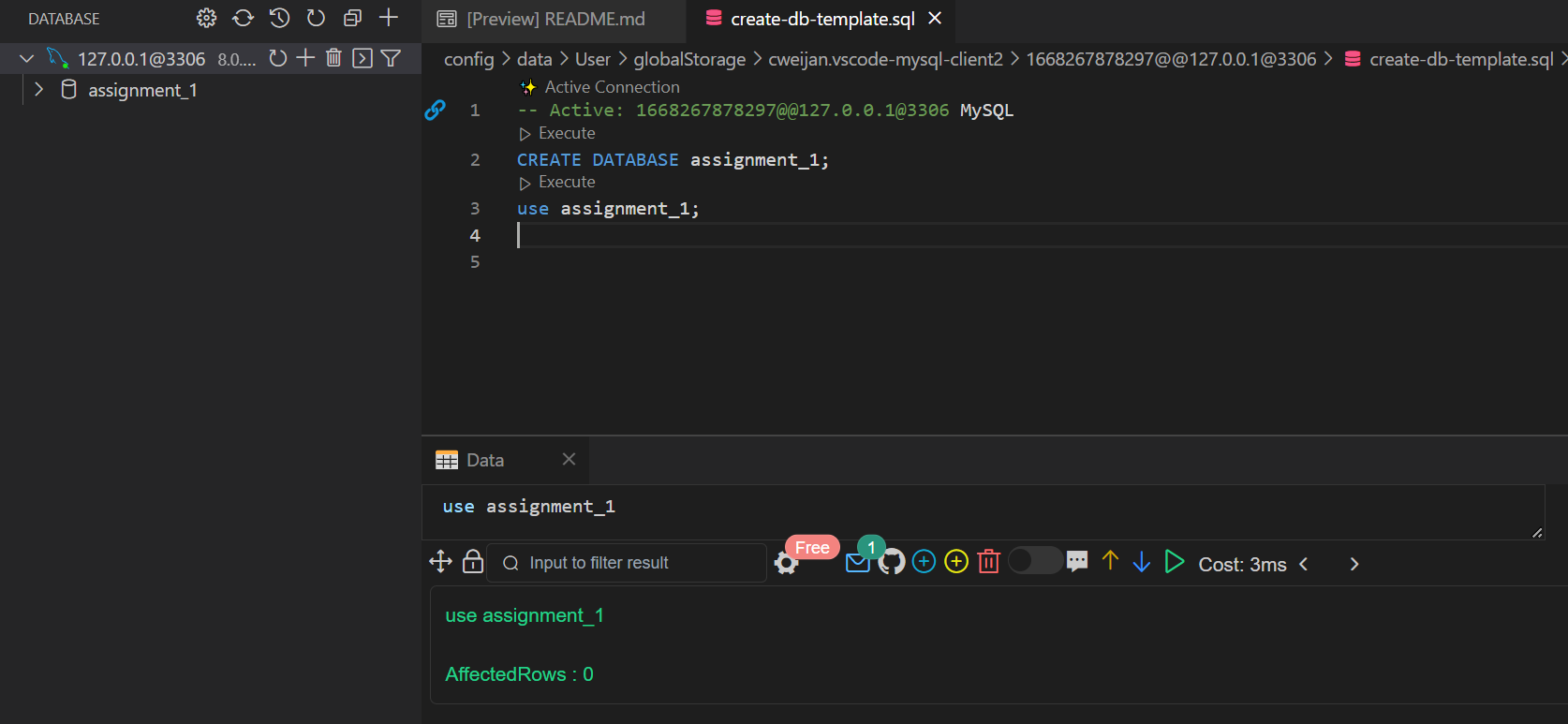
**SQL ASSIGNMENT 1:**

Create Database => assignment\_1

**CREATE DATABASE assignment\_1;**

**use assignment\_1;**



**1. Write a query to display the columns in a specific order, such as order date,**

**salesman ID, order number, and purchase amount for all orders.**

**ord\_no purch\_amt ord\_date customer\_id salesman\_id**

---------- ---------- ---------- ----------- ----------- ----------- ----------- ----

70001 150.5 2012-10-05 3005 5002

70009 270.65 2012-09-10 3001 5005

70002 65.26 2012-10-05 3002 5001

70004 110.5 2012-08-17 3009 5003

70007 948.5 2012-09-10 3005 5002

70005 2400.6 2012-07-27 3007 5001

70008 5760 2012-09-10 3002 5001

70010 1983.43 2012-10-10 3004 5006

70003 2480.4 2012-10-10 3009 5003

70012 250.45 2012-06-27 3008 5002

70011 75.29 2012-08-17 3003 5007

70013 3045.6 2012-04-25 3002 5001

**create table order\_data(**

**ord\_no int,**

**purch\_amt int,**

**ord\_date DATE,**

**customer\_id int,**

**salesman\_id int**

**);**

**insert into order\_data values**

**(70001,150.5,'2012-10-05',3005,5002),**

**(70009,270.65,'2012-09-10',3001,5005),**

**(70002,65.26,'2012-10-05',3002,5001),**

**(70004,110.5,'2012-08-17',3009,5003),**

**(70007,948.5,'2012-09-10',3005,5002),**

**(70005,2400.6,'2012-07-27', 3007,5001),**

**(70008,5760,'2012-09-10',3002,5001),**

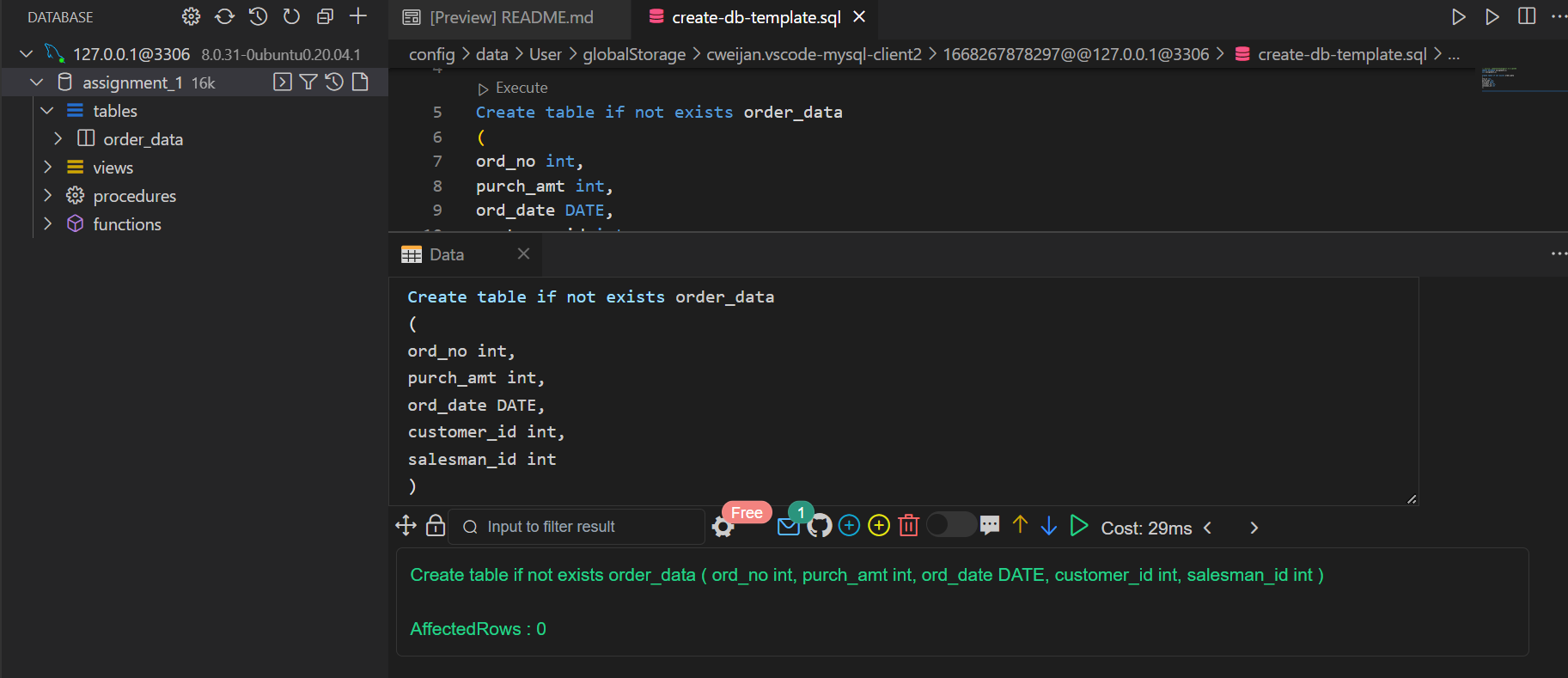
**(70010,1983.43,'2012-10-10',3004,5006),**

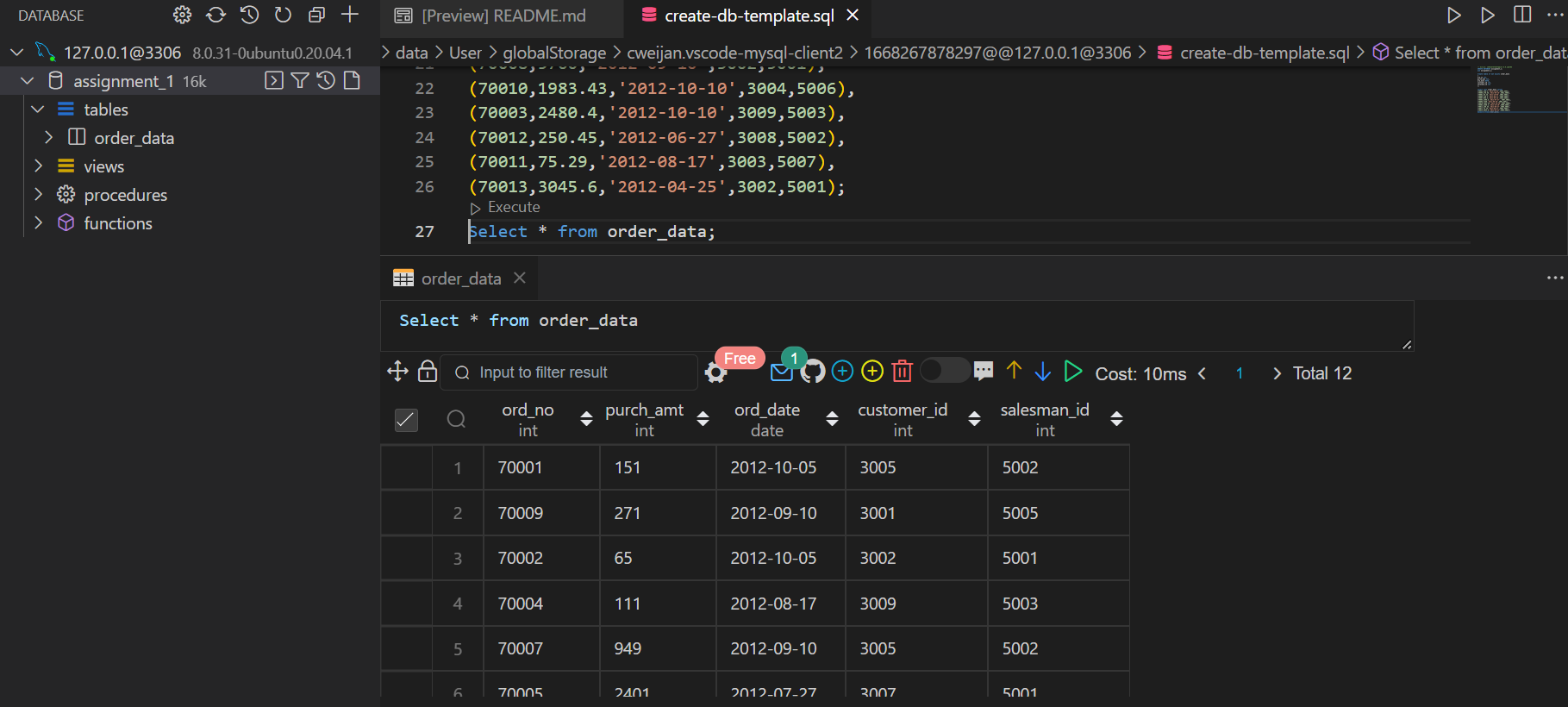
**(70003,2480.4,'2012-10-10',3009,5003),**

**(70012,250.45,'2012-06-27',3008,5002),**

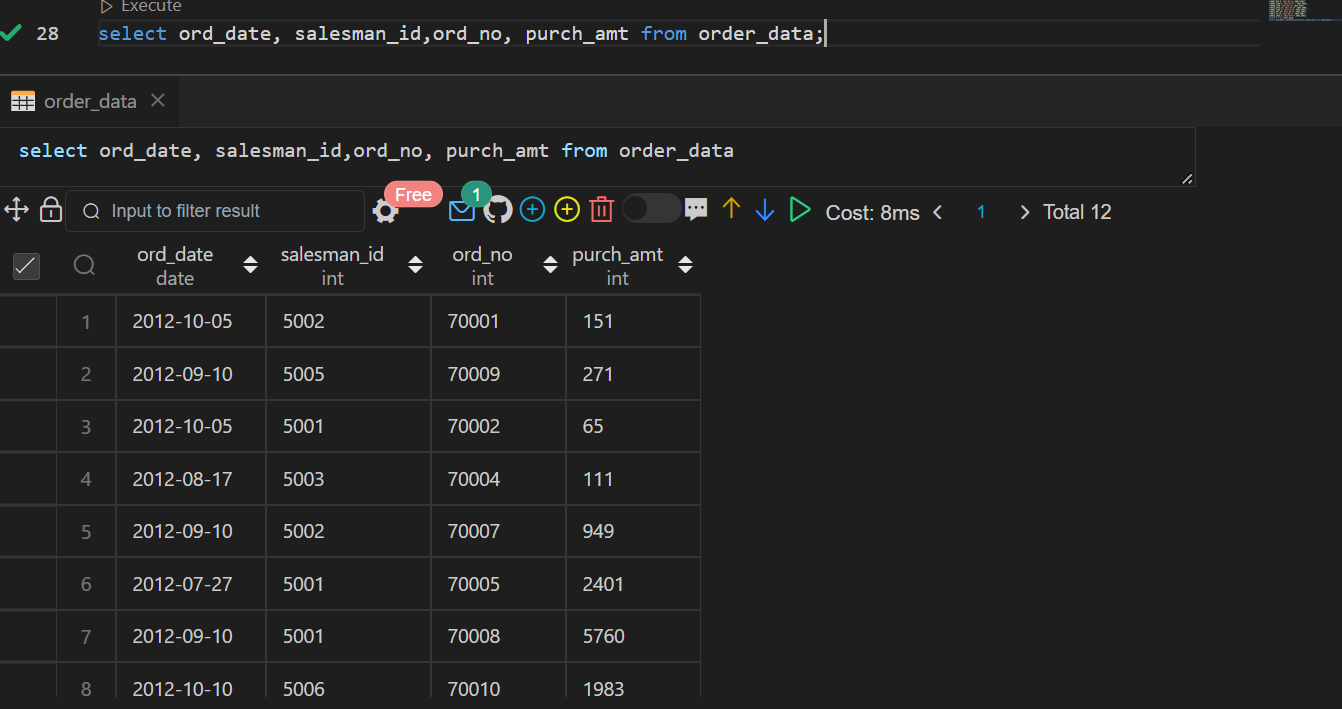
**(70011,75.29,'2012-08-17',3003,5007),**

**(70013,3045.6,'2012-04-25',3002,5001);**





**select ord\_date, salesman\_id,ord\_no, purch\_amt from order\_data;**



**2. From the following table, write a SQL query to locate salespeople who live in the city**

**of 'Paris'. Return salesperson's name, city.**

**salesman\_id | name | city | commission**

-------------+------------+----------+------------

5001 | James Hoog | New York | 0.15

5002 | Nail Knite | Paris | 0.13

5005 | Pit Alex | London | 0.11

5006 | Mc Lyon | Paris | 0.14

5007 | Paul Adam | Rome | 0.13

5003 | Lauson Hen | San Jose | 0.12

**create table sales(**

**salesman\_id int primary key,**

**name varchar(15),**

**city varchar(15),**

**commission float**

**);**

**insert into sales values**

**(5001,'James Hoog','New York',0.15),**

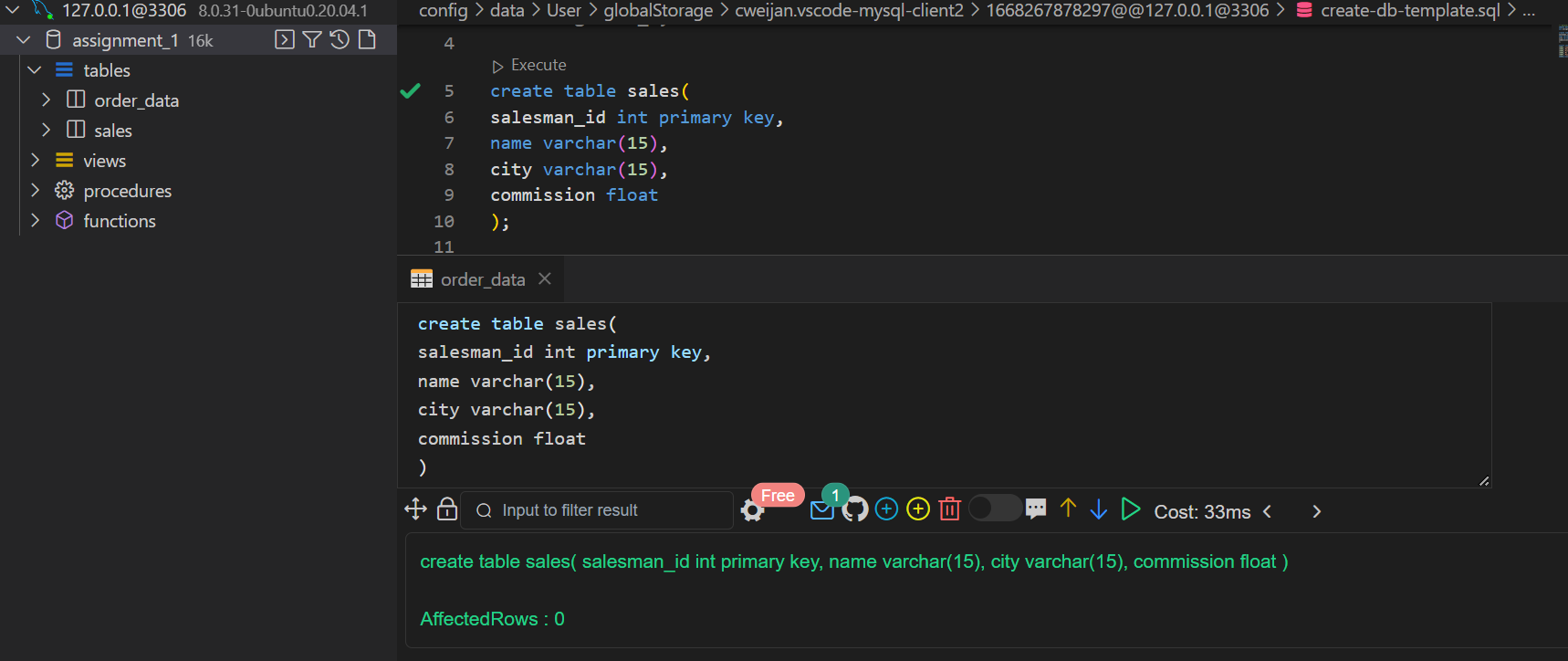
**(5002,'Nail Knite','Paris',0.13),**

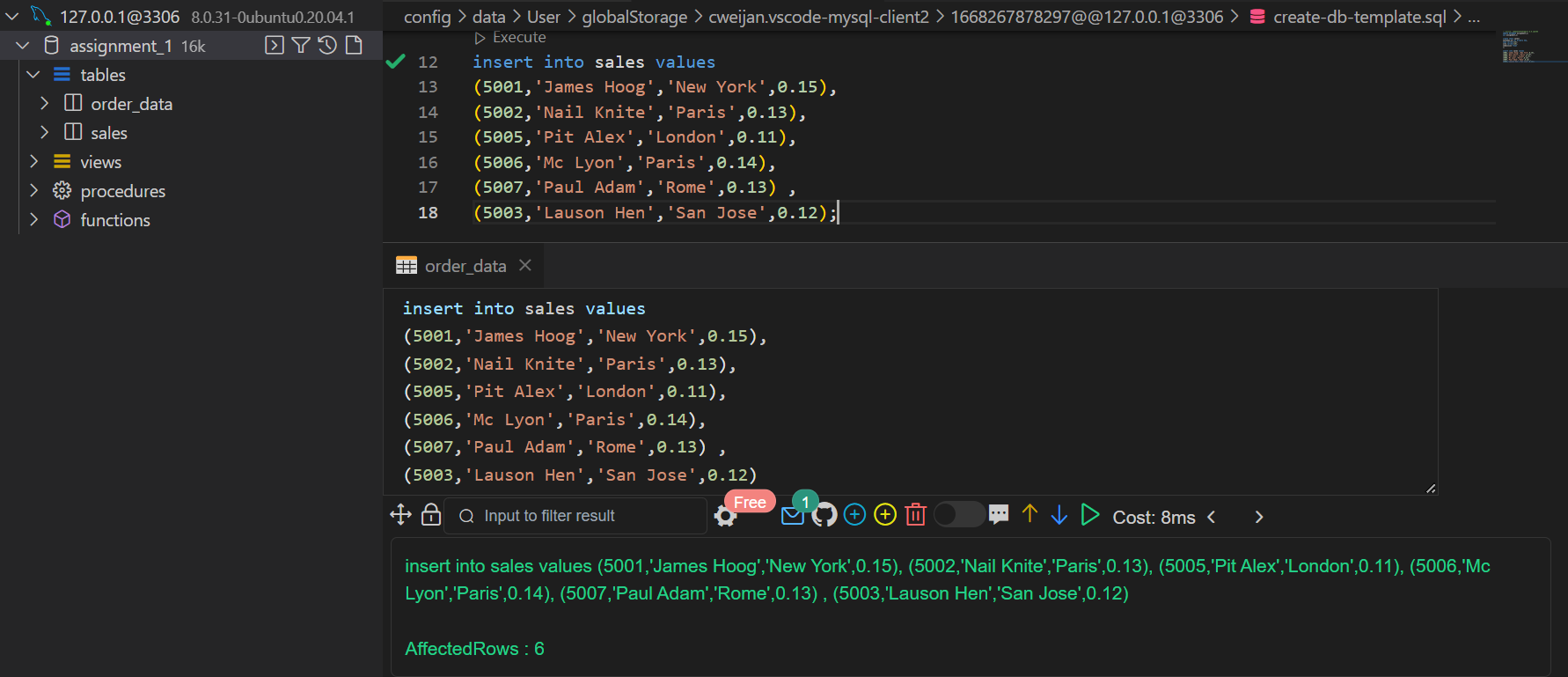
**(5005,'Pit Alex','London',0.11),**

**(5006,'Mc Lyon','Paris',0.14),**

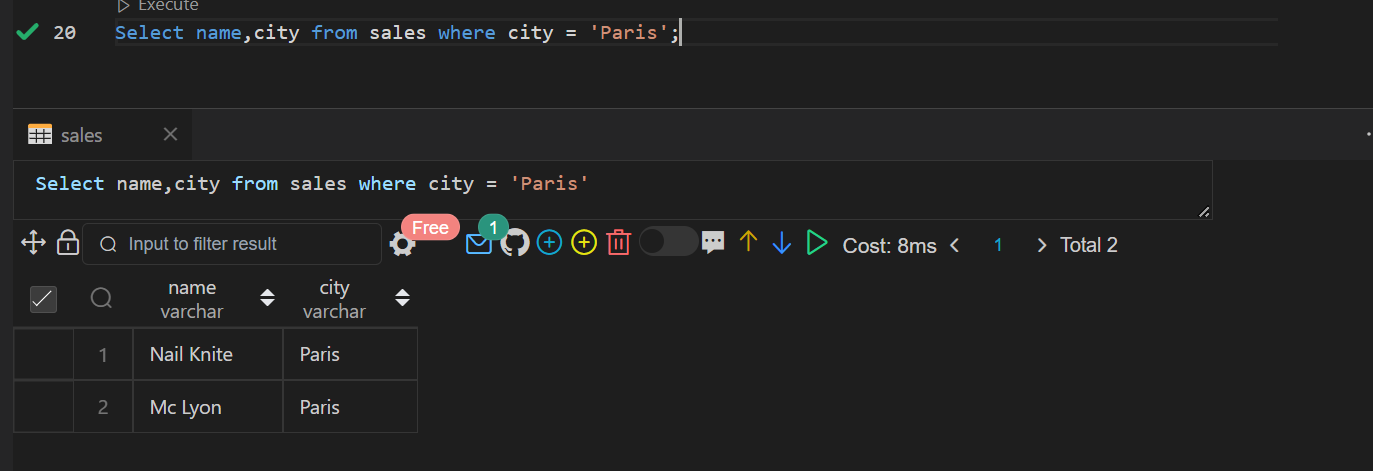
**(5007,'Paul Adam','Rome',0.13) ,**

**(5003,'Lauson Hen','San Jose',0.12);**





**Select name,city from sales where city = 'Paris';**



**3. From the following table, write a SQL query to select a range of products whose**

**price is in the range Rs.200 to Rs.600. Begin and end values are included. Return**

**pro\_id, pro\_name, pro\_price, and pro\_com.**

**PRO\_ID PRO\_NAME PRO\_PRICE PRO\_COM**

------- ------------------------- ------------------- --------------

101 Motherboard 3200.00 15

102 Keyboard 450.00 16

103 ZIP drive 250.00 14

104 Speaker 550.00 16

105 Monitor 5000.00 11

106 DVD drive 900.00 12

107 CD drive 800.00 12

108 Printer 2600.00 13

109 Refill cartridge 350.00 13

110 Mouse 250.00 12

**create table if not exists products**

**(**

**prod\_id int primary key,**

**prod\_name varchar(30),**

**prod\_price float,**

**prod\_com int);**

**insert into products values**

**(101, 'Motherboard', 3200.00, 15),**

**(102, 'Keyboard', 450.00, 16),**

**(103, 'ZIP drive', 250.00, 14),**

**(104, 'Speaker', 550.00, 16),**

**(105, 'Monitor', 5000.00, 11),**

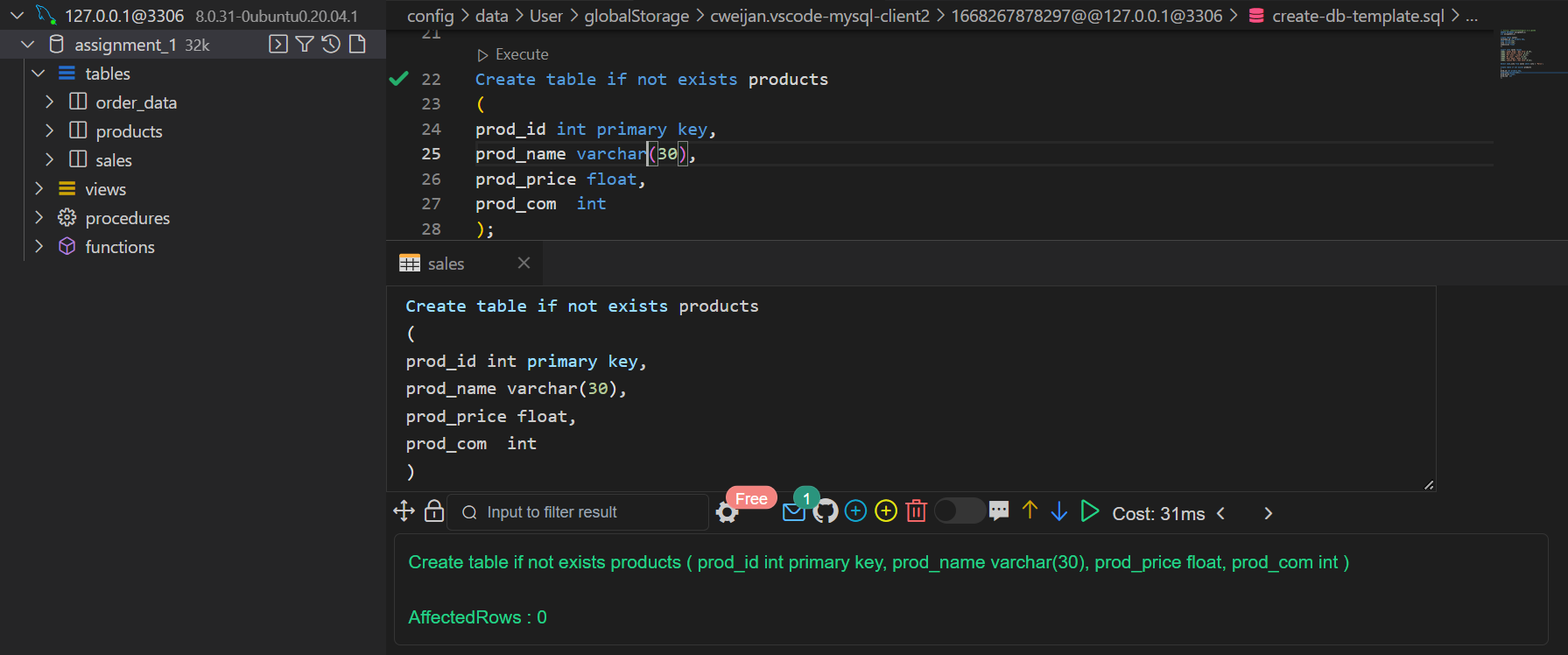
**(106, 'DVD drive', 900.00, 12),**

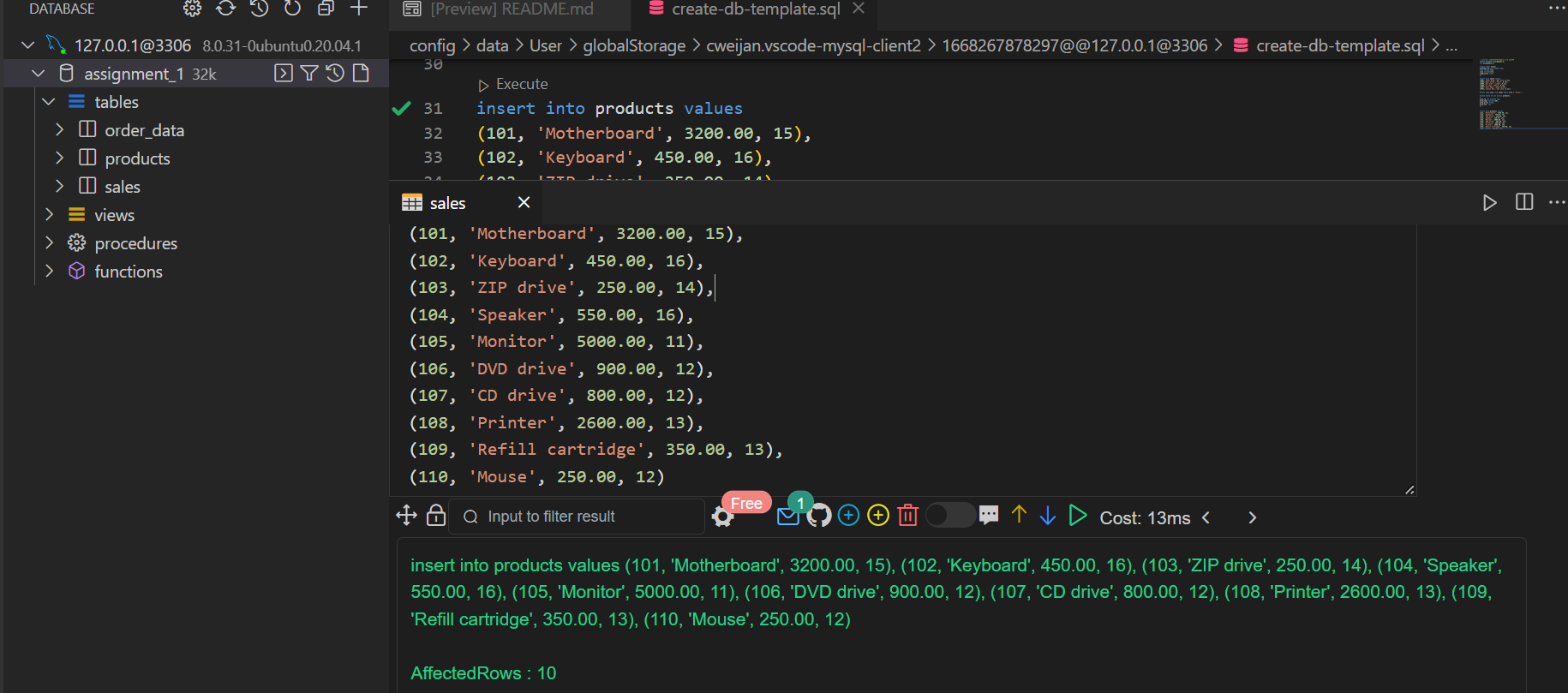
**(107, 'CD drive', 800.00, 12),**

**(108, 'Printer', 2600.00, 13),**

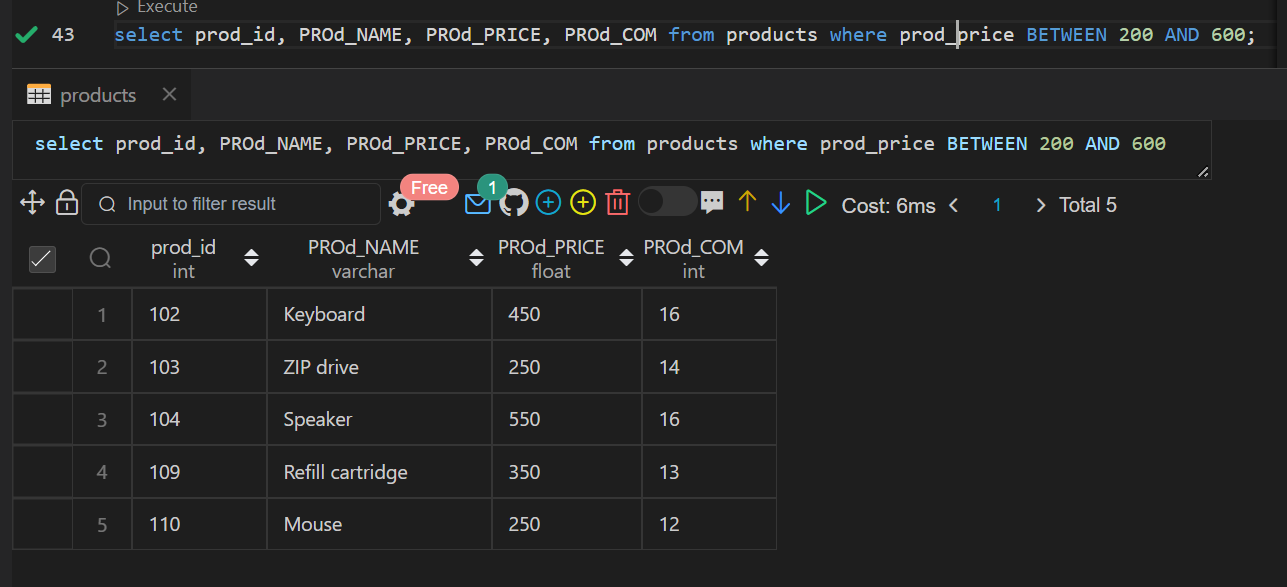
**(109, 'Refill cartridge', 350.00, 13),**

**(110, 'Mouse', 250.00, 12);**





**select prod\_id, PROd\_NAME, PROd\_PRICE, PROd\_COM from products where prod\_price BETWEEN 200 AND 600;**



**4. From the following table, write a SQL query to find the items whose prices are higher than or equal to $550. Order the result by product price in descending, then product name in ascending. Return pro\_name and pro\_price.**

**PRO\_ID PRO\_NAME PRO\_PRICE PRO\_COM**

------- ------------------------- -------------- ----------

101 Motherboard 3200.00 15

102 Keyboard 450.00 16

103 ZIP drive 250.00 14

104 Speaker 550.00 16

105 Monitor 5000.00 11

106 DVD drive 900.00 12

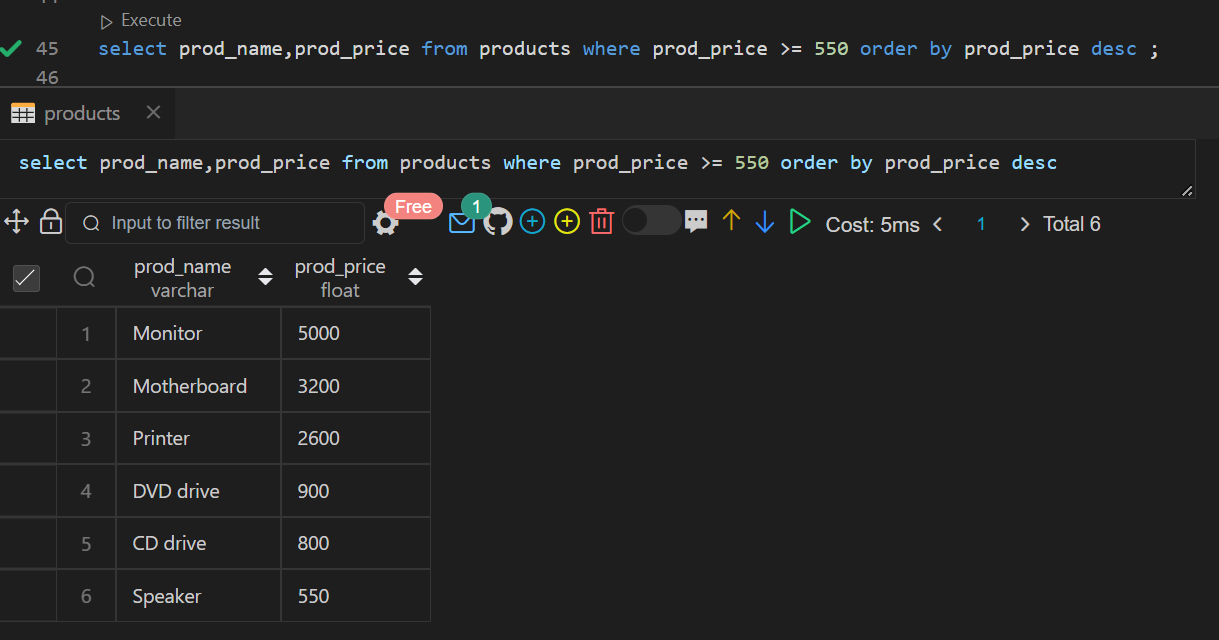
107 CD drive 800.00 12

108 Printer 2600.00 13

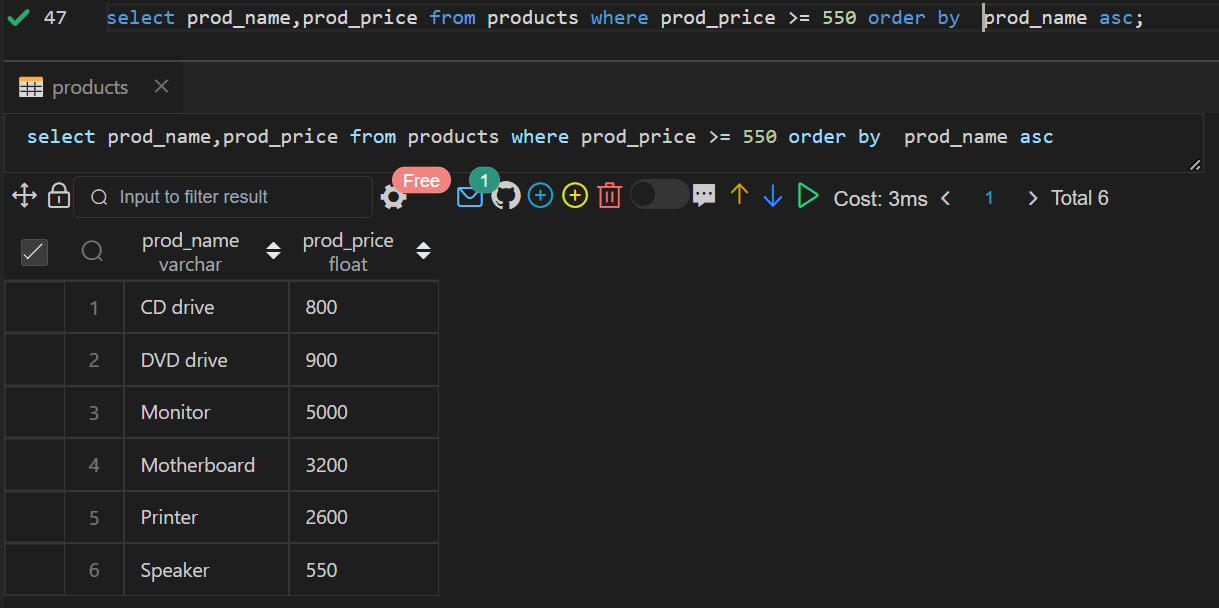
109 Refill cartridge 350.00 13

110 Mouse 250.00 12

**select prod\_name,prod\_price from products where prod\_price >= 550 order by prod\_price desc ;**



**select prod\_name,prod\_price from products where prod\_price >= 550 order by prod\_name asc;**



**5. From the following table, write a SQL query to find details of all orders excluding those with ord\_date equal to '2012-09-10' and salesman\_id higher than 5005 or purch\_amt greater than 1000.Return ord\_no, purch\_amt, ord\_date, customer\_id and salesman\_id.**

**ord\_no purch\_amt ord\_date customer\_id salesman\_id**

**---------- ---------- ---------- ----------- ----------- —--------------**

70001 150.5 2012-10-05 3005 5002

70009 270.65 2012-09-10 3001 5005

70002 65.26 2012-10-05 3002 5001

70004 110.5 2012-08-17 3009 5003

70007 948.5 2012-09-10 3005 5002

70005 2400.6 2012-07-27 3007 5001

70008 5760 2012-09-10 3002 5001

70010 1983.43 2012-10-10 3004 5006

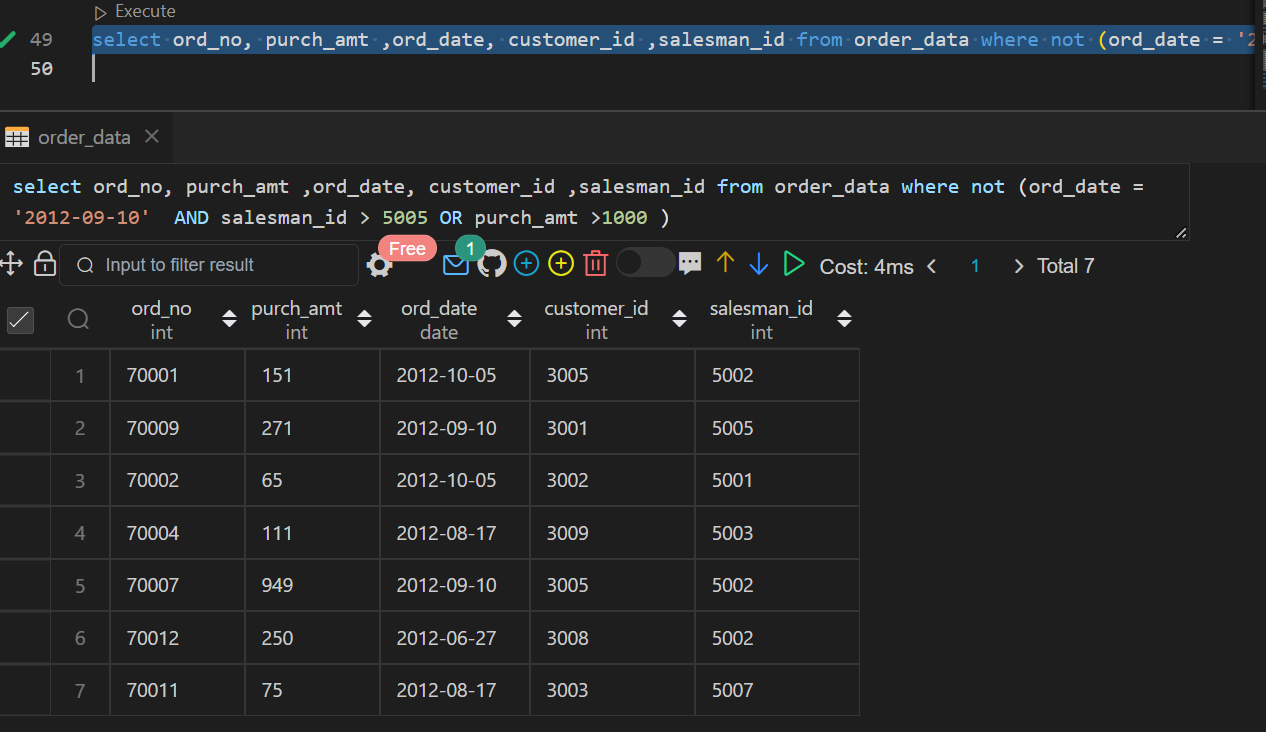
70003 2480.4 2012-10-10 3009 5003

70012 250.45 2012-06-27 3008 5002

70011 75.29 2012-08-17 3003 5007

70013 3045.6 2012-04-25 3002 5001

**select ord\_no, purch\_amt ,ord\_date, customer\_id ,salesman\_id from order\_data where not (ord\_date = '2012-09-10' AND salesman\_id > 5005 OR purch\_amt >1000 );**



**6. Create the table world with your schema and find the below queries !**

**name continent area population gdp**

Afghanistan Asia 652230 25500100 20343000000

Albania Europe 28748 2831741 12960000000

Algeria Africa 2381741 37100000 188681000000

Andorra Europe 468 78115 3712000000

Angola Africa 1246700 20609294 100990000000

Dominican Republic Caribbean 48671 9445281 58898000000

China Asia 9596961 1365370000 8358400000000

Colombia South America 1141748 47662000 369813000000

Comoros Africa 1862 743798 616000000

Denmark Europe 43094 5634437 314889000000

Djibouti Africa 23200 886000 1361000000

Dominica Caribbean 751 71293 499000000

**create table if not exists country\_data (**

**name varchar(30),**

**continent varchar(30),**

**area int ,**

**population bigint,**

**gdp bigint);**

**insert into country\_data values**

**('Afghanistan', 'Asia', 652230, 25500100, 20343000000),**

**('Albania', 'Europe', 28748, 2831741, 12960000000),**

**('Algeria', 'Africa', 2381741, 37100000, 188681000000),**

**('Andorra', 'Europe', 468, 78115, 3712000000),**

**('Angola', 'Africa', 1246700, 20609294, 100990000000),**

**('Dominican Republic', 'Caribbean', 48671, 9445281, 58898000000),**

**('China', 'Asia', 9596961, 1365370000, 8358400000000),**

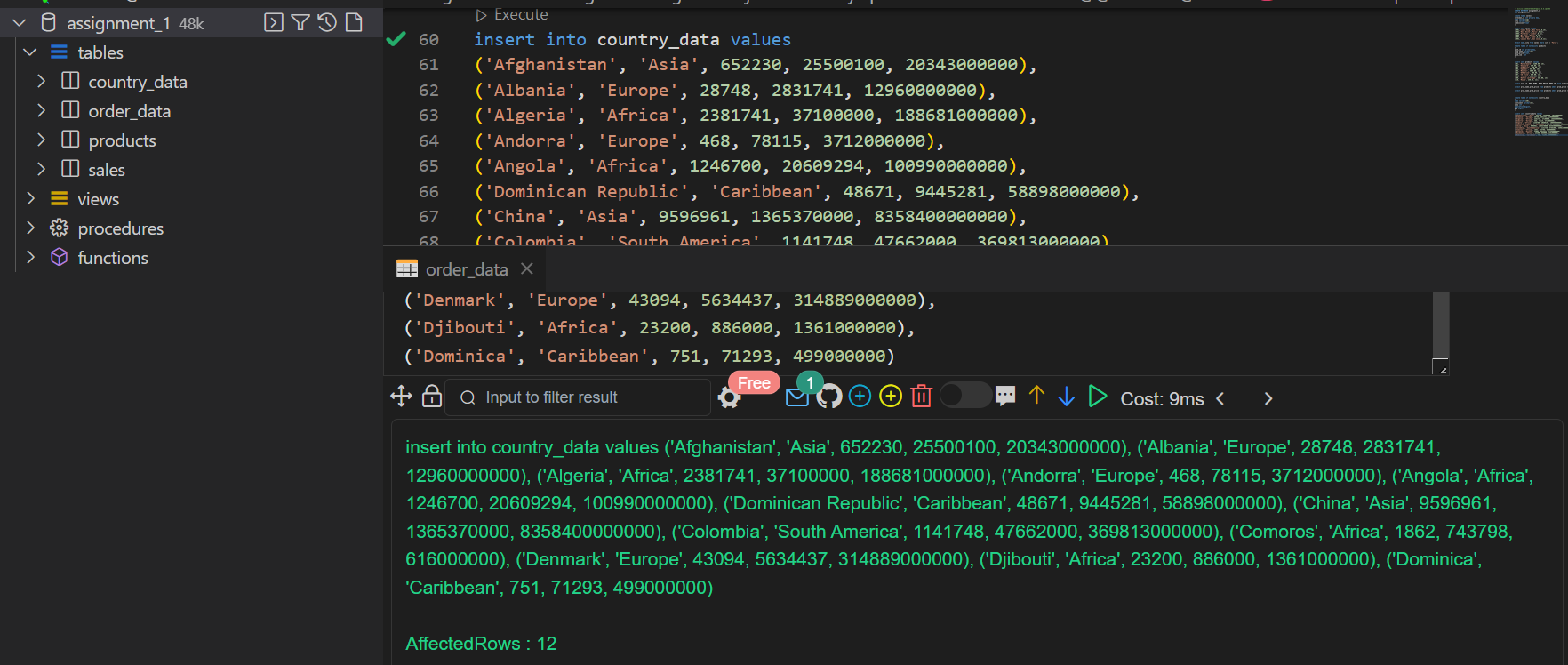
**('Colombia', 'South America', 1141748, 47662000, 369813000000),**

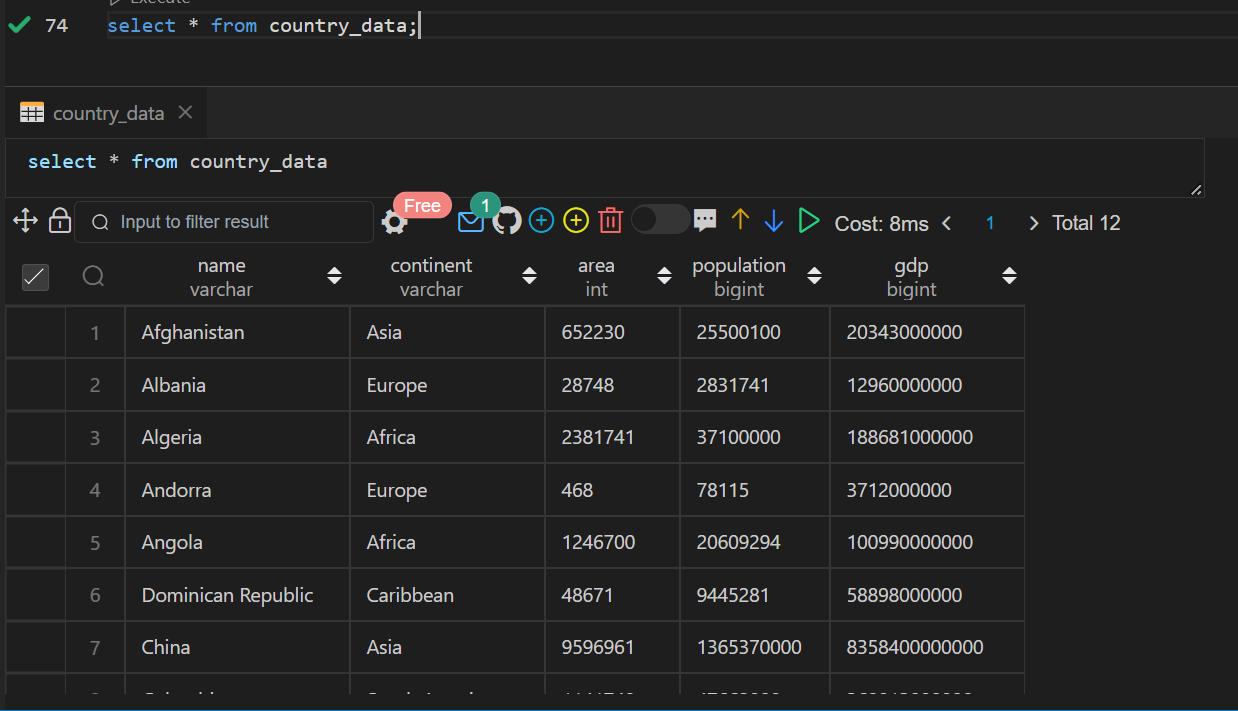
**('Comoros', 'Africa', 1862, 743798, 616000000),**

**('Denmark', 'Europe', 43094, 5634437, 314889000000),**

**('Djibouti', 'Africa', 23200, 886000, 1361000000),**

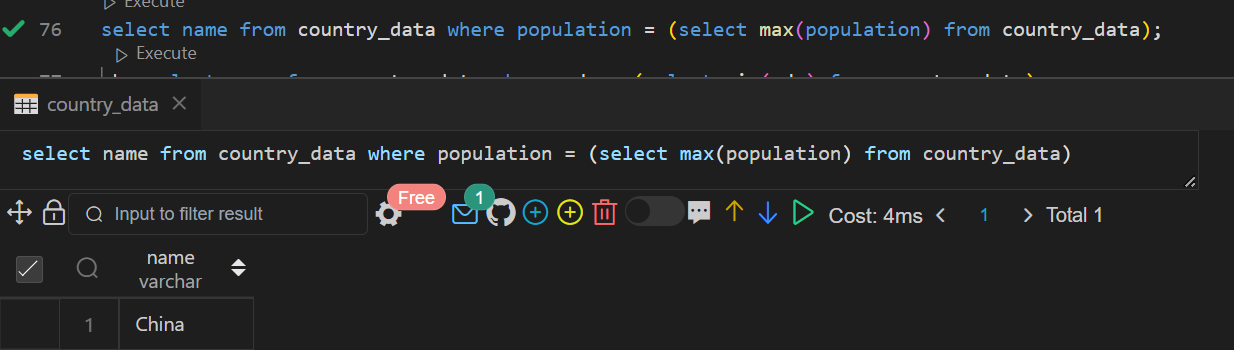
**('Dominica', 'Caribbean', 751, 71293, 499000000);**





1. **Write a query to fetch which country has the highest population?**

**select name from country\_data where population = (select max(population) from country\_data);**



**2.write a query to fetch the name of the country which has the least gdp?**

**select name from country\_data where gdp = (select min(gdp) from country\_data);**

**3.Write a query to fetch the name of the country which ends with letter C?**

**Select name from country\_data where name like '%C';**

**4.write a query to fetch the name of the country which starts with letter D?**

**Select name from country\_data where name like 'D%';**

**5.write query to fetch which continent has highest gdp?**

**select continent from country\_data where gdp = (select max(gdp) from country\_data);**

**6.Give the total GDP of Africa?**

**select sum (gdp) gdp from country\_data where continent = 'Africa';**

**7.write a query to fetch the total population for each continent?**

**select continent,sum (population)population from country\_data group by continent;**

**8. For each relevant continent show the number of countries that has a population of at least**

**200000000?**

**select continent, count(name)no\_of\_countries from country\_data where population >= 200000000 group by continent;**

**Hint: Can be solved using aggregate function**

**7. Problem statement: Suppose we have two table students and course**

**create table students(student\_id int,**

**student\_name varchar(60) not null,**

**city varchar(60) not null,**

**primary key(student\_id));**

**create table course(student\_id int,**

**course\_name varchar(60) not null,**

**Marks int not null,**

**primary key(student\_id),**

**foreign key(student\_id) references students(student\_id));**

**insert into students values(200,'John Doe','Delhi'),**

**(210,'John Doe','Delhi'),(220,'Moon ethan','Rajasthan'),**

**(230,'Jessie','Bangalore'),(240,'Benbrook','Bihar'),(250,'Ethan','Bihar'),**

**(260,'Johnnie','Bangalore'),(270,'Goh','Delhi'),(380,'John Doe','Delhi'),**

**(280,'Pavi','Delhi'),(290,'Sanvi','Rajasthan'),(300,'Navyaa','Bangalore'),**

**(310,'Ankul','Bihar'),(311,'Hitanshi','Bihar'),(312,'Aayush','Bangalore'),**

**(313,'Rian','Delhi');**

**insert into course values(200,'Datascience',75),**

**(210,'Datascience',75),(220,'Dataanalyst',80),(230,'Dataanalyst',80),**

**(240,'Dataanalyst',84),(250,'Dataanalyst',50),(260,'Datascience',80),**

**(270,'Datascience',99),(380,'Datascience',45),(280,'Datascience',78),**

**(290,'Dataanalyst',78),(300,'Computer vision',90),(310,'Computer vision',90),**

**(311,'Computer vision',75),(312,'Computer vision',39);**

**Questions :**

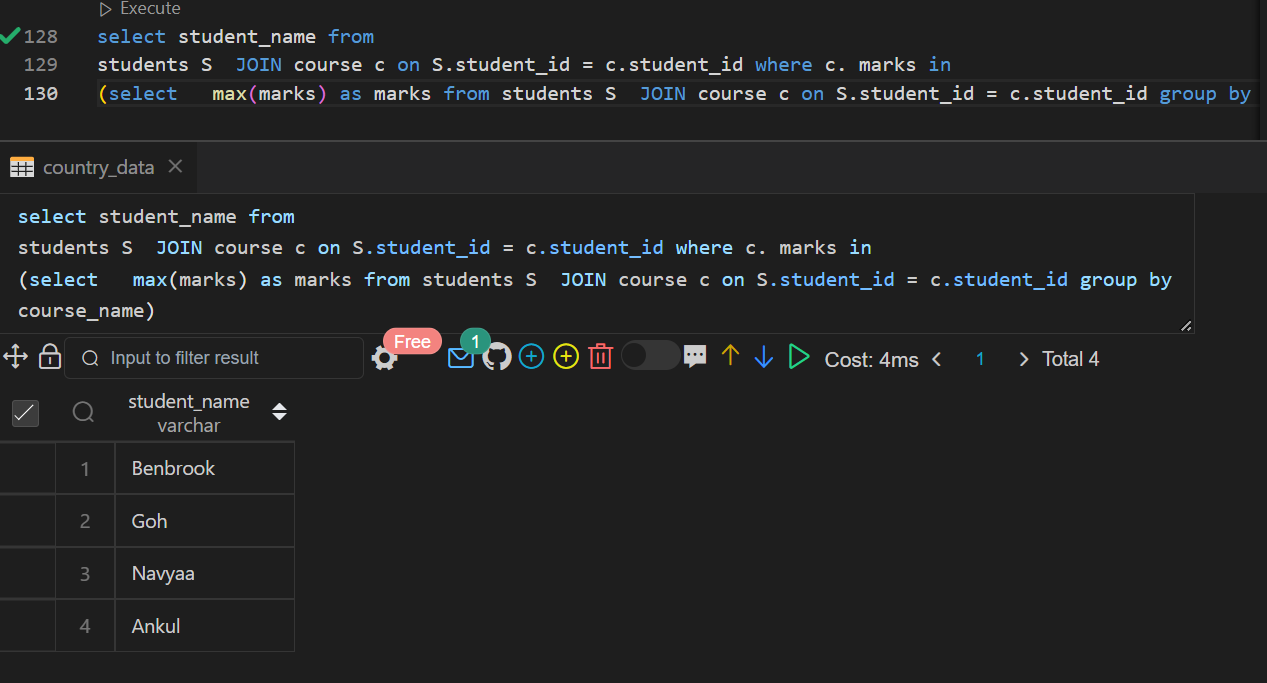
**q1. write a query to fetch the names of the students having maximum marks in each course?**

**\*\* Using JOINS**

**select student\_name from**

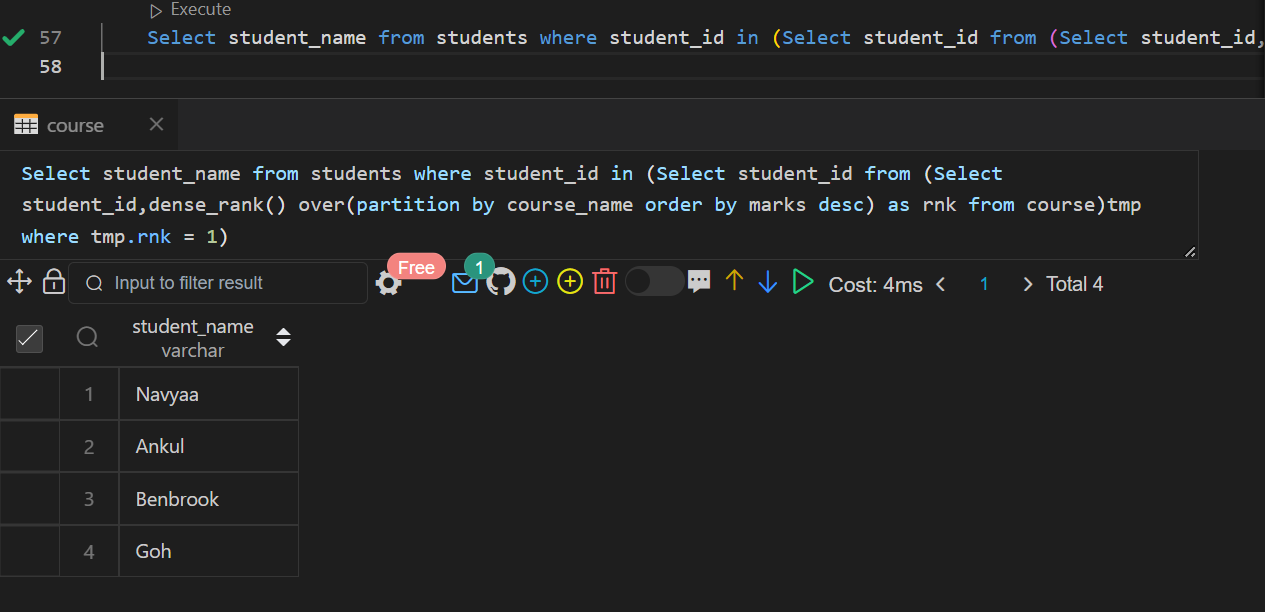
**students S JOIN course c on S.student\_id = c.student\_id where c. marks in**

**(select max(marks) as marks from students S JOIN course c on S.student\_id = c.student\_id group by course\_name);**

****

**\*\* USING Window Function**

**Select student\_name from students where student\_id in (Select student\_id from (Select student\_id,dense\_rank() over(partition by course\_name order by marks desc) as rnk from course)tmp where tmp.rnk = 1);**



**q2. write a query to fetch the names of the students having 3th highest marks from each course?**

**\*\*Using CTE,Joins and Window Function**

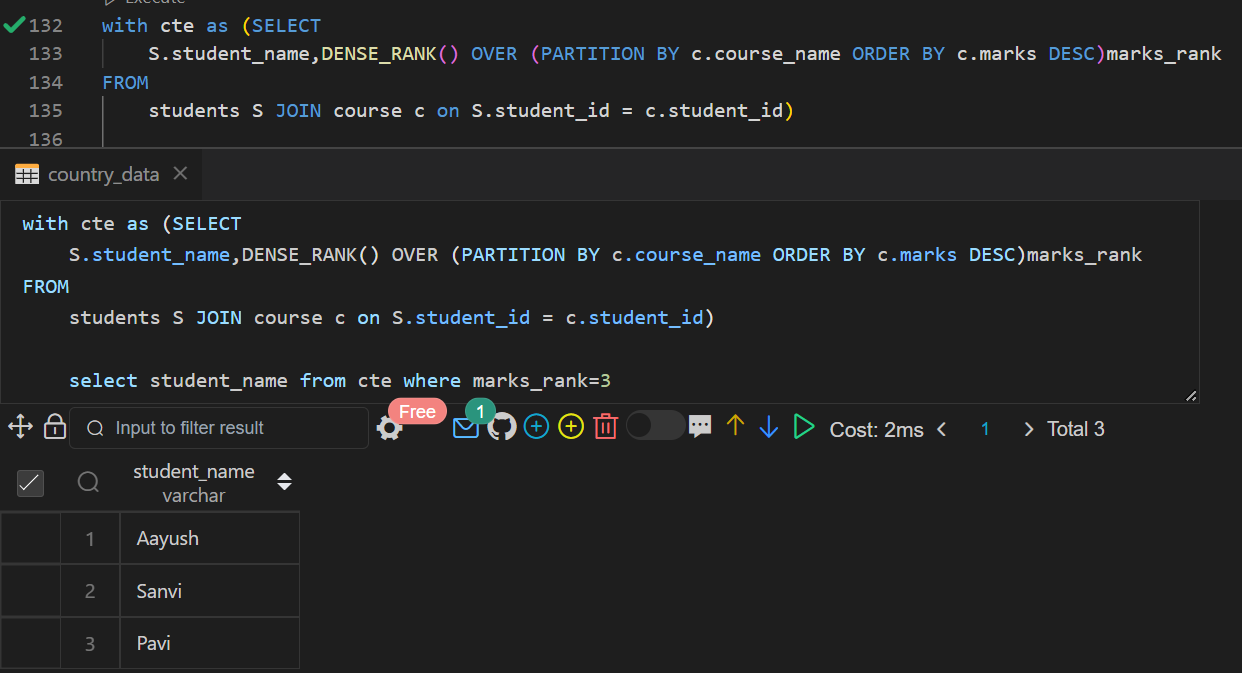
**with cte as (SELECT**

**S.student\_name,DENSE\_RANK() OVER (PARTITION BY c.course\_name ORDER BY c.marks DESC)marks\_rank**

**FROM**

**students S JOIN course c on S.student\_id = c.student\_id)**

**select student\_name from cte where marks\_rank=3;**

****

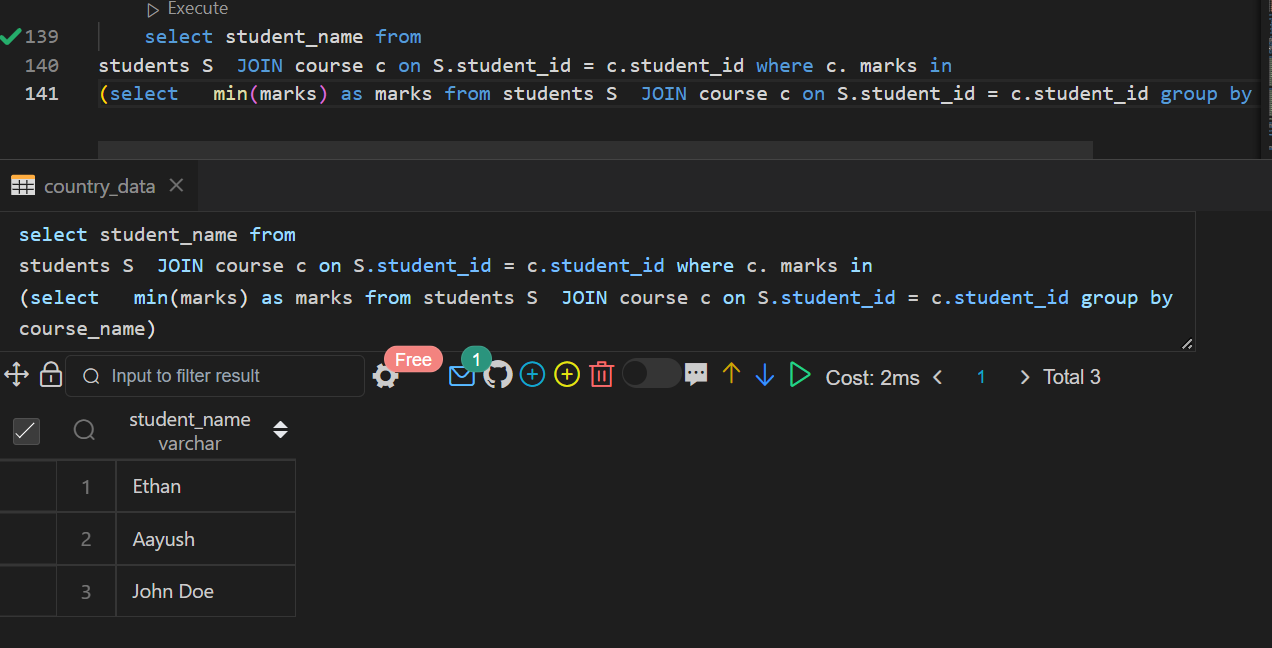
**q3. write a query to fetch the names of the students having minimum marks in each course?**

**\*\*Using JOINS**

**select student\_name from**

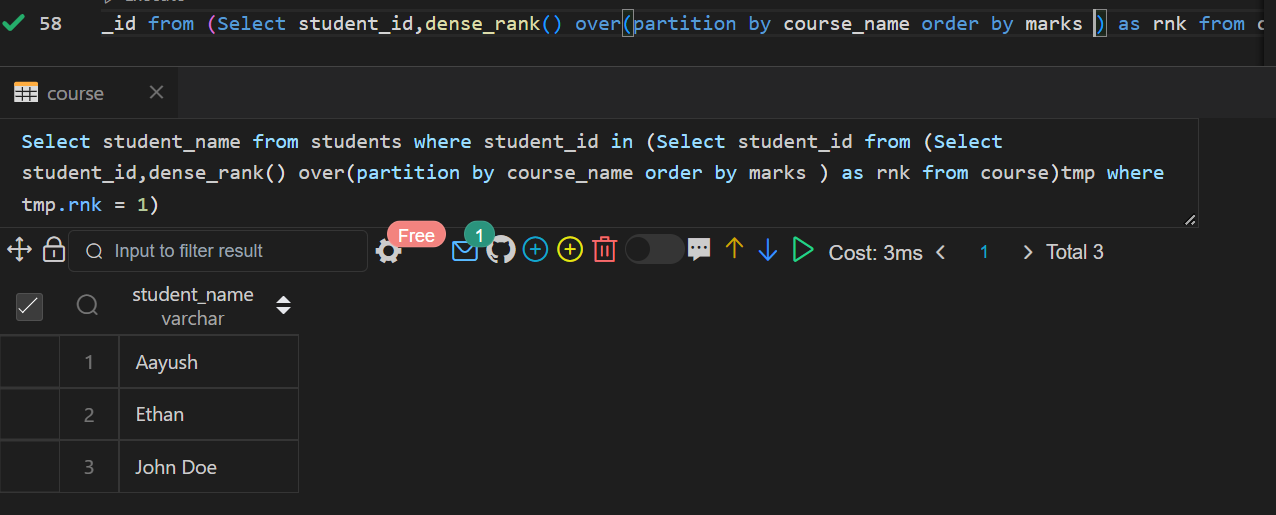
**students S JOIN course c on S.student\_id = c.student\_id where c. marks in**

**(select min(marks) as marks from students S JOIN course c on S.student\_id = c.student\_id group by course\_name);**

****

**\*\*Using Window Function**

**Select student\_name from students where student\_id in (Select student\_id from (Select student\_id,dense\_rank() over(partition by course\_name order by marks ) as rnk from course)tmp where tmp.rnk = 1);**



**q4. write a query to fetch the names of the students having 4th least marks from each course?**

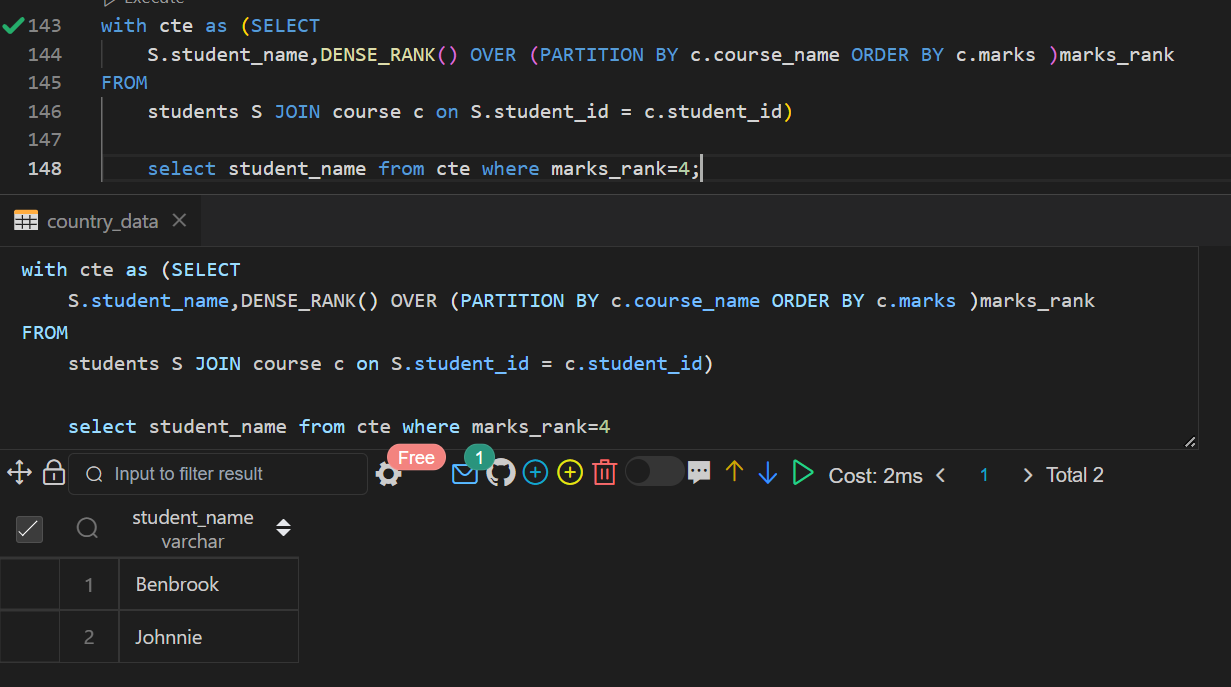
**with cte as (SELECT**

**S.student\_name,DENSE\_RANK() OVER (PARTITION BY c.course\_name ORDER BY c.marks )marks\_rank**

**FROM**

**students S JOIN course c on S.student\_id = c.student\_id)**

**select student\_name from cte where marks\_rank=4;**

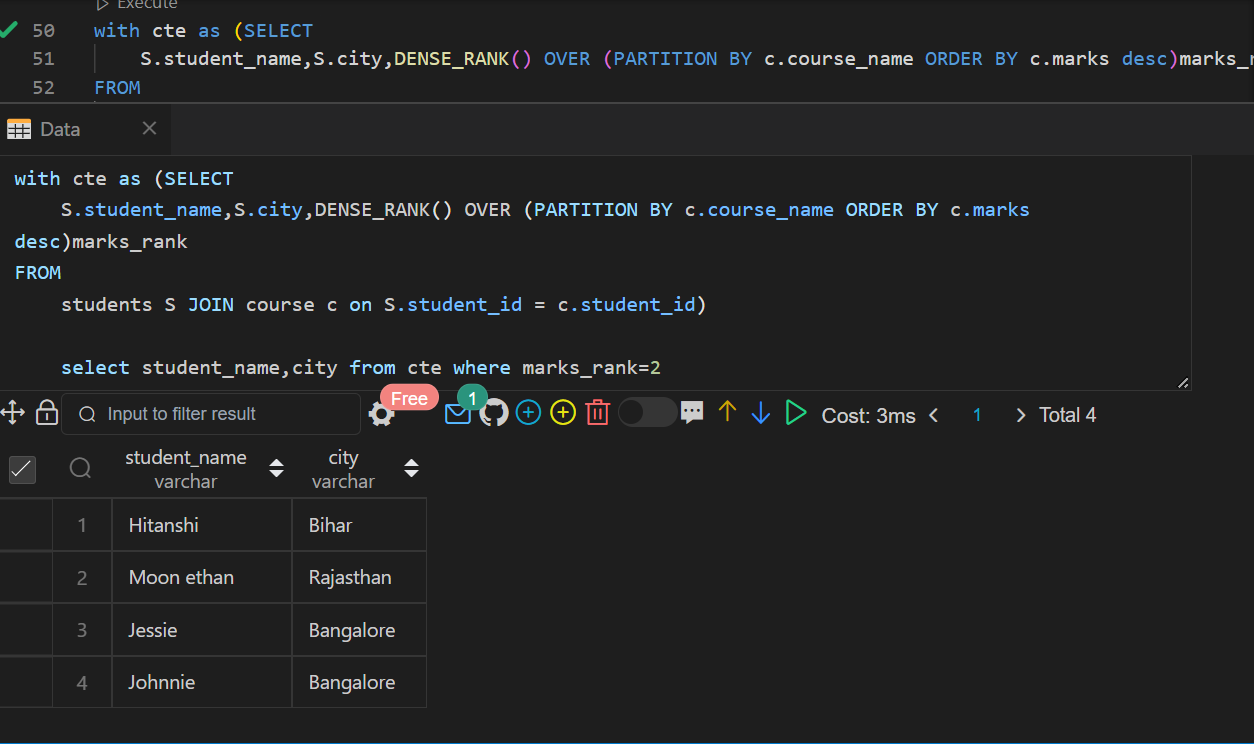
****

**q5. write a query to fetch the city name of the students who have 2nd highest marks?**

**with cte as**

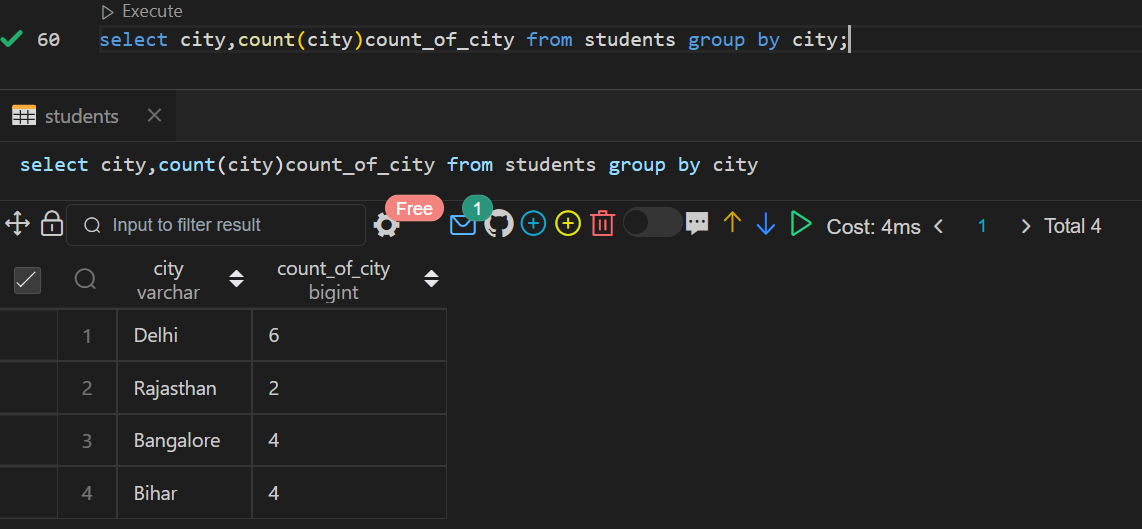
**(SELECT S.student\_name,S.city,DENSE\_RANK() OVER (PARTITION BY c.course\_name ORDER BY c.marks desc)marks\_rank FROM students S JOIN course c on S.student\_id = c.student\_id)**

**select student\_name,city from cte where marks\_rank=2;**

****

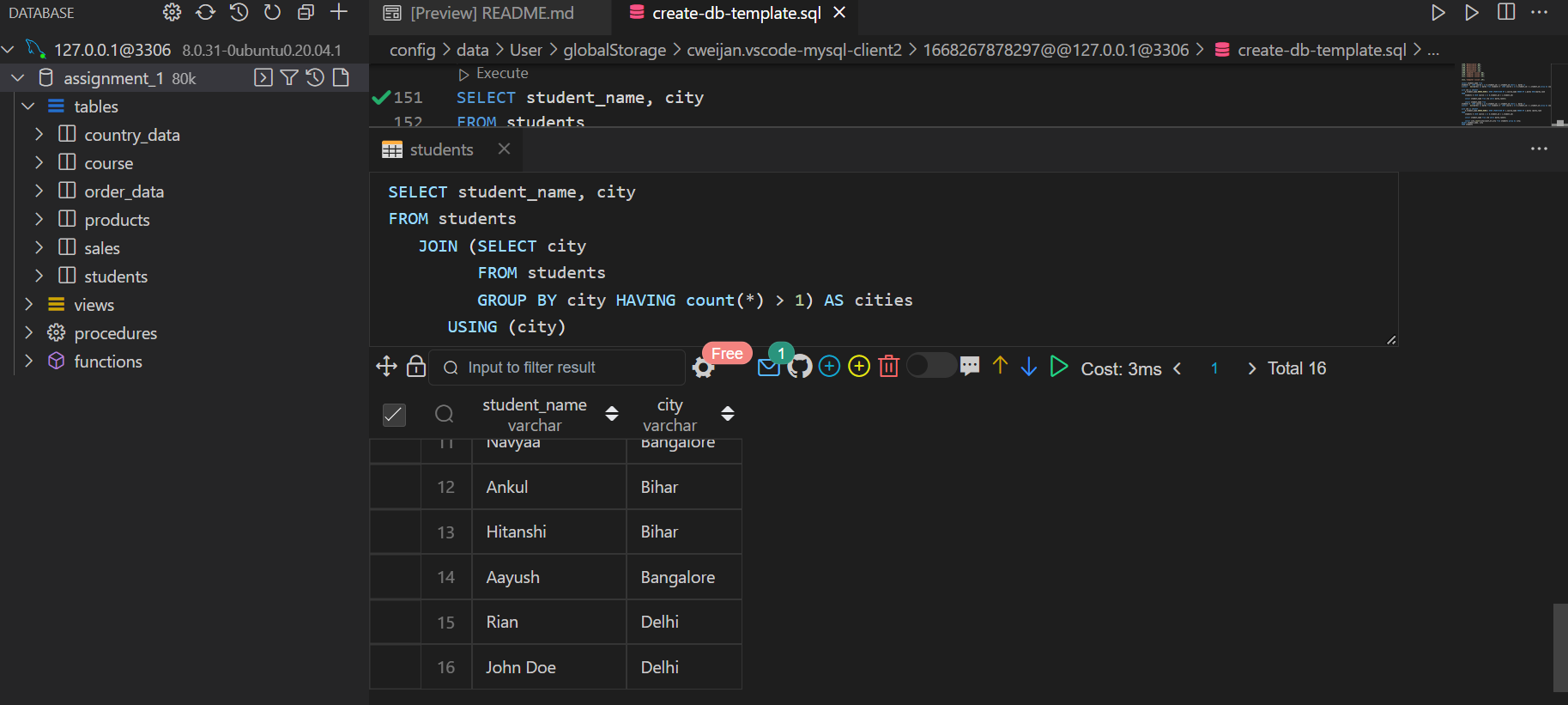
**q6. write a query to fetch the count of each city?**

**select city,count(city)count\_of\_city from students group by city;**



**q7. write a query to fetch the names of the students who are from the same city?**

**SELECT student\_name, city FROM students JOIN (SELECT city FROM students GROUP BY city HAVING count(\*) > 1) AS cities USING (city);**

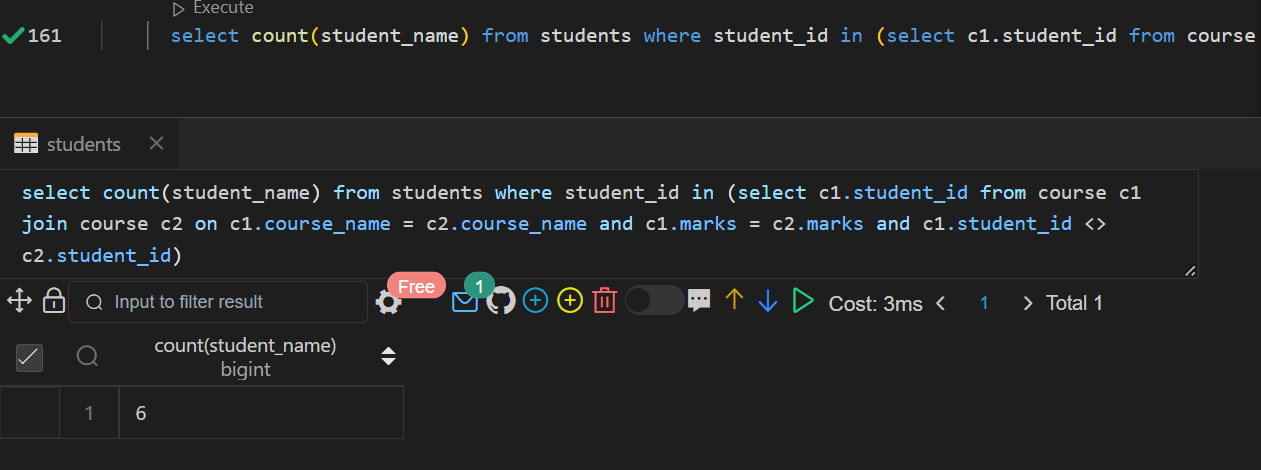
****

**q8.write a query to fetch the names of students starting with 'A'?**

**select student\_name from students where student\_name like 'A%';**

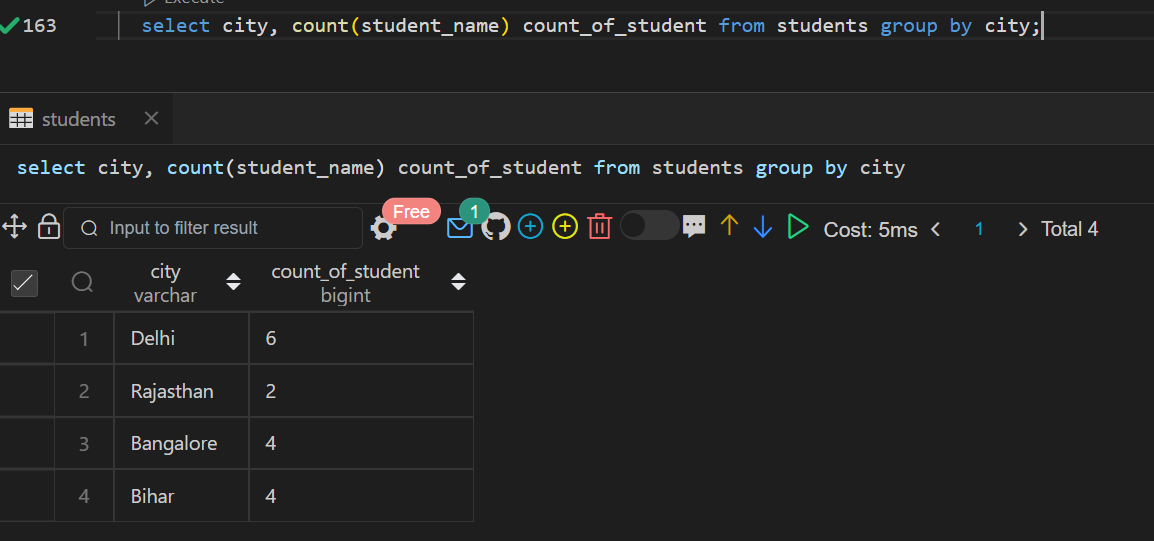
**q9.write a query to fetch the count of students' names having the same marks in each course?**

**select count(student\_name) from students where student\_id in (select c1.student\_id from course c1 join course c2 on c1.course\_name = c2.course\_name and c1.marks = c2.marks and c1.student\_id <> c2.student\_id);**

****

**q10.write a query to fetch the count of students from each city?**

**select city, count(student\_name) count\_of\_student from students group by city;**



**Hint : You must use Joins, Windows functions and CTE**

**8. Create a table below.**

| Column Name | Type |

+--------------+---------+

| player\_id | int |

| device\_id | int |

| event\_date | date |

| games\_played | int |

(player\_id, event\_date) is the primary key of this table. This table shows the activity of players of some games.Each row is a record of a player who logged in and played a number of games (possibly 0)before logging out on someday using some device.

**Write an SQL query to report the first login date for each player. Return the result table in any order.**

The query result format is in the following example.

Input:

Activity table:

+-----------+-----------+------------+--------------+

| player\_id | device\_id | event\_date | games\_played |

+-----------+-----------+------------+--------------+

| 1 | 2 | 2016-03-01 | 5 |

| 1 | 2 | 2016-05-02 | 6 |

| 2 | 3 | 2017-06-25 | 1 |

| 3 | 1 | 2016-03-02 | 0 |

| 3 | 4 | 2018-07-03 | 5 |

+-----------+-----------+------------+--------------+

Output:

+-----------+-------------+

| player\_id | first\_login |

+-----------+-------------+

| 1 | 2016-03-01 |

| 2 | 2017-06-25 |

| 3 | 2016-03-02 |

+-----------+-------------+

**create table if not exists activity**

**(**

**player\_id int ,**

**device\_id int,**

**event\_date date ,**

**games\_played int**

**primary key (player\_id,event\_date)**

**);**

**insert into activity values**

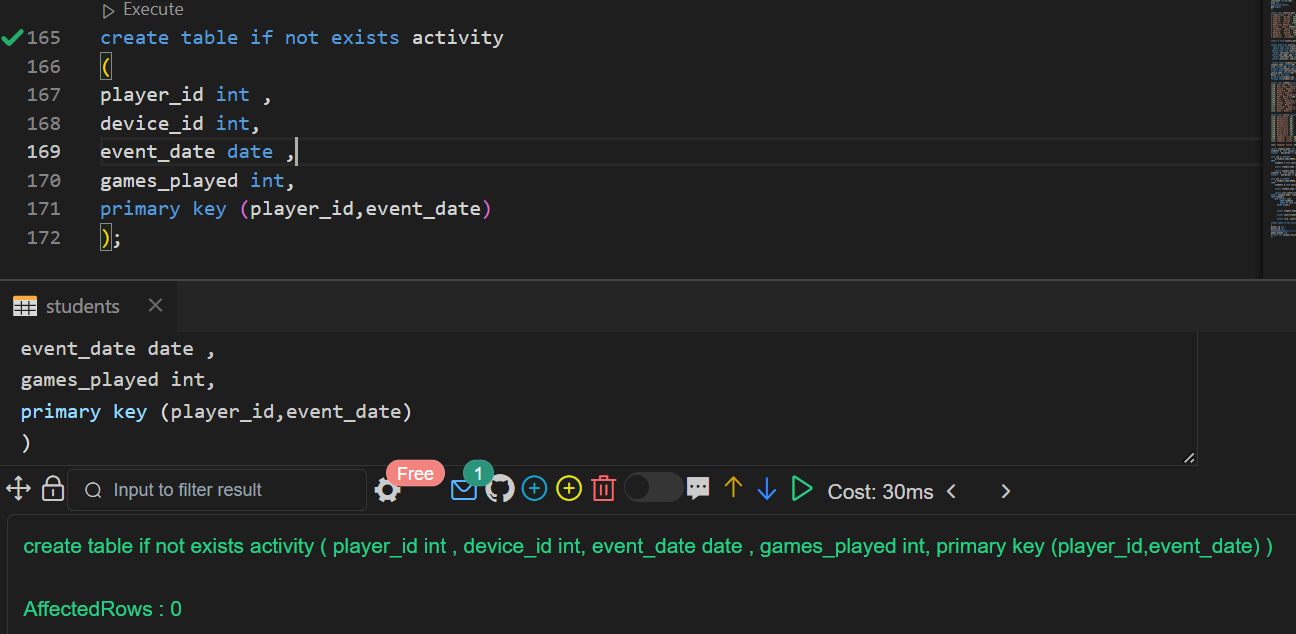
**(1,2,'2016-03-01',5),**

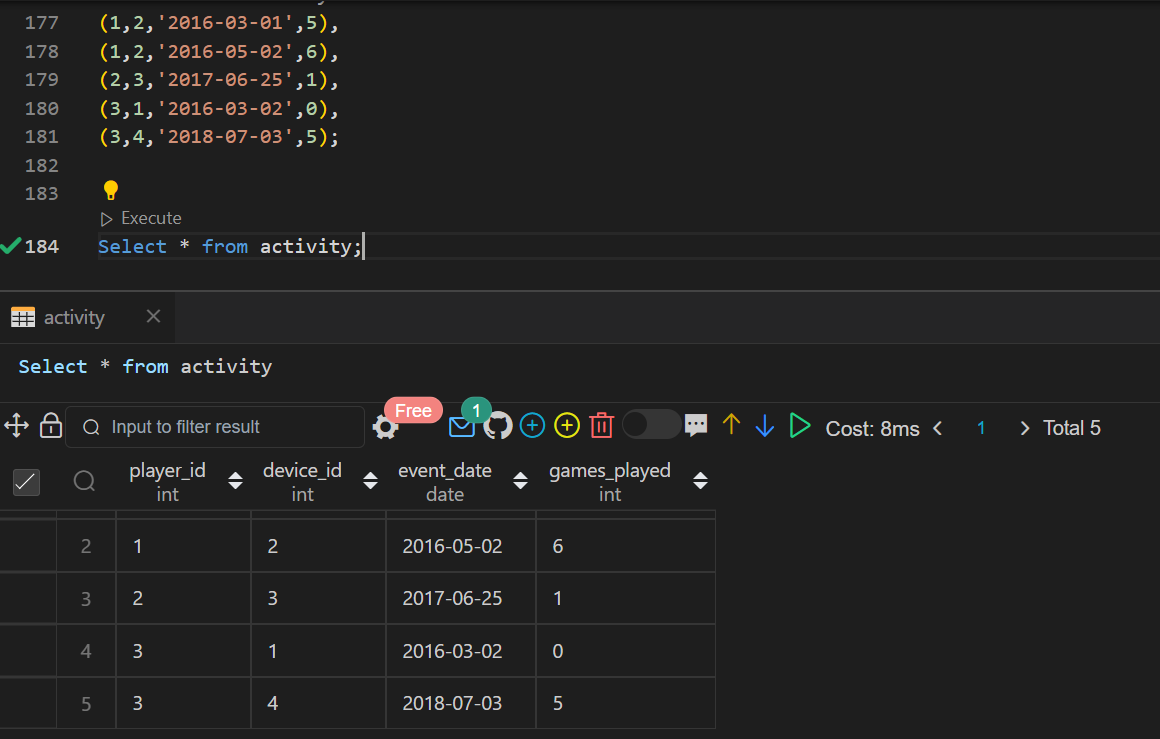
**(1,2,'2016-05-02',6),**

**(2,3,'2017-06-25',1),**

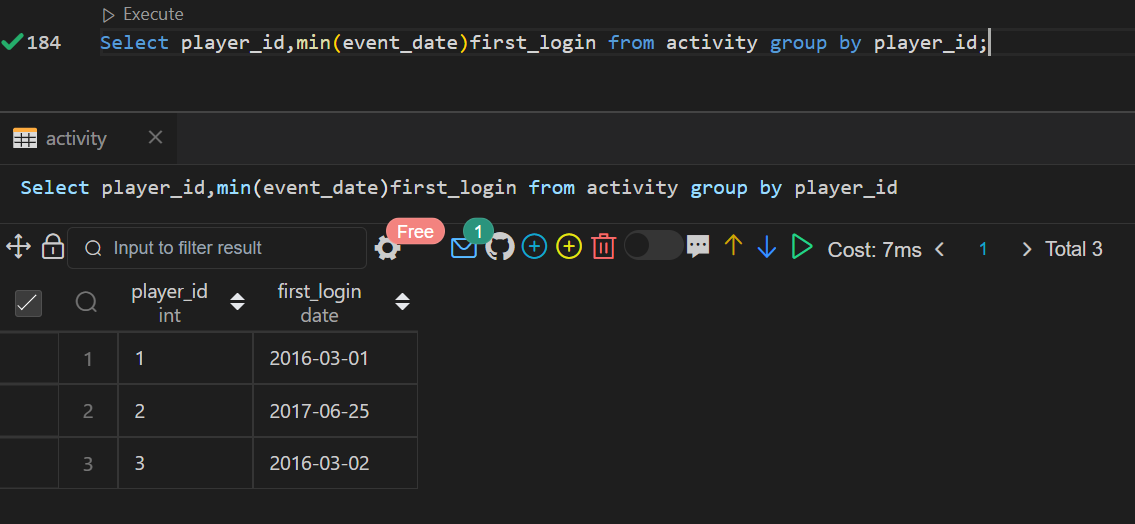
**(3,1,'2016-03-02',0),**

**(3,4,'2018-07-03',5);**





**Select player\_id,min(event\_date)first\_login from activity group by player\_id;**



**9. Create a table below.**

+-------------+---------+

| Column Name | Type |

+-------------+---------+

| product\_id | int |

| low\_fats | enum |

| recyclable | enum |

+-------------+---------+

product\_id is the primary key for this table. low\_fats is an ENUM of type ('Y', 'N') where 'Y' means this product is low fat and 'N' means it is not. recyclable is an ENUM of types ('Y', 'N') where 'Y' means this product is recyclable and 'N' means it is not.

**Write an SQL query to find the ids of products that are both low fat and recyclable. Return the result table in any order. The query result format is in the following example.**

Input:

Products table:

+-------------+----------+------------+

| product\_id | low\_fats | recyclable |

+-------------+----------+------------+

| 0 | Y | N |

| 1 | Y | Y |

| 2 | N | Y |

| 3 | Y | Y |

| 4 | N | N |

+-------------+----------+------------+

Output:

+-------------+

| product\_id |

+-------------+

| 1 |

| 3 |

+-------------+

**create table product\_data**

**(**

**product\_id int primary key,**

**low\_fats enum('Y','N'),**

**recyclable enum('Y','N')**

**);**

**insert into product\_data values**

**(0,'Y','N'),**

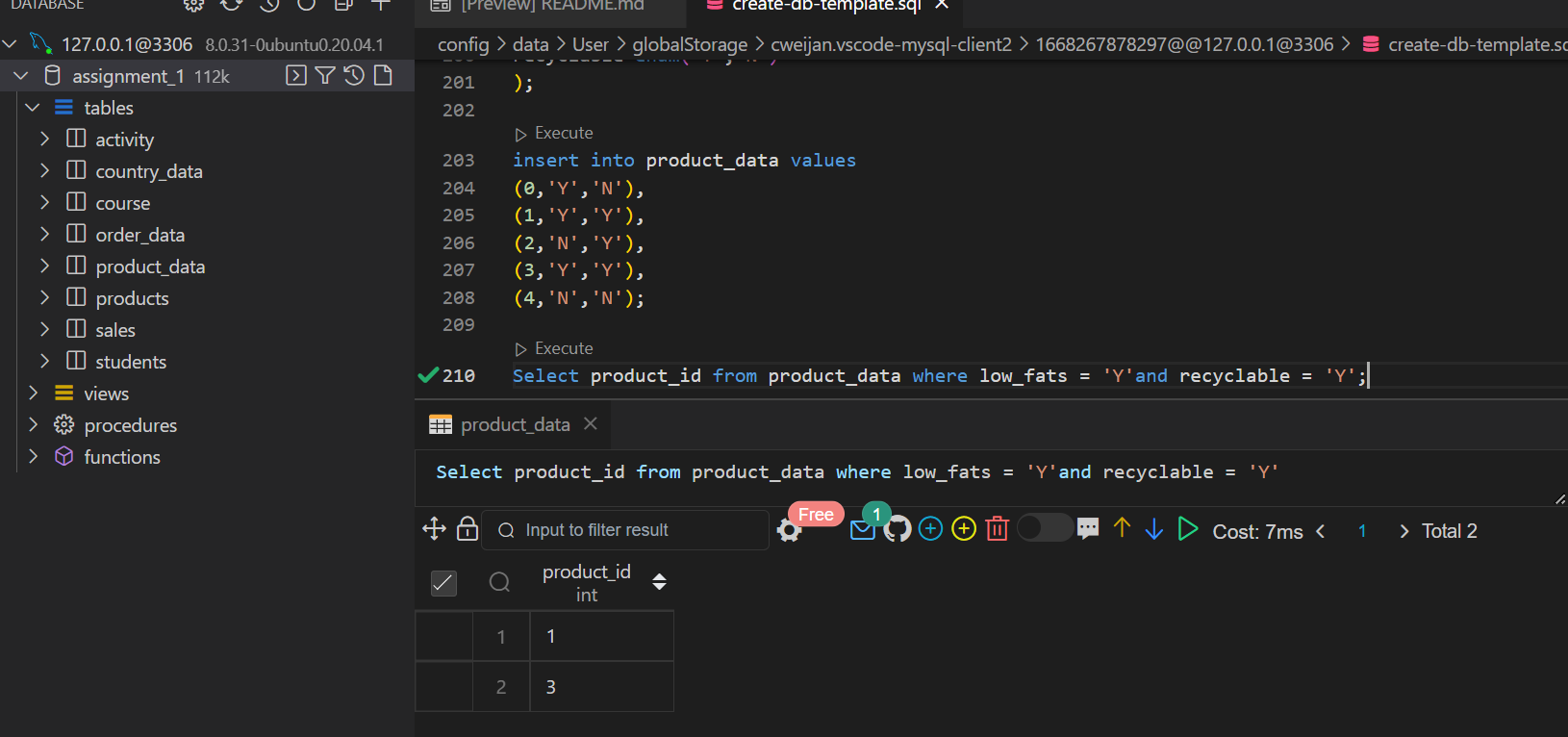
**(1,'Y','Y'),**

**(2,'N','Y'),**

**(3,'Y','Y'),**

**(4,'N','N');**

**Select product\_id from product\_data where low\_fats = 'Y'and recyclable = 'Y';**



**10. Create a table below.**

name region area population gdp

Afghanistan South Asia 652225 26000000

Albania Europe 28728 3200000 6656000000

Algeria MiddleEast 2400000 32900000 75012000000

Andorra Europe 468 64000

...

**1.Select the statement that shows the sum of population of all countries .**

**Select name,sum(population) from countries group by (name);**

**2. Select the statement that shows the number of countries with population smaller than 150000**

**SELECT COUNT(name) FROM countries WHERE population < 150000;**

**3. Select the list of core SQL aggregate functions**

**AVG(), COUNT(), MAX(), MIN(), SUM()**

**4. Select the result that would be obtained from the following code:**

**5. Select the statement that shows the average population of 'Poland', 'Germany' and 'Denmark'**

**SELECT AVG(population) FROM countries WHERE name IN ('Poland', 'Germany', 'Denmark');**

**6. Select the statement that shows the medium population density of each region**

**SELECT region, SUM(population)/SUM(area) AS density FROM countries GROUP BY region;**

**7. Select the statement that shows the name and population density of the country with the largest population**

**SELECT name, population/area AS density FROM countries WHERE population = (SELECT MAX(population) FROM countries);**