1. Problem : <https://leetcode.com/problems/department-highest-salary/>

Solution:

Select a.Department, a.Employee, a.Salary from

(SELECT department.Name AS Department, employee.Name AS Employee, employee.salary as Salary,

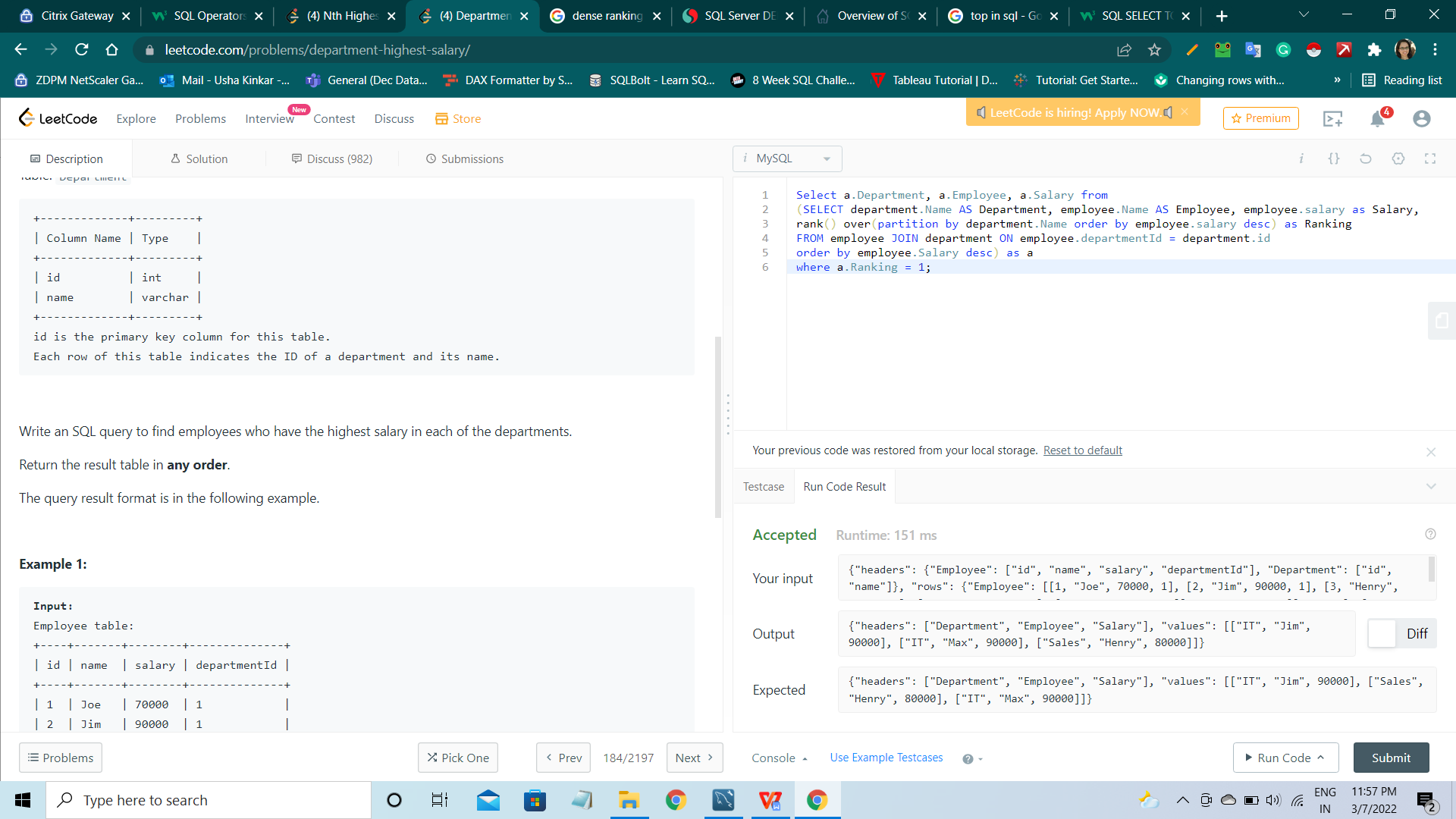
rank() over(partition by department.Name order by employee.salary desc) as Ranking

FROM employee JOIN department ON employee.departmentId = department.id

order by employee.Salary desc) as a

where a.Ranking = 1;

**Output Screenshot** :-



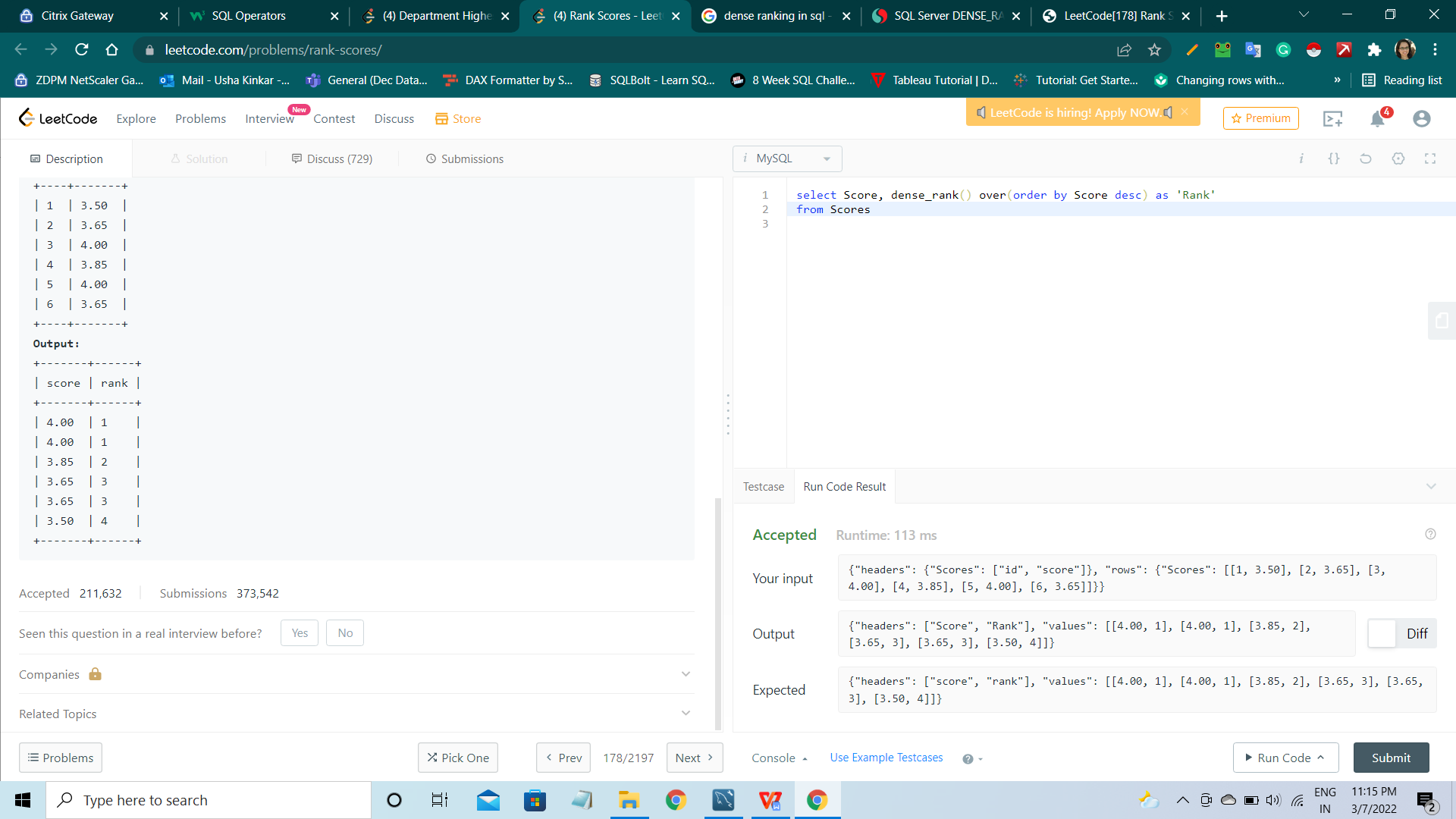
1. Problem: <https://leetcode.com/problems/rank-scores/>

Solution:

select Score, dense\_rank() over(order by Score desc) as 'Rank'

from Scores

**Output Screenshot** :-



1. Problem: <https://leetcode.com/problems/nth-highest-salary/>

Solution:

CREATE FUNCTION getNthHighestSalary(N INT) RETURNS INT

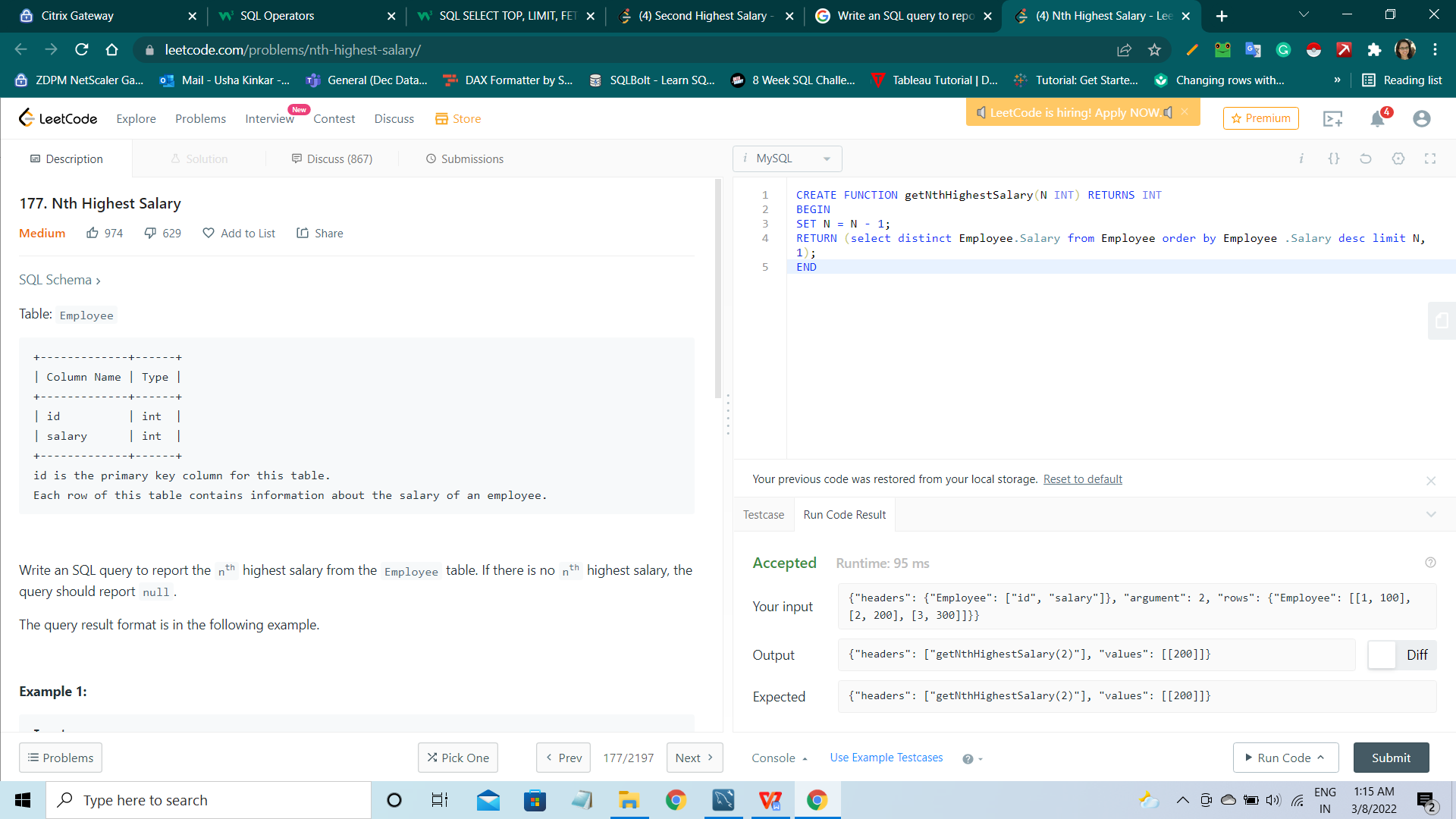
BEGIN

SET N = N - 1;

RETURN (select distinct Employee.Salary from Employee order by Employee .Salary desc limit N, 1);

END

**Output Screenshot** :-



1. Problem: <https://leetcode.com/problems/second-highest-salary/>

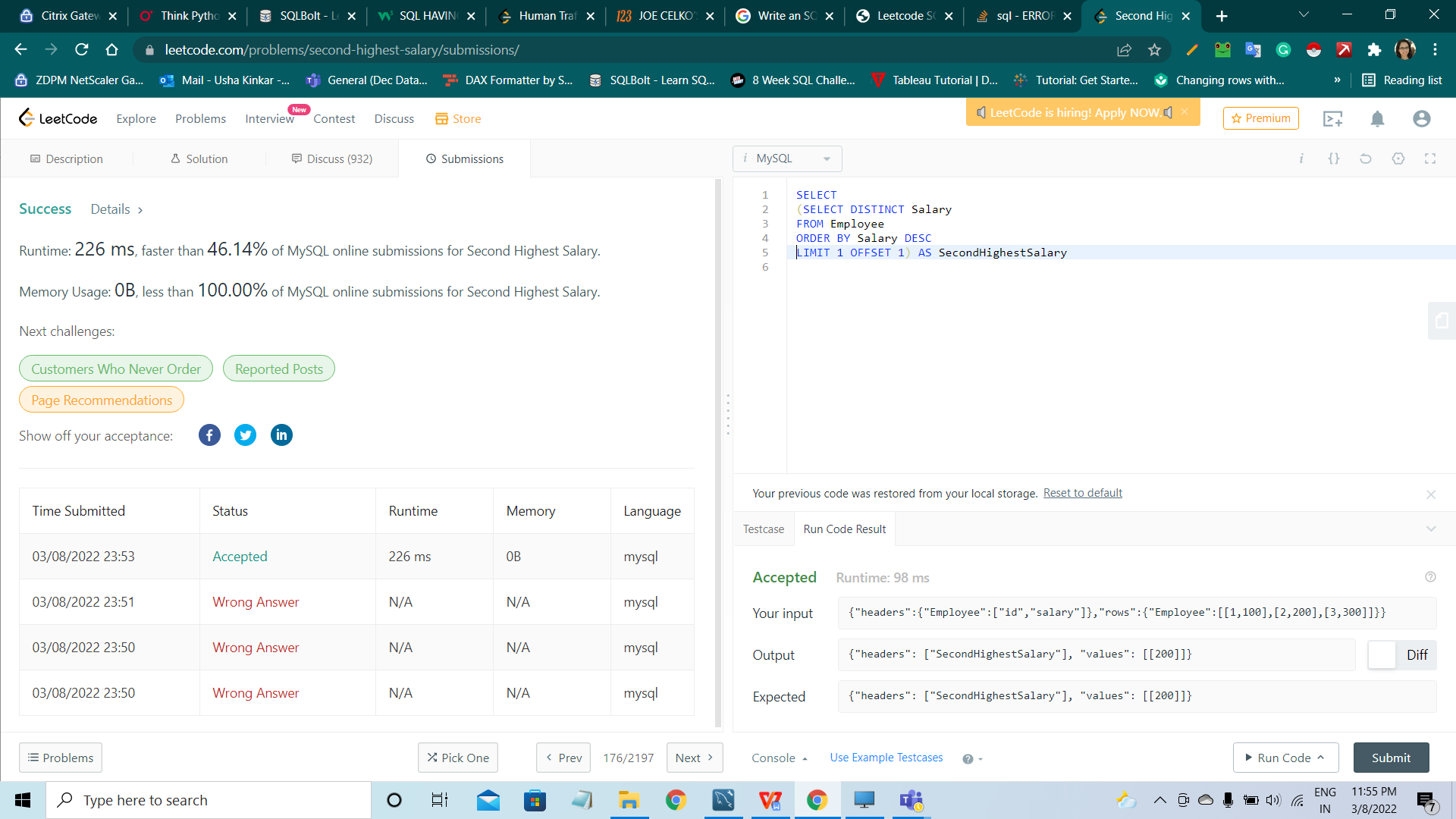
Solution:

SELECT (SELECT DISTINCT Salary FROM Employee1

ORDER BY Salary DESC

LIMIT 1 OFFSET 1) AS SecondHighestSalary;

**Output Screenshot** :-



1. Problem: <https://leetcode.com/problems/trips-and-users/>

Solution:

select a.Request\_at as Day, round(1 - sum(a.status = "Completed") / a.all\_count, 2) as "Cancellation Rate"

from

(

select \*, count(status) over (partition by request\_at) as all\_count from Trips where client\_id in

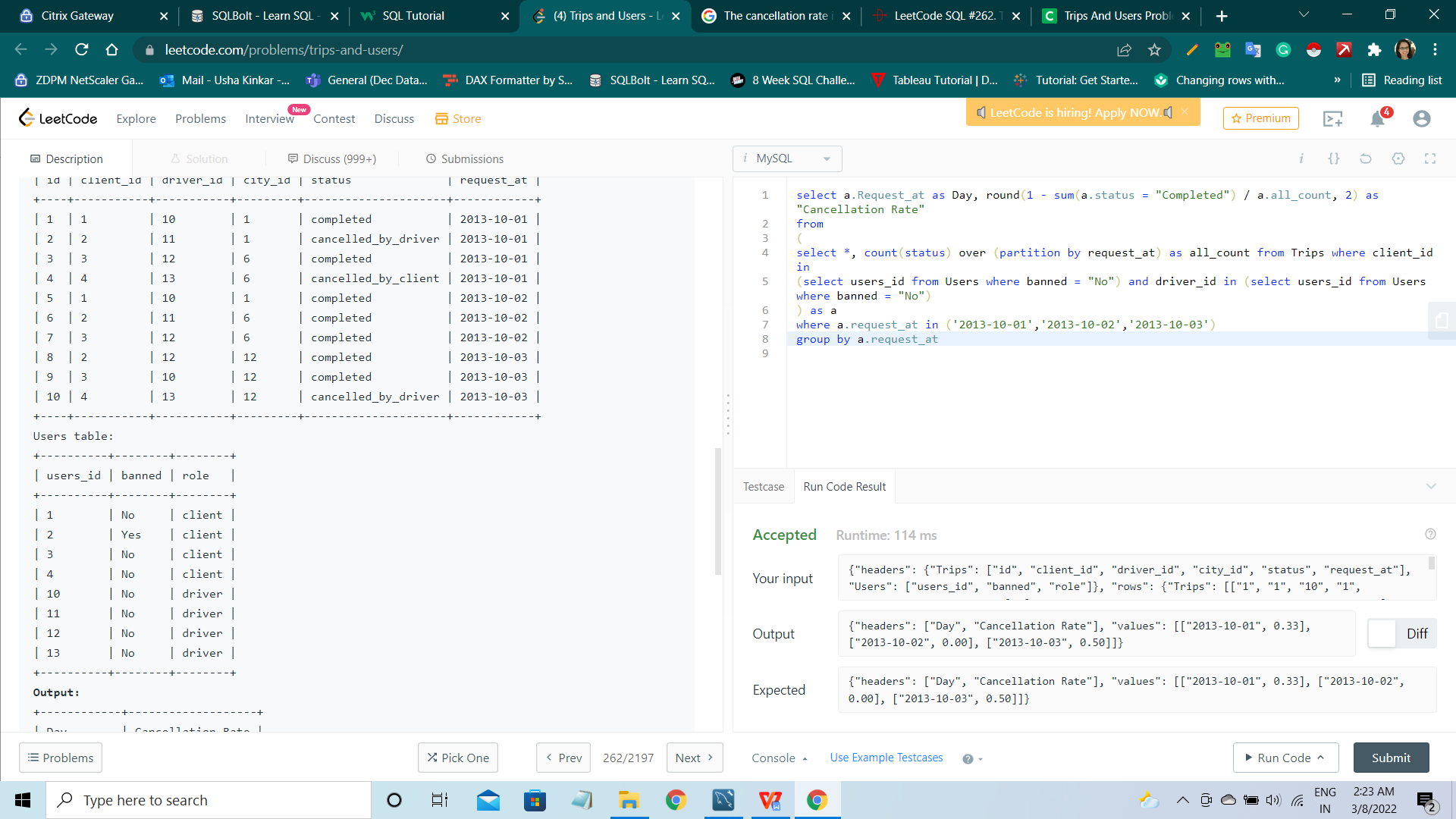
(select users\_id from Users where banned = "No") and driver\_id in (select users\_id from Users where banned = "No")

) as a

where a.request\_at in ('2013-10-01','2013-10-02','2013-10-03')

group by a.request\_at

**Output Screenshot** :-



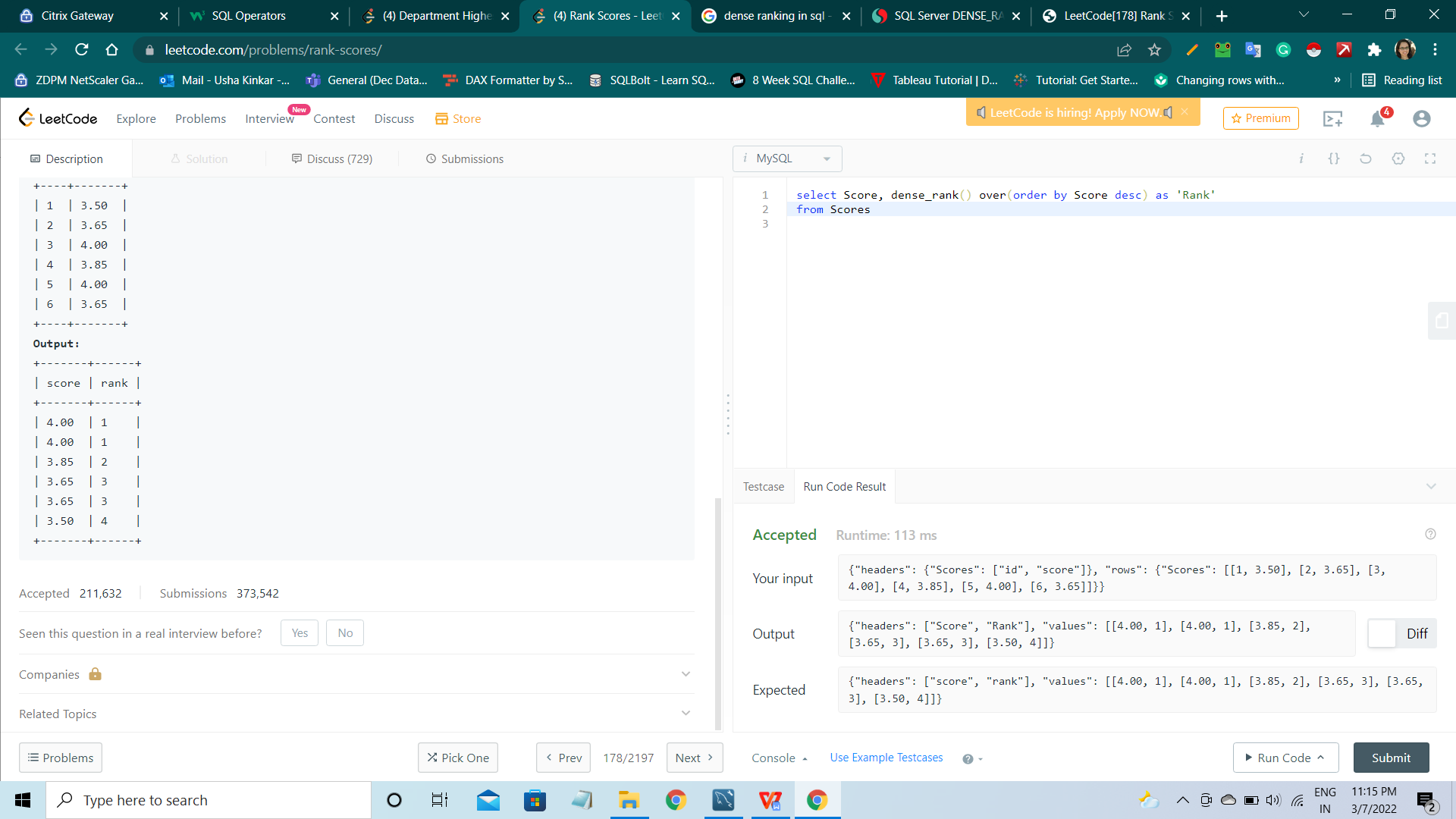
1. Problem: <https://leetcode.com/problems/rank-scores/>

Solution:

select Score, dense\_rank() over(order by Score desc) as 'Rank'

from Scores

**Output Screenshot** :-



1. Problem: <https://leetcode.com/problems/department-top-three-salaries/>

Solution:

Select a.Department, a.Employee, a.Salary from

(SELECT Department1.Name AS Department, Employee3.Name AS Employee, Employee3.salary as Salary,

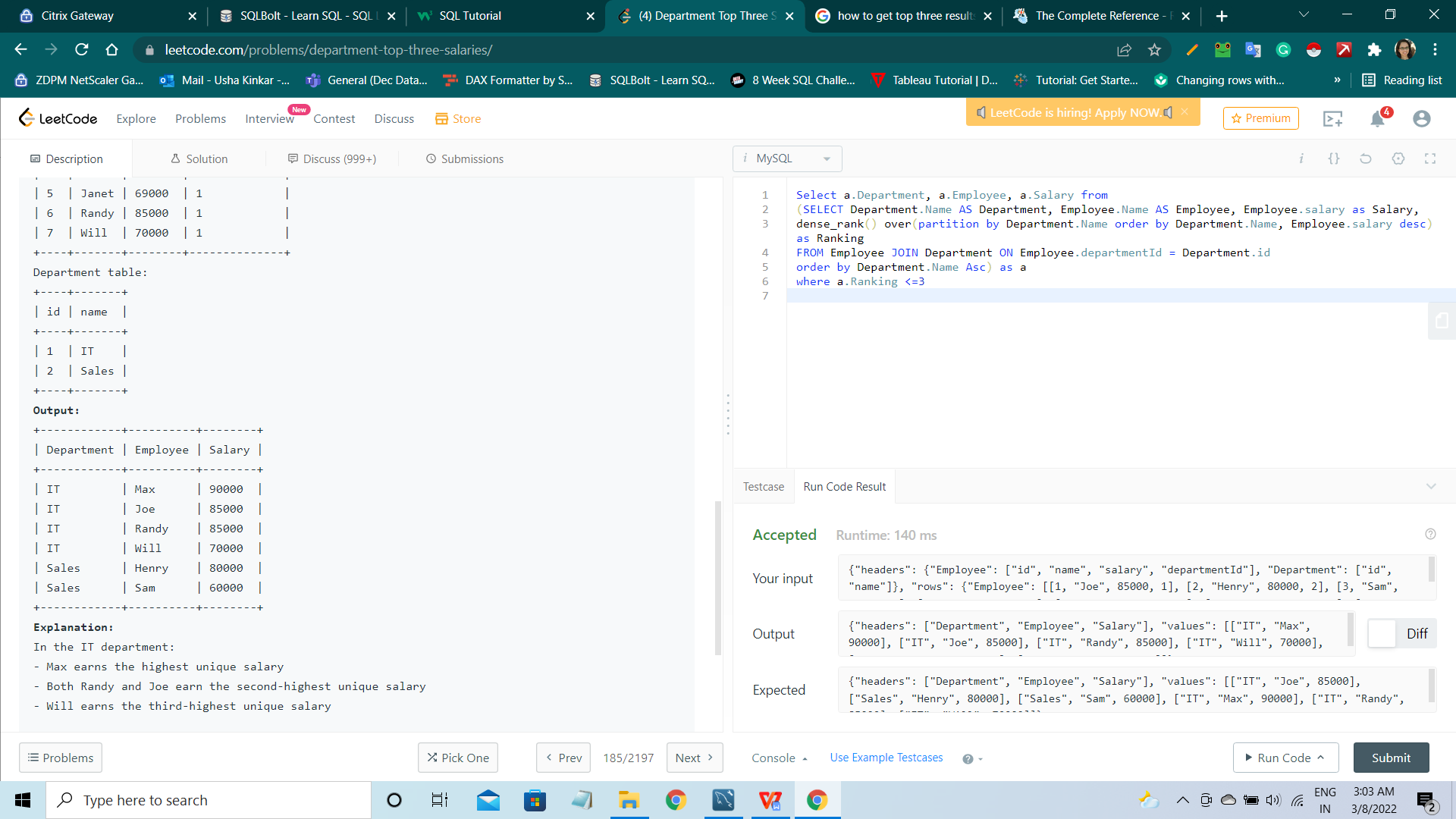
dense\_rank() over(partition by Department1.Name order by Department1.Name, Employee3.salary desc) as Ranking

FROM Employee3 JOIN Department1 ON Employee3.departmentId = Department1.id

order by Department1.Name Asc) as a

where a.Ranking <=3

**Output Screenshot** :-



1. Problem: <https://leetcode.com/problems/human-traffic-of-stadium/>

Solution:

select id, visit\_date, people from

(select id, visit\_date, people, count(\*) over(partition by difference) as a from

(select id, visit\_date, people, (id - row\_number() over(order by visit\_date)) as difference

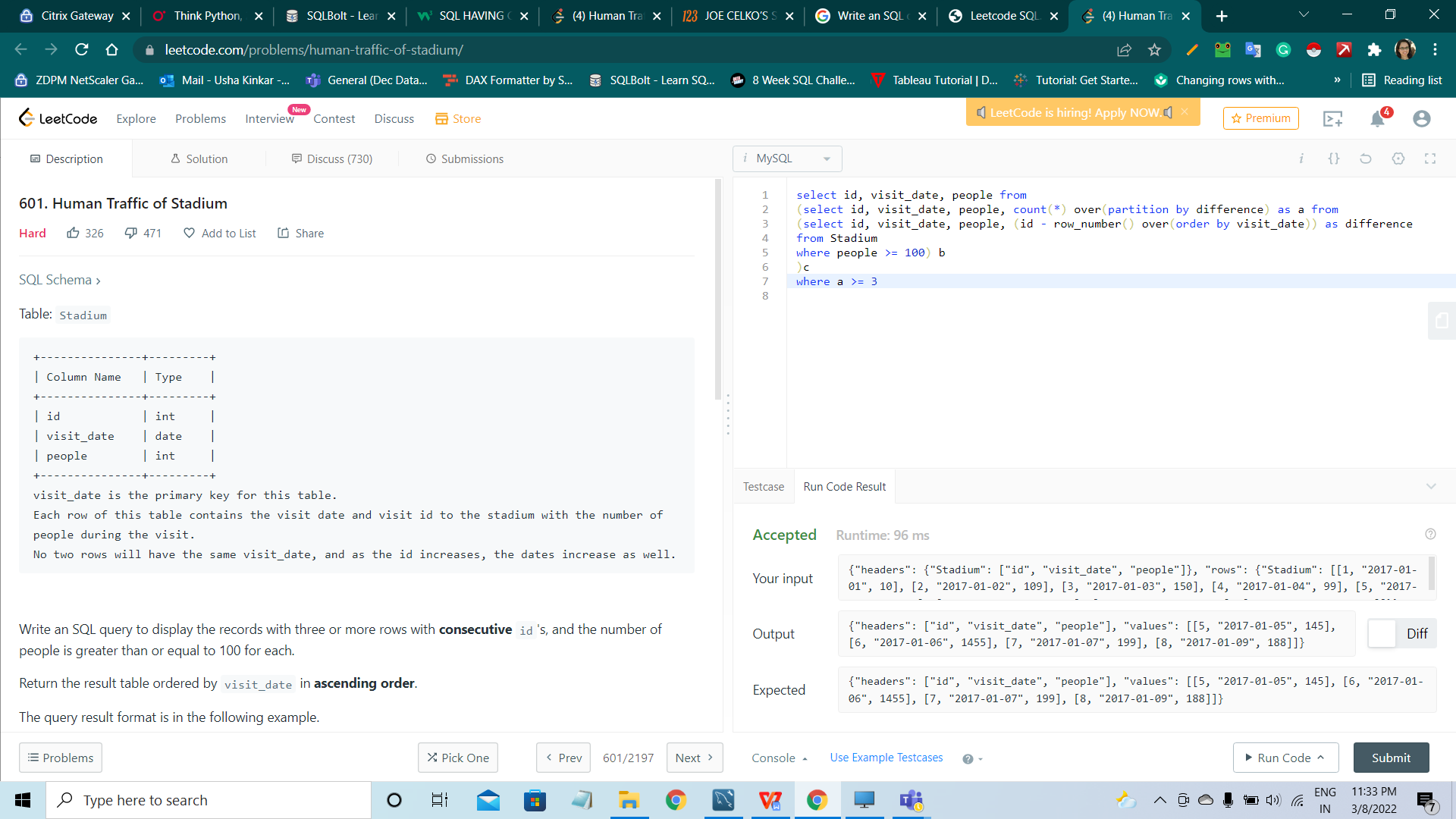
from Stadium

where people >= 100) b

)c

where a >= 3

**Output Screenshot** :-



Following questions are marked 5 marks each

1. Let’s write some CREATE TABLE statements that are as complete as possible. This little exercise is important because SQL is a declarative language and you need to learn how to specify things in the database instead of in the code. The table looks like this:

CREATE TABLE FiscalYearTable1 (fiscal\_year INTEGER, start\_date DATE, end\_date DATE);

It stores date ranges for determining what fiscal year any given date belongs to. For example, the federal government runs its fiscal year from October 1 until the end of September.

The scalar subquery you would use to do this table lookup is:

(SELECT F1.fiscal\_year FROM FiscalYearTable1 AS F1 WHERE outside\_date BETWEEN F1.start\_date AND F1.end\_date)

Your assignment is to add all the constraints you can think of to the table to guarantee that it contains only correct information.

While vendors all have different date and time functions, let’s assume that all we have is the SQL-92 temporal arithmetic and the function EXTRACT ([YEAR | MONTH | DAY] FROM ), which returns an integer that represents a field within a date.

Solution:

drop table FiscalYearTable1;

CREATE TABLE FiscalYearTable1

(fiscal\_year INTEGER NOT NULL PRIMary KEY UNIqUE,

start\_date DATE NOT NULL UNIqUE,

end\_date DATE NOT NULL UNIqUE );

alter table FiscalYearTable1

add Constraint vstart\_date CHECK (start\_date < End\_date);

alter table FiscalYearTable1

add Constraint vend\_date CHECK (End\_date > start\_date);

ALTER TABLE FISCALYEARTABLE1

ADD Constraint valid\_start\_date\_1

Check ( YEAR(start\_date) = fiscal\_year - 1) ;

ALTER TABLE FISCALYEARTABLE1

ADD Constraint valid\_start\_date\_2

CHECK ( MONTH(start\_date) = 10 ) ;

ALTER TABLE FISCALYEARTABLE1

ADD Constraint valid\_start\_date\_3

CHECK (DAY(start\_date) = 01);

ALTER TABLE FISCALYEARTABLE1

ADD Constraint valid\_end\_date\_1

Check ( YEAR(end\_date) = fiscal\_year );

ALTER TABLE FISCALYEARTABLE1

ADD Constraint valid\_end\_date\_2

Check (MONTH(end\_date) = 09);

ALTER TABLE FISCALYEARTABLE1

ADD Constraint valid\_end\_date\_3

Check ( DAY(end\_date) = 30) ;

Insert Into FiscalYearTable1 Values (2010,'2009-10-01','2010-09-30') ;

SELECT \* FROM FISCALYEARTABLE1;

1. In a factory, a project is described in a work order, which has a series of steps that it must go through.

A step\_nbr on the work order is either completed or awaiting the completion of one or more of the steps that come before it.

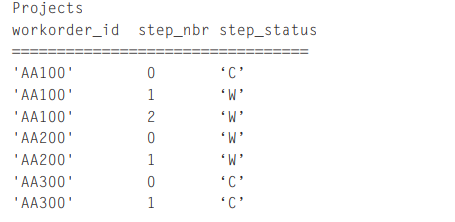
Table looks like this:

CREATE TABLE Projects (workorder\_id CHAR(5) NOT NULL,

step\_nbr INTEGER NOT NULL CHECK (step\_nbr BETWEEN 0 AND 1000), step\_status CHAR(1) NOT NULL CHECK (step\_status IN ('C', 'W')), -- complete, waiting

PRIMARY KEY (workorder\_id, step\_nbr));

With some sample data like this:



We would like to get the work orders where the step\_nbr is zero and the step\_status is ‘C’, but all other legs for that work order have a step\_status of ‘W’.

For example, the query should return only 'AA100' in the sample data.

Solution:

CREATE TABLE Projects (workorder\_id CHAR(5) NOT NULL,

step\_nbr INTEGER NOT NULL CHECK (step\_nbr BETWEEN 0 AND 1000), step\_status CHAR(1) NOT NULL CHECK (step\_status IN ('C', 'W')), -- complete, waiting

PRIMARY KEY (workorder\_id, step\_nbr));

INSERT INTO Projects VALUES ('AA100',0,'C');

INSERT INTO Projects VALUES ('AA100',1,'W');

INSERT INTO Projects VALUES ('AA100',2,'W');

INSERT INTO Projects VALUES ('AA200',0,'W');

INSERT INTO Projects VALUES ('AA200',1,'W');

INSERT INTO Projects VALUES ('AA300',0,'C');

INSERT INTO Projects VALUES ('AA300',1,'C');

SELECT workorder\_id

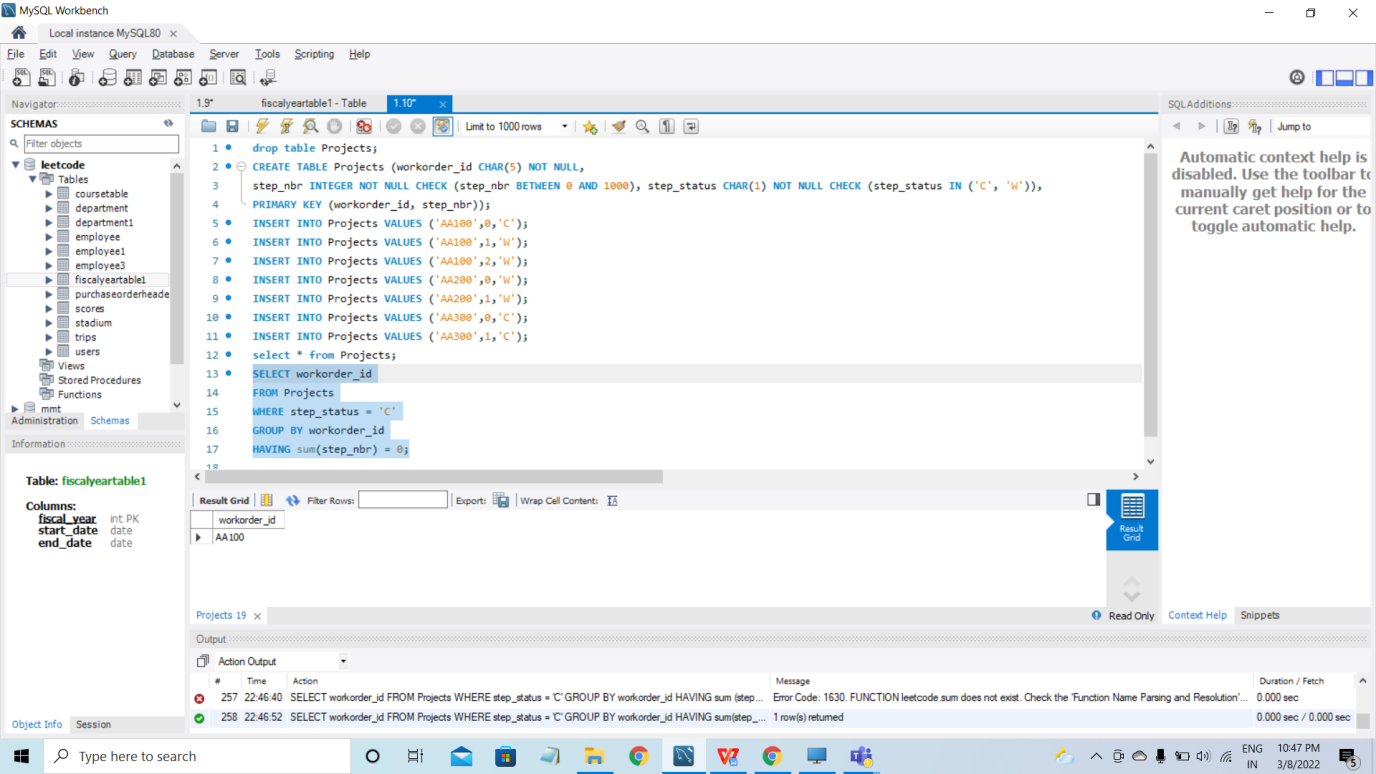
FROM Projects

WHERE step\_status = 'C'

GROUP BY workorder\_id

HAVING sum(step\_nbr) = 0;

**Output Screenshot** :-



1. Create the following table

Create Table coursetable

(

CourseName varchar(50),

CourseCategory varchar(50),

Price int

)

Insert into coursetable values('C', 'PROGRAMMING', 5000) ;

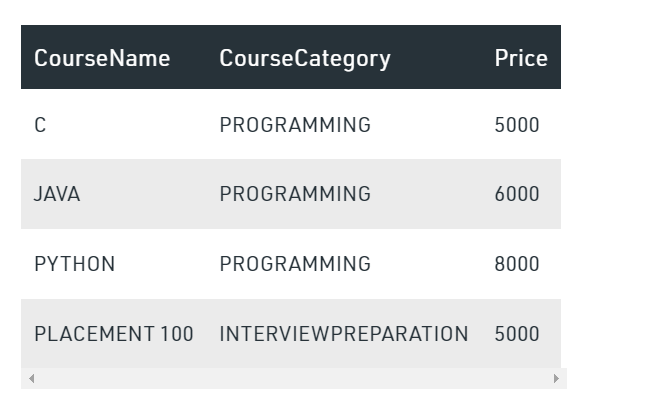
Insert into coursetable values('JAVA', 'PROGRAMMING', 6000) ;

Insert into coursetable values('PYTHON', 'PROGRAMMING', 8000) ;

Insert into coursetable values('PLACEMENT 100', 'INTERVIEWPREPARATION', 5000) ;

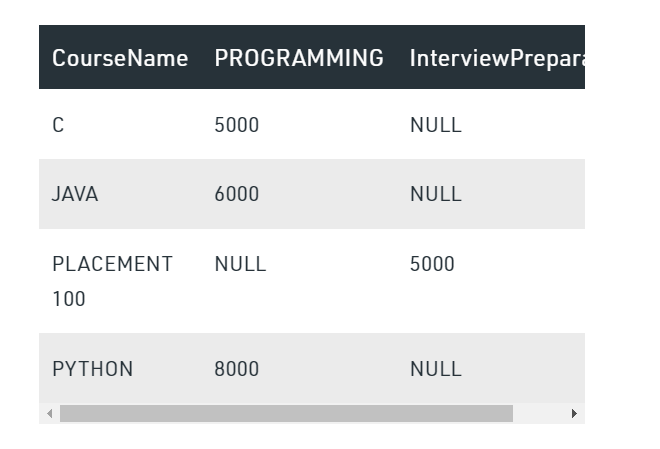
SELECT \* FROM coursetable ;

Following will be the output



What query can you write to get the following output ?

Hint : Use pivot table



Solution:

SELECT CourseName,

(case

when CourseCategory='PROGRAMMING' and price=5000 then 5000

when CourseCategory='PROGRAMMING' and price=6000 then 6000

when CourseCategory='PROGRAMMING' and price=8000 then 8000

else 'null'

end ) as PROGRAMMING,

(case when CourseCategory='INTERVIEWPREPARATION' and price=5000 then 5000 else null end) as INTERVIEWPREPARATION

FROM coursetable

**Output Screenshot** :-

