City-Dataset:https://docs.google.com/spreadsheets/d/1dk9kRwcMxj5USuJqxtITD05S-aOUD6fzNzV W41dcpgc/edit?usp=sharing

**Table for city data**

create table if not exists city

(

id int,

name varchar(17),

countrycode varchar(3),

district varchar(20),

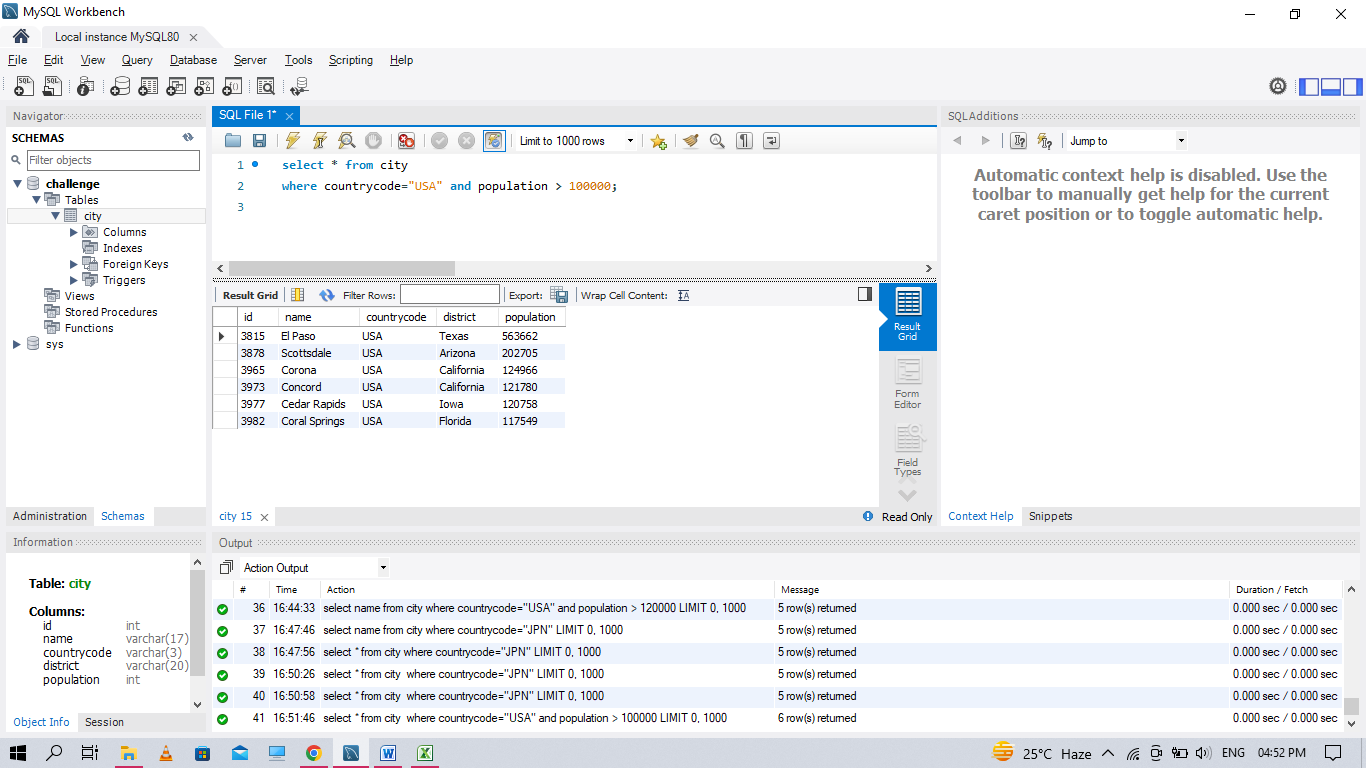
population int

);

Q1. Query all columns for all American cities in the CITY table with populations larger than 100000. The CountryCode for America is USA.

select \* from city

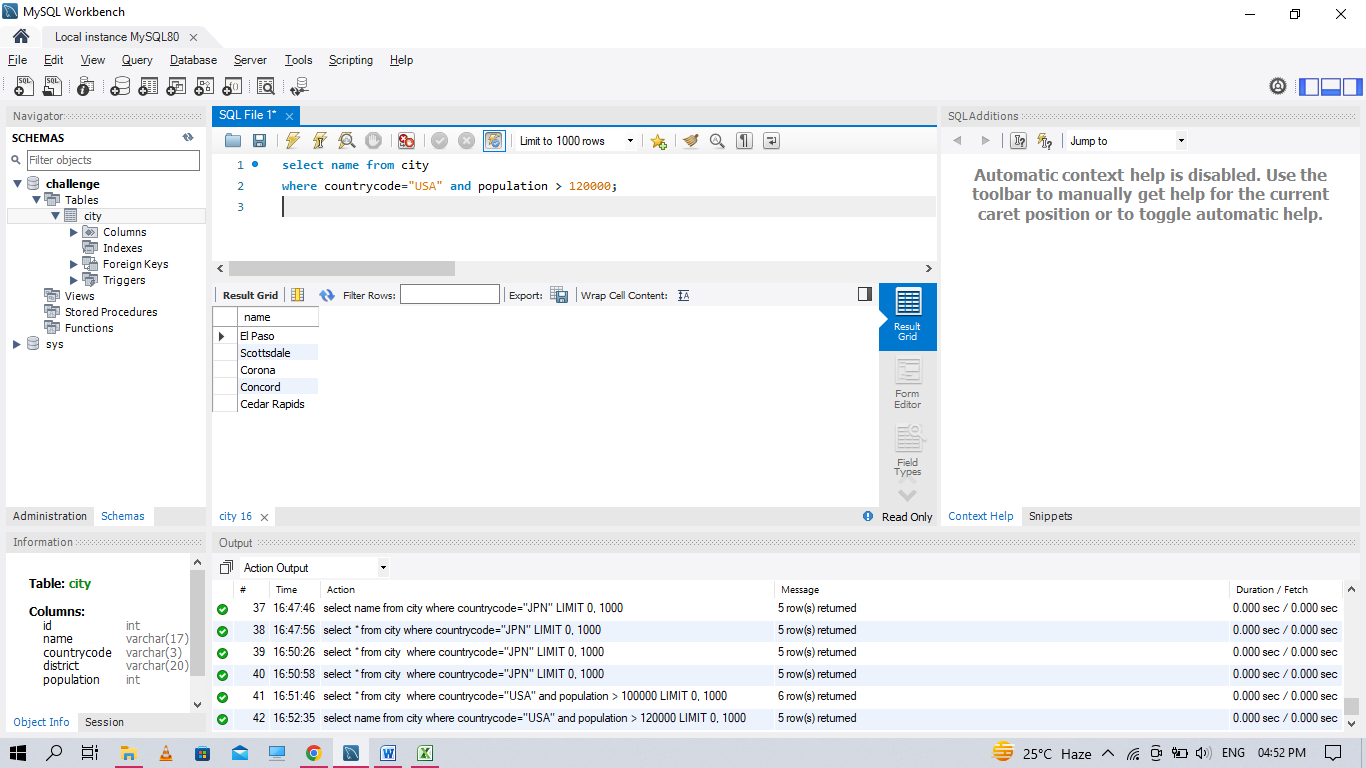
where countrycode="USA" and population > 100000;



Q2. Query the NAME field for all American cities in the CITY table with populations larger than 120000. The CountryCode for America is USA.

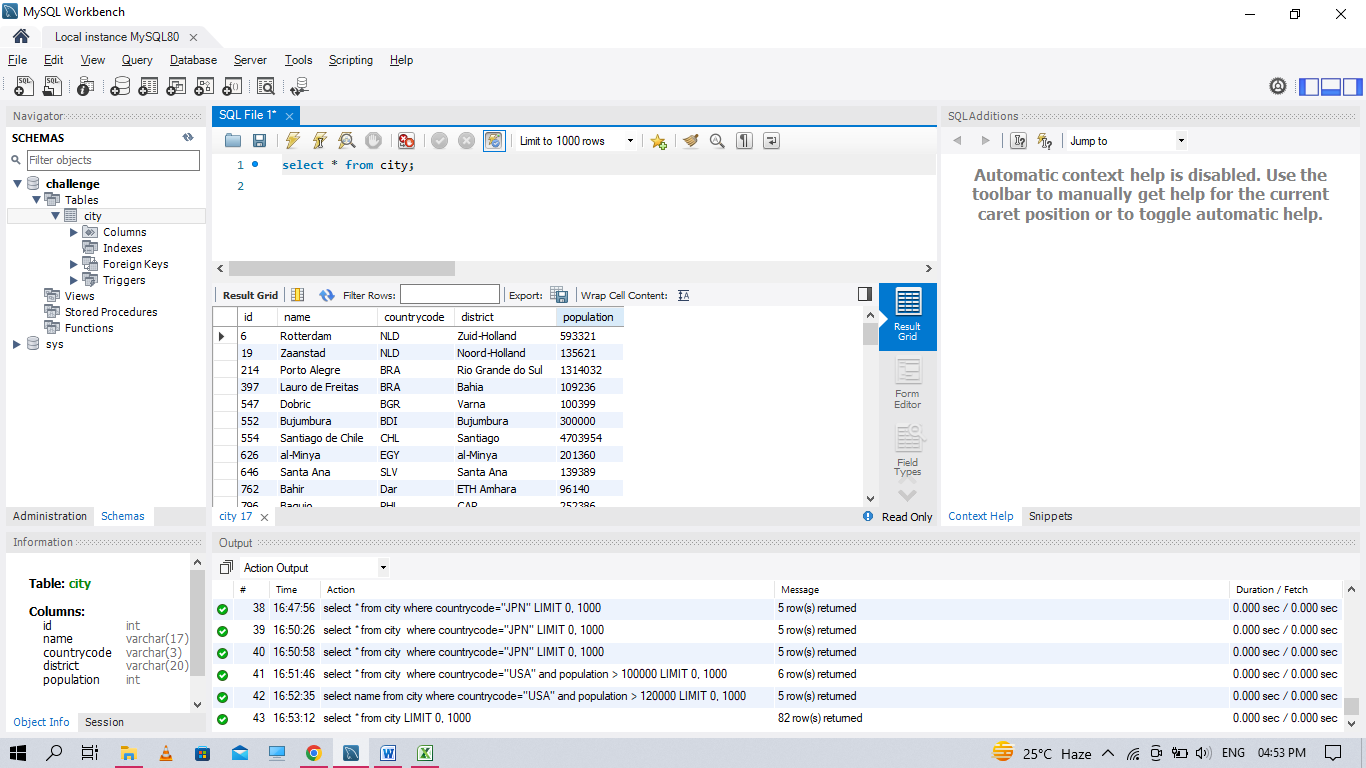
select name from city

where countrycode="USA" and population > 120000;



Q3. Query all columns (attributes) for every row in the CITY table. The CITY table is described as follows:

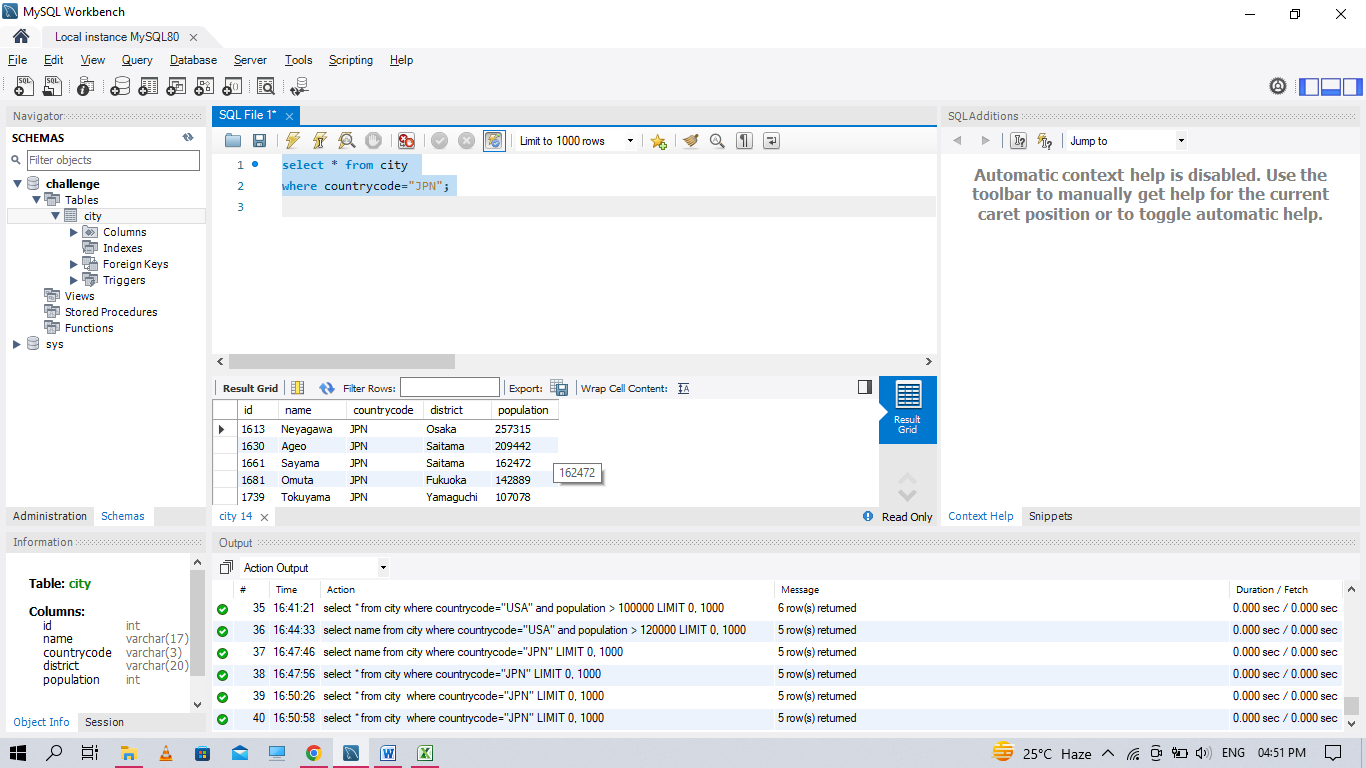
select \* from city ;



Q5. Query all attributes of every Japanese city in the CITY table. The COUNTRYCODE for Japan is JPN.

select \* from city

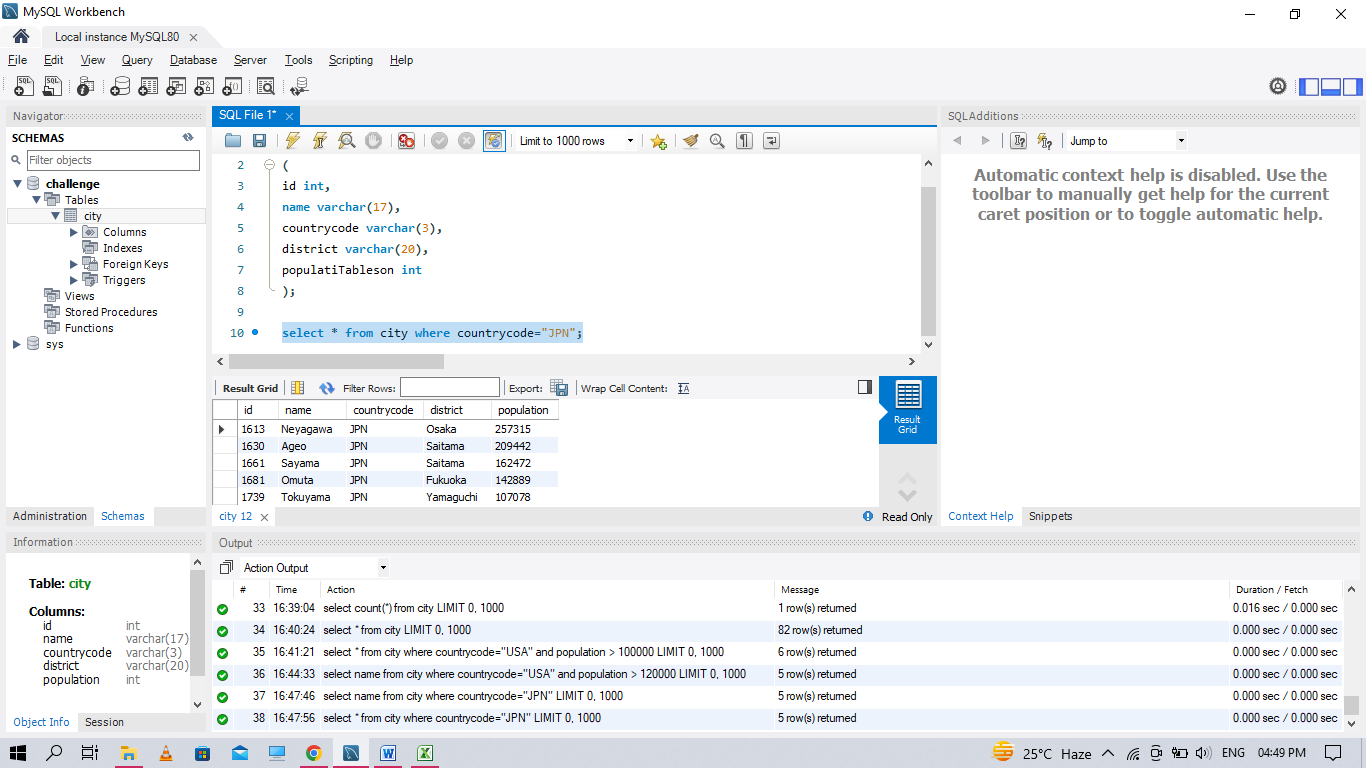
where countrycode="JPN";



Q6. Query the names of all the Japanese cities in the CITY table. The COUNTRYCODE for Japan is JPN.

select name from city

where countrycode="JPN";



station-table:https://docs.google.com/spreadsheets/d/1sHPhE7walQD5mL7ppFNqybyoOJY3E51N0 cWYzhp2UH4/edit?usp=sharing

**Table for station data**

create table if not exists station

(

id int,

city varchar(21),

state varchar(2),

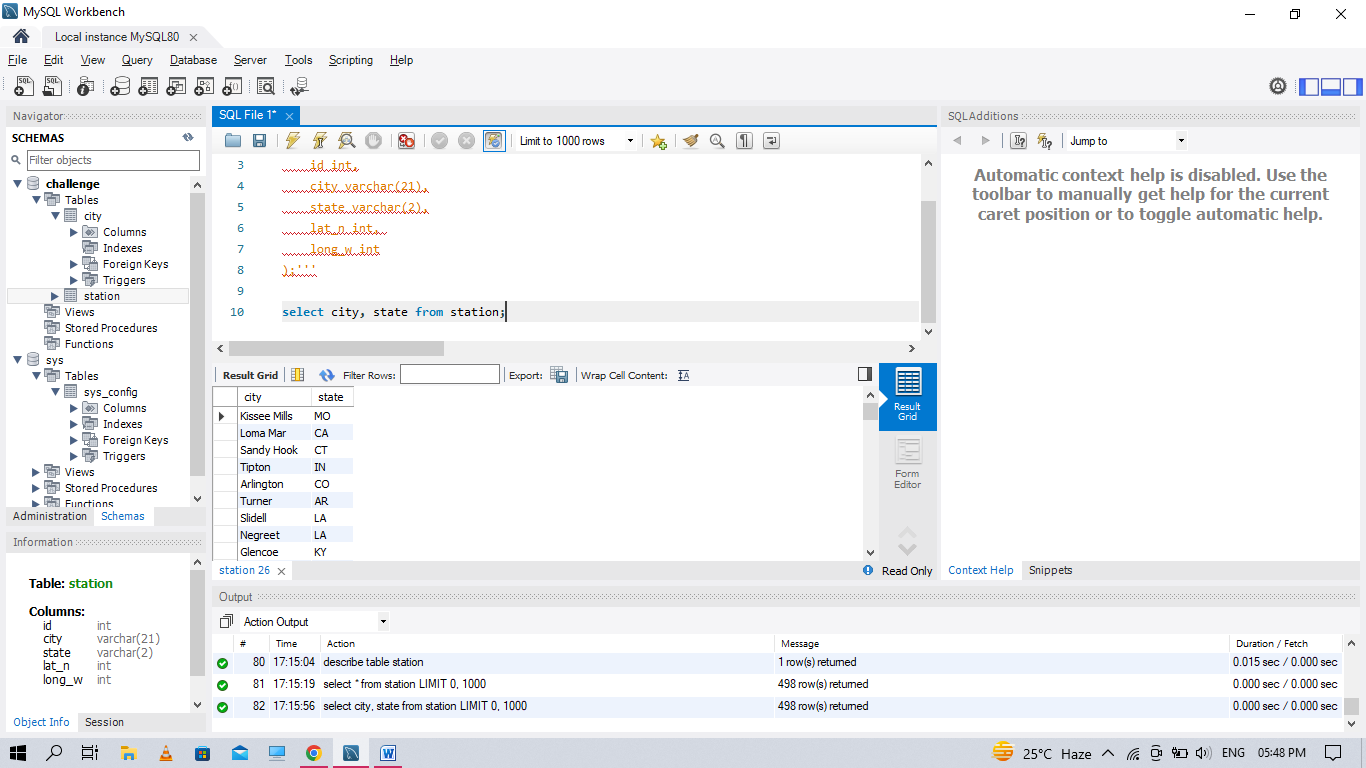
lat\_n number,

long\_w number

)

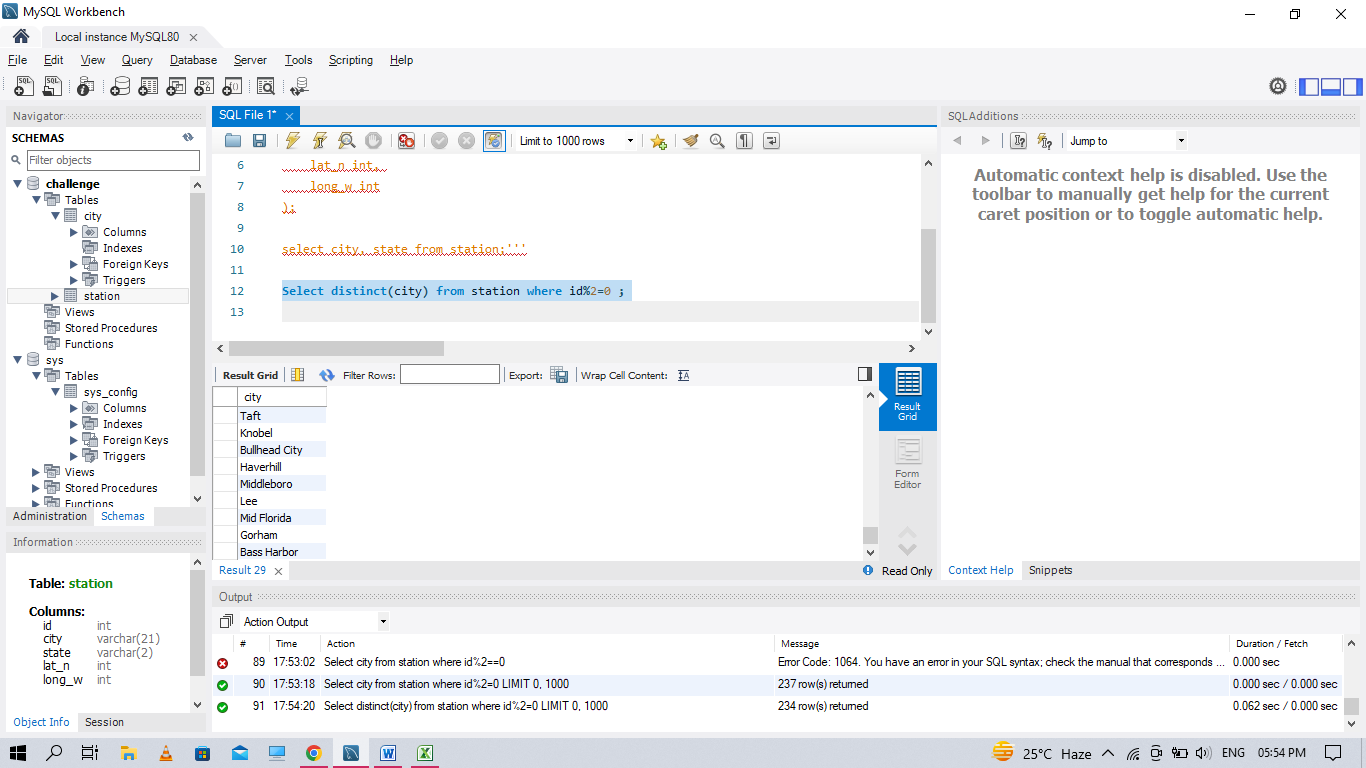
Q7. Query a list of CITY and STATE from the STATION table. The STATION table is described as follows:

select city, state from station;



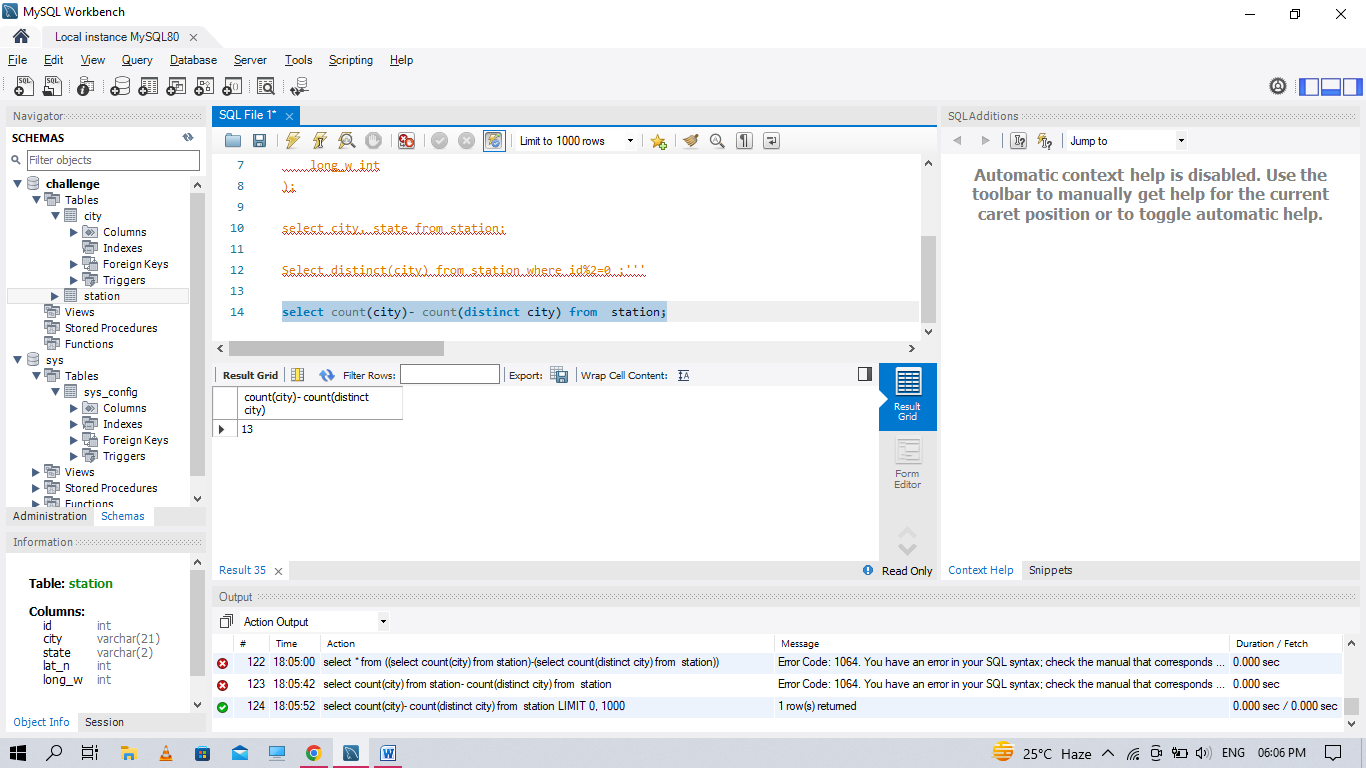
Q8. Query a list of CITY names from STATION for cities that have an even ID number. Print the results in any order, but exclude duplicates from the answer.

Select distinct(city) from station where id%2=0 ;



Q9. Find the difference between the total number of CITY entries in the table and the number of distinct CITY entries in the table.

select count(city)- count(distinct city) from station;

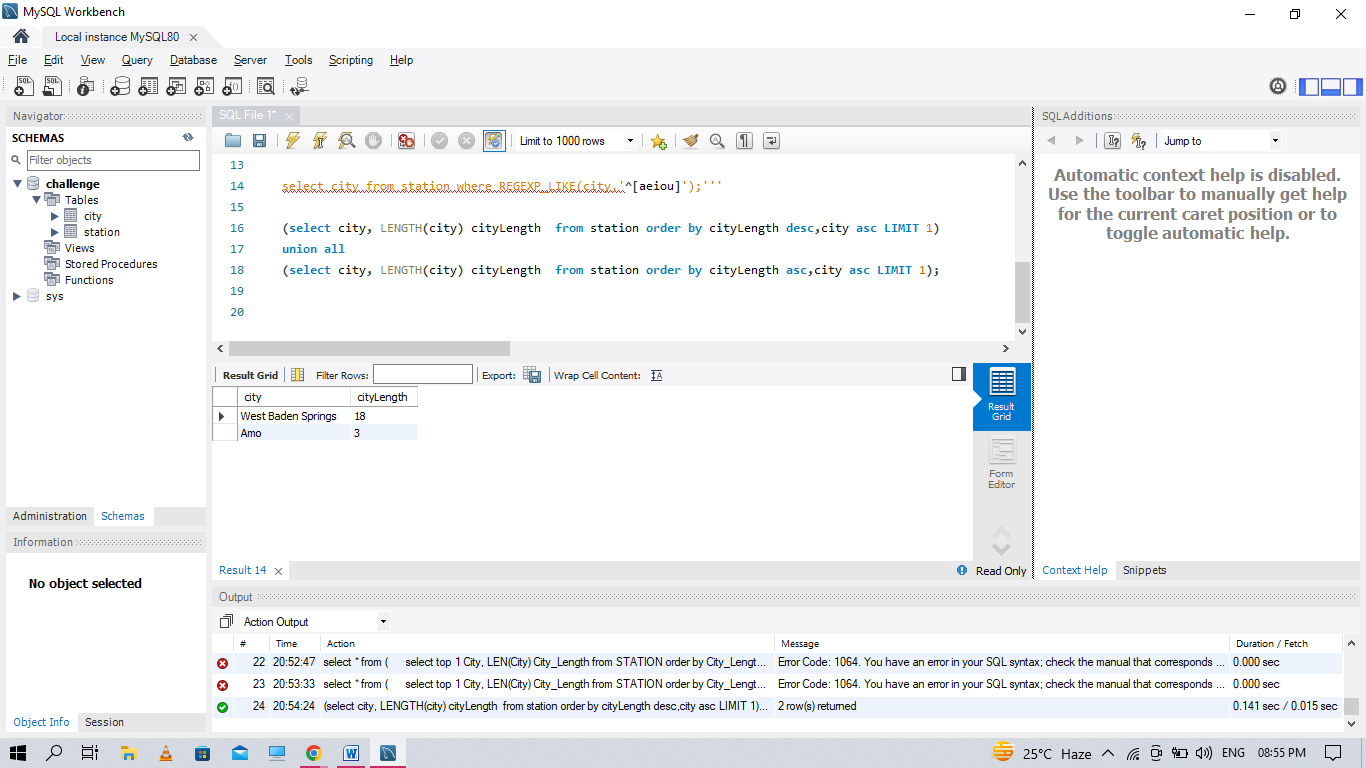


Q10. Query the two cities in STATION with the shortest and longest CITY names, as well as their respective lengths (i.e.: number of characters in the name). If there is more than one smallest or largest city, choose the one that comes first when ordered alphabetically.

(select city, LENGTH(city) cityLength from station order by cityLength desc,city asc LIMIT 1)

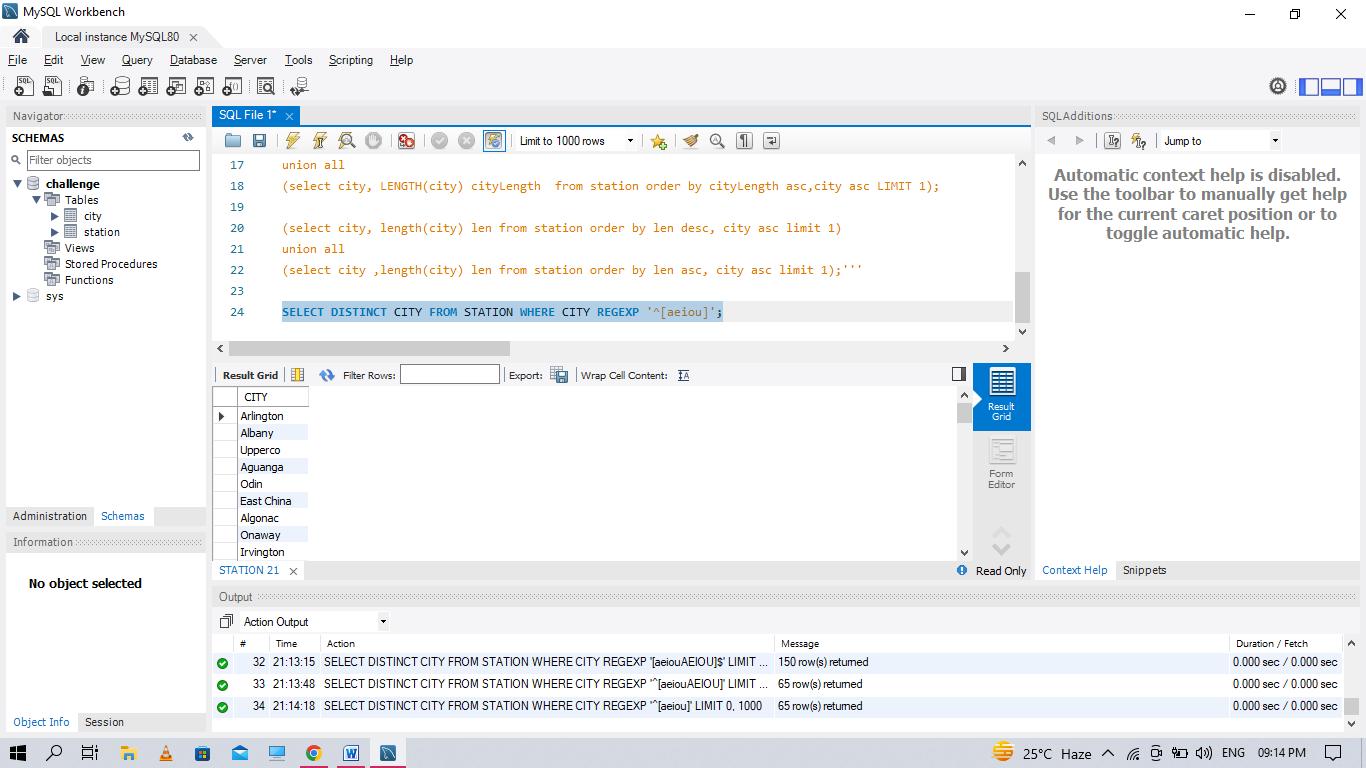
union all

(select city, LENGTH(city) cityLength from station order by cityLength asc,city asc LIMIT 1)



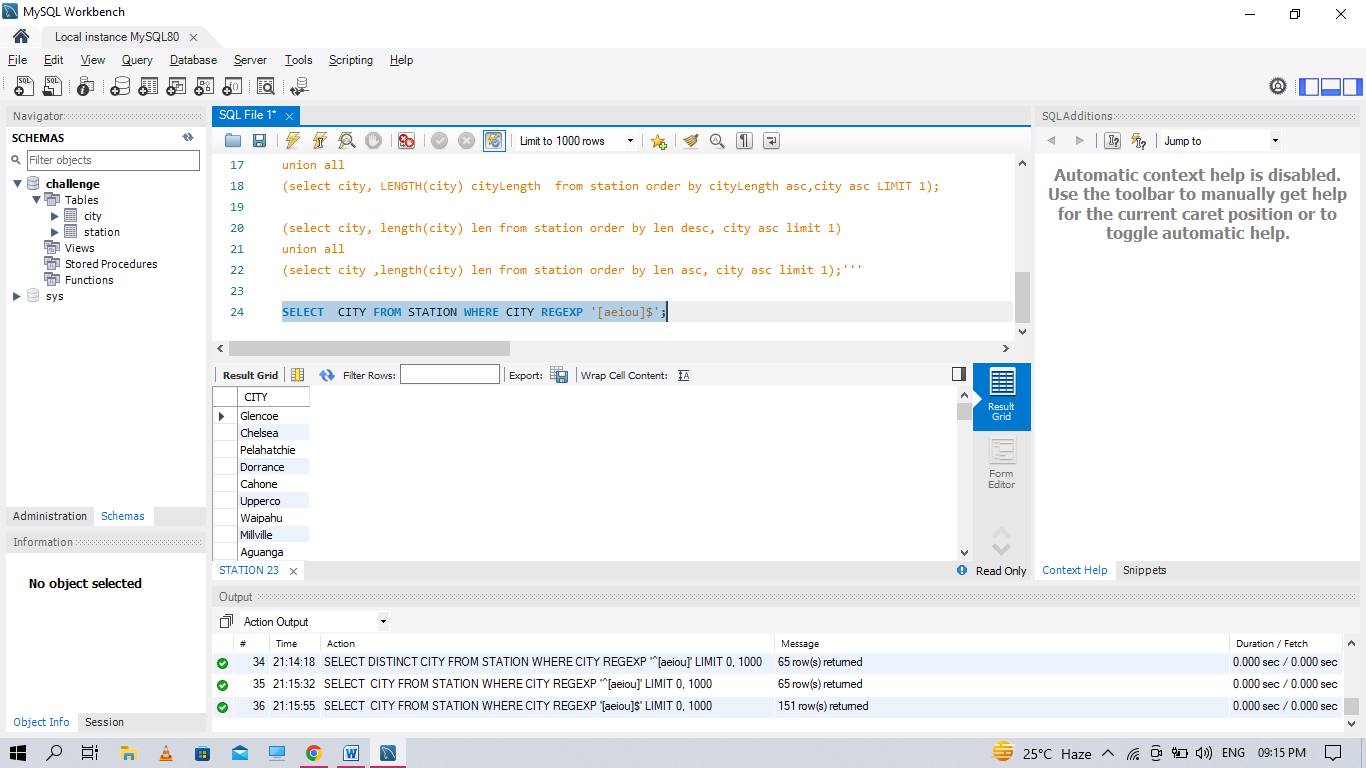
Q11. Query the list of CITY names starting with vowels (i.e., a, e, i, o, or u) from STATION. Your result cannot contain duplicates.

SELECT DISTINCT CITY FROM STATION WHERE CITY REGEXP '^[aeiou]';



Q12. Query the list of CITY names ending with vowels (a, e, i, o, u) from STATION. Your result cannot contain duplicates.

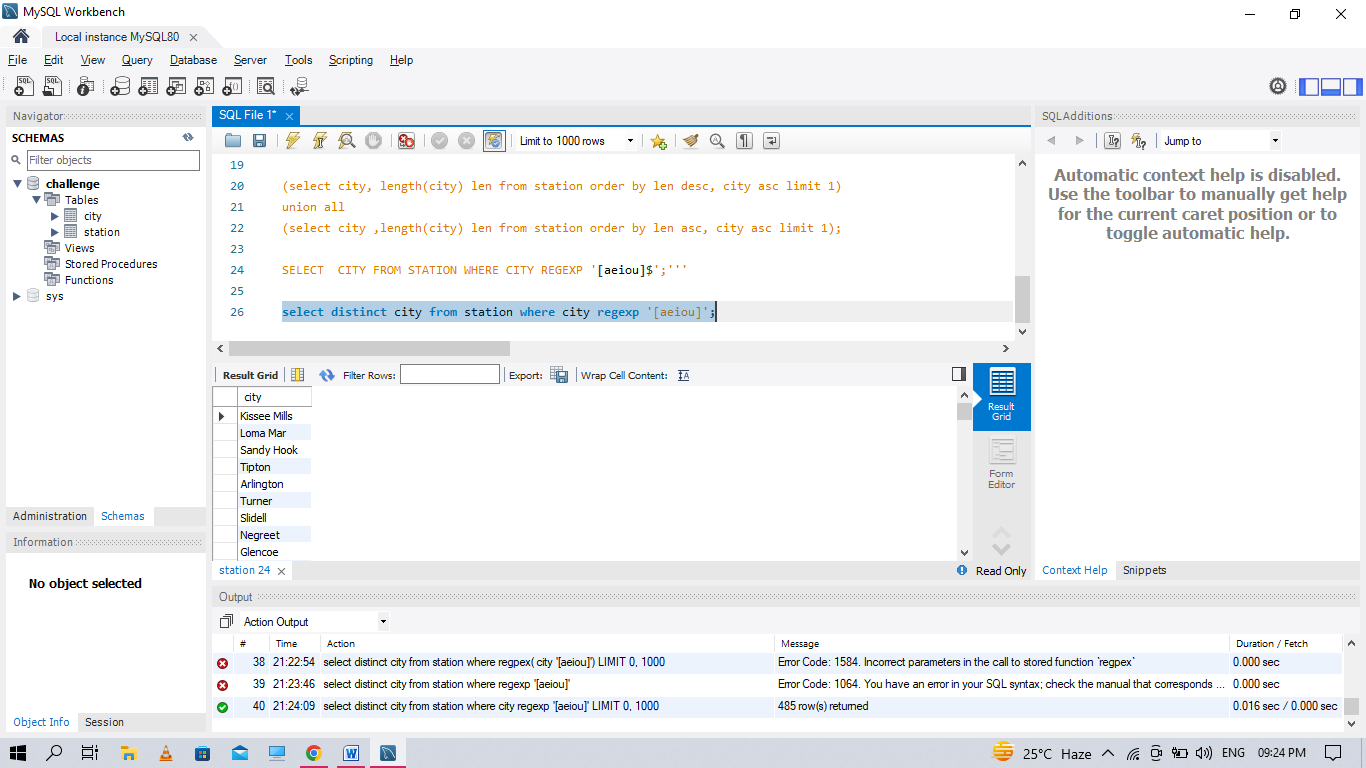
SELECT CITY FROM STATION WHERE CITY REGEXP '[aeiou]$';



Q13.Query the list of CITY names from STATION that do not start with vowels. Your result cannot contain duplicates.

select distinct city from station

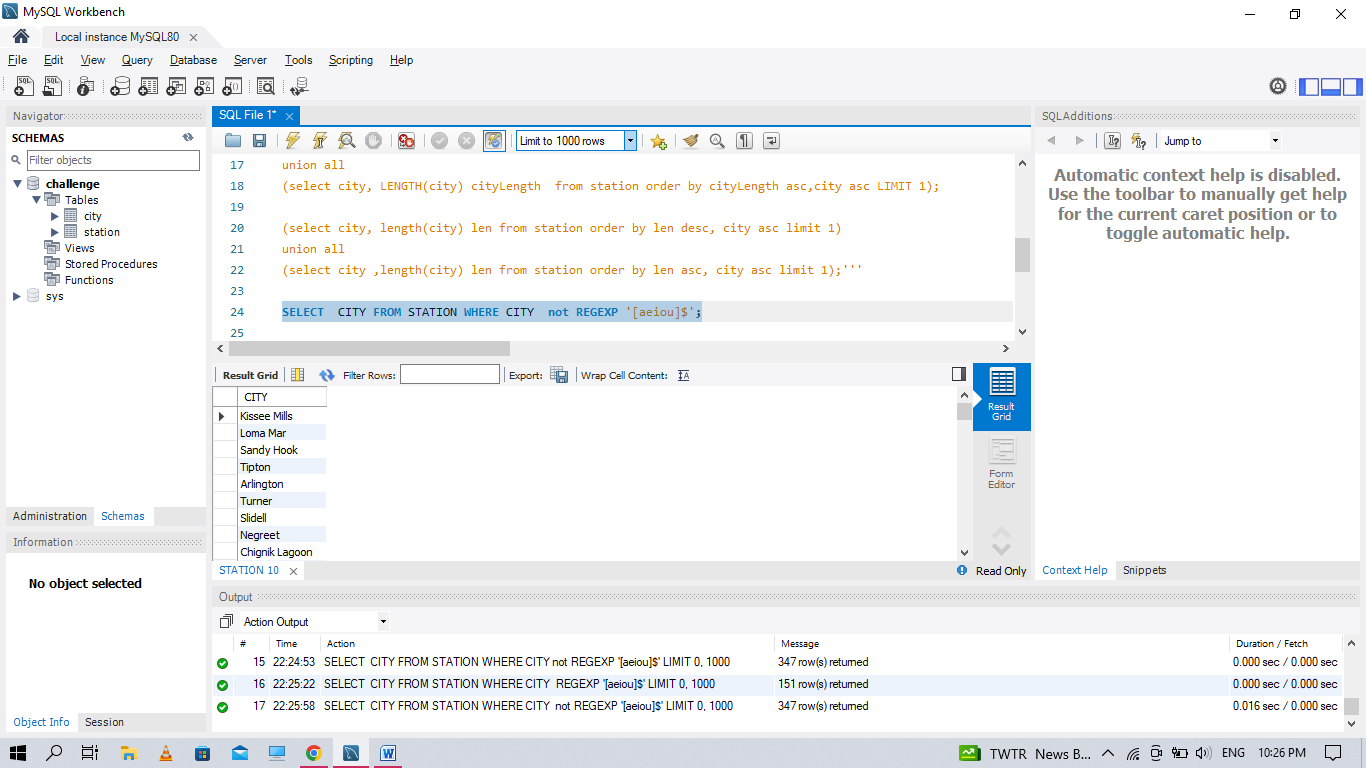
where city regexp '[aeiou]';



Q14. Query the list of CITY names from STATION that do not end with vowels. Your result cannot contain duplicates.

SELECT CITY FROM STATION

WHERE CITY not REGEXP '[aeiou]$';



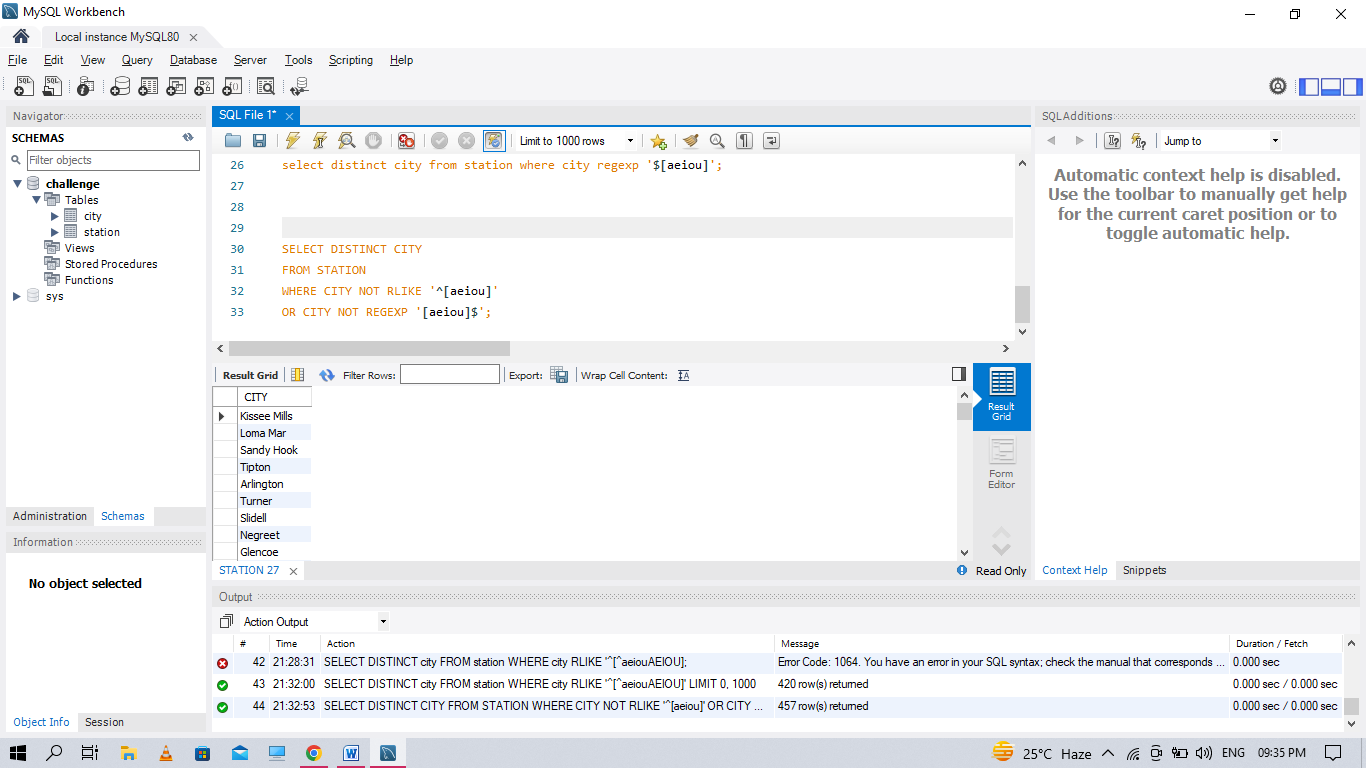
Q15. Query the list of CITY names from STATION that either do not start with vowels or do not end with vowels. Your result cannot contain duplicates.

SELECT DISTINCT CITY

FROM STATION

WHERE CITY NOT RLIKE '^[aeiou]'

OR CITY NOT REGEXP '[aeiou]$';



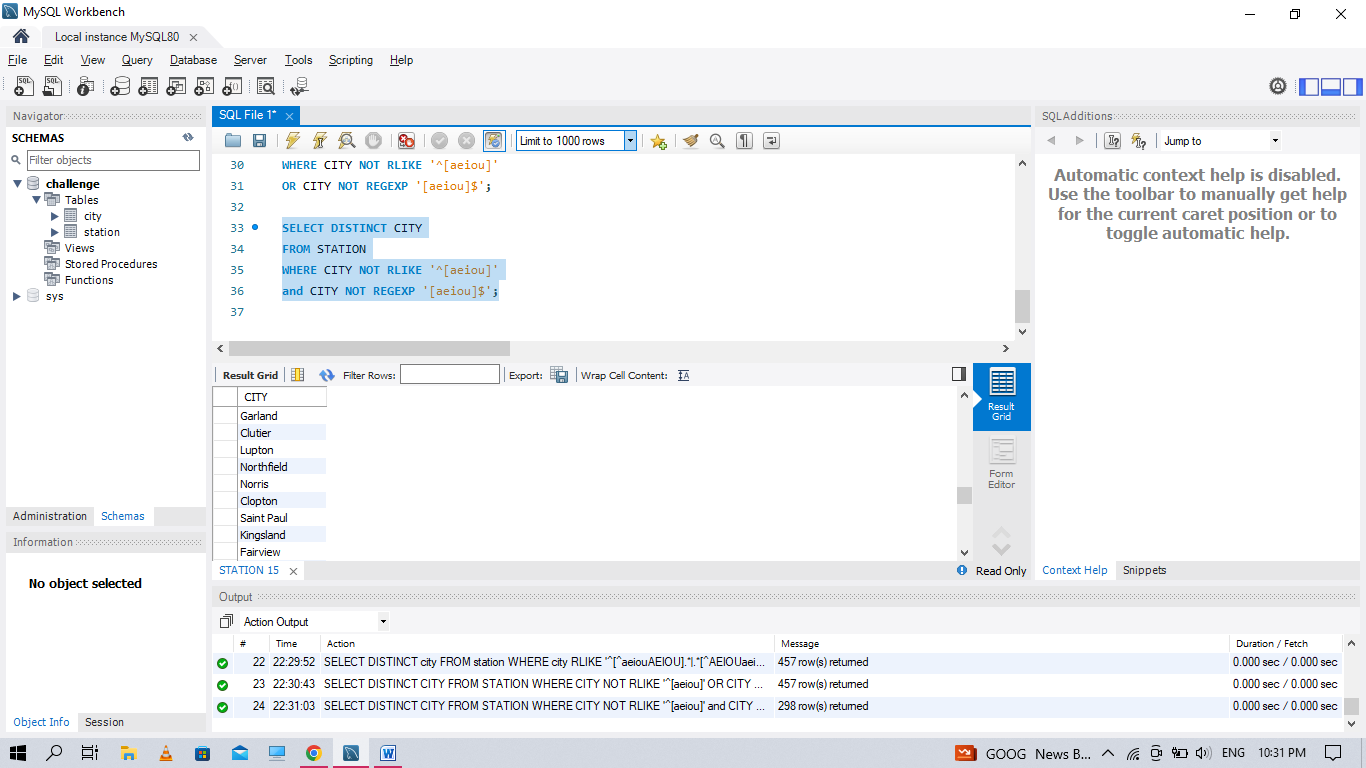
Q16. Query the list of CITY names from STATION that do not start with vowels and do not end with vowels. Your result cannot contain duplicates.

SELECT DISTINCT CITY

FROM STATION

WHERE CITY NOT RLIKE '^[aeiou]'

and CITY NOT REGEXP '[aeiou]$';



**Q17. Table: Product**

create table product

(

product\_id int,

product\_name varchar(20),

unit\_price int,

primary key(product\_id))

insert into product values(1,'S8',1000),

(2,'G4',800),

(3,'IPhone',1400)

**Table : sales**

create table sales(

seller\_id int,

product\_id int,

buyer\_id int,

sale\_date date,

quantity int,

price int,

constraint fk foreign key (product\_id) references product(product\_id)

);

insert into sales values(1,1,2,'2019-01-21',2,2000),

(1,2,2,'2019-02-17',1,800),

(2,2,3,'2019-06-02',1,800),

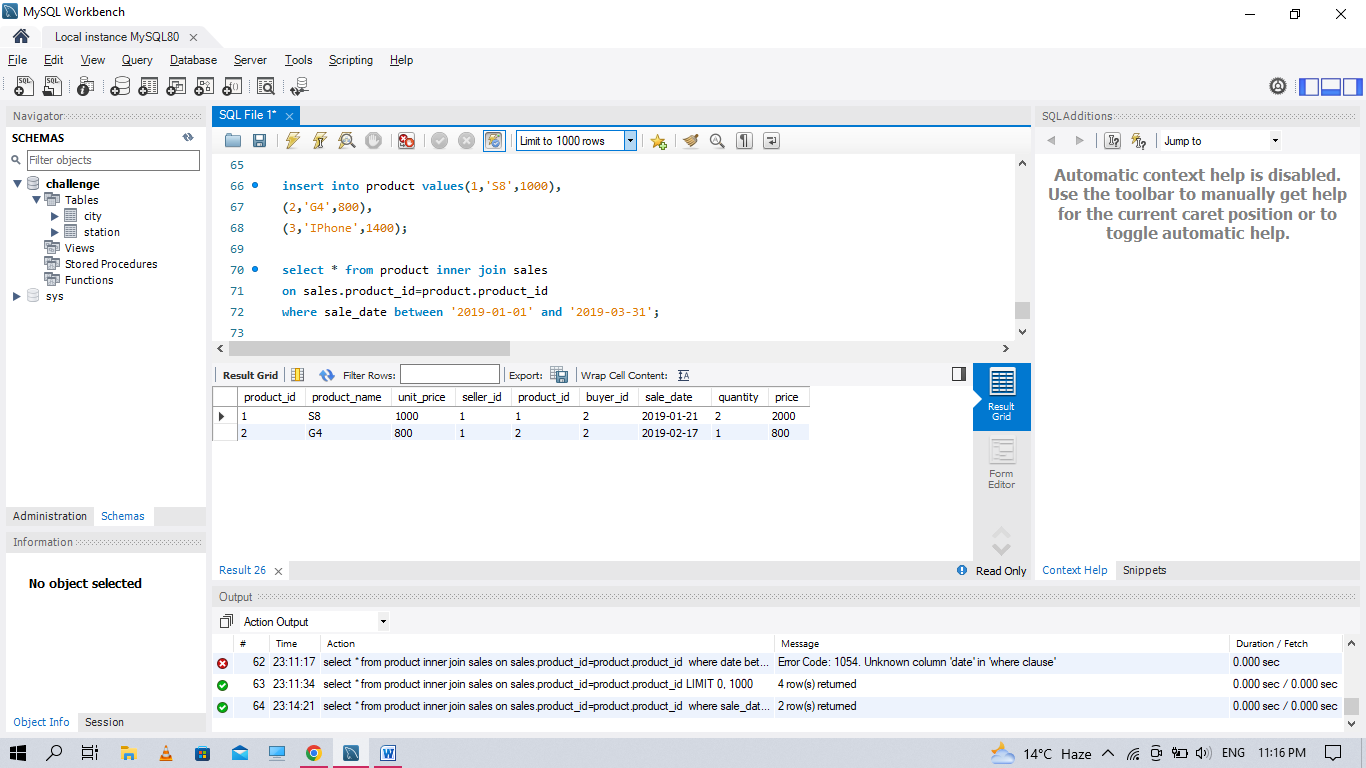
(3,3,4,'2019-05-13',2,2800)

Write an SQL query that reports the products that were only sold in the first quarter of 2019. That is, between 2019-01-01 and 2019-03-31 inclusive.

select \* from product inner join sales

on sales.product\_id=product.product\_id

where sale\_date between '2019-01-01' and '2019-03-31';



**Q18. Table: Views**

create table views

(

article\_id int,

author\_id int,

viewer\_id int,

view\_date date

);

insert into views values

(1 ,3, 5 ,'2019-08-01'),

(1 ,3 ,6,'2019-08-02'),

(2 ,7 ,7,'2019-08-01'),

(2 ,7 ,6,'2019-08-02'),

(4 ,7 ,1,'2019-07-22'),

(3 ,4 ,4,'2019-07-21'),

(3 ,4 ,4,'2019-07-21');

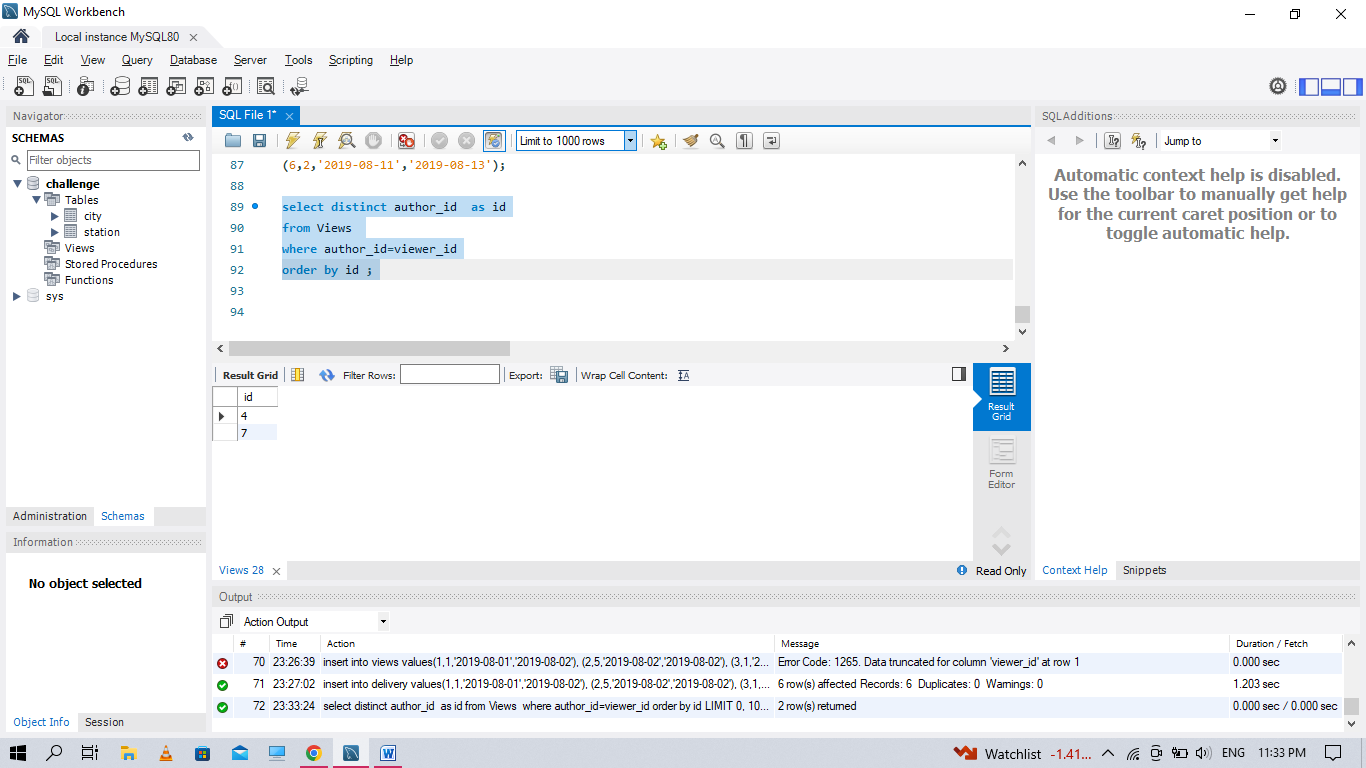
Write an SQL query to find all the authors that viewed at least one of their own articles. Return the result table sorted by id in ascending order.

select distinct author\_id as id

from Views

where author\_id=viewer\_id

order by id ;



**Q19. Table: Delivery**

create table delivery

(

delivery\_id int,

customer\_id int,

order\_date date,

customer\_pref\_delivery\_date date

);

insert into delivery values(1,1,'2019-08-01','2019-08-02'),

(2,5,'2019-08-02','2019-08-02'),

(3,1,'2019-08-11','2019-08-11'),

(4,3,'2019-08-24','2019-08-26'),

(5,4,'2019-08-21','2019-08-22'),

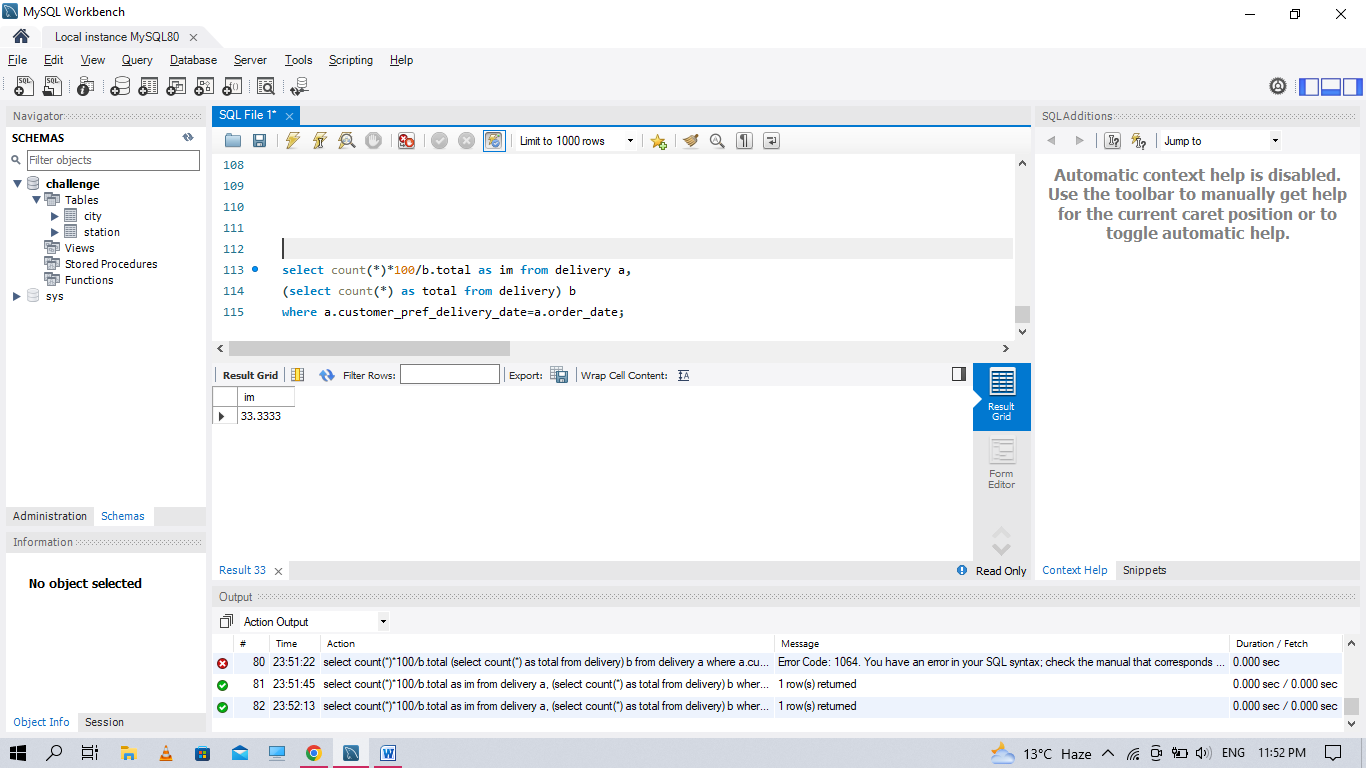
(6,2,'2019-08-11','2019-08-13');

If the customer's preferred delivery date is the same as the order date, then the order is called immediately; otherwise, it is called scheduled. Write an SQL query to find the percentage of immediate orders in the table, rounded to 2 decimal places.

select count(\*)\*100.0/b.total as immdediate\_percentage from Delivery a,

(select count(\*) as total from Delivery) b

where a.order\_date = a.customer\_pref\_delivery\_date;



**Q20. Table: Ads**

create table ads

(

ad\_id int,

user\_id int,

action\_ enum('Viewed','ignored','clicked'),

constraint pk primary key(ad\_id,user\_id)

);

insert into ads values

(1 ,1, 'Clicked'),

(2 ,2, 'Clicked'),

(3 ,3, 'Viewed'),

(5 ,5, 'Ignored'),

(1 ,7, 'Ignored'),

(2 ,7, 'Viewed'),

(3 ,5, 'Clicked'),

(1 ,4, 'Viewed'),

(2 ,11, 'Viewed'),

(1 ,2, 'Clicked');

Write an SQL query to find the ctr of each Ad. Round ctr to two decimal points. Return the result table ordered by ctr in descending order and by ad\_id in ascending order in case of a tie

**Q21. Table: Employee**

create table employee

(

employee\_id int,

team\_id int

);

insert into employee values

(1 ,8),

(2 ,8),

(3 ,8),

(4 ,7),

(5 ,9),

(6 ,9);

Write an SQL query to find the team size of each of the employees. Return result table in any order.

SELECT employee\_id, COUNT(team\_id) OVER (PARTITION BY team\_id) team\_size

FROM Employee;



**Q22. Table: Countries**

create table countries

(

country\_id int,

country\_name varchar(20)

);

insert into countries values

(2, 'USA'),

(3, 'Australia'),

(7, 'Peru'),

(5, 'China'),

(8, 'Morocco'),

(9, 'Spain');

**Table: Weather**

create table Weather

(

country\_id int,

weather\_state int,

day date

PRIMARY KEY (country\_id ,DAY)

);

insert into weather values

(2, 12, '2019-10-27'),

(3, -2, '2019-11-10'),

(3, 0, '2019-11-11'),

(3, 3, '2019-11-12'),

(5, 16, '2019-11-07'),

(5, 18, '2019-11-09'),

(5, 21, '2019-11-23'),

(7, 25, '2019-11-28'),

(7 ,22, '2019-12-01'),

(7 ,20, '2019-12-02'),

(8 ,25, '2019-11-05'),

(8 ,27,'2019-11-15'),

(8 ,31,'2019-11-25'),

(9 ,7, '2019-10-23'),

(9 ,3, '2019-12-23');

Write an SQL query to find the type of weather in each country for November 2019. The type of weather is:

● Cold if the average weather\_state is less than or equal 15,

● Hot if the average weather\_state is greater than or equal to 25, and

● Warm otherwise.

select country\_name,

case

when avg(weather\_state) <= 15 then "Cold"

when avg(weather\_state) >= 25 then "Hot"

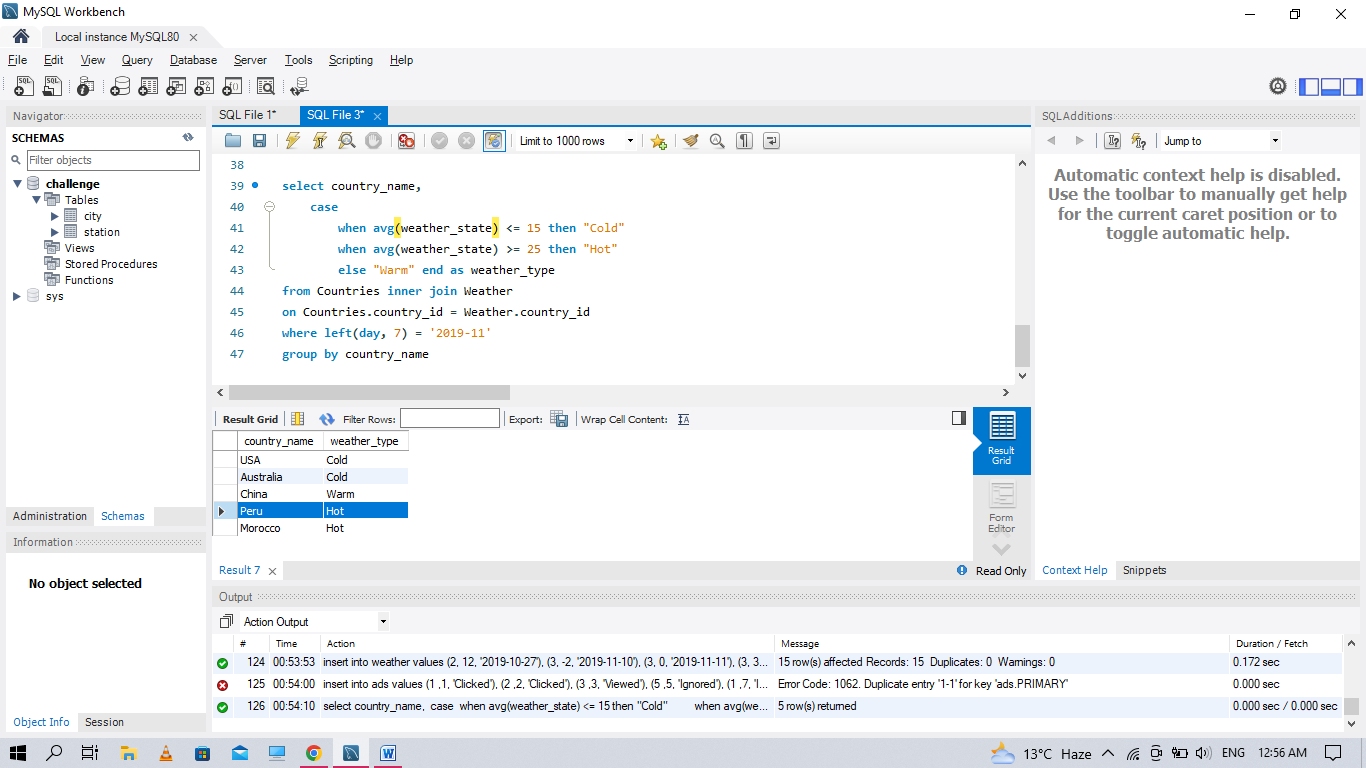
else "Warm" end as weather\_type

from Countries inner join Weather

on Countries.country\_id = Weather.country\_id

where left(day, 7) = '2019-11'

group by country\_name



**Q23 Table: Prices**

CREATE TABLE PRICES

(

product\_id int,

start\_date date,

end\_date date,

price int

);

**Table: UnitsSold**

CREATE TABLE UNITDSOLD

(product\_id int,

purchase\_date date,

units int);

select product\_id, ifnull(round(sum(prices\_sum) / sum(units), 2), 0) as average\_price

from (

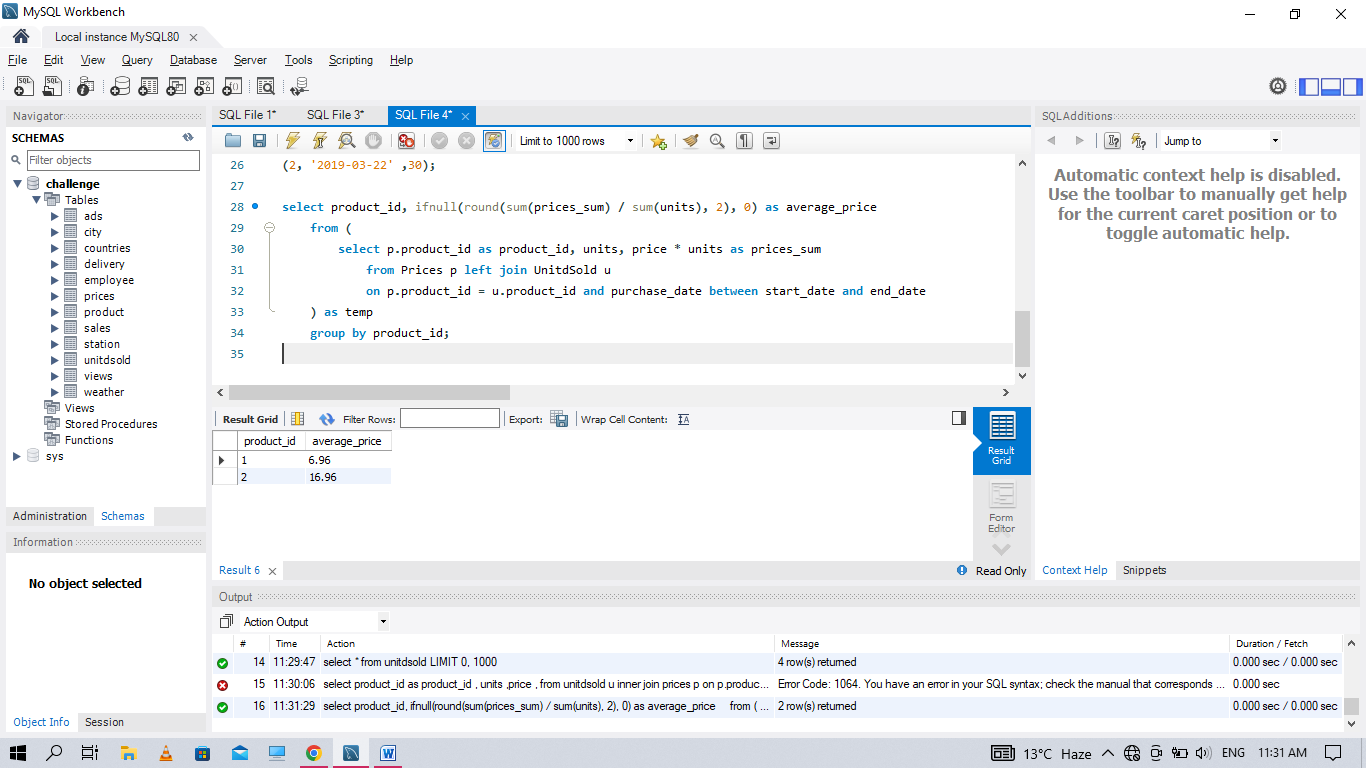
select p.product\_id as product\_id, units, price \* units as prices\_sum

from Prices p inner join UnitdSold u

on p.product\_id = u.product\_id and purchase\_date between start\_date and end\_date

) as temp

group by product\_id;



**Q24. Table: Activity**

create table activity

(

player\_id int,

device\_id int,

event\_date date,

games\_played int,

constraint pk 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);

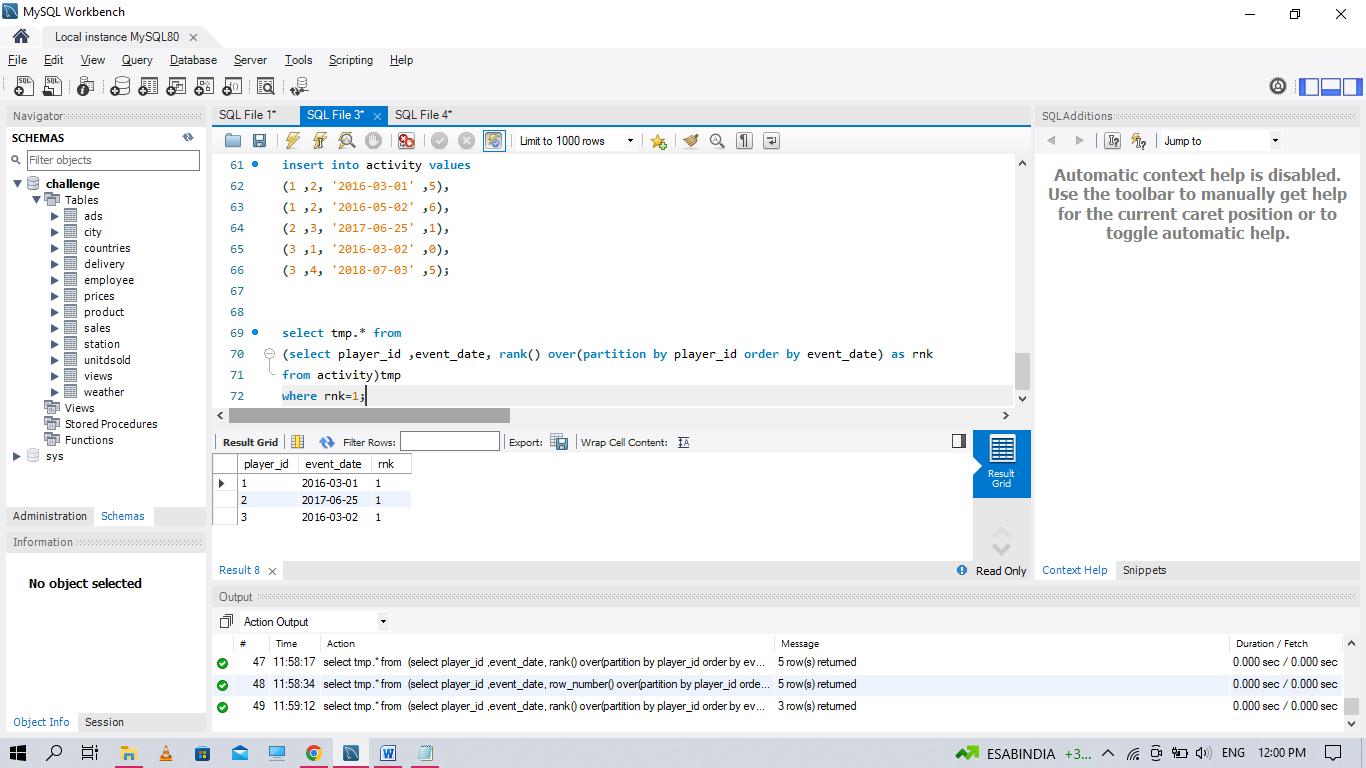
**Write an SQL query to report the first login date for each player. Return the result table in any order**

**select tmp.\* from**

(select player\_id ,event\_date, rank() over(partition by player\_id order by event\_date) as rnk

from activity)tmp

where rnk=1;



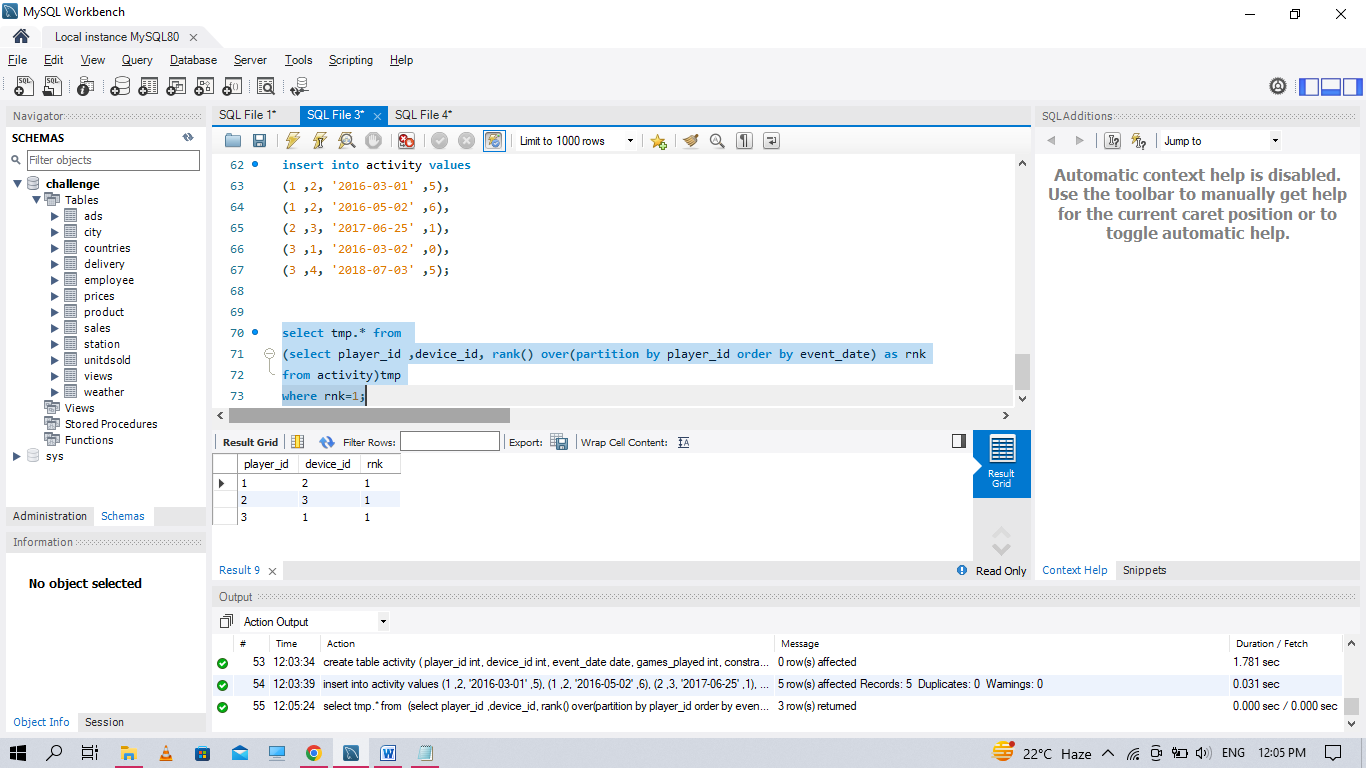
**Q25. Write an SQL query to report the device that is first logged in for each player. Return the result table in any order**

select tmp.\* from

(select player\_id ,device\_id, rank() over(partition by player\_id order by event\_date) as rnk

from activity)tmp

where rnk=1;

**.**

**Q26 Write an SQL query to get the names of products that have at least 100 units ordered in February 2020 and their amount..**

**Table: Products**

create table products

(

product\_id int,

product\_name varchar(30),

product\_category varchar(30),

constraint pk primary key(product\_id)

);

insert into products values

(1,'Leetcode Solutions' ,'Book'),

(2,'Jewels of Stringologys' ,'Book'),

(3, 'HP', 'Laptop'),

(4,'Lenovo', 'Laptop'),

(5,'Leetcode', 'Kit T-shirt')

create table orders

(

product\_id int,

order\_date date,

unit int,

constraint fork foreign key (product\_id) references products(product\_id))

insert into orders values

(1 ,'2020-02-05', 60),

(1 ,'2020-02-10', 70),

(2 ,'2020-01-18', 30),

(2 ,'2020-02-11', 80),

(3 ,'2020-02-17', 2),

(3 ,'2020-02-24', 3),

(4 ,'2020-03-01', 20),

(4 ,'2020-03-04', 30),

(4 ,'2020-03-04', 60),

(5 ,'2020-02-25', 50),

(5 ,'2020-02-27', 50),

(5 ,'2020-03-01', 50);

select tmp.\* from

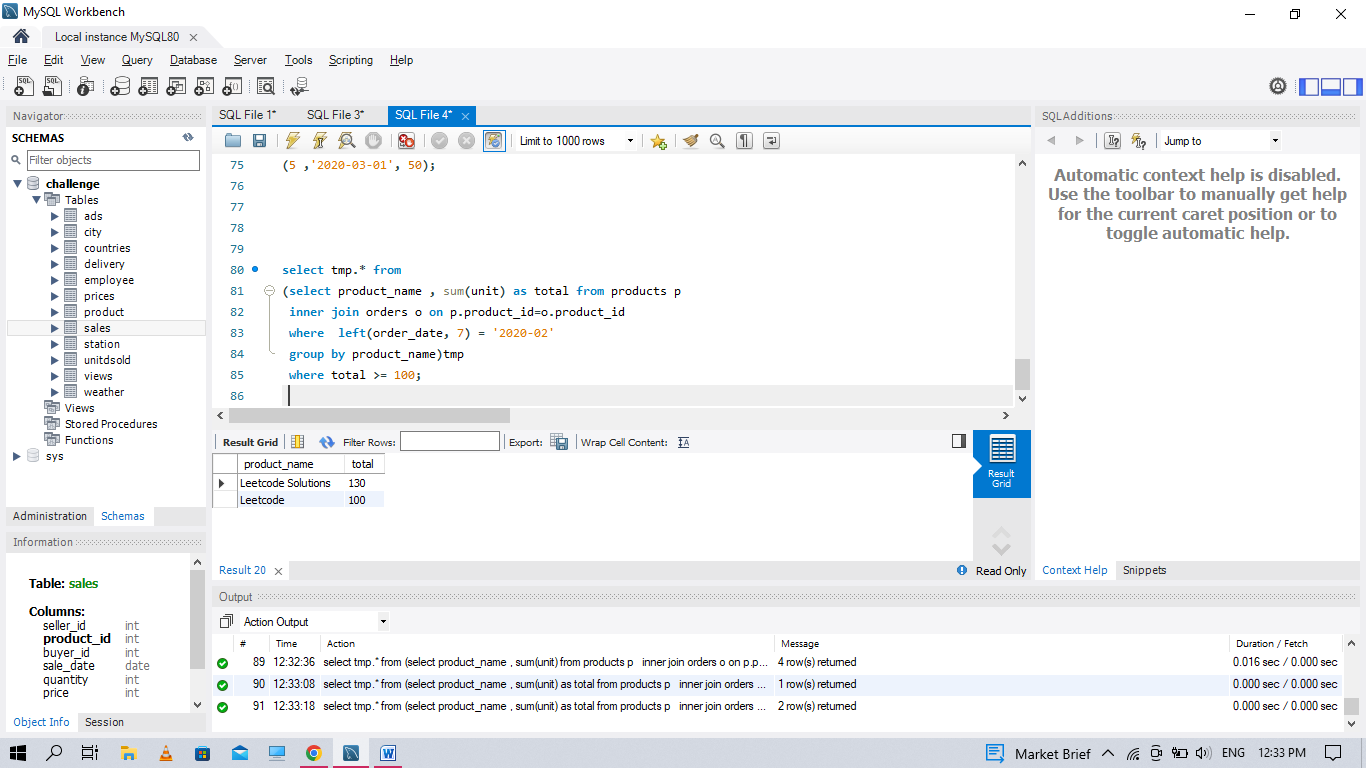
(select product\_name , sum(unit) as total from products p

inner join orders o on p.product\_id=o.product\_id

where left(order\_date, 7) = '2020-02'

group by product\_name)tmp

where total >= 100;



**Q27. Table: Users**

create table users(

user\_id int,

name varchar(30),

mail varchar(50),

constraint pkk primary key(user\_id)

);

insert into users values

(1, 'Winston' ,'winston@leetcode.com'),

(2, 'Jonathan' ,'jonathanisgreat'),

(3, 'Annabelle' ,'bella-@leetcode.com'),

(4, 'Sally' ,'sally.come@leetcode.com'),

(5, 'Marwan' ,'quarz#2020@leetcode.com'),

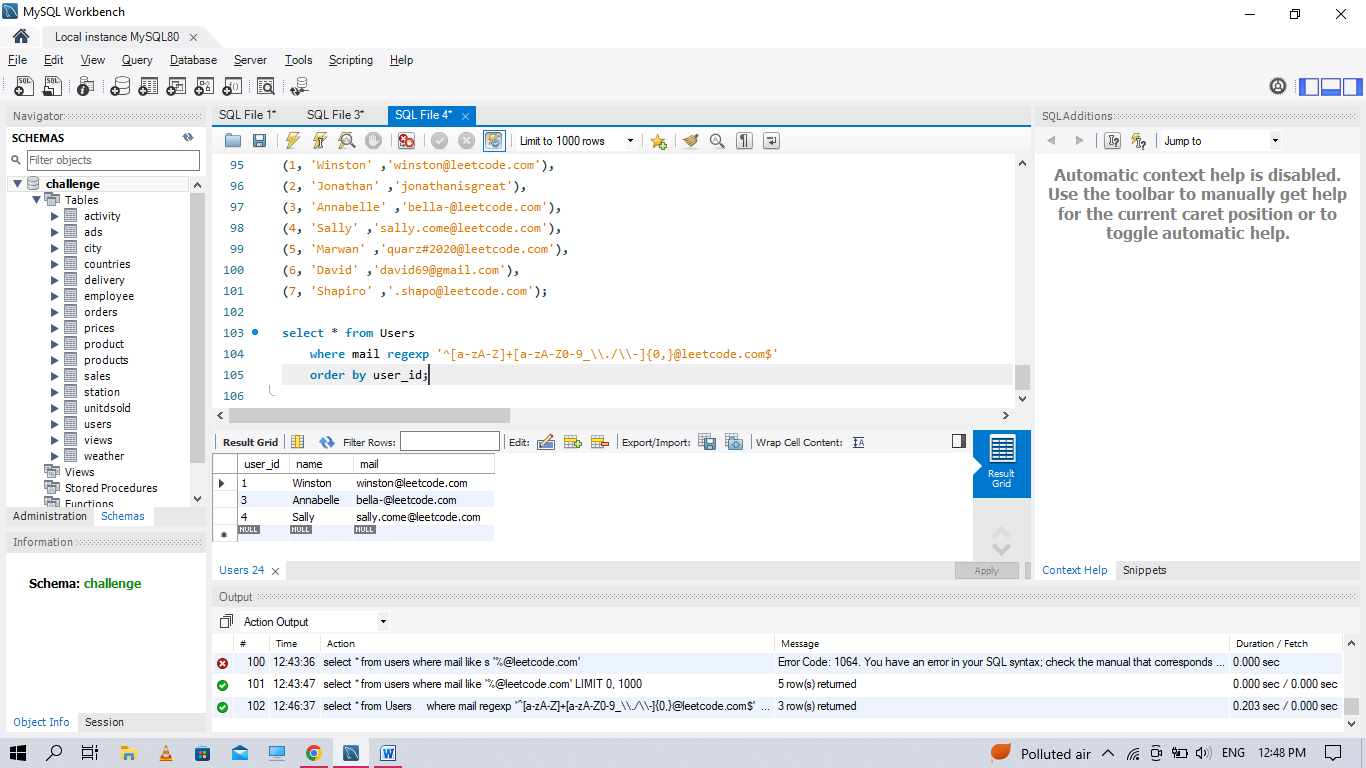
(6, 'David' ,'david69@gmail.com'),

(7, 'Shapiro' ,'.shapo@leetcode.com');

select \* from Users

where mail regexp '^[a-zA-Z]+[a-zA-Z0-9\_\\./\\-]{0,}@leetcode.com$'

order by user\_id;



**Q28. Table: Customers**

create table customers

(customer\_id int,

name varchar(30),

country varchar(30),

primary key(customer\_id)

)

create table orderss

(

order\_id int,

customer\_id int,

product\_id int,

order\_date date,

quantity int,

primary key(order\_id)

)

create table prooduct

(

product\_id int,

description varchar(30),

price int,

primary key(product\_id)

);

insert into customers values

(1, 'Winston' ,'USA'),

(2, 'Jonathan', 'Peru'),

(3, 'Moustafa' ,'Egypt');

insert into prooduct values

(10 ,'LC Phone' ,300),

(20 ,'LC T-Shirt' ,10),

(30 ,'LC Book' ,45),

(40 ,'LC Keychain' ,2)

insert into orderss values

(1, 1 ,10 ,'2020-06-10' ,1),

(2 ,1 ,20 ,'2020-07-01' ,1),

(3 ,1 ,30 ,'2020-07-08' ,2),

(4 ,2 ,10 ,'2020-06-15' ,2),

(5 ,2 ,40 ,'2020-07-01' ,10),

(6 ,3 ,20 ,'2020-06-24' ,2),

(7 ,3 ,30 ,'2020-06-25' ,2),

(9 ,3 ,30, '2020-05-08', 3);

**Write an SQL query to report the customer\_id and customer\_name of customers who have spent at least $100 in each month of June and July 2020.**

select o.customer\_id, c.name

from Customers c, Prooduct p, Orderss o

where c.customer\_id = o.customer\_id and p.product\_id = o.product\_id

group by o.customer\_id

having

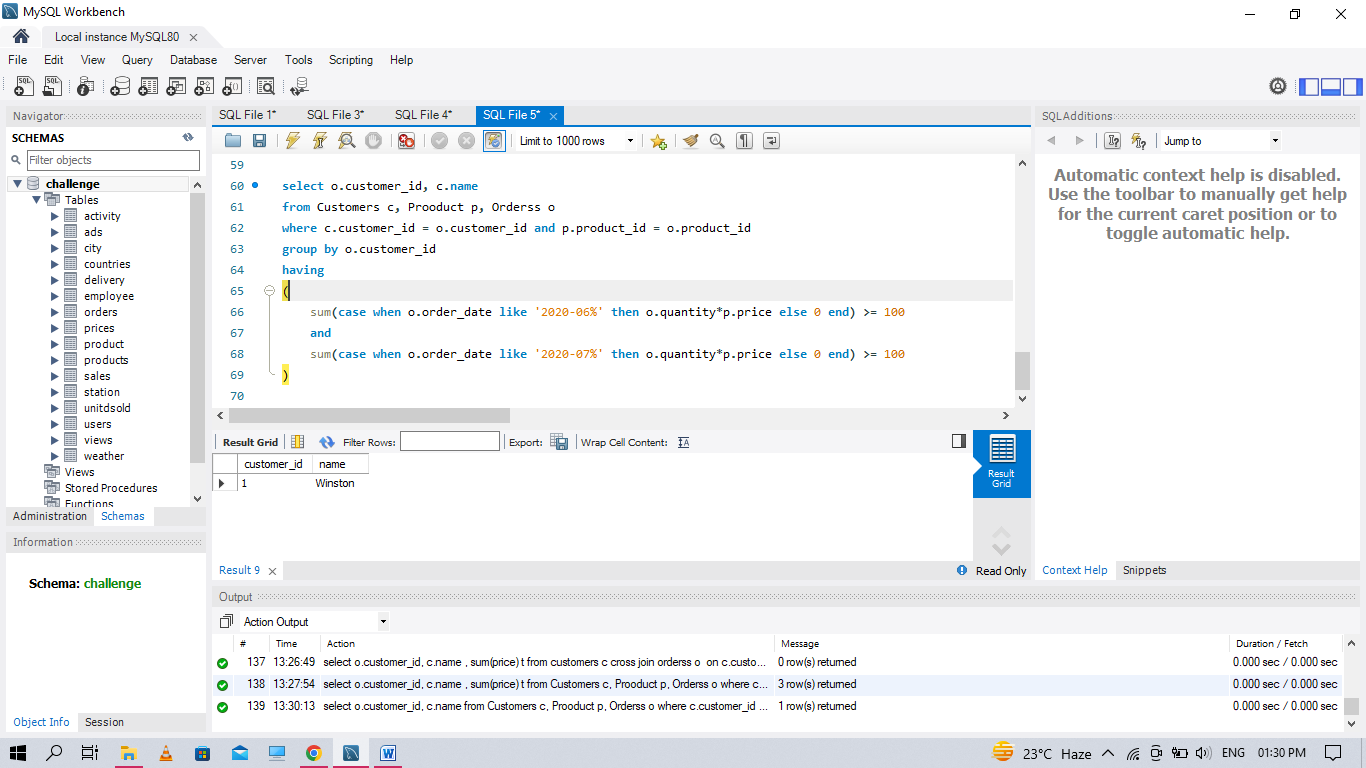
(

sum(case when o.order\_date like '2020-06%' then o.quantity\*p.price else 0 end) >= 100

and

sum(case when o.order\_date like '2020-07%' then o.quantity\*p.price else 0 end) >= 100

)

\

**Q29. Table: TVProgram**

CREATE TABLE TVPROGRAM

(

program\_date date,

content\_id int,

channel varchar(30),

PRIMARY KEY (program\_date, content\_id))

CREATE TABLE CONTENT

(

content\_id varchar(20),

title varchar(20),

Kids\_content enum('N','Y'),

content\_type varchar(20),

PRIMARY KEY(content\_id)

)

INSERT INTO TVPROGRAM VALUES

('2020-06-10 08:00' ,1 ,'LC-Channel'),

('2020-05-11 12:00' ,2 ,'LC-Channel'),

('2020-05-12 12:00' ,3 ,'LC-Channel'),

('2020-05-13 14:00' ,4, 'Disney Ch'),

('2020-06-18 14:00' ,4 ,'Disney Ch'),

('2020-07-15 16:00' ,5, 'Disney Ch')

INSERT INTO CONTENT VALUES

(1, 'Leetcode Movie' ,'N', 'Movies'),

(2, 'Alg. for Kids' ,'Y' ,'Series'),

(3, 'Database Sols', 'N' ,'Series'),

(4, 'Aladdin' ,'Y' ,'Movies'),

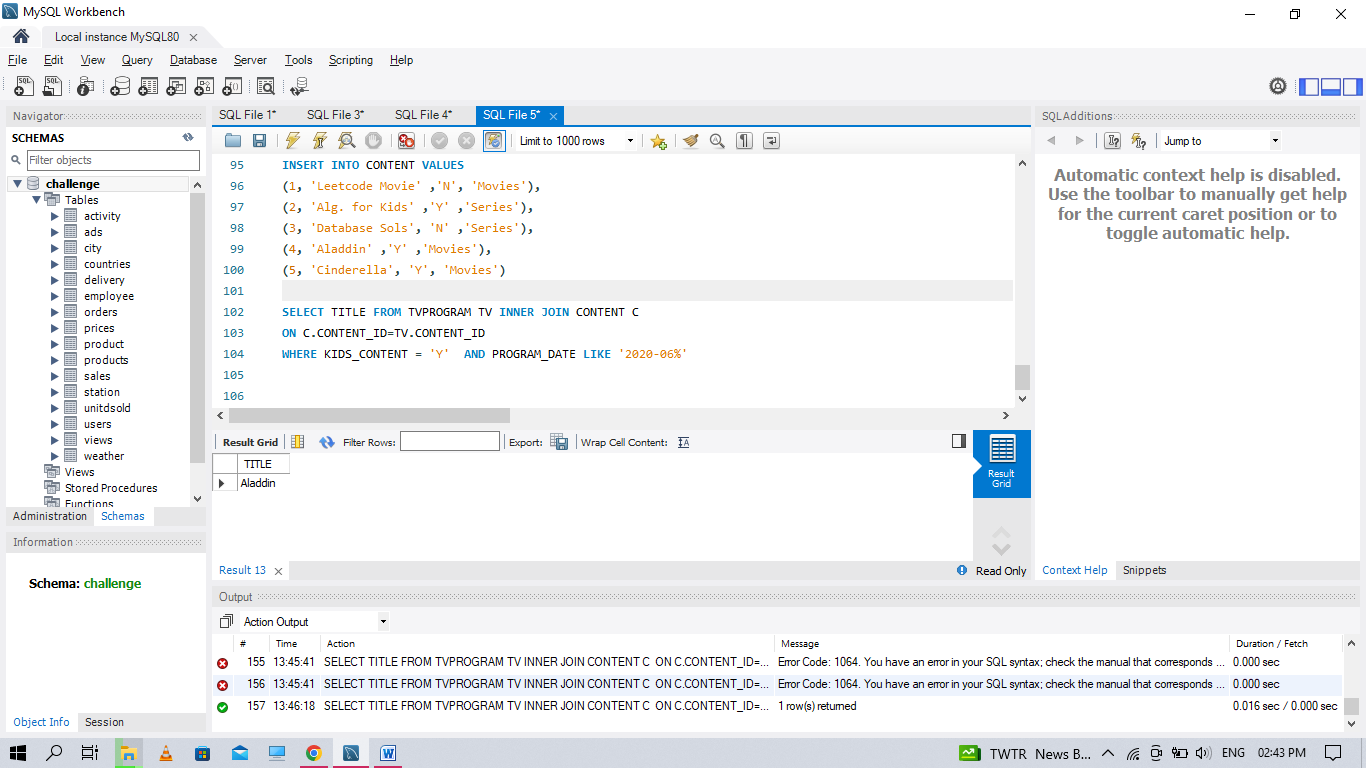
(5, 'Cinderella', 'Y', 'Movies')

**Write an SQL query to report the distinct titles of the kid-friendly movies streamed in June 2020. Return the result table in any order.**

SELECT TITLE FROM TVPROGRAM TV INNER JOIN CONTENT C

ON C.CONTENT\_ID=TV.CONTENT\_ID

WHERE KIDS\_CONTENT = 'Y' AND PROGRAM\_DATE LIKE '2020-06%'



**Q30 Write an SQL query to find the npv of each query of the Queries table. Return the result table in any order.**

create table npv

(

id int,

year int,

npv int,

primary key(id, year)

)

create table queries

(

id int,

year int,

primary key(id , year)

);

insert into npv values

(1, 2018, 100),

(7 ,2020 ,30),

(13, 2019, 40),

(1 ,2019 ,113),

(2 ,2008 ,121),

(3 ,2009 ,12),

(11, 2020, 99),

(7 ,2019, 0)

insert into queries values

(1 ,2019),

(2 ,2008),

(3 ,2009),

(7 ,2018),

(7 ,2019),

(7 ,2020),

(13 ,2019)

selelect q.id ,q.year, n.npv from queries q inner join

npv n on (q.id,q.year)=(n.id,n.year) ;



**Q31 Write an SQL query to find the npv of each query of the Queries table. Return the result table in any order**

selelect q.id ,q.year, n.npv from queries q inner join

npv n on (q.id,q.year)=(n.id,n.year) ;



**Q32. Write an SQL query to show the unique ID of each user, If a user does not have a unique ID replace just show null.**

create table employees

(

id int,

name varchar(30),

primary key(id)

)

create table employeeUNI

(

id int,

unique\_id int,

primary key(id, unique\_id)

)

insert into employees values

(1 ,'Alice'),

(7 ,'Bob'),

(11 ,'Meir'),

(90 ,'Winston'),

(3 ,'Jonathan')

insert into employeeuni values

(3 ,1),

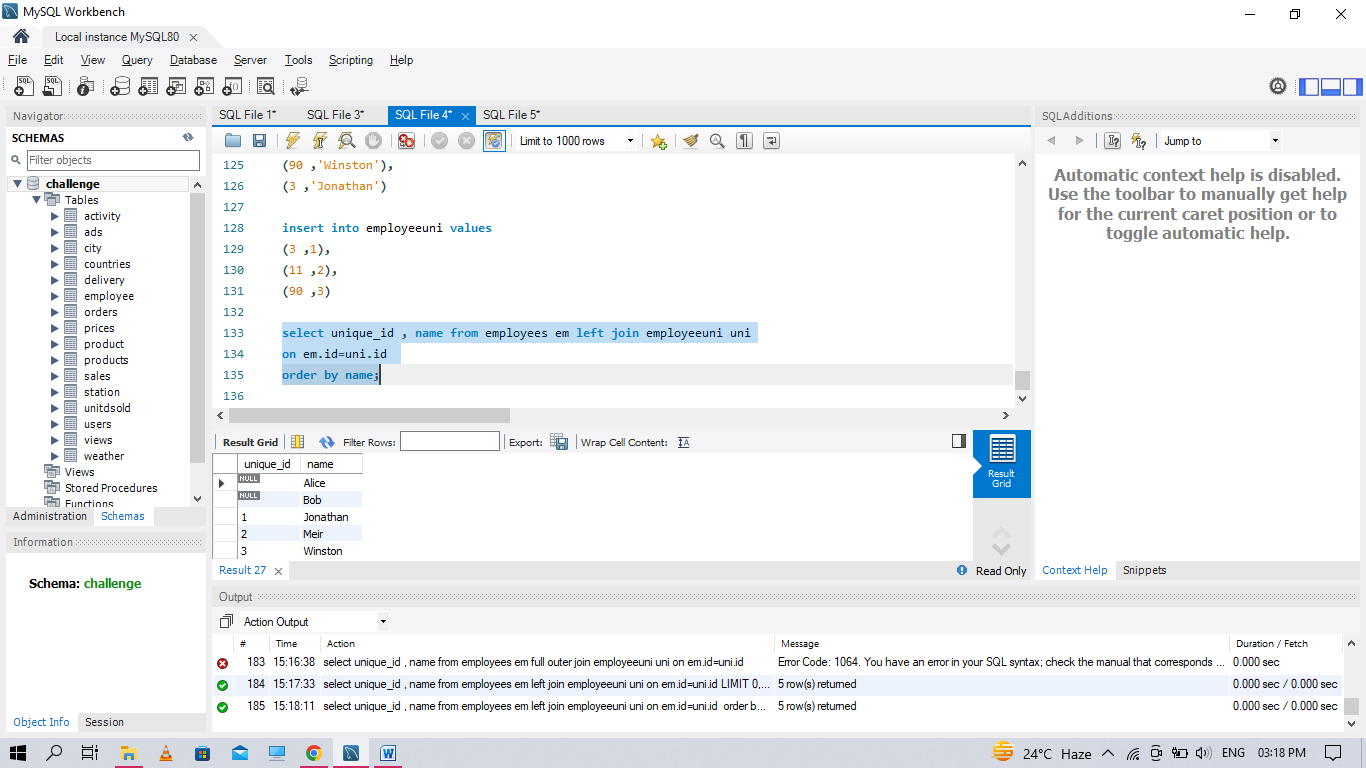
(11 ,2),

(90 ,3)

select unique\_id , name from employees em left join employeeuni uni

on em.id=uni.id

order by name;



**Q33. Write an SQL query to report the distance travelled by each user. Return the result table ordered by travelled\_distance in descending order, if two or more users travelled the same distance, order them by their name in ascending order**

create table userss

(

id int,

name varchar(20),

primary key(id)

)

create table rides

(

id int,

user\_id int,

distance int,

primary key(id))

insert into userss values

(1, 'Alice'),

(2, 'Bob'),

(3, 'Alex'),

(4, 'Donald'),

(7, 'Lee'),

(13, 'Jonathan'),

(19, 'Elvis')

insert into rides values

(1, 1 ,120),

(2 ,2 ,317),

(3 ,3 ,222),

(4 ,7 ,100),

(5 ,13 ,312),

(6 ,19 ,50),

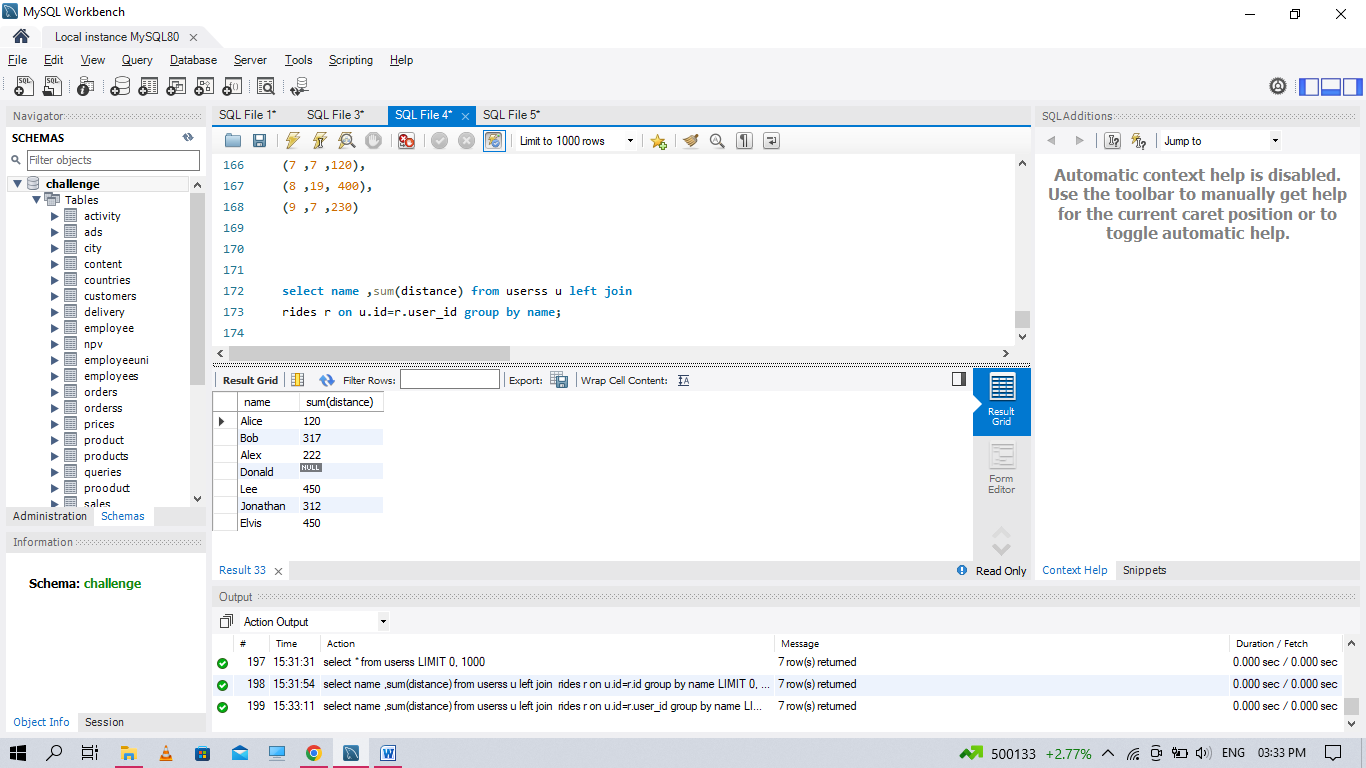
(7 ,7 ,120),

(8 ,19, 400),

(9 ,7 ,230)

select name ,sum(distance) from userss u left join

rides r on u.id=r.user\_id group by name;



**Q34. Write an SQL query to get the names of products that have at least 100 units ordered in February 2020 and their amount.**

select tmp.\* from

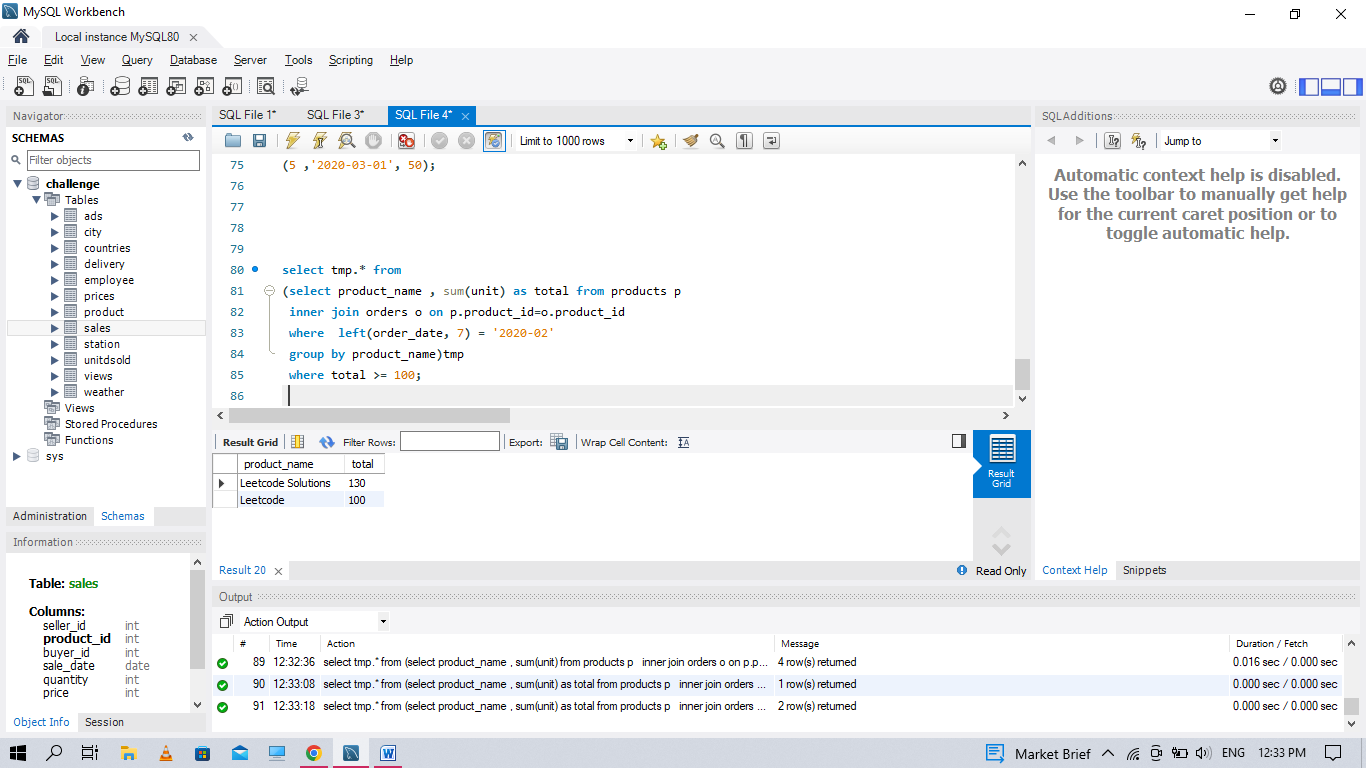
(select product\_name , sum(unit) as total from products p

inner join orders o on p.product\_id=o.product\_id

where left(order\_date, 7) = '2020-02'

group by product\_name)tmp

where total >= 100;



**Q35 . Write an SQL query to:**

**● Find the name of the user who has rated the greatest number of movies. In case of a tie, return the lexicographically smaller user name.**

**● Find the movie name with the highest average rating in February 2020. In case of a tie, return the lexicographically smaller movie name.**

SELECT user\_name AS results FROM

(

SELECT a.name AS user\_name, COUNT(\*) AS counts FROM MovieRating AS b

JOIN Uzers AS a

on a.user\_id = b.user\_id

GROUP BY b.user\_id

ORDER BY counts DESC, user\_name ASC LIMIT 1

) first\_query

UNION

SELECT movie\_name AS results FROM

(

SELECT c.title AS movie\_name, AVG(d.rating) AS rate FROM MovieRating AS d

JOIN Movies AS c

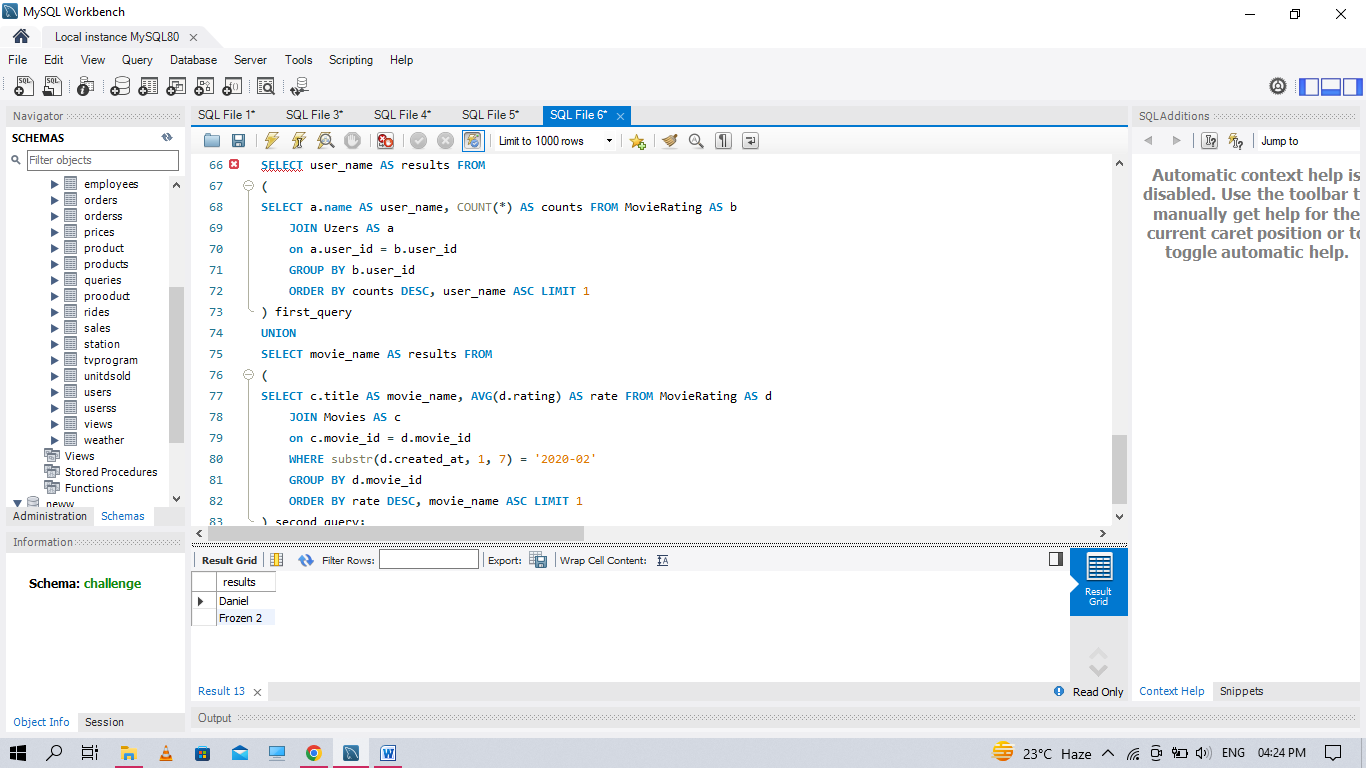
on c.movie\_id = d.movie\_id

WHERE substr(d.created\_at, 1, 7) = '2020-02'

GROUP BY d.movie\_id

ORDER BY rate DESC, movie\_name ASC LIMIT 1

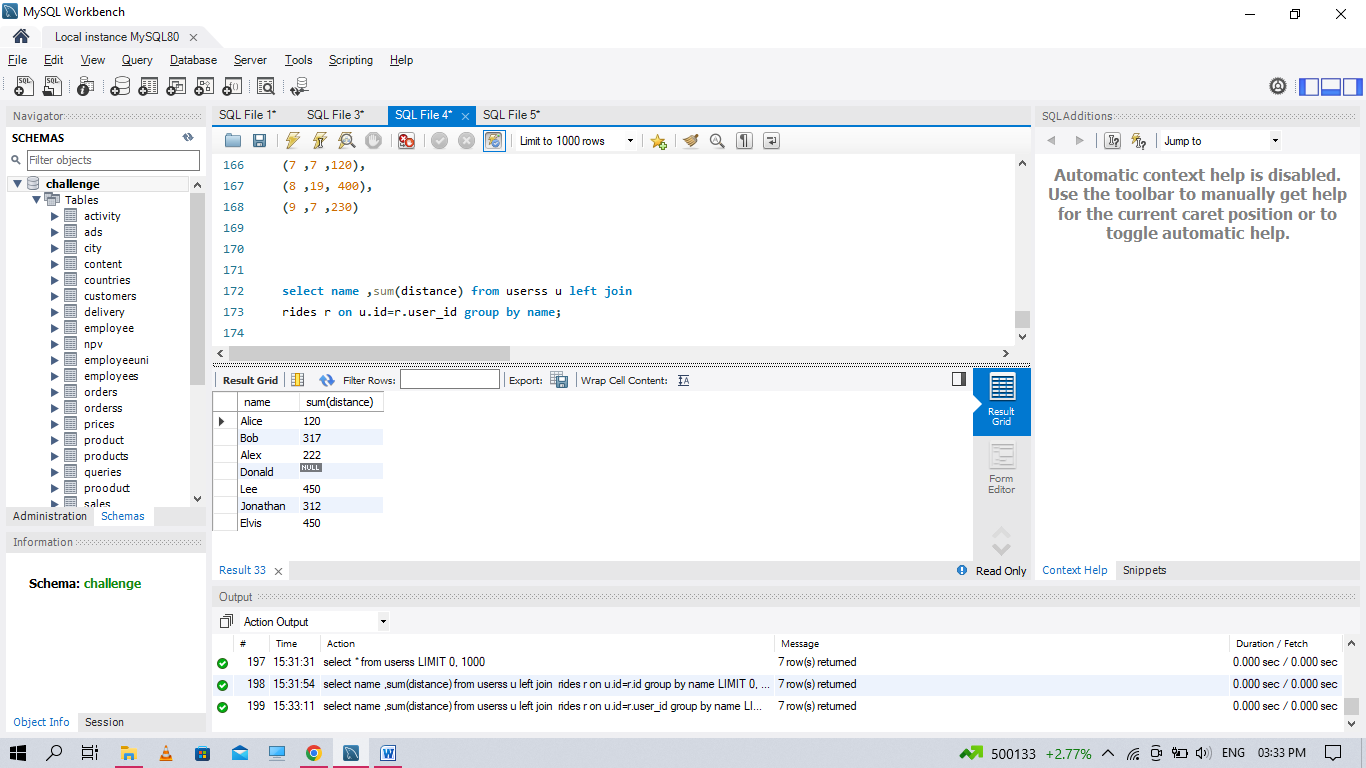
) second\_query;



**Q36. Write an SQL query to report the distance travelled by each user. Return the result table ordered by travelled\_distance in descending order, if two or more users travelled the same distance, order them by their name in ascending order**

select name ,sum(distance) from userss u left join

rides r on u.id=r.user\_id group by name;

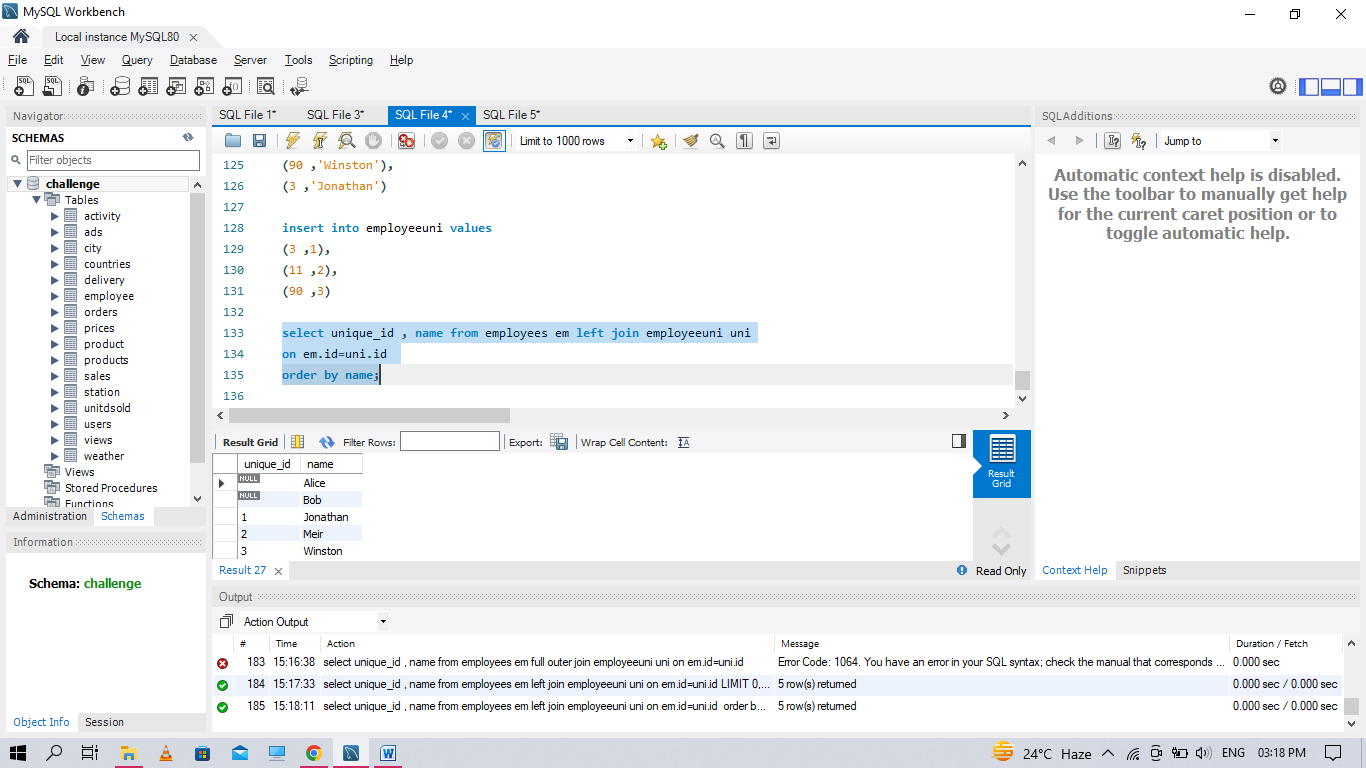


**Q37. Write an SQL query to show the unique ID of each user, If a user does not have a unique ID replace just show null.**

select unique\_id , name from employees em left join employeeuni uni

on em.id=uni.id

order by name;



**Q38. Write an SQL query to find the id and the name of all students who are enrolled in departments that no longer exist.**

create table departments

(

id int,

name varchar(40),

primary key(id)

)

create table students

(

id int,

name varchar(20),

department\_id int,

primary key(id)

)

insert into departments values

(1, 'Electrical Engineering'),

(7, 'Computer Engineering'),

(13, 'Business Administration')

insert into students values

(23, 'Alice', 1),

(1 ,'Bob' ,7),

(5 ,'Jennifer', 13),

(2 ,'John' ,14),

(4 ,'Jasmine' ,77),

(3 ,'Steve' ,74),

(6 ,'Luis' ,1),

(8 ,'Jonathan' ,7),

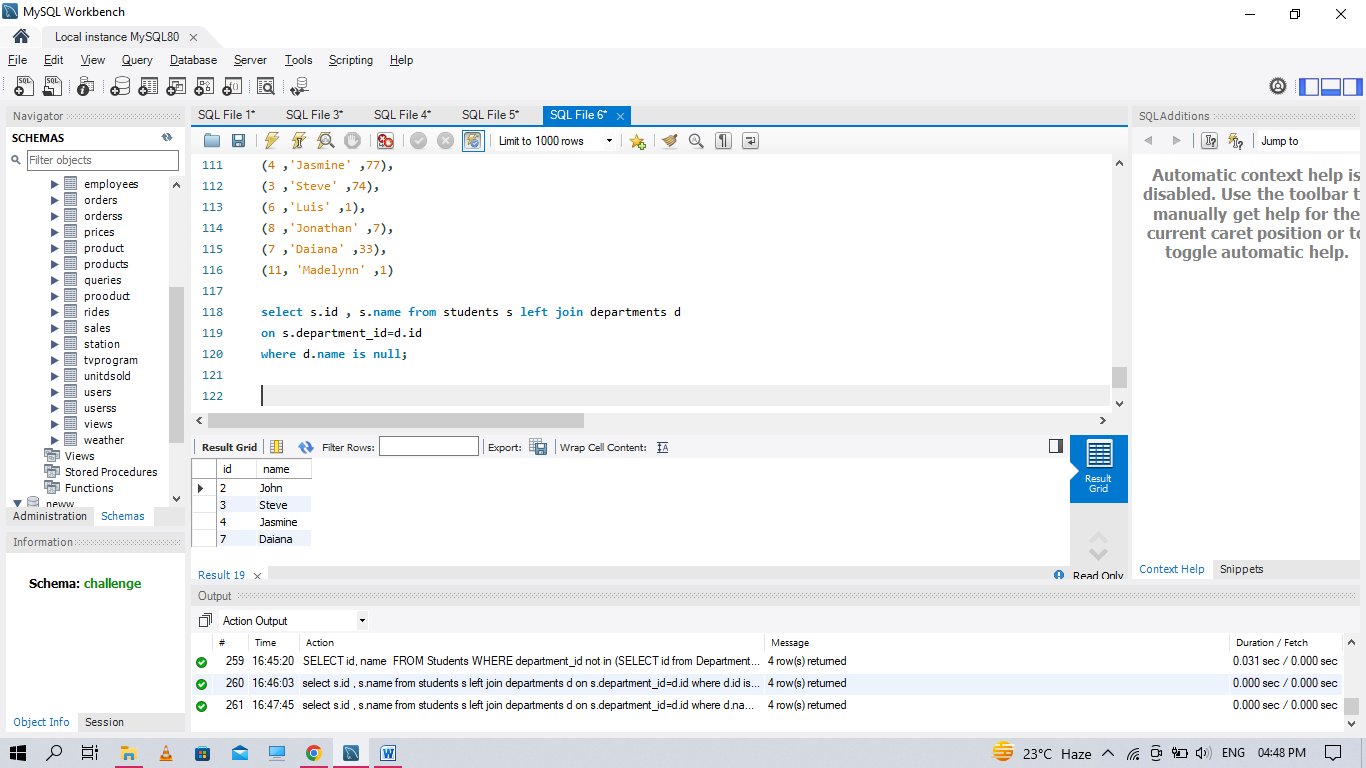
(7 ,'Daiana' ,33),

(11, 'Madelynn' ,1)

select s.id , s.name from students s left join departments d

on s.department\_id=d.id

where d.name is null;



**Q39. Write an SQL query to report the number of calls and the total call duration between each pair of distinct persons (person1, person2) where person1 < person2**

create table calls

(

from\_id int,

to\_id int,

duration int

)

insert into calls values

(1 ,2 ,59),

(2 ,1 ,11),

(1 ,3 ,20),

(3 ,4 ,100),

(3 ,4 ,200),

(3 ,4 ,200),

(4 ,3 ,499)

select from\_id as person1,to\_id as person2,

count(duration) as call\_count, sum(duration) as total\_duration

from (select \*

from Calls

union all

select to\_id, from\_id, duration

from Calls) t1

where from\_id < to\_id

group by person1, person2



**Q40. Write an SQL query to find the average selling price for each product. average\_price should be rounded to 2 decimal places**

select product\_id, ifnull(round(sum(prices\_sum) / sum(units), 2), 0) as average\_price

from (

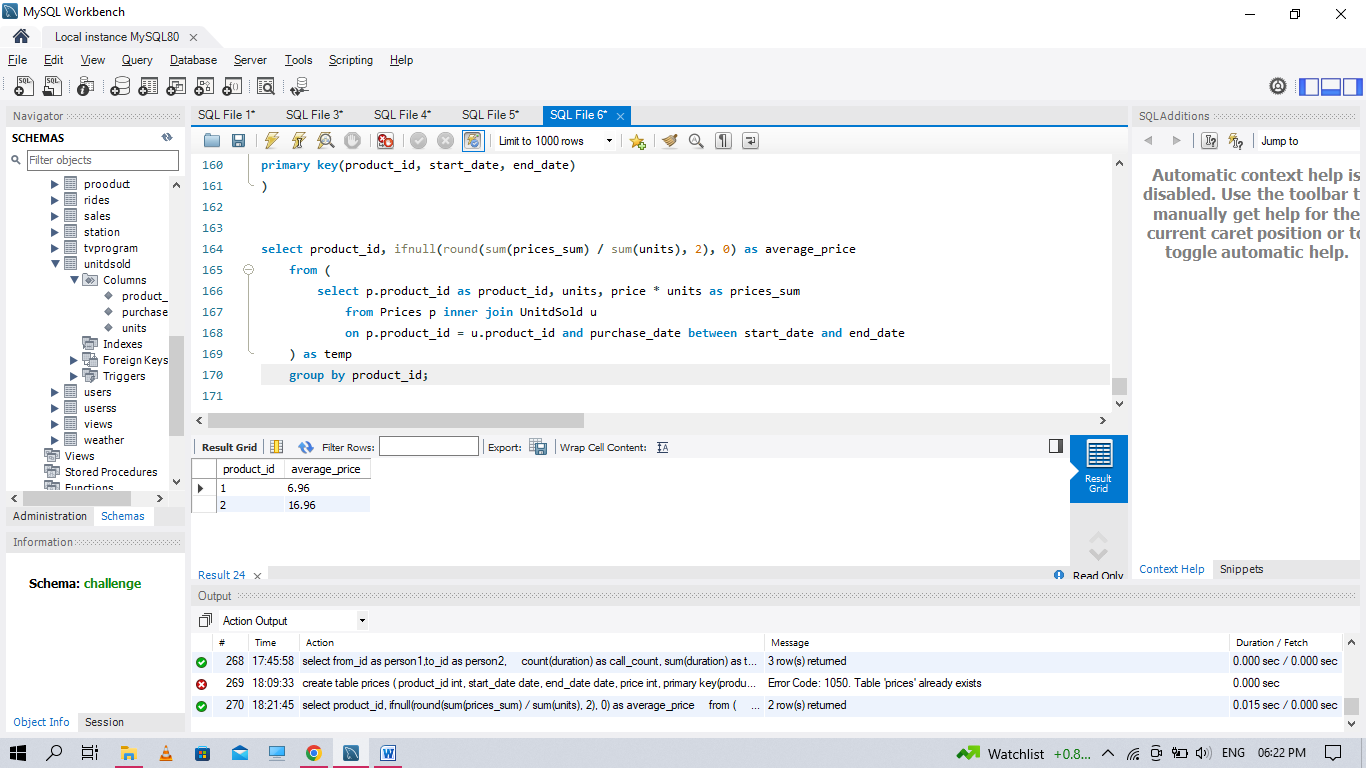
select p.product\_id as product\_id, units, price \* units as prices\_sum

from Prices p inner join UnitdSold u

on p.product\_id = u.product\_id and purchase\_date between start\_date and end\_date

) as temp

group by product\_id;



**Q41 Write an SQL query to report the number of cubic feet of volume the inventory occupies in each warehouse.**

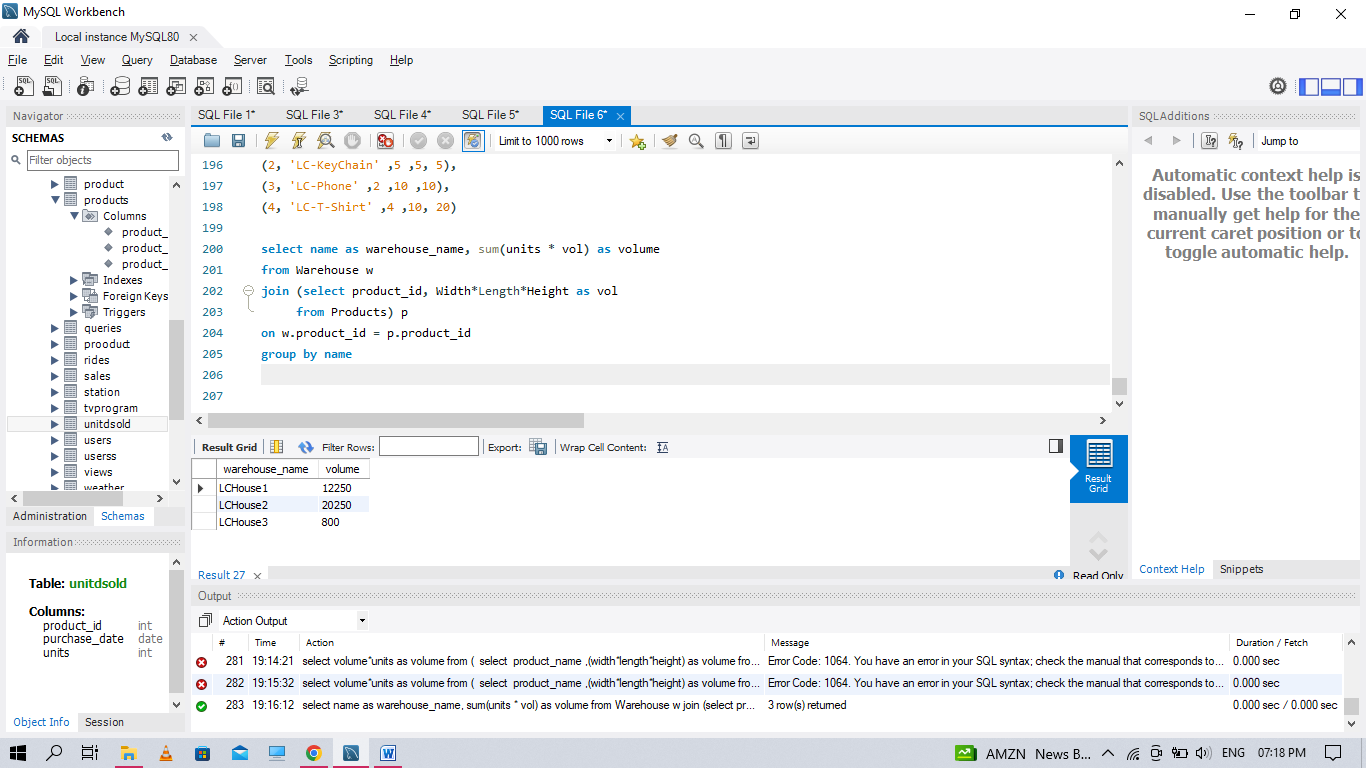
select name as warehouse\_name, sum(units \* vol) as volume

from Warehouse w

join (select product\_id, Width\*Length\*Height as vol from Products) p

on w.product\_id = p.product\_id

group by name



**Q42 Write an SQL query to report the difference between the number of apples and oranges sold each day. Return the result table ordered by sale\_date.**

create table saless

(

sale\_date date,

fruit enum('apples','oranges'),

sold\_num int,

primary key(sale\_date, fruit))

insert into saless values

('2020-05-01' ,'apples' ,10),

('2020-05-01' ,'oranges' ,8),

('2020-05-02' ,'apples' ,15),

('2020-05-02' ,'oranges', 15),

('2020-05-03' ,'apples' ,20),

('2020-05-03' ,'oranges', 0),

('2020-05-04' ,'apples' ,15),

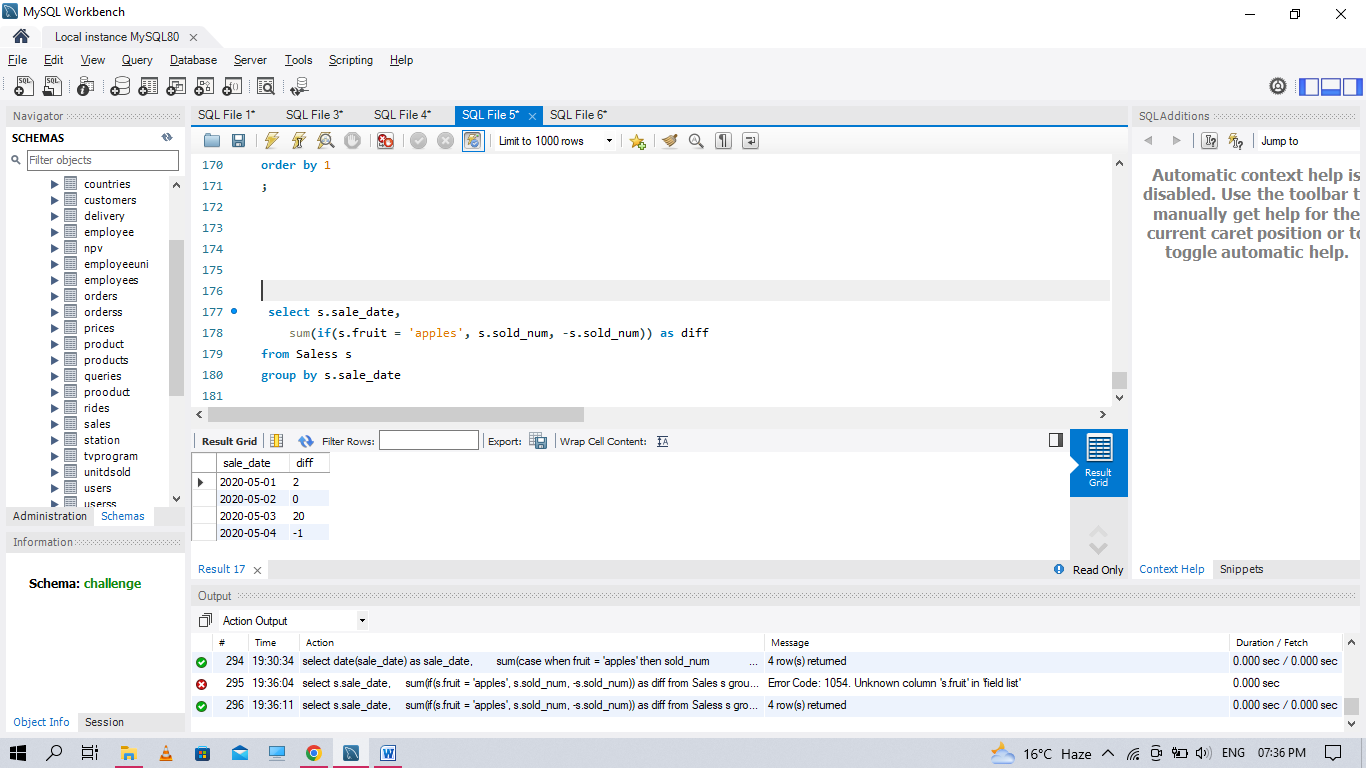
('2020-05-04' ,'oranges' ,16)

select s.sale\_date,

sum(if(s.fruit = 'apples', s.sold\_num, -s.sold\_num)) as diff

from Saless s

group by s.sale\_date



**Q43. Write an SQL query to report the fraction of players that logged in again on the day after the day they first logged in, rounded to 2 decimal places. In other words, you need to count the number of players that logged in for at least two consecutive days starting from their first login date, then divide that number by the total number of players**

create table activity1

(

player\_id int,

device\_id int,

event\_date date,

games\_played int,

primary key (player\_id, event\_date)

)

insert into activity1 values

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

(1, 2, '2016-03-02', 6),

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

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

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

WITH cte AS (

SELECT player\_id, MIN(event\_date) as first\_login

FROM Activity1

GROUP BY player\_id

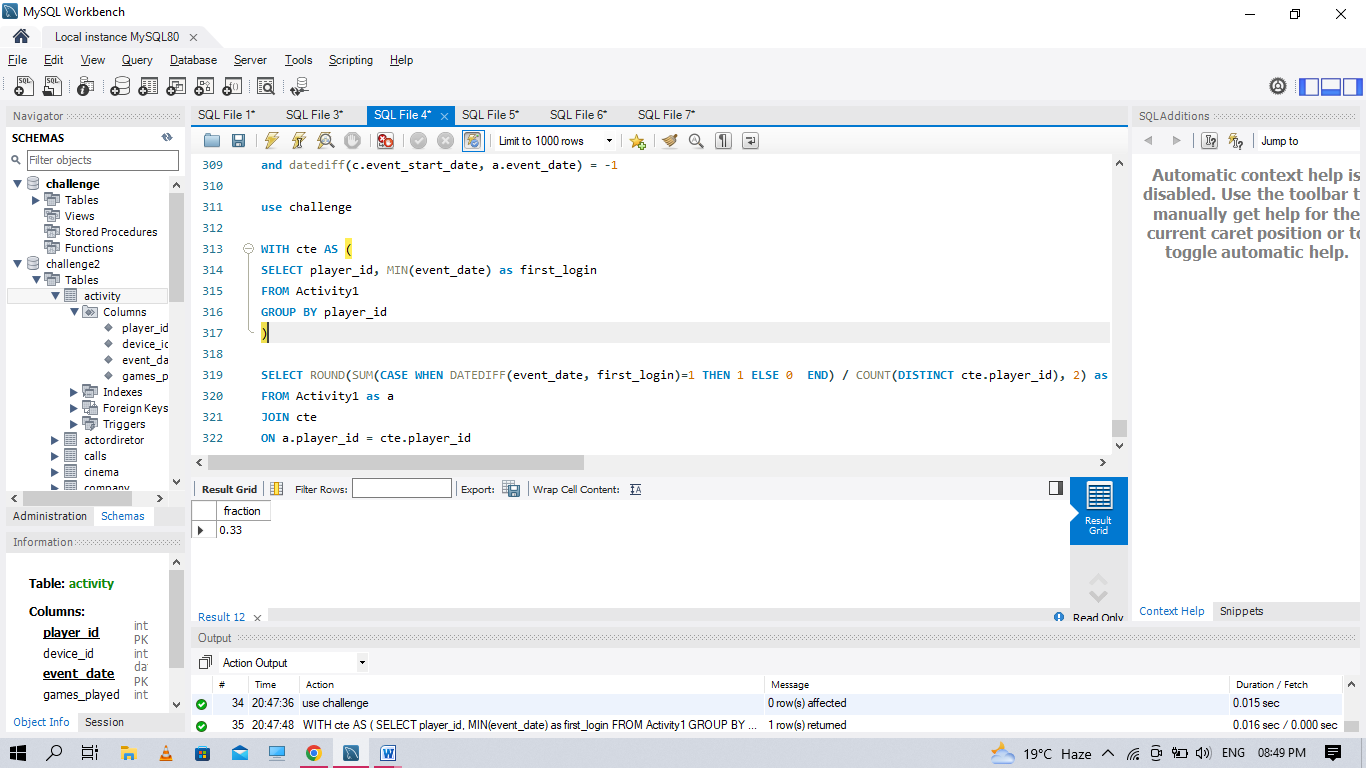
)

SELECT ROUND(SUM(CASE WHEN DATEDIFF(event\_date, first\_login)=1 THEN 1 ELSE 0 END) / COUNT(DISTINCT cte.player\_id), 2) as fraction

FROM Activity1 as a

JOIN cte

ON a.player\_id = cte.player\_id



**Q44. Write an SQL query to report the managers with at least five direct reports. Return the result table in any order.**

create table if not exists emp

(

id int,

name varchar(20),

department varchar(20),

managerId int,

primary key (id)

)

insert into emp values

(101, 'John','A' ,NULL),

(102, 'Dan' ,'A' ,101),

(103, 'James', 'A' ,101),

(104, 'Amy' ,'A' ,101),

(105, 'Anne' ,'A' ,101),

(106, 'Ron' ,'B', 101)

select Name from Emp

where Id in

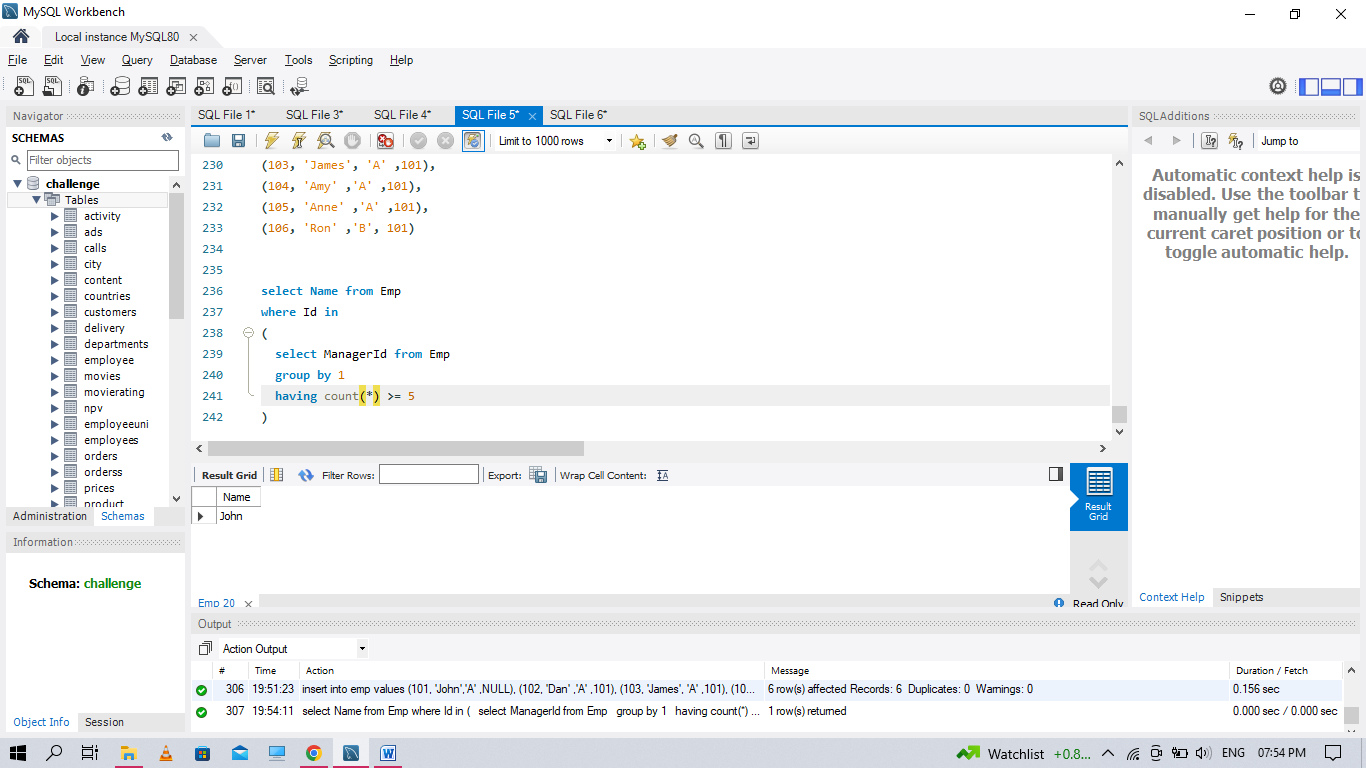
(

select ManagerId from Emp

group by 1

having count(\*) >= 5

)



**Q45. Write an SQL query to report the respective department name and number of students majoring in each department for all departments in the Department table (even ones with no current students). Return the result table ordered by student\_number in descending order. In case of a tie, order them by dept\_name alphabetically.**

create table student

(

student\_id int,

student\_name varchar(30),

gender varchar(20),

dept\_id int,

primary key (student\_id)

)

create table department

(

dept\_id int,

dept\_name varchar(20),

primary key (dept\_id)

)

insert into student values

(1, 'Jack', 'M', 1),

(2, 'Jane', 'F', 1),

(3, 'Mark', 'M', 2)

insert into department values

(1, 'Engineering'),

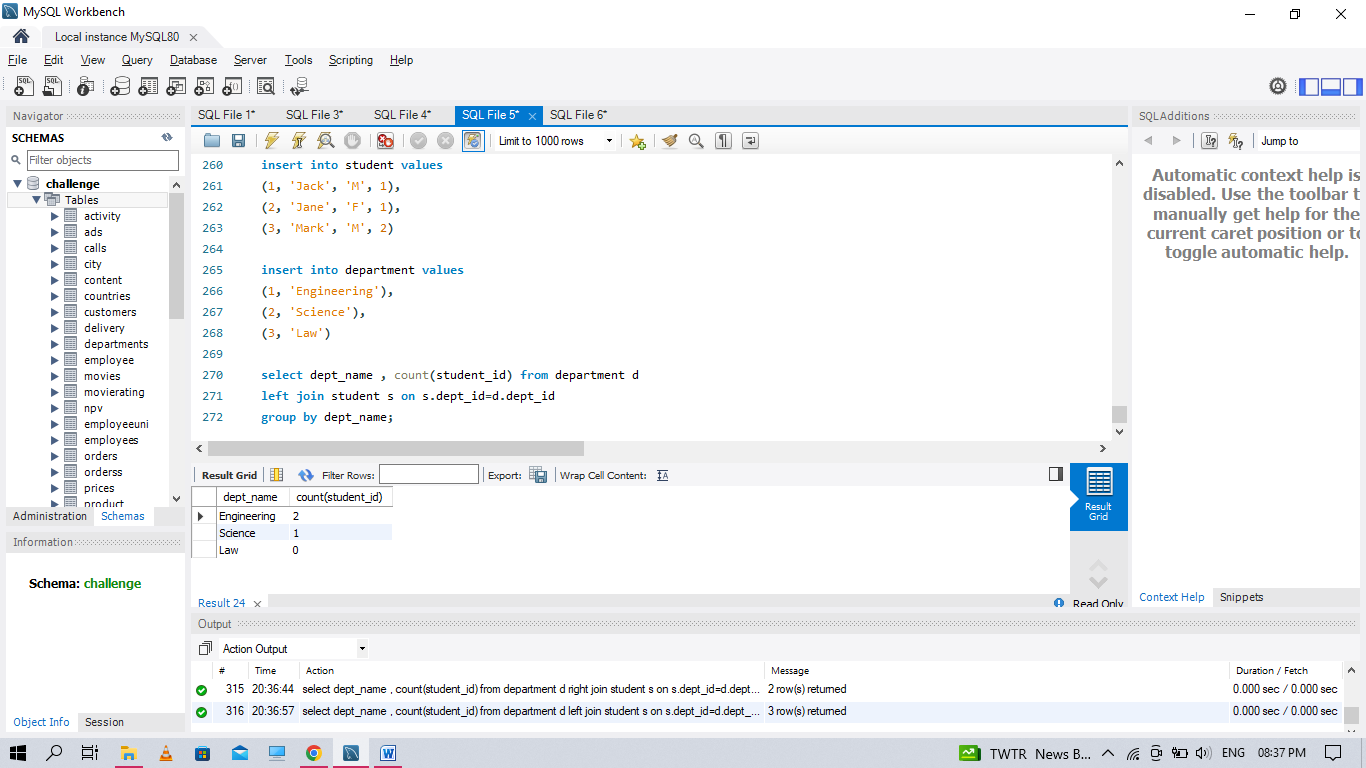
(2, 'Science'),

(3, 'Law')

select dept\_name , count(student\_id) from department d

left join student s on s.dept\_id=d.dept\_id

group by dept\_name;



**Q46. Write an SQL query to report the customer ids from the Customer table that bought all the products in the Product table.**

create table customer

(customer\_id int,

product\_key int,

foreign key (product\_key) references prodct(product\_key))

create table prodct

(product\_key int,

primary key (product\_key)

)

insert into prodct values

(5),(6)

insert into customer values

(1, 5),

(2, 6),

(3, 5),

(3, 6),

(1, 6)

SELECT customer\_id FROM customer

GROUP BY customer\_id

HAVING COUNT( DISTINCT product\_key) = (SELECT COUNT(\*) FROM prodct)

