\_password()

RETURNS TRIGGER AS $$

BEGIN

  NEW.password = MD5(NEW.password);

  RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER password\_hasher

BEFORE INSERT ON users

FOR EACH ROW

EXECUTE FUNCTION hash\_password();

و در نهایت یوزر را اضافه می‌کنیم:

INSERT INTO

    users

VALUES

    (

        'idris babu',

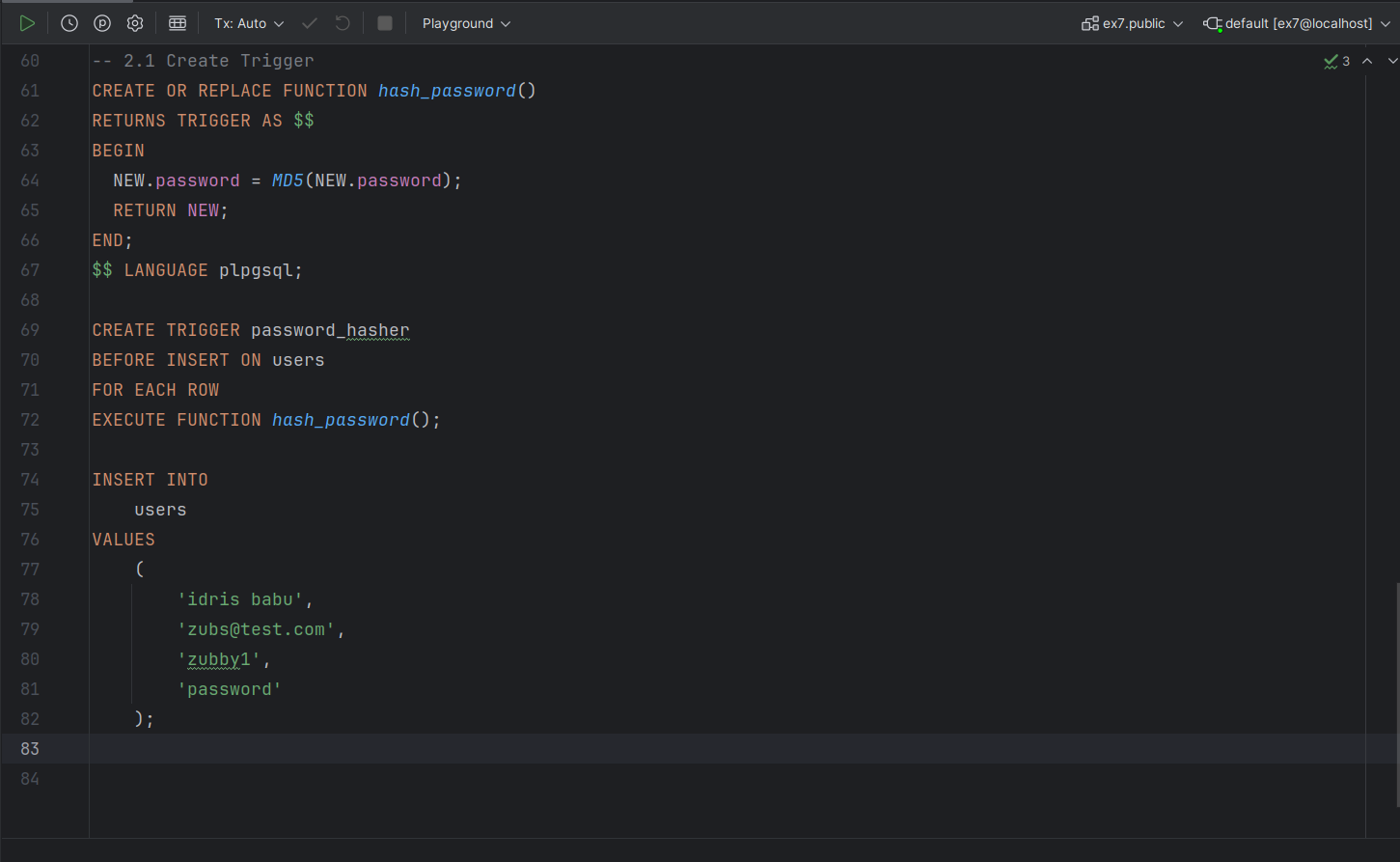
        'zubs@test.com',

        'zubby1',

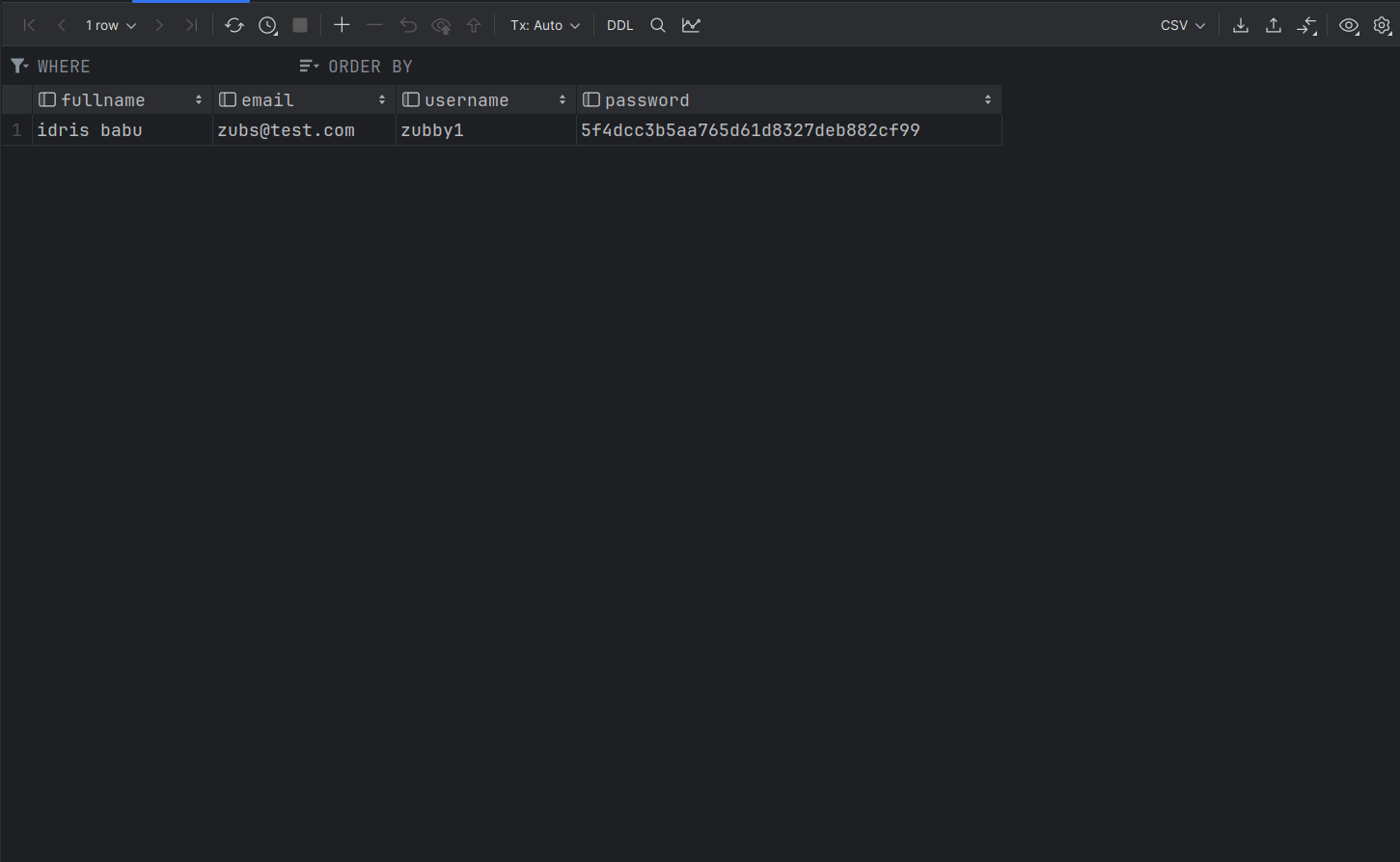
        'password'

    );

دستورهای اجرا شده به صورت کلی در تصویر زیر واضح است:



پس از اضافه کردن یوزر، جدول را مشاهده کرده و فیلد password را بررسی می‌کنیم:



می‌بینیم که به جای عبارت password، هش آن در جدول قرار گرفته است.

## ٢-٢. حذف تریگر

در این بخش، تریگر ایجاد شده در بخش قبل را حذف می‌کنیم:

DROP TRIGGER

    IF EXISTS password\_hasher

    ON users;

و سپس یک یوزر جدید اضافه می‌کنیم:

INSERT INTO

    users

VALUES

    (

        'idris babu',

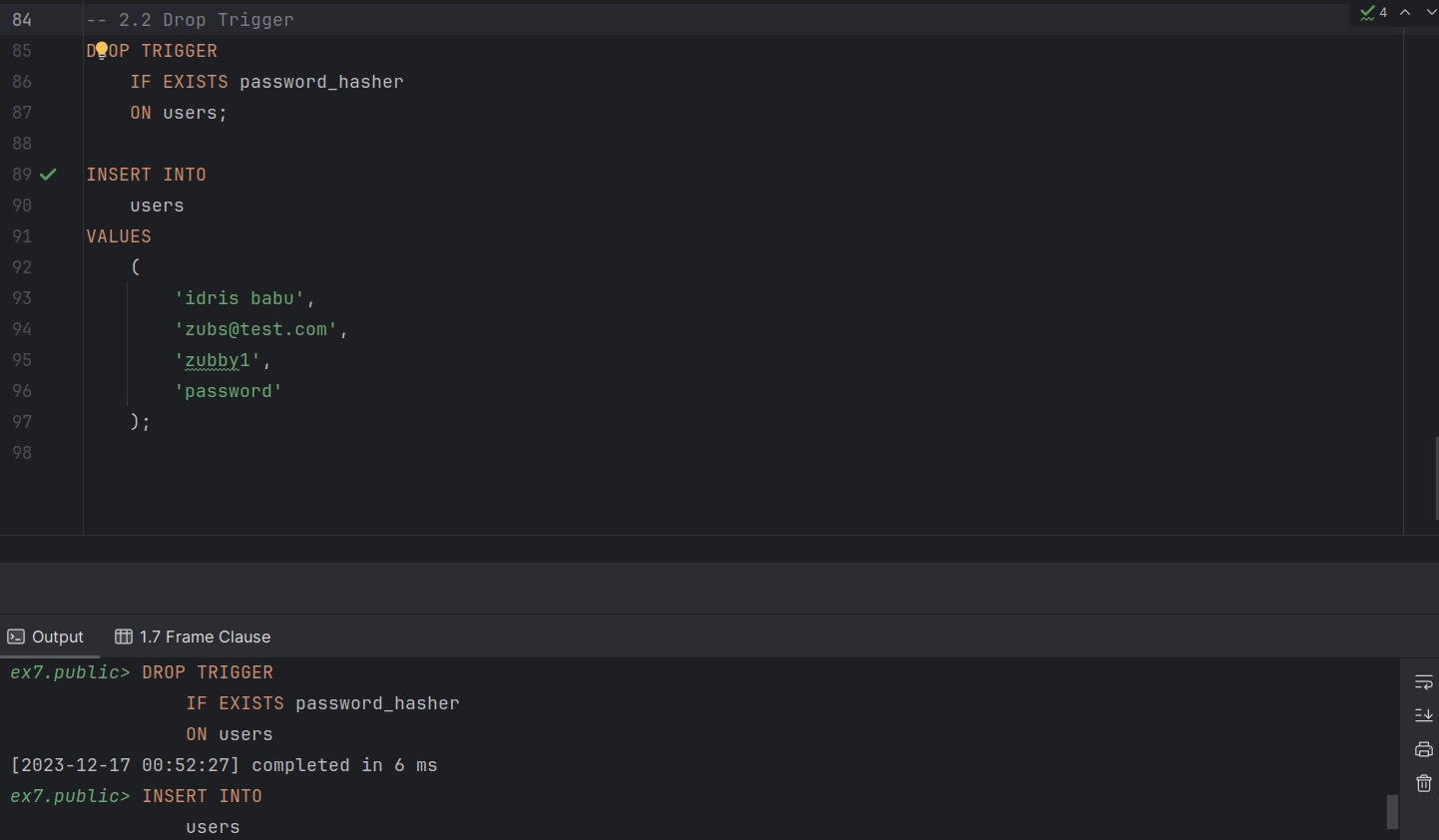
        'zubs@test.com',

        'zubby1',

        'password'

    );

خروجی به صورت زیر است:



حال اگر جدول users را بررسی کنیم، می‌بینیم که مقدار password در جدول قرار گرفته و دیگر هش آن به جای خود پسورد در جدول نیست:

