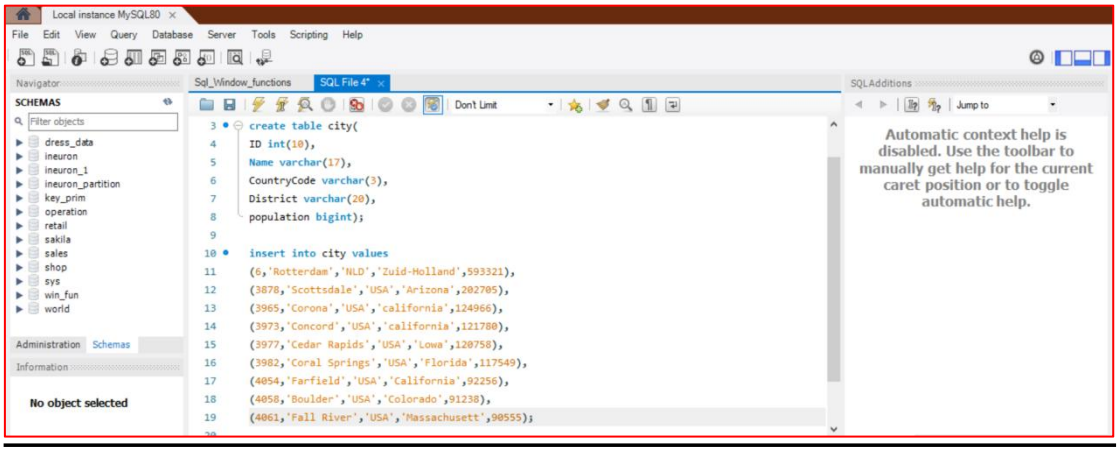
SQL ASSIGNMENT SET 1

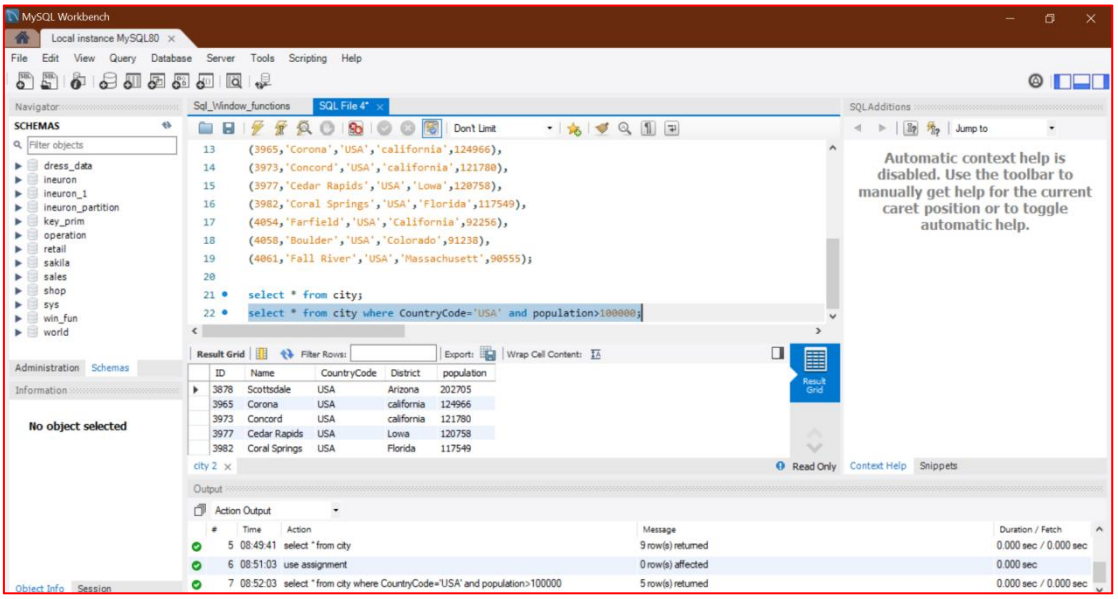
Sample Dataset 1

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

Sol) **select * from city where CountryCode='USA' and population>100000;**

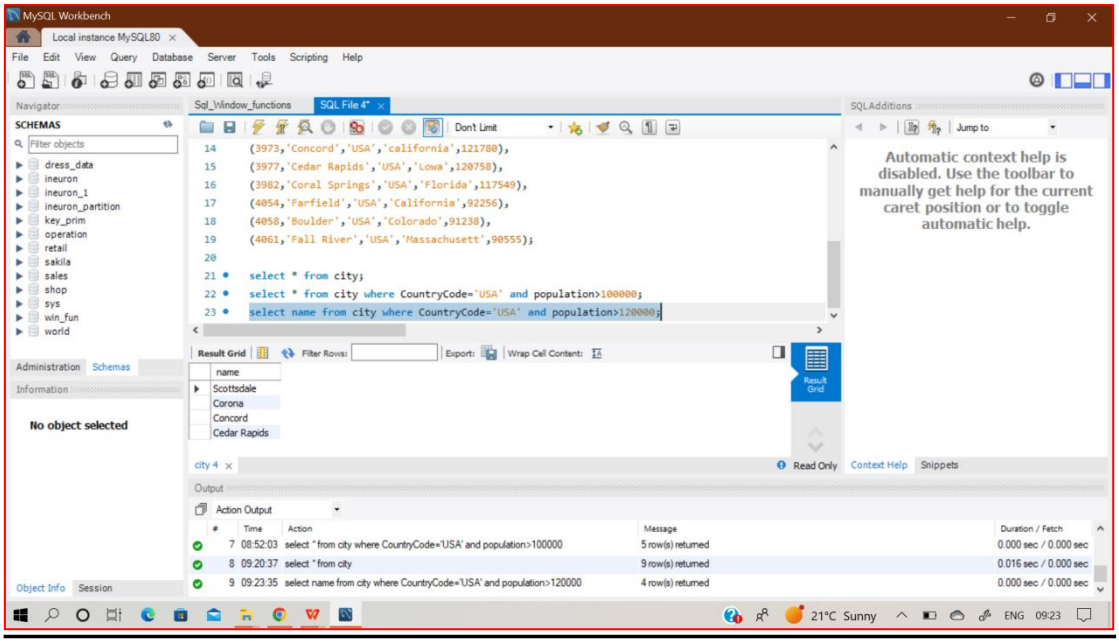
First of all, I have created a database named 'assignment' and then created a table inside it named 'City'. Then Inserted records using insert functions and then performed the desired query.



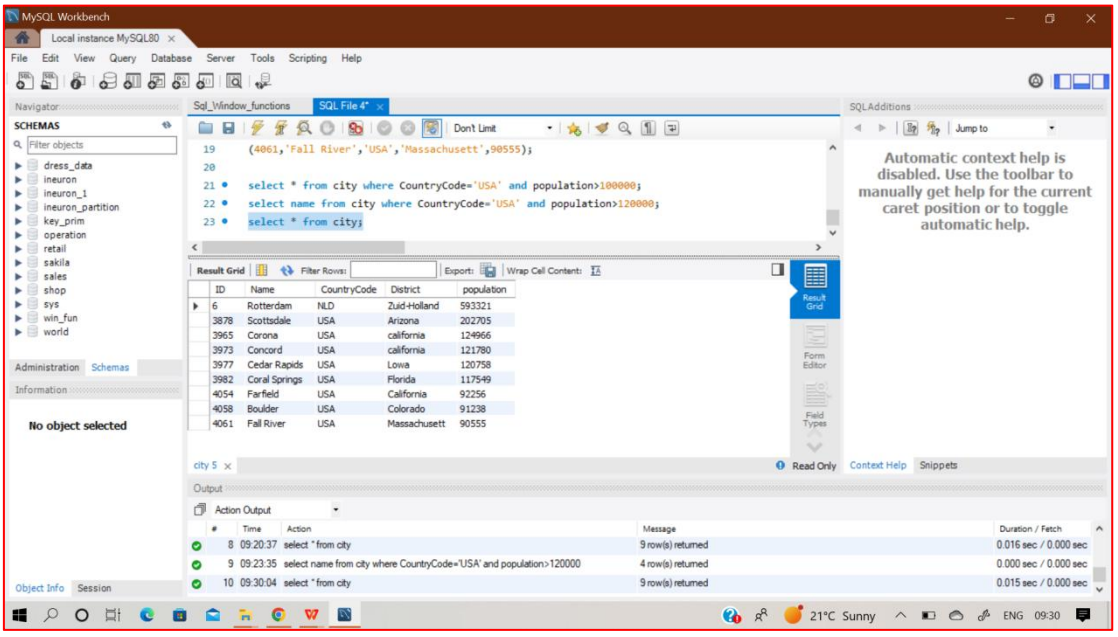


Q2. Query the NAME field for all American cities in the CITY table with populations larger than 120000.

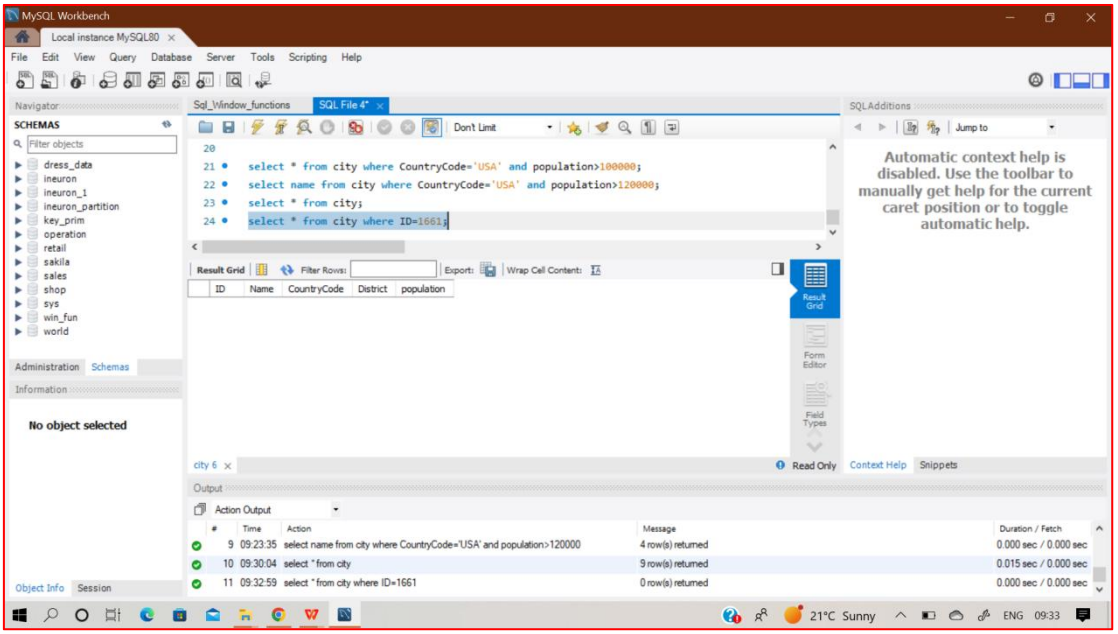
Sol) **select name from city where CountryCode='USA' and population>120000;**



Q3. Query all columns (attributes) for every row in the CITY table.
Sol) **Select * from City;**



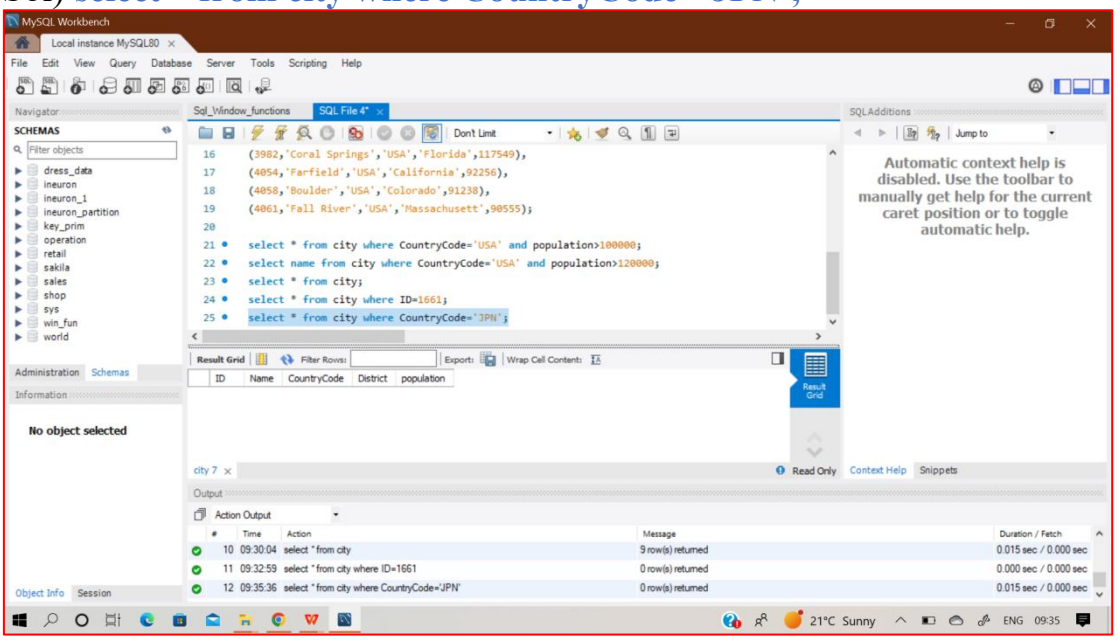
Q4) Query all columns for a city in CITY with the ID 1661.
Sol) **Select * from city where ID = 1661;**



Output - No rows returned.

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

Sol) **select * from city where CountryCode='JPN';**



Output - No rows returned.

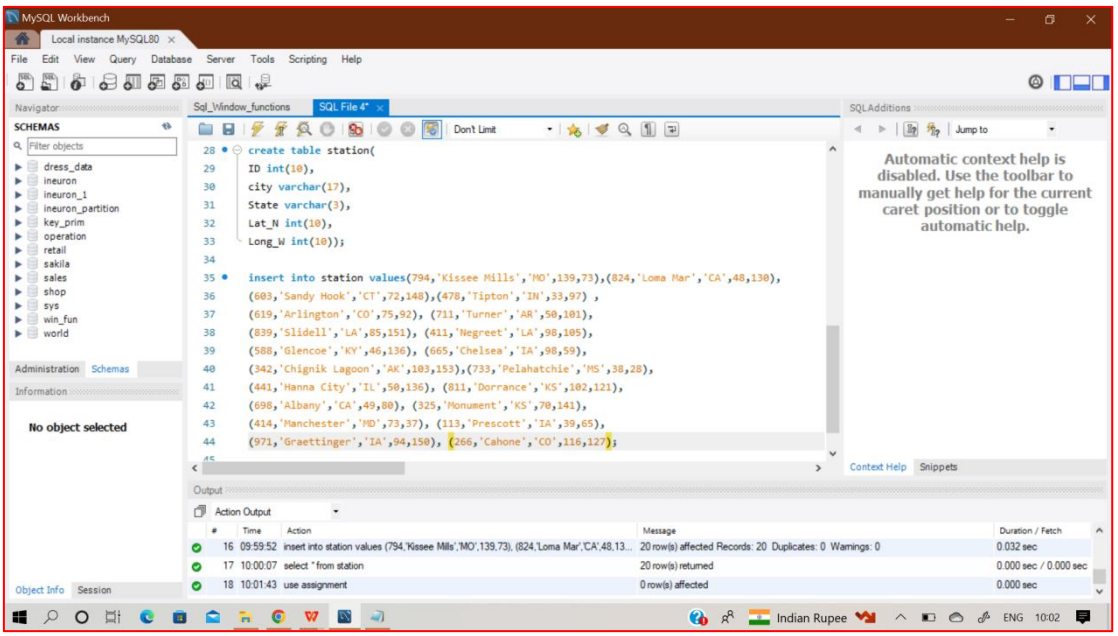
Q6) Query the names of all the Japanese cities in the CITY table. The COUNTRYCODE for Japan is JPN.

Sol) **Select name from City where CountryCode='JPN';**

Output - No rows returned.

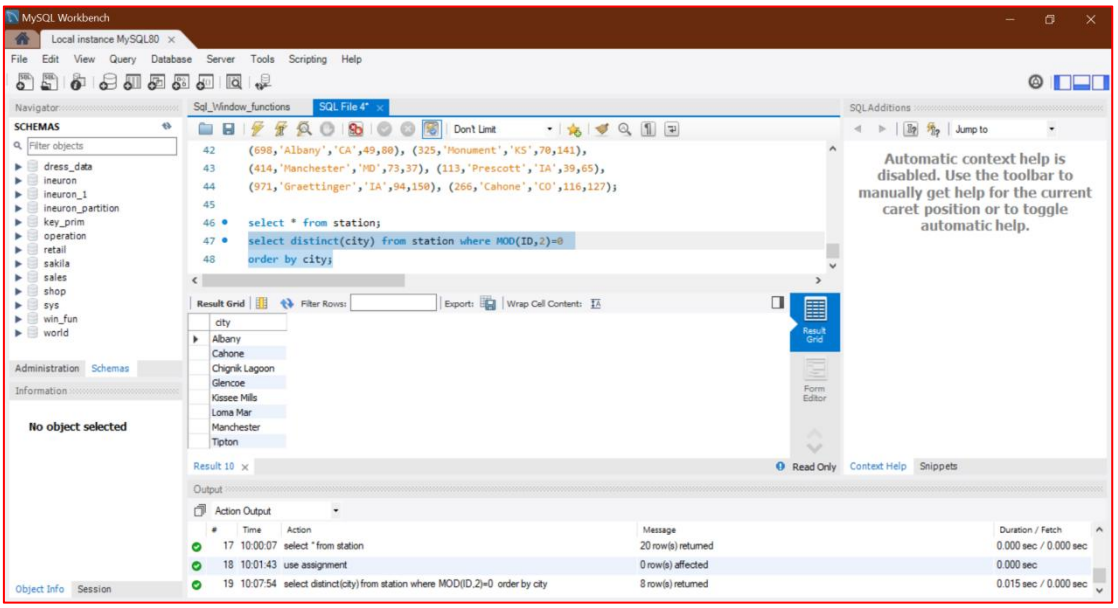
Sample Dataset 2

First of all , I created a new table ‘station’ and then inserted the records in it using multi-insert statement. The glimpse of it is shown in the image.



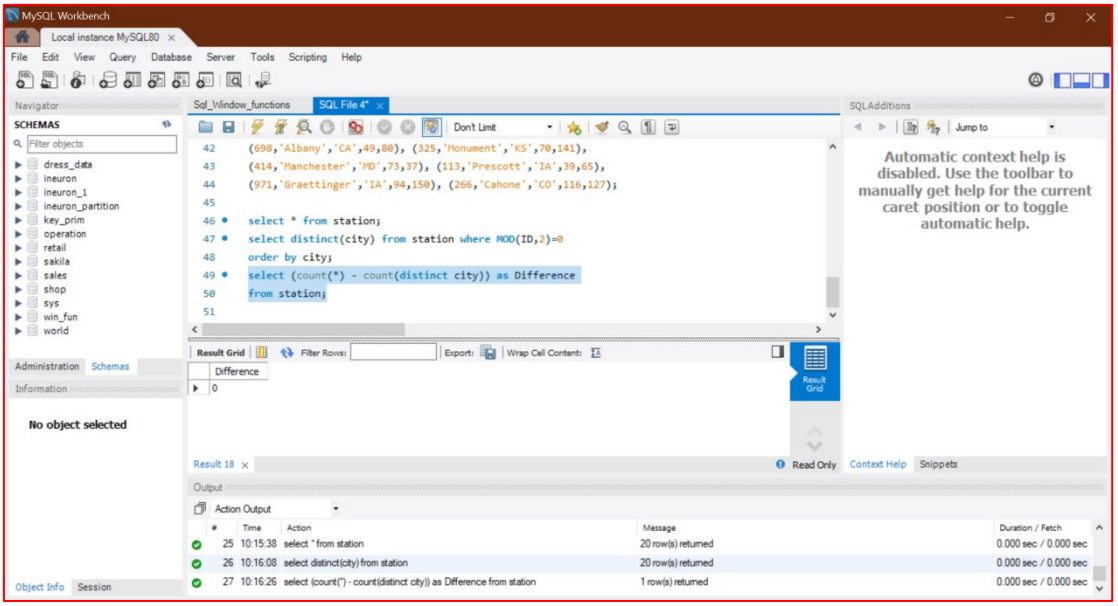
Q8. Query a list of CITY names from STATION for cities that have an even ID number. Print the results in any order, but exclude duplicates from the answer.

Sol) **Select distinct(city) from station where MOD(ID,2)=0 order by city;**



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

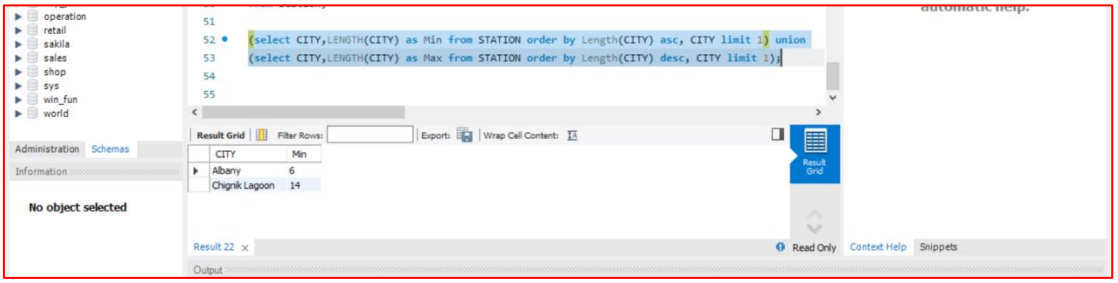
Sol) **select (count(*) - count(distinct city)) as Difference from station;**



There is zero difference between total no of cities and distinct cities. All the city records in the station table are unique.

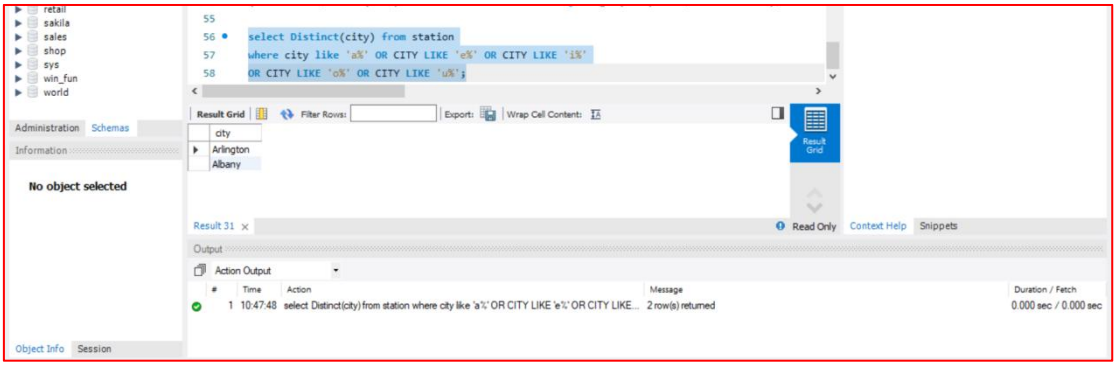
Q10) Query the two cities in STATION with the shortest and longest CITY names, as well as their respective lengths (i.e.: number of characters in the name). If there is more than one smallest or largest city, choose the one that comes first when ordered alphabetically.

Sol) **(Select CITY,LENGTH(CITY) as Min from STATION order by Length(CITY) asc, CITY limit 1) union (Select CITY,LENGTH(CITY) as Max from STATION order by Length(CITY) desc, CITY limit 1);**



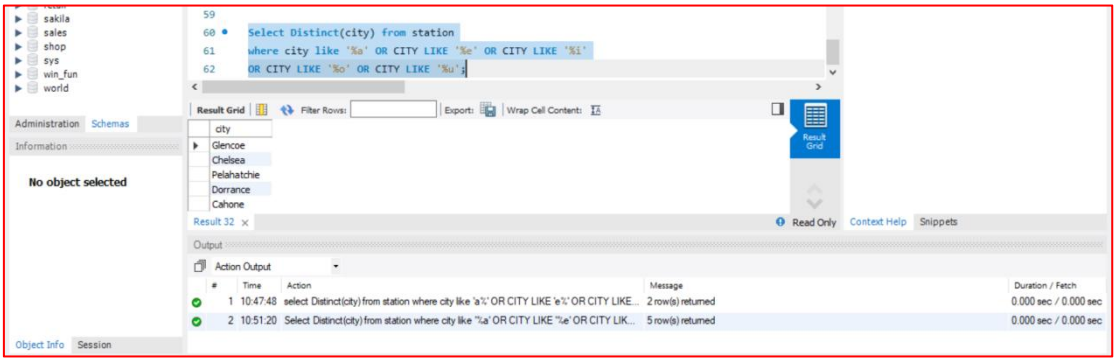
Q11) Query the list of CITY names starting with vowels (i.e., a, e, i, o, or u) from STATION. Your result cannot contain duplicates.

Sol) **Select Distinct(city) from station where city like 'a%' OR CITY LIKE 'e%' OR CITY LIKE 'i%' OR CITY LIKE 'o%' OR CITY LIKE 'u%';**



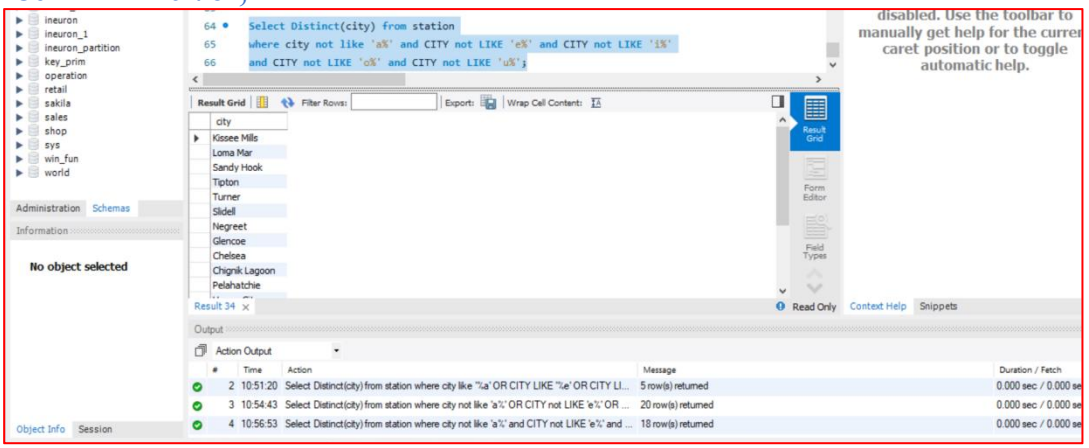
Q12) Query the list of CITY names ending with vowels (a, e, i, o, u) from STATION. Your result cannot contain duplicates.

Sol) **Select Distinct(city) from station**
where city like '%a' OR CITY LIKE '%e' OR CITY LIKE '%i'
OR CITY LIKE '%o' OR CITY LIKE '%u';



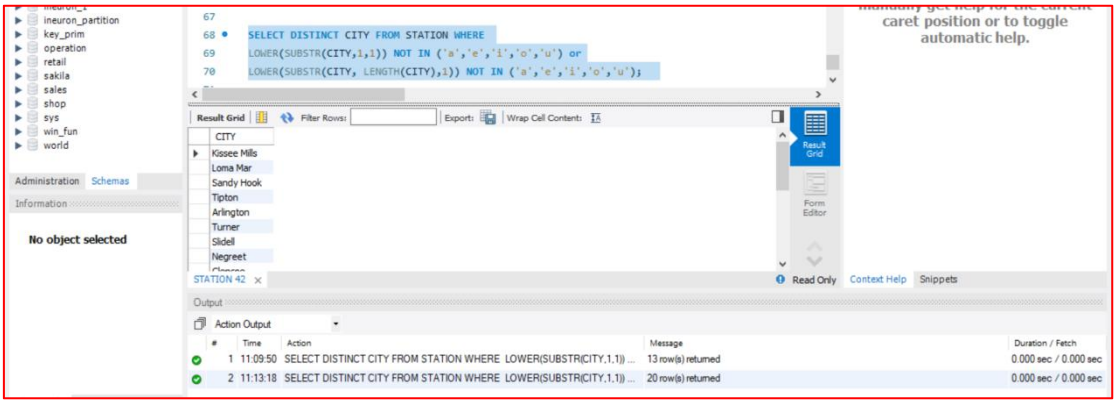
Q13) Query the list of CITY names from STATION that do not start with vowels. Your result cannot contain duplicates.

Select Distinct(city) from station where city not like 'a%' and CITY not LIKE 'e%' and CITY not LIKE 'i%' and CITY not LIKE 'o%' and CITY not LIKE 'u%';



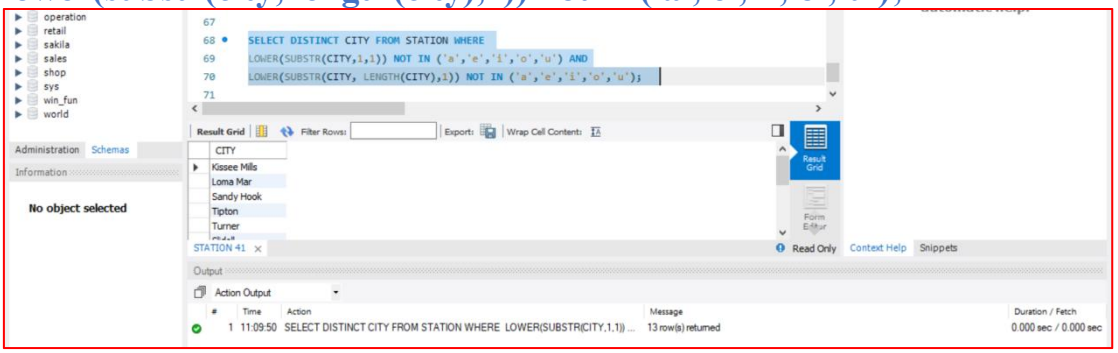
Q15) Query the list of CITY names from STATION that do not start with vowels or do not end with vowels. Your result cannot contain duplicates.

Sol) Select distinct city from station where lower(substr(city,1,1)) not in ('a','e','i','o','u') or lower(substr(city, length(city),1)) not in ('a','e','i','o','u');



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

Sol) Select distinct city from station where lower(substr(city,1,1)) not in ('a','e','i','o','u') and lower(substr(city, length(city),1)) not in ('a','e','i','o','u');



Q17) Table: Product

Column Name	**Type**
Product_id	int
Product_name	varchar
Unit_price	int

product_id is the primary key of this table.
Each row of this table indicates the name and the price of each product

Table: Sales

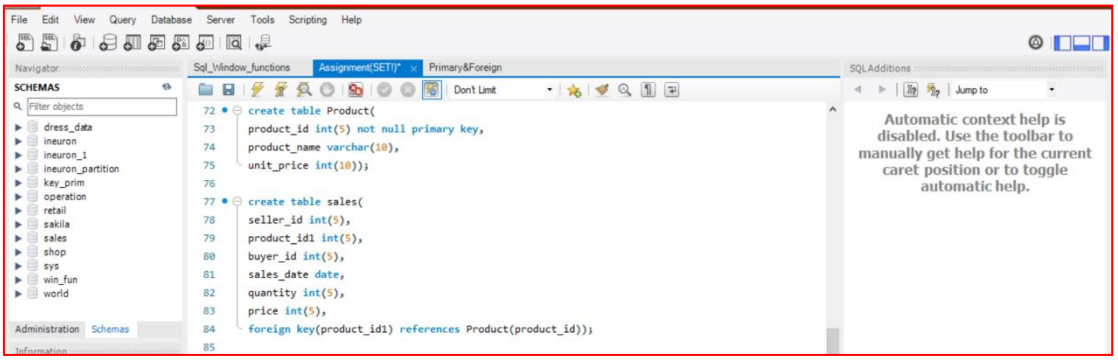
Column Name	Type
Seller_id	int
Product_id	int
Buyer_id	int
Sales_date	date

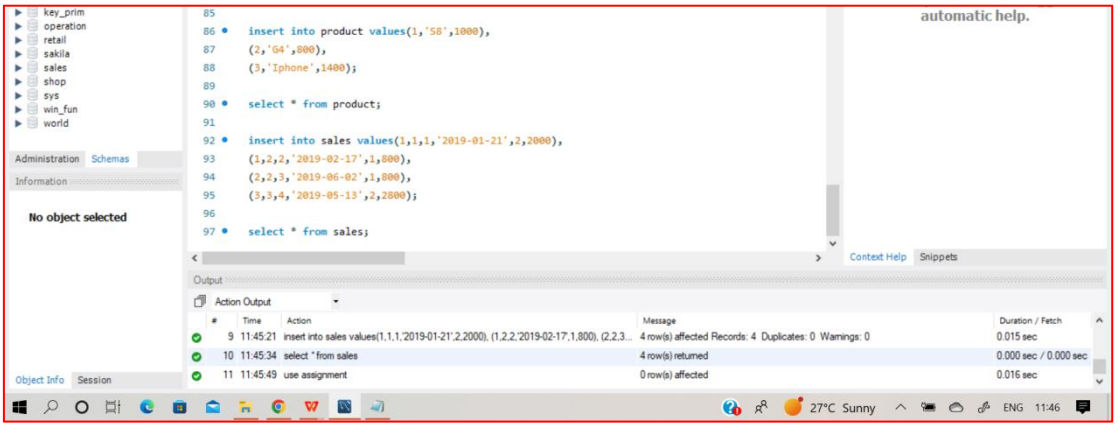
Quantity	int
Price	int

This table has no primary key, it can have repeated rows. product_id is a foreign key to the Product table. Each row of this table contains some information about one sale.

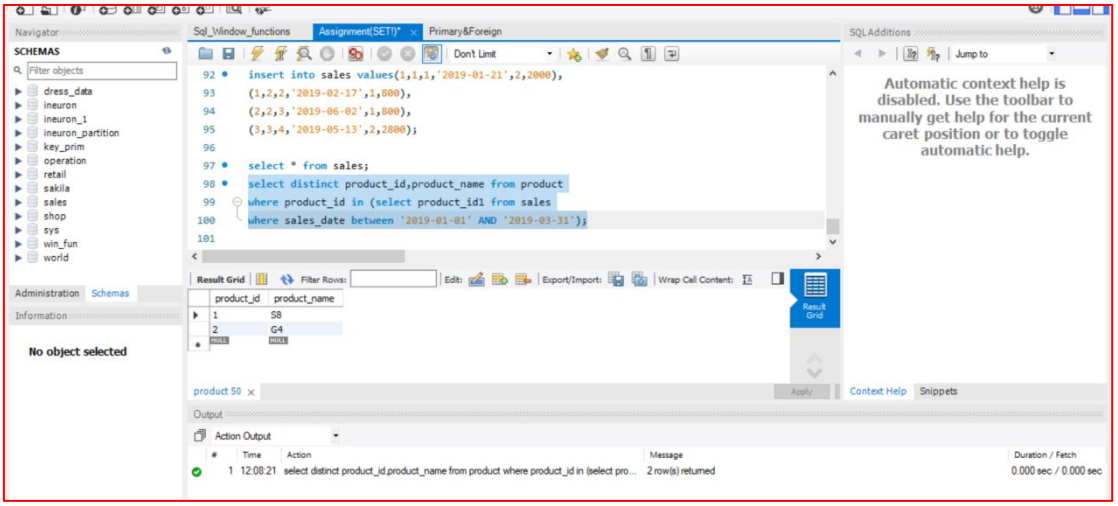
Write an SQL query that reports the products that were only sold in the first quarter of 2019. That is, between 2019-01-01 and 2019-03-31 inclusive.

First of all , I have created both tables Product and Sales and then inserted the records manually using insert command. The glimpse of it is shown below.





Sol) Select distinct product_id,product_name from product where product_id in (select product_id1 from sales where sales_date between '2019-01-01' AND '2019-03-31');



Q18) Table Views:

Column_Name	Type
Article_id	int
Author_id	int
Viewer_id	int
View_date	date

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. Return the resulted table in ascending order.

Sol) First of all , I created a table 'views' and then inserted the records manually using insert function.

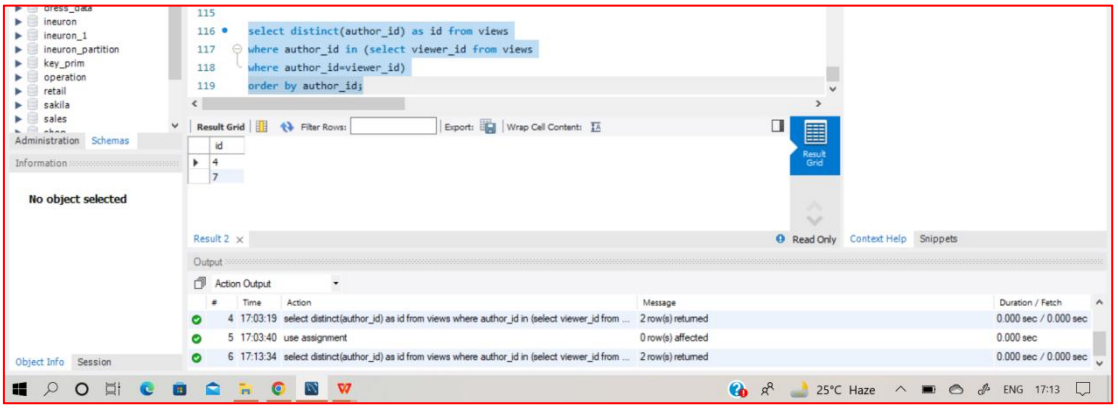
Input:

Article_id	Author_id	Viewer_id	View_date
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

Output:

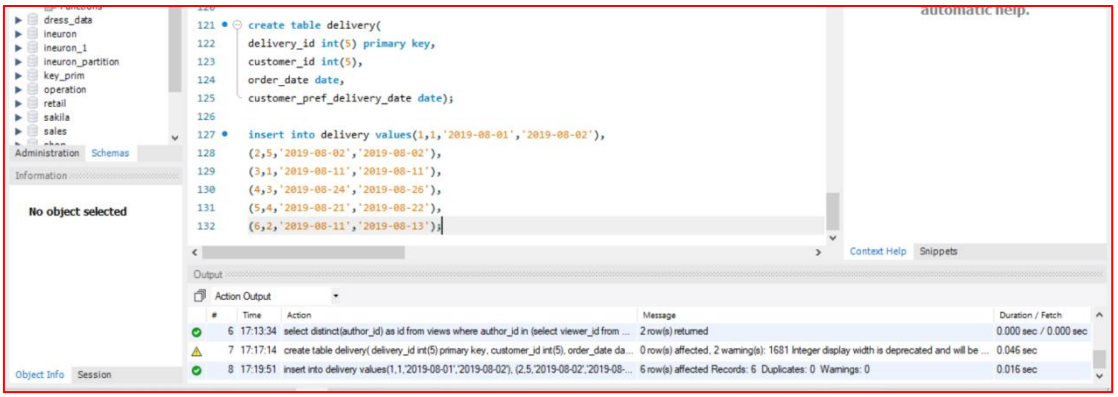
id
4
7

```
select distinct(author_id) as id from views
where author_id in (select viewer_id from views
where author_id=viewer_id)
Order by author_id;
```



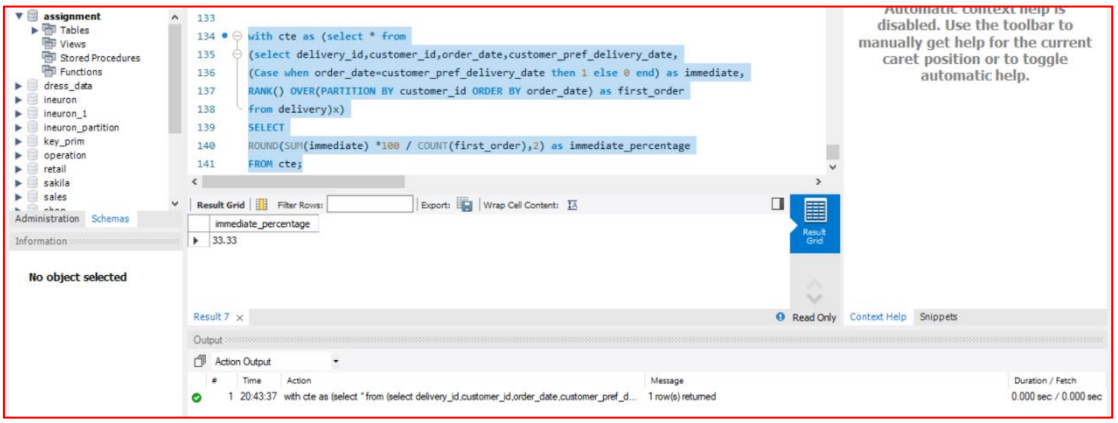
Q19) The table holds information about food delivery to customers that make orders at some date and specify a preferred delivery date (on the same order date or after it). If the customer's preferred delivery date is the same as the order date, then the order is called immediately; otherwise, it is called scheduled. Write an SQL query to find the percentage of immediate orders in the table, rounded to 2 decimal places.

Sol) First of all , I created a table 'delivery' and then inserted the records manually using insert function.




```
with cte as (select * from
(select delivery_id,customer_id,order_date,customer_pref_delivery_date,
(Case when order_date=customer_pref_delivery_date then 1 else 0 end) as
immediate,
```

```
RANK() OVER(PARTITION BY customer_id ORDER BY order_date) as first_order
from delivery)x)
SELECT
round(sum(immediate) *100 / count(first_order),2) as immediate_percentage
from cte;
```



Q20) A company is running Ads and wants to calculate the performance of each Ad. Performance of the Ad is measured using Click-Through Rate (CTR) where: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.

Input:

Column	Data type
Ad_id	int
User_id	int
Action	enum

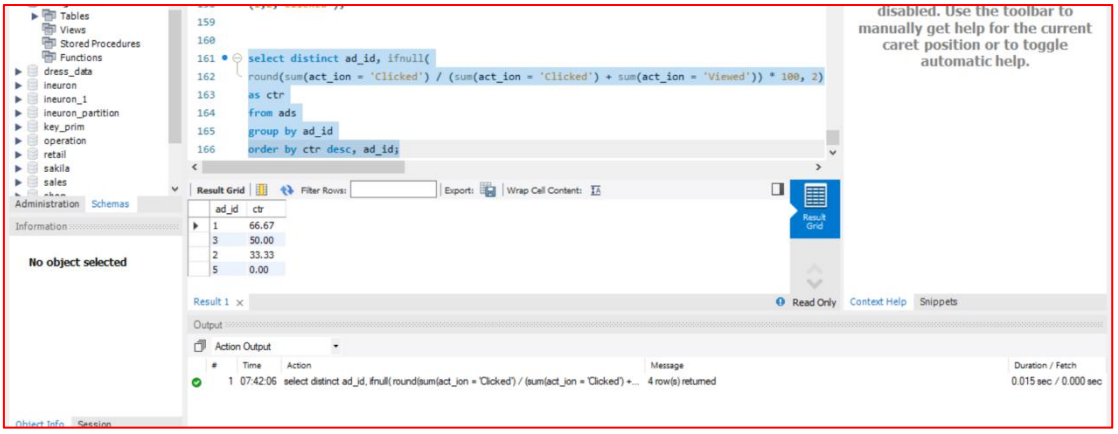
Output:

Ad_id	CTR
1	66.67
3	50
2	33.33
5	0

First of all , I have created a table 'ads' and then inserted the data manually using multi insert function.
Here, We have to Calculate Click Through Rate(CTR) which is used to calculate performance of the ads. The formula to calculate it is:

CTR = {0, Ad(Total clicks)/Ad(total clicks) + Ad(Total views)}
Note- Ignored ads is not considered.


```
Select distinct ad_id, ifnull(
round(sum(act_ion = 'Clicked') / (sum(act_ion = 'Clicked') + sum(act_ion =
'Viewed')) * 100, 2),0)
as ctr
from ads
group by ad_id
order by ctr desc, ad_id;
```



Q21) Table: Employee

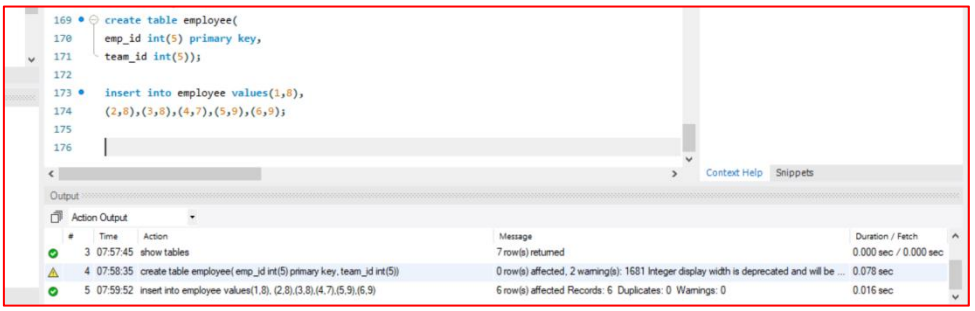
Column Name	Type
Employee_id	int
Team_id	int

Employee_id is the primary key for this table. Each row of this table contains the ID of each employee and their respective team. Write an SQL query to find the team size of each of the employees. Return result table in any order. The query result format is in the following example.

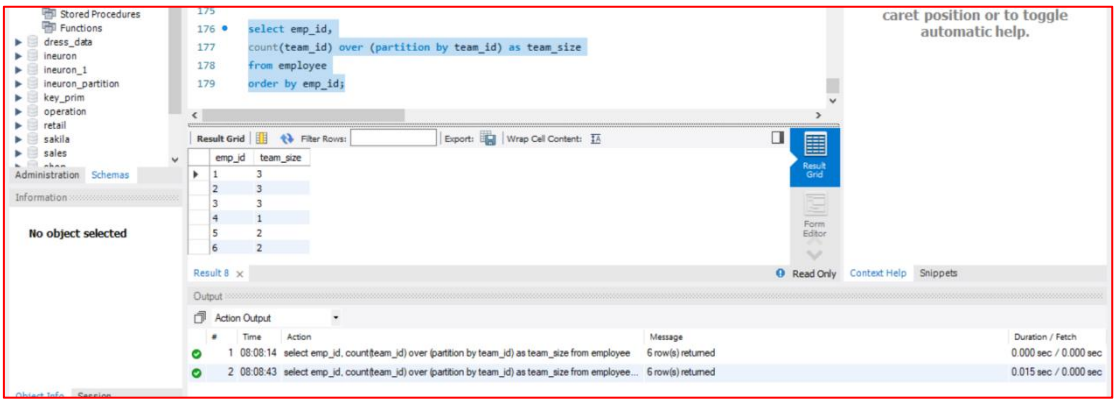
Output:

Employee_id	Team_size

Sol) As similar to above procedures, I have created an empty table first named 'employee' and then added records in it.And then performed the desired query over it. Here we have to find out team size of each employee of the table.



Select emp_id,
count(team_id) over (partition by team_id) as team_size
from employee
order by emp_id;



Q22) 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.
Return result table in any order

Input: Countries table

Column Name	Type
Country_id	int
Country_name	varchar

Weather Table:

Column Name	Type

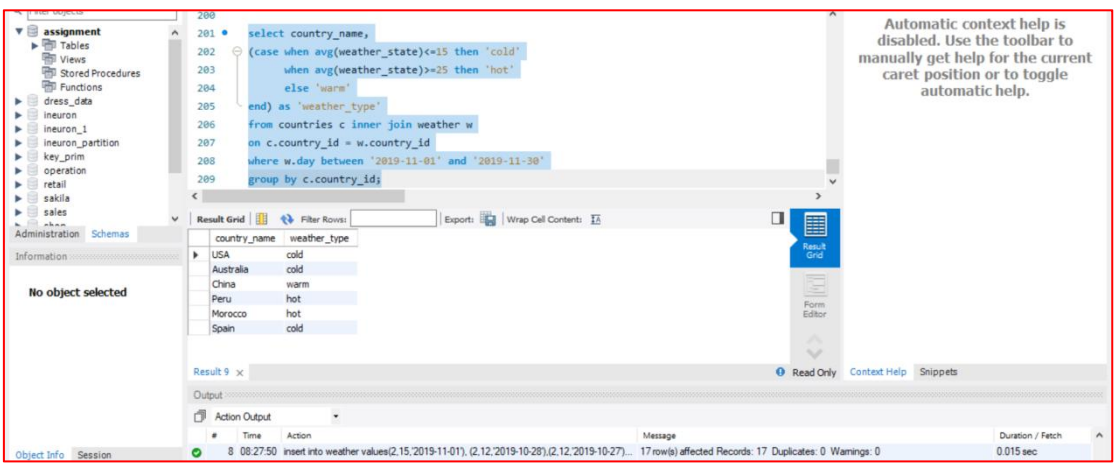
Country_id	int
Weather state	int
day	date

Output:

Country Name	**Weather Type**
USA	Cold
Australia	Cold
Peru	Hot
Morocco	Hot
China	Warm

Sol) At first, I created two empty table named 'countries' and 'weather' and then inserted the records row by row. Here we have to find out the Country name and Weather type of every country.


```
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 c inner join weather w
on c.country_id = w.country_id
where w.day between '2019-11-01' and '2019-11-30'
group by c.country_id;
```



Q23) Write an SQL query to find the average selling price for each product. average_price should be
rounded to 2 decimal places.
Return the result table in any order

Input:
Prices Table

Column Name	Type
Product_id	int
Start_date	date
End_date	date
price	int

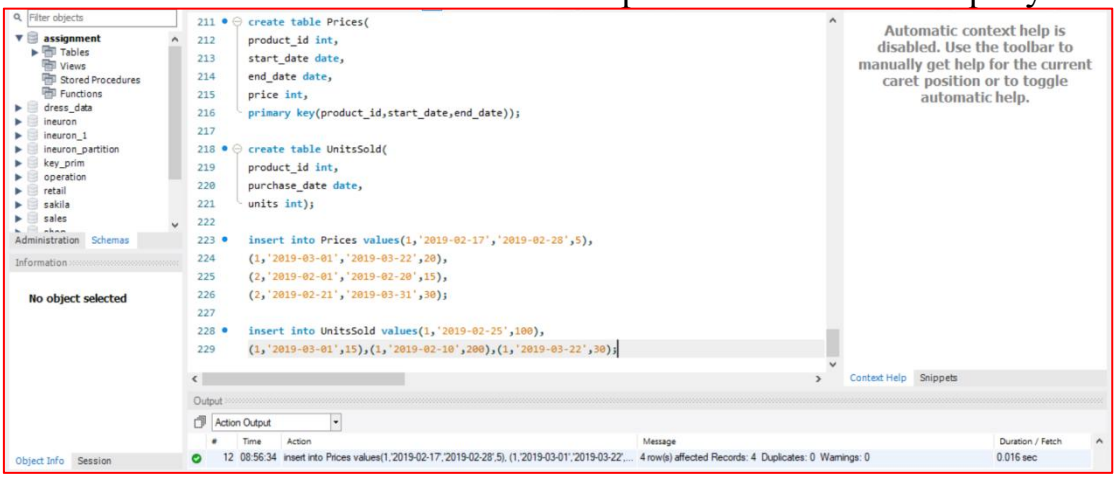
UnitsSold Table

Column Name	Type
Product_id	int
purchase_date	date
Units	int

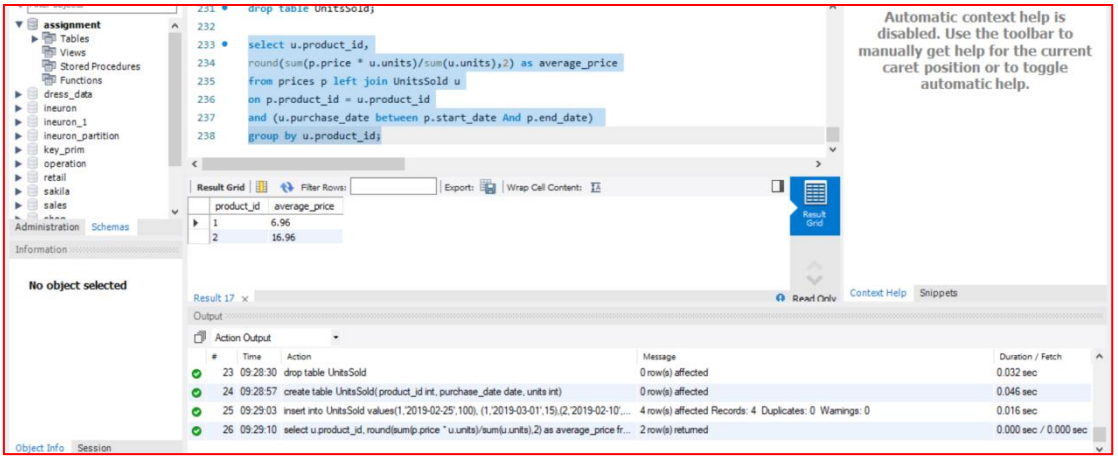
Output:

Product_id	Average_price
1	6.96
2	16.96

Sol) Initially,I created two empty tables named ‘Price’ and ‘UnitsSold’ And then inserted records in it. And then performed the desired query.




```
select u.product_id,
round(sum(p.price * u.units)/sum(u.units),2) as average_price
from prices p left join UnitsSold u
on p.product_id = u.product_id and (u.purchase_date between p.start_date
And p.end_date) group by u.product_id;
```



Q24) Input:
Activity Table

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.

Output:

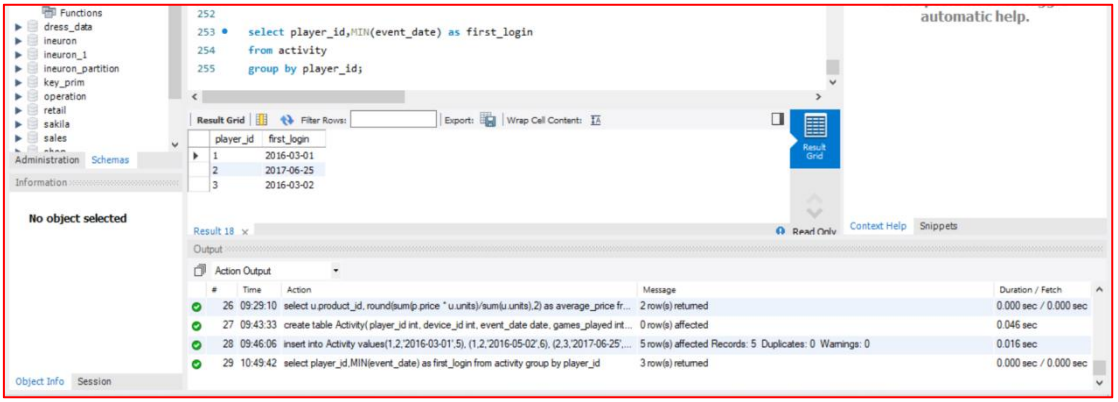
Player_id	First_login
1	2016-03-01
2	2017-06-25
3	2016-03-02

Sol) Initially, I created a empty table 'Activity' and then loaded the data inside it.

And then performed the desired query over it.

select player_id,MIN(event_date) as first_login
from activity

group by player_id;

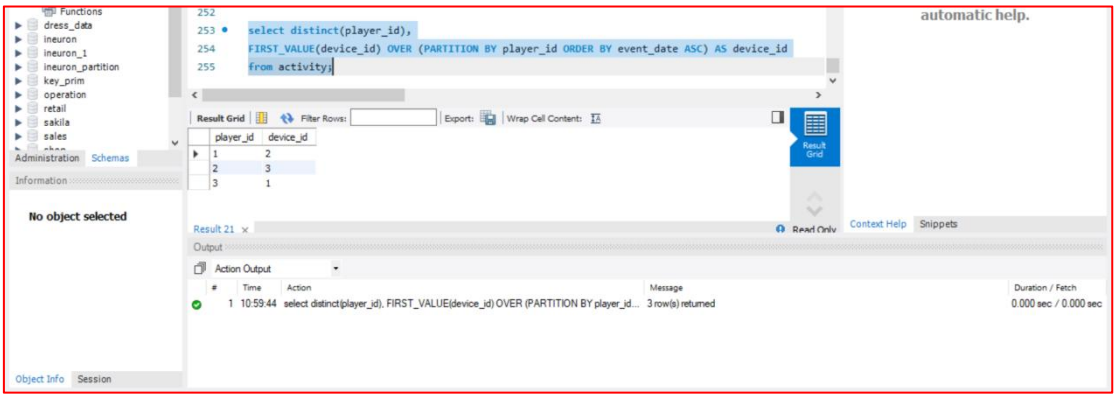


Q25) Write a SQL query to report the device that is first logged in for each player. Return the table in any order.

Output:

Player_id	Device_id
1	2
2	3
3	1

Sol) select distinct(player_id),
FIRST_VALUE(device_id) OVER (PARTITION BY player_id ORDER BY event_date ASC) AS device_id
from activity;



Q26) Input:
Products Table

Column Name	Type
Product_id	int
Product_name	varchar
Product_category	varchar

product_id is the primary key for this table. This table contains data about the company's products.

Orders Table

Column Name	Type
Product_id	int
Order date	date
unit	int

There is no primary key for this table. It may have duplicate rows. product_id is a foreign key to the Products table. unit is the number of products ordered in order_date.

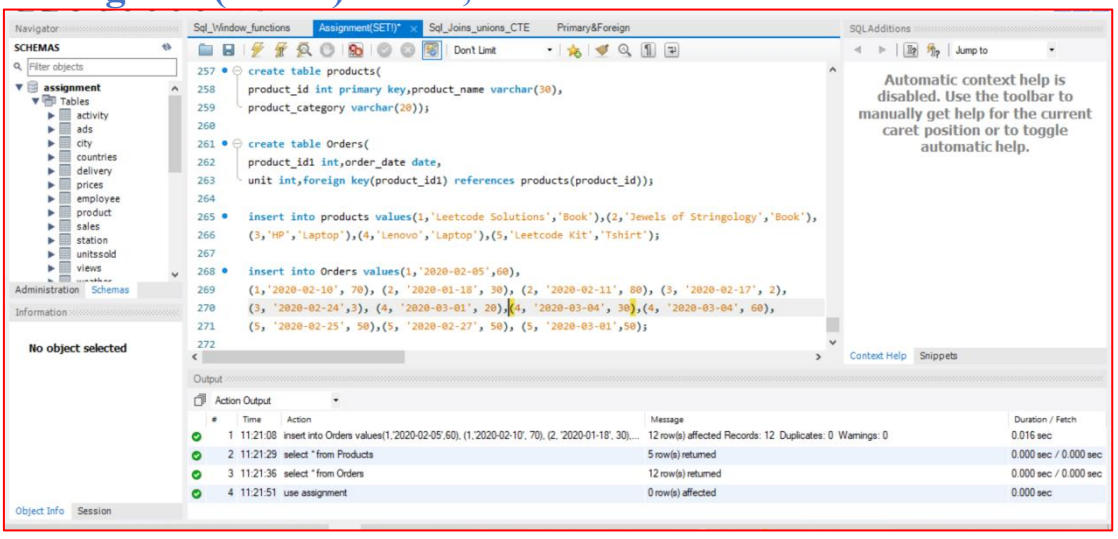
Write an SQL query to get the names of products that have at least 100 units ordered in February 2020 and their amount. Return result table in any order. The query result format is in the following example.

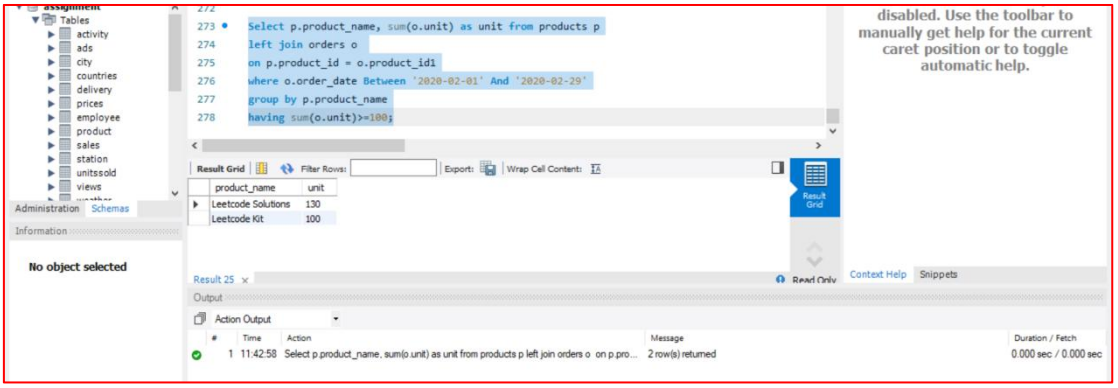
Output:

Product_name	Unit
Leetcode solutions	130
Leetcode kit	100

Sol) Create two empty table 'Products' & 'Orders' table and then inserted records in it row by row. And then perform the desired query.

```
Select p.product_name, sum(o.unit) as unit from products p
left join orders o
on p.product_id = o.product_id1
where o.order_date Between '2020-02-01' And '2020-02-29'
group by p.product_name
having sum(o.unit)>=100;
```





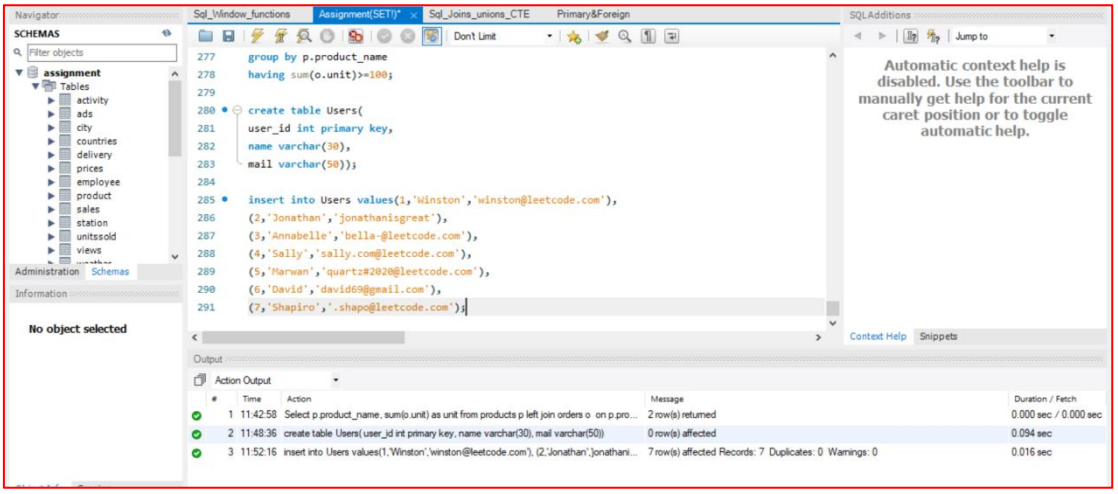
Q27) Input:

Users Table

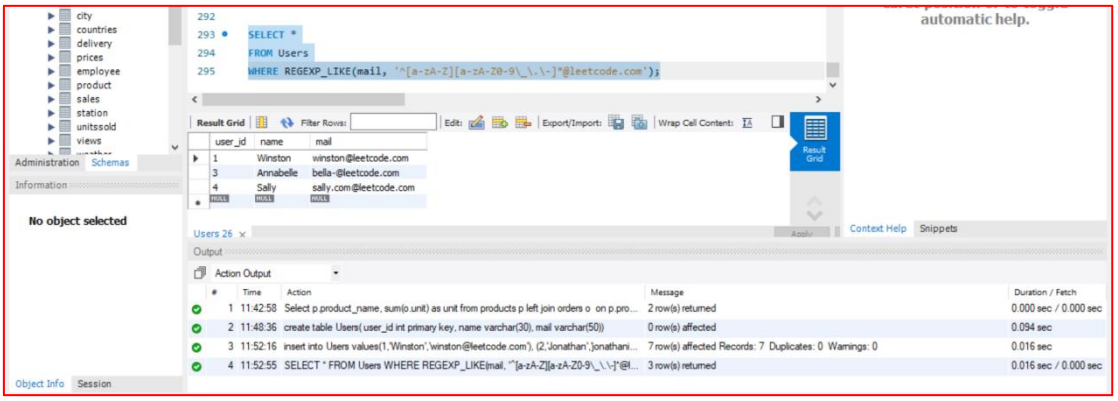
Column Name	Type
user_id	int
name	varchar
mail	varchar

user_id is the primary key for this table.
This table contains information of the users signed up in a website. Some emails are invalid.
Write an SQL query to find the users who have valid emails.
A valid e-mail has a prefix name and a domain where:
● The prefix name is a string that may contain letters (upper or lower case), digits, underscore
'_', period '.', and/or dash '-'. The prefix name must start with a letter.
● The domain is '@leetcode.com'.
Return the result table in any order.
The query result format is in the following example

Sol) Create an empty table 'Users' and then inserted the records manually row by row.
Then performed the desired query.

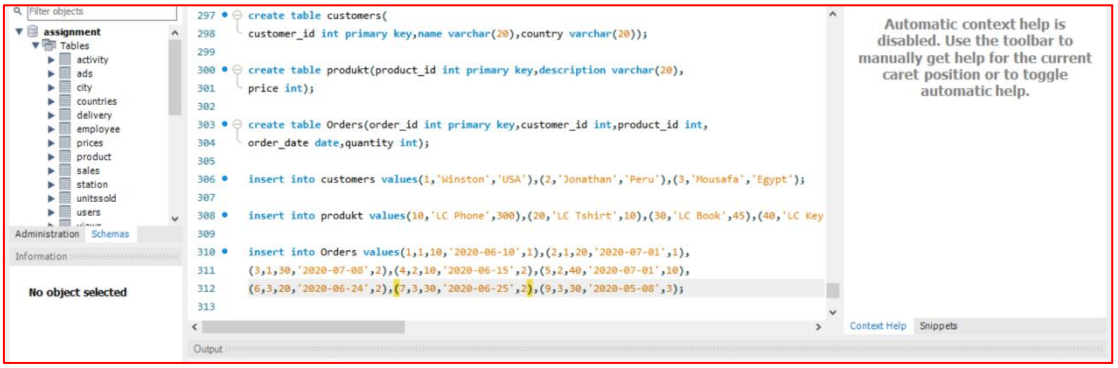


```
select *
from users
where regexp_like(mail, '^[a-za-z][a-za-z0-9\_\.\-]*@leetcode.com');
```

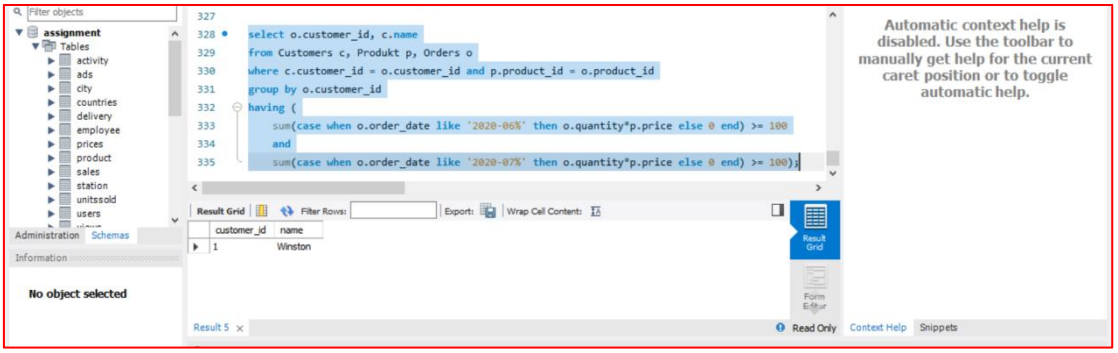


Q28) Write an SQL query to report the customer_id and customer_name of customers who have spent at least $100 in each month of June and July 2020. Return the result table in any order.
The query result format is in the following example

Sol) First of all , I have created 3 empty tables namely ‘Customers’,’Produkt’ and ‘Orders’. Then manually inserted records in it row by row and then performed the query. Here all the three tables are connected to each other, through some column.
We have to use it as a condition and find the desired output.




```
select o.customer_id, c.name
from Customers c, Produkt p, Orders 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) Input:
Table: TVProgram

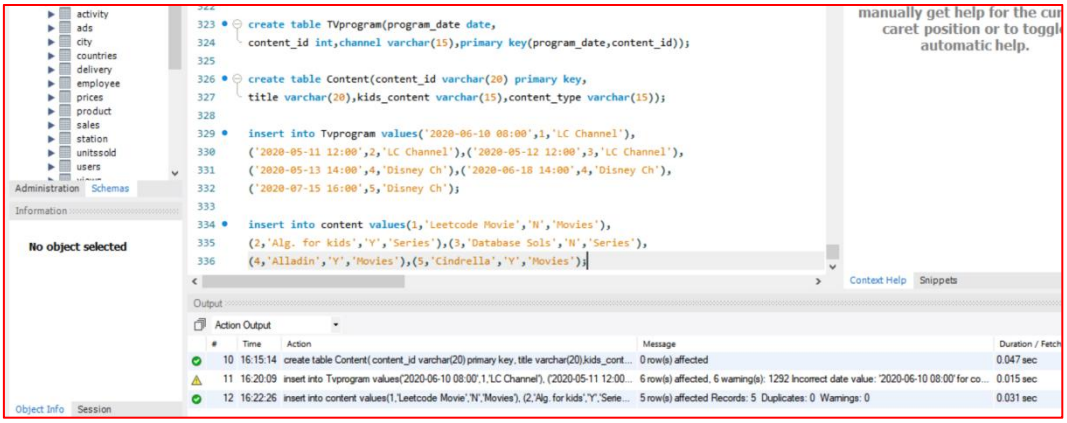
Column Name	Type
program_date	date
content_id	int
channel	varchar

Table: Content

Column Name	Type
content_id	varchar
title	varchar
Kids_content	enum
content_type	varchar

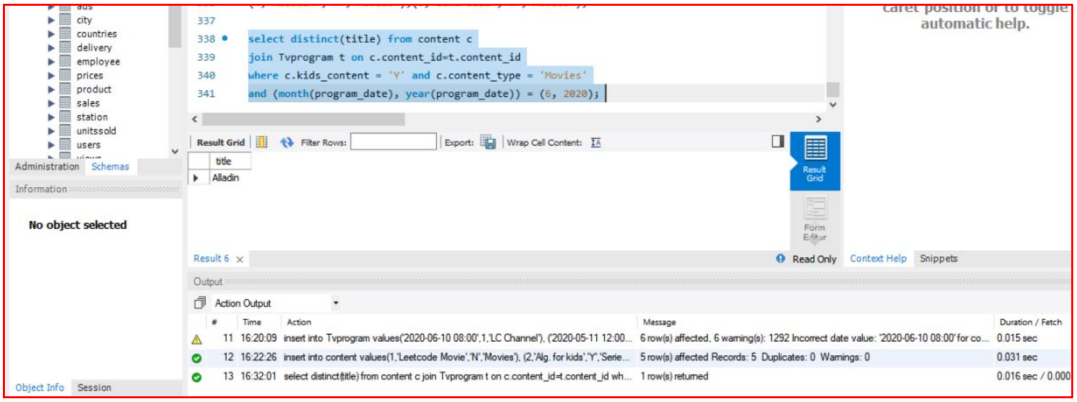
Write an SQL query to report the distinct titles of the kid-friendly movies streamed in June 2020. Return the result table in any order.
The query result format is in the following example.

Sol) Create an empty table 'TvProgram' and 'Content' then inserted the records manually row by row.Then performed the desired query.




```
Select distinct(title) from content c
join Tvprogram t on c.content_id=t.content_id
where c.kids_content = 'Y' and c.content_type = 'Movies'
```

and (month(program_date), year(program_date)) = (6, 2020);



Q30) Table: NPV

Column Name	Type
id	int
year	int
npv	int

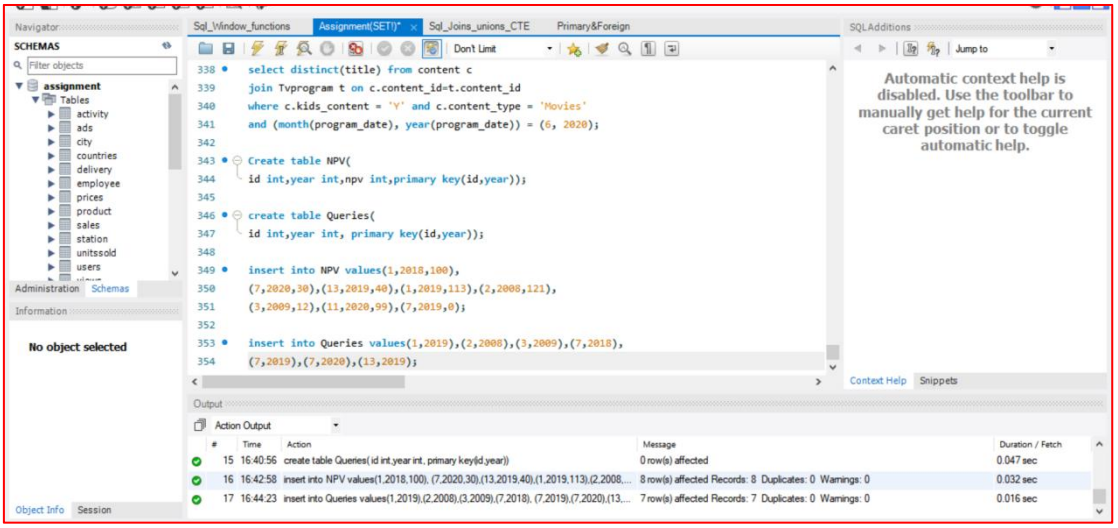
(id, year) is the primary key of this table.

Table: Queries

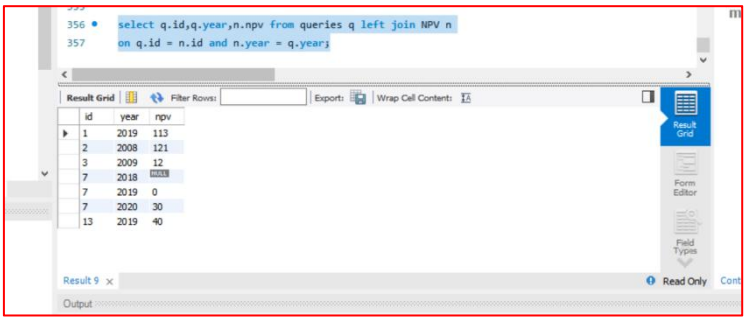
Column Name	Type
id	int
year	int

Write an SQL query to find the npv of each query of the Queries table.
Return the result table in any order. The query result format is in the following example.

Sol) Create two empty table 'NPV' and 'Queries' then inserted the records manually row by row.Then performed the desired query.



Select q.id,q.year,n.npv from queries q left join NPV n on q.id = n.id and n.year = q.year;



Q31) SAME QUESTION AS Q30

Q32) Table: Employees

Column Name	Type
id	int
name	varchar

id is the primary key for this table.
Each row of this table contains the id and the name of an employee in a company.

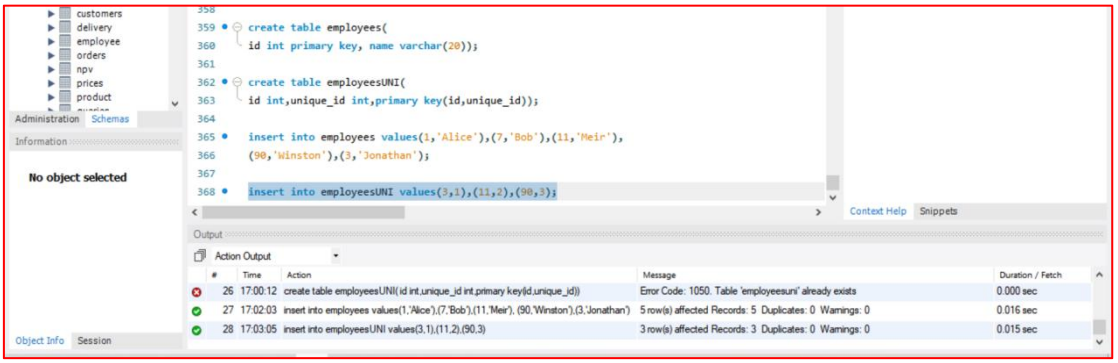
Table: EmployeeUNI

Column Name	Type
id	int
unique_id	int

(id, unique_id) is the primary key for this table.

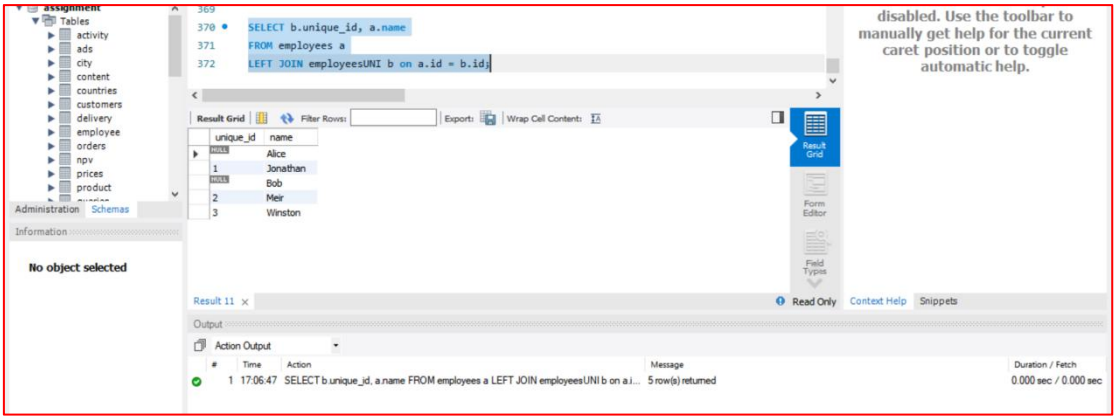
Write an SQL query to show the unique ID of each user, If a user does not have a unique ID replace just show null.

Sol) Create two empty table ‘Employees’ and ‘EmployeesUNI’ then inserted the records manually row by row.Then performed the desired query.



In this query, I have used Left join function because it is asked to show null values if a user does not have unique id.

```
SELECT b.unique_id, a.name
FROM employees a
LEFT JOIN employeesUNI b on a.id = b.id;
```



Q33) Table: Users

Column Name	Type
id	int
name	varchar

id is the primary key for this table. name is the name of the user.

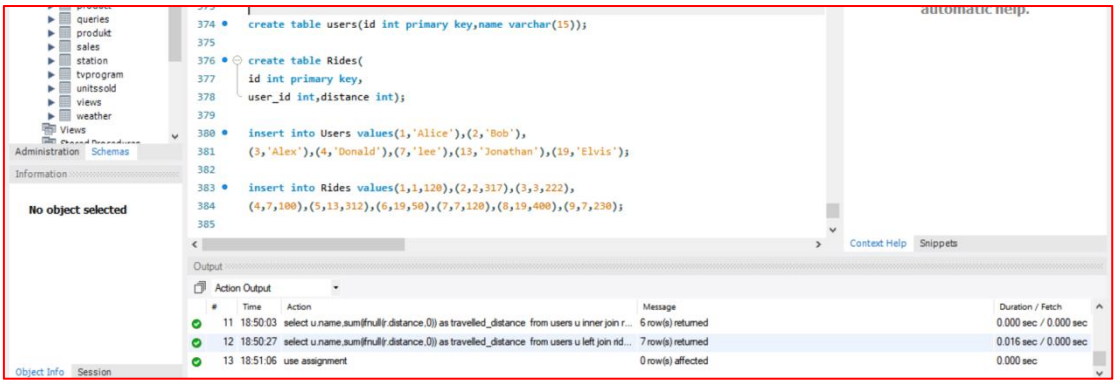
Table: Rides

Column Name	Type
id	int
user_id	int
distance	int

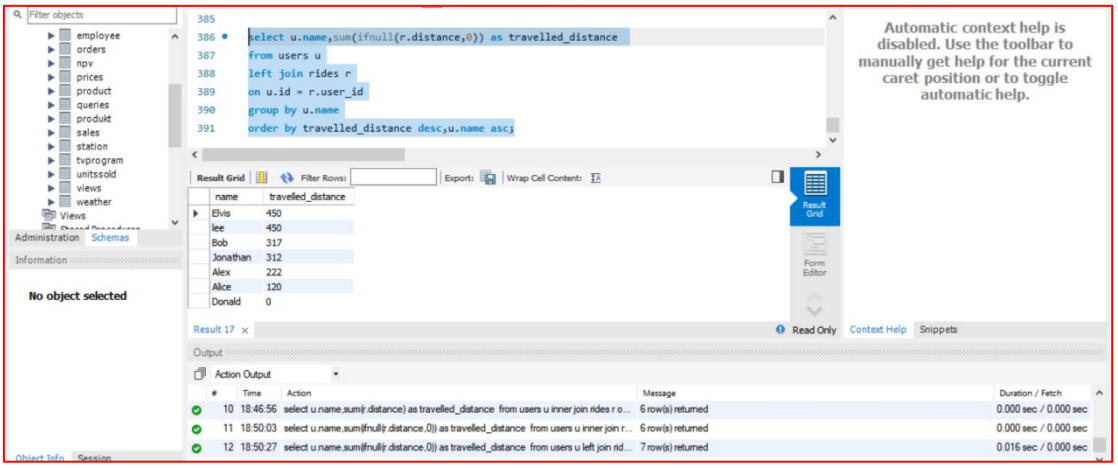
id is the primary key for this table.
user_id is the id of the user who travelled the distance "distance".

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. The query result format is in the following example

Sol) Create two empty table 'Users' and 'Rides' then inserted the records manually row by row.Then performed the desired query.




```
select u.name,sum(ifnull(r.distance,0)) as travelled_distance
from users u
left join rides r
on u.id = r.user_id
group by u.name
order by travelled_distance desc,u.name asc;
```



Q34) SAME QUESTION AS Q26

Q35) Table: Movies

Column Name	Type
movie_id	int
title	varchar

Table: Users

Column Name	Type
user_id	int
name	varchar

Table: MovieRating

Column Name	Type
movie_id	int
user_id	int
rating	int
created_at	date

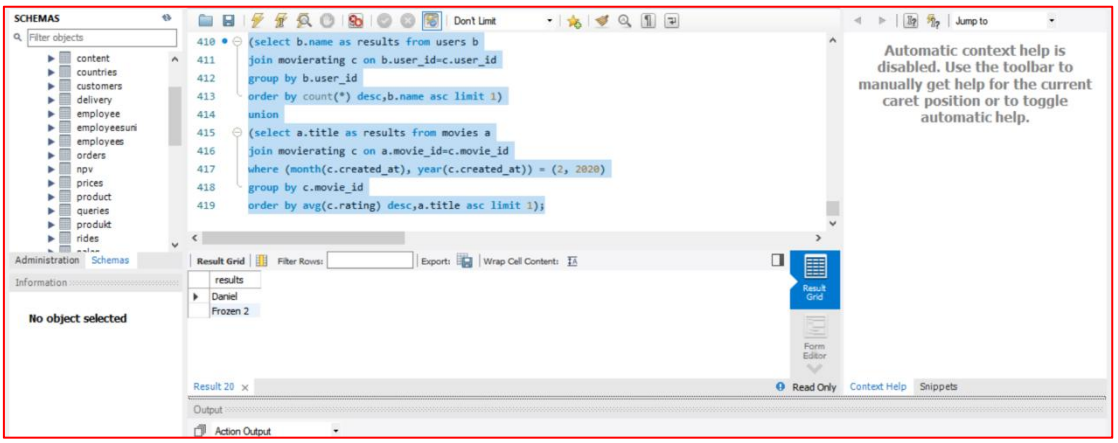
Write an SQL query to:

- **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.**

Sol) Create three empty table 'Movies','Users' and 'Movierating' then inserted the records manually row by row.Then performed the desired query.

(select b.name as results from users b
join movierating c on b.user_id=c.user_id
group by b.user_id

```
order by count(*) desc,b.name asc limit 1)
UNION
(select a.title as results from movies a
join movierating c on a.movie_id=c.movie_id
where (month(c.created_at), year(c.created_at)) = (2, 2020)
group by c.movie_id
order by avg(c.rating) desc,a.title asc limit 1);
```



Q36) SAME QUESTION AS Q33

Q37) SAME QUESTION AS Q32

Q38) Table: Departments

Column Name	Type
id	int
name	varchar

id is the primary key of this table.The table has information about the id of each department of a university.

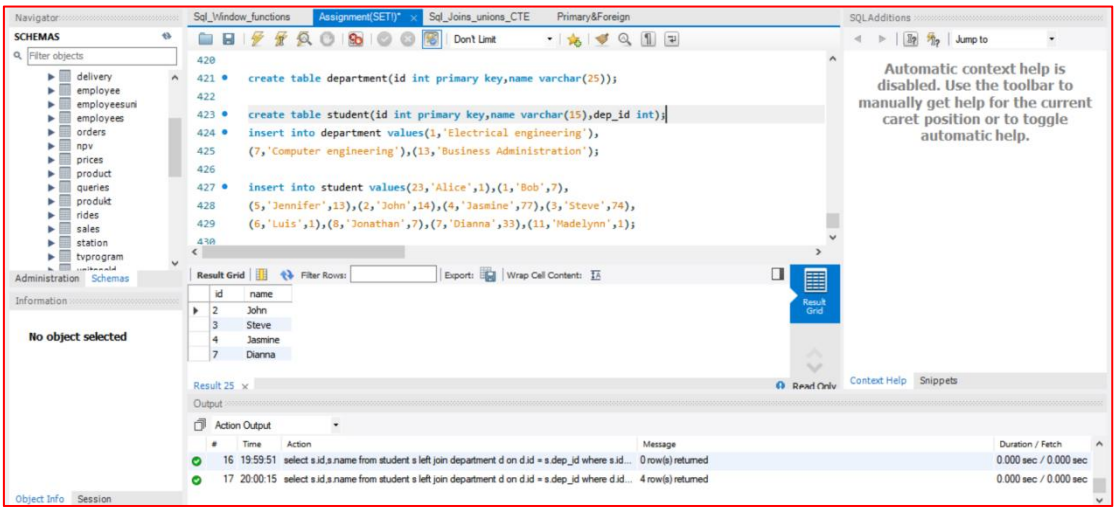
Table: Students

Column Name	Type
id	int
name	varchar
department_id	int

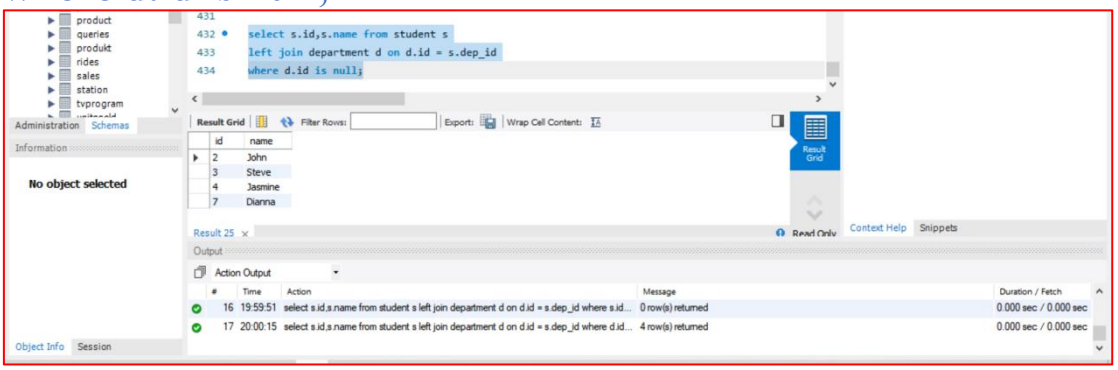
id is the primary key of this table. The table has information about the id of each student at a university and the id of the department he/she studies at.

Write an SQL query to find the id and the name of all students who are enrolled in departments that no longer exist. Return the result table in any order.

Sol) Create two empty table 'Departments' and 'Student' then inserted the records manually row by row.Then performed the desired query.




```
select s.id,s.name from student s
left join department d on d.id = s.dep_id
where d.id is null;
```



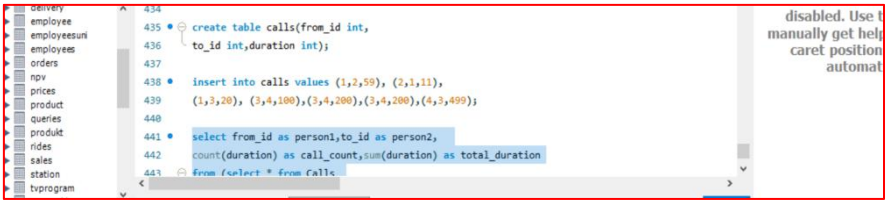
Q39) Table: Calls

Column Name	Type
from_id	int
to_id	int
duration	int

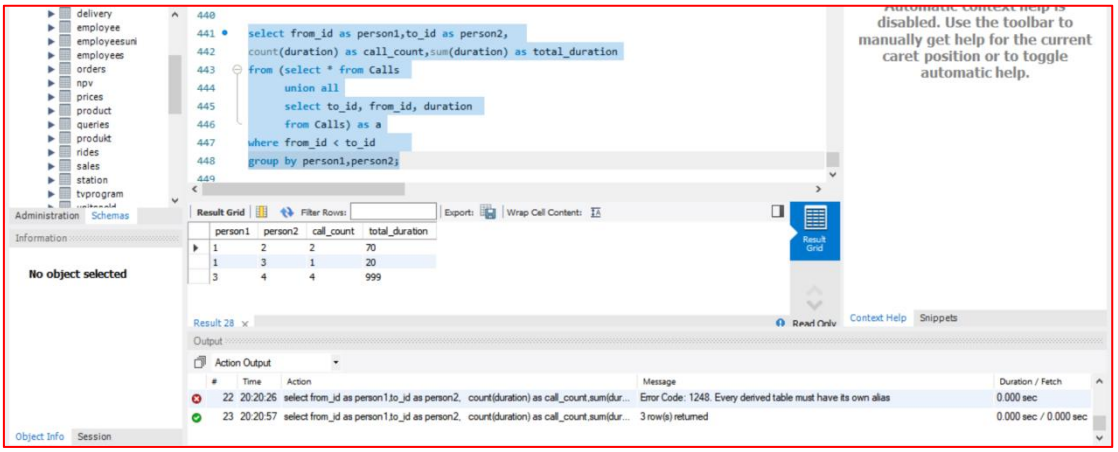
This table does not have a primary key, it may contain duplicates.
This table contains the duration of a phone call between from_id and to_id.from_id != to_id

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.

Sol) Create an empty table 'Calls' then inserted the records manually row by row.Then performed the desired query.




```
Select from_id as person1,to_id as person2,
Count(duration) as call_count,sum(duration) as total_calls
From (select * from calls
        Union all
        Select to_id,from_id,duration
        From calls) as a
Where from_id < to_id
Group by person1,person2;
```



Q40) SAME QUESTION AS Q23

Q41.
Table: Warehouse

Column Name	Type
name	varchar
product_id	int
units	int

(name, product_id) is the primary key for this table.Each row of this table contains the information of the products in each warehouse.

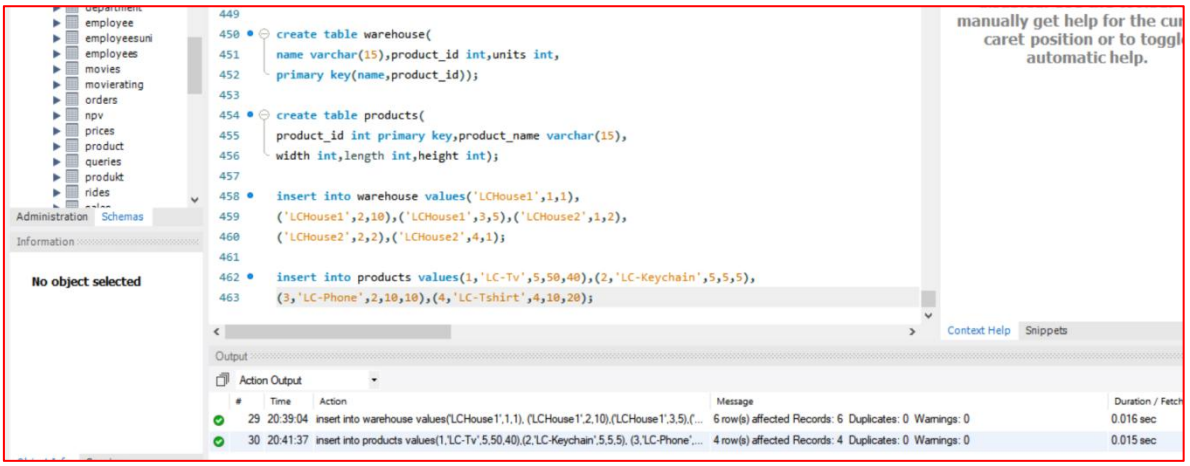
Table: Products

Column Name	Type
product_id	int
product_name	varchar
Width	int
Length	int
Height	int

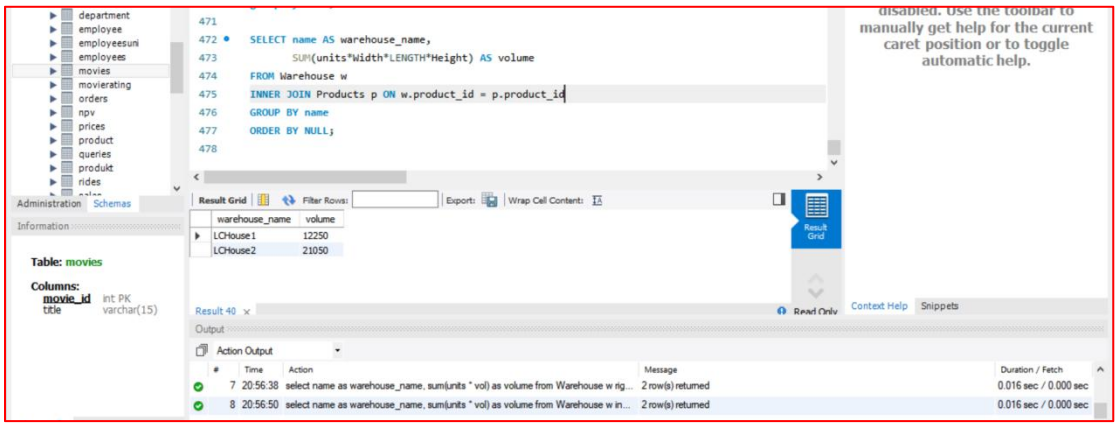
product_id is the primary key for this table.Each row of this table contains information about the product dimensions (Width, Length, and Height) in feets of each product.

Write an SQL query to report the number of cubic feet of volume the inventory occupies in each warehouse. Return the result table in any order.The query result format is in the following example.

Sol) Create two empty table 'Warehouse' and 'Products' then inserted the records manually row by row.Then performed the desired query.



```
select name as warehouse_name,
       sum(units*width*length*height) as volume
from warehouse w
inner join products p on w.product_id = p.product_id
group by name
order by null;
```



Q42) Table: Sales

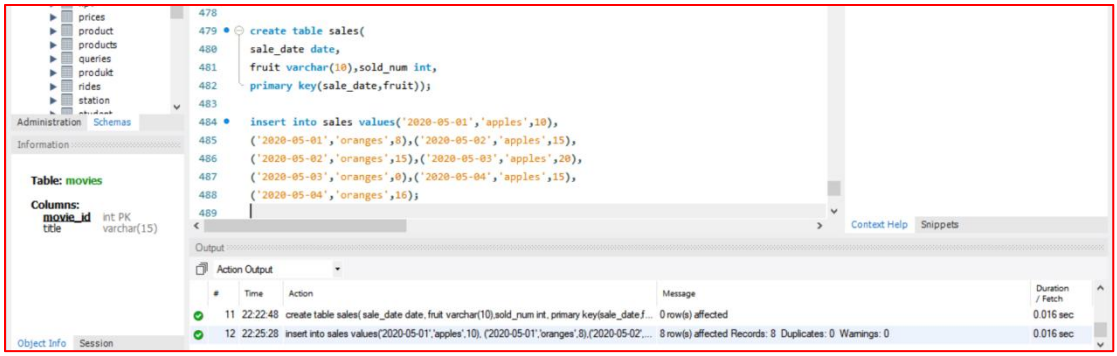
Column Name	Type
sale_date	date
fruit	enum
sold_num	int

(sale_date, fruit) is the primary key for this table.
This table contains the sales of "apples" and "oranges" sold each day.

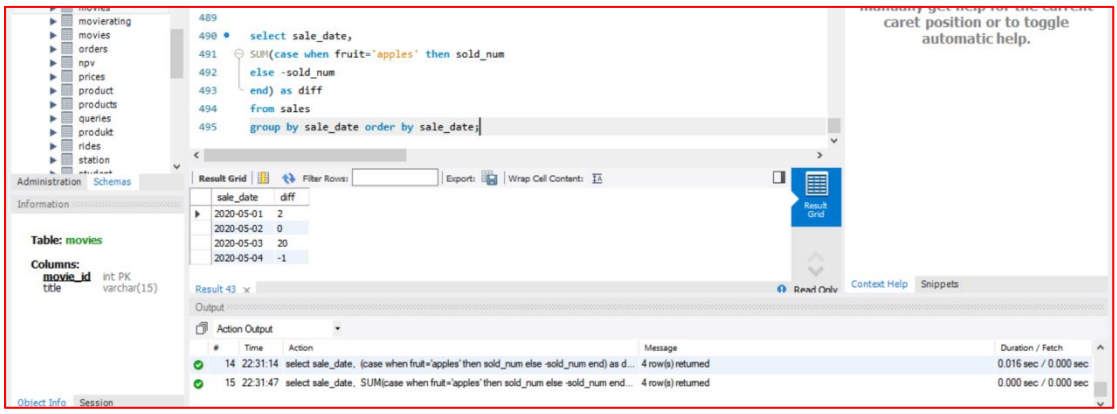
Write an SQL query to report the difference between the number of apples and oranges sold each day. Return the result table ordered by sale_date.

The query result format is in the following example

Sol) create an empty table 'sales' and insert records inside it row by row manually. And then perform the desired query in it.




```
select sale_date,
SUM(case when fruit='apples' then sold_num
else -sold_num
end) as diff
from sales
group by sale_date order by sale_date;
```



Q43) Table: Activity

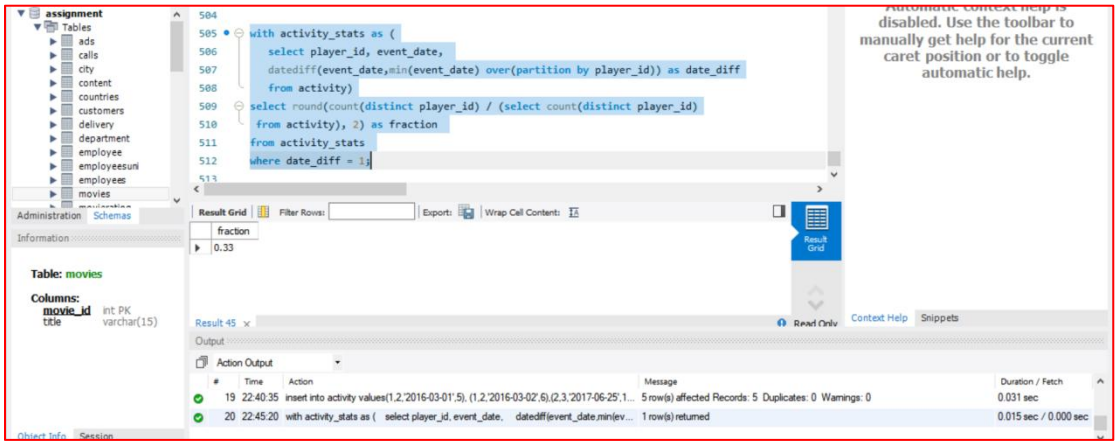
Column Name	Type
player_id	int
device_id	int
event_date	date
games_played	int

(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 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 play.

Sol) create an empty table 'Activity' and insert records inside it row by row manually. And then perform the desired query in it.


```
with activity_stats as (
   select player_id, event_date,
   datediff(event_date,min(event_date) over(partition by player_id))
as date_diff
   from activity)
select round(count(distinct player_id) / (select count(distinct
player_id)
 from activity), 2) as fraction
from activity_stats
where date_diff = 1;
```



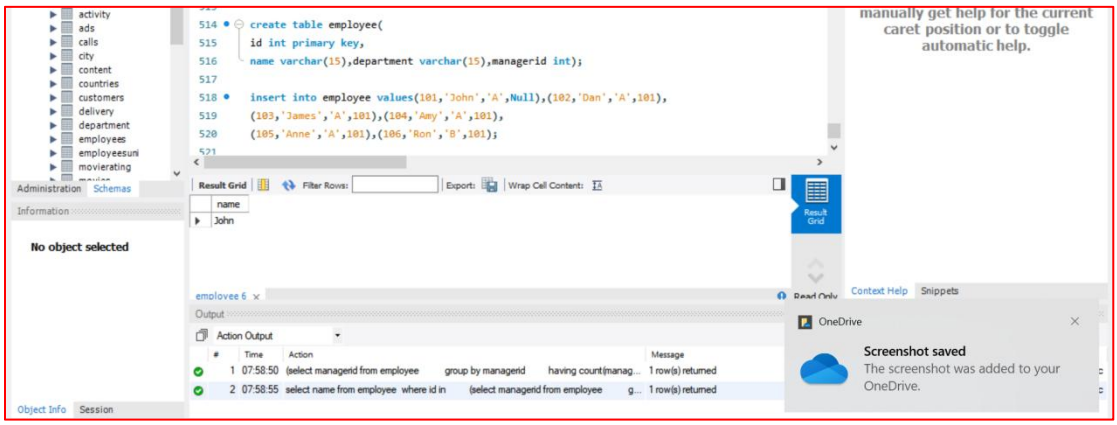
Q44) Table: Employee

Column Name	Type
id	int
name	varchar
department	varchar
managerId	int

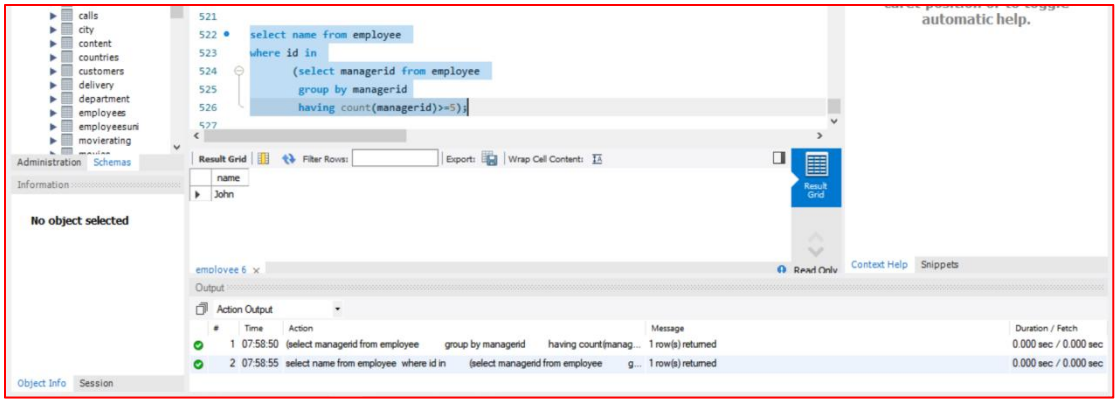
id is the primary key column for this table.
Each row of this table indicates the name of an employee, their department, and the id of their manager.If manager_Id is null, then the employee does not have a manager. No employee will be the manager of themself.

Write an SQL query to report the managers with at least five direct reports. Return the result table in any order. The query result format is in the following example.

Sol) create an empty table 'Employee' and insert records inside it row by row manually. And then perform the desired query in it.




```
select name from employee
where id in
      (select managerid from employee
       group by managerid
       having count(managerid)>=5);
```



Q45) Table: Student

Column Name	Type
student_id	int
student_name	varchar
gender	varchar
dept_id	int

student_id is the primary key column for this table.dept_id is a foreign key to dept_id in the Department tables.Each row of this table indicates the name of a student, their gender, and the id of their department.

Table: Department

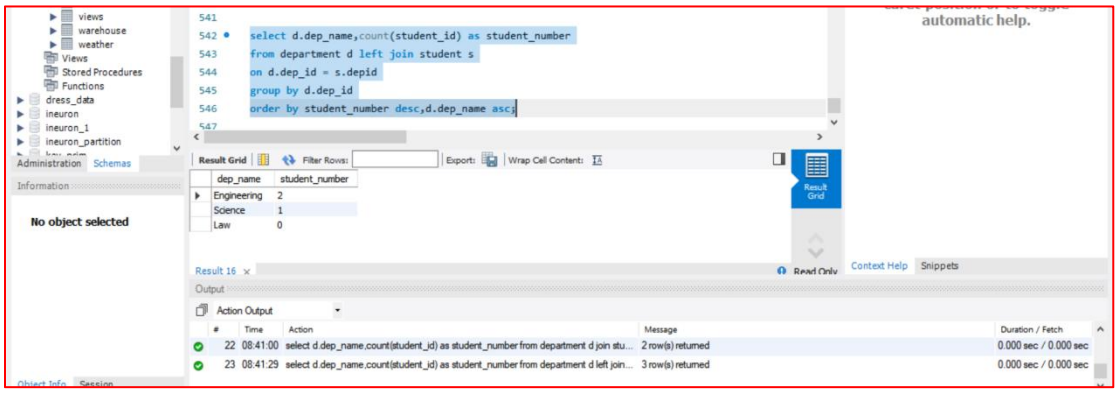
Column Name	Type
dept_id	int
dept_name	varchar

dept_id is the primary key column for this table. Each row of this table contains the id and the name of a department.

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. The query result format is in the following example.

Sol) create two empty table 'Student' & 'Employee' and insert records inside it row by row manually. And then perform the desired query in it.


```
select d.dep_name,count(student_id) as student_number
from department d left join student s
on d.dep_id = s.depid
group by d.dep_id
order by student_number desc,d.dep_name asc;
```



Q46)

Table: Customer

Column Name	Type
customer_id	int
product_key	int

There is no primary key for this table. It may contain duplicates. Product_key is a foreign key to the Product table.

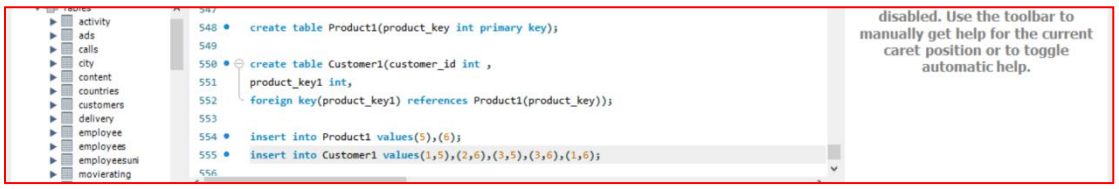
Table: Product

Column Name	Type
product_key	int

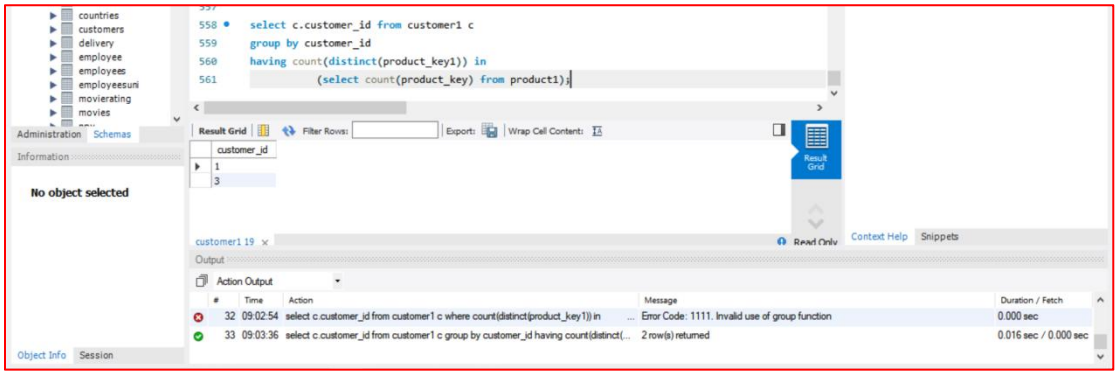
Product_key is a primary key for this table.

Write an SQL query to report the customer ids from the Customer table that bought all the products in the Product table. Return the result table in any order. The query result format is in the following example.

Sol) create two empty table 'Student' & 'Employee' and insert records inside it row by row manually. And then perform the desired query in it.




```
select c.customer_id from customer1 c
group by customer_id
having count(distinct(product_key1)) in
        (select count(product_key) from product1);
```



Q47) Table: Project

Column Name	Type
Project_id	int
Employee_id	int

(project_id, employee_id) is the primary key of this table. employee_id is a foreign key to the Employee table. Each row of this table indicates that the employee with employee_id is working on the project with project_id.

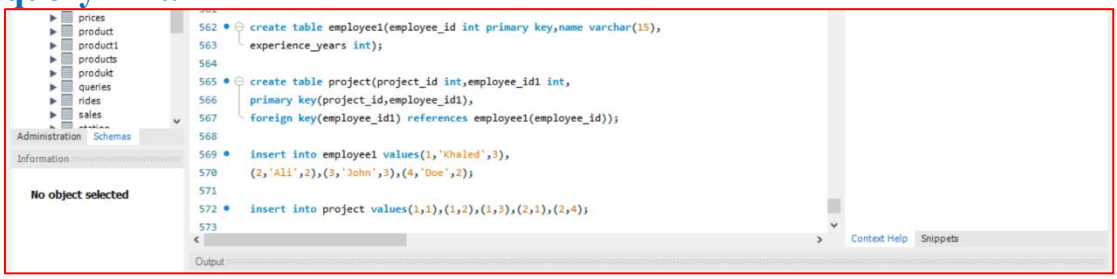
Table: Employee

Column Name	Type
employee_id	int
name	varchar
experience_years	int

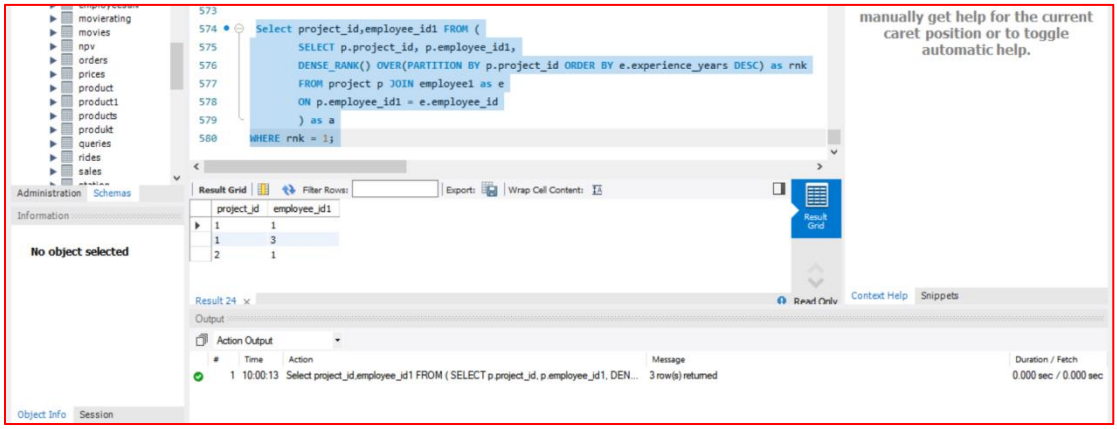
employee_id is the primary key of this table. Each row of this table contains information about one employee.

Write an SQL query that reports the most experienced employees in each project. In case of a tie, report all employees with the maximum number of experience years. Return the result table in any order. The query result format is in the following example.

Sol) create two empty table 'Project' & 'Employee' and insert records inside it row by row manually. And then perform the desired query in it.




```
Select project_id,employee_id1 FROM (
            SELECT p.project_id, p.employee_id1,
            DENSE_RANK() OVER(PARTITION BY p.project_id
ORDER BY e.experience_years DESC) as rnk
      FROM project p JOIN employee1 as e
      ON p.employee_id1 = e.employee_id
            ) as a
WHERE rnk = 1;
```



Q48) Table: Books

Column Name	Type
book_id	int
name	varchar
available_from	date

book_id is the primary key of this table.

Table: Orders

Column Name	Type
order_id	int
book_id	int
quantity	int
dispatch_date	date

order_id is the primary key of this table. Book_id is a foreign key to the Books table.

Write an SQL query that reports the books that have sold less than 10 copies in the last year, excluding books that have been available for less than one month from today. Assume today is 2019-06-23.
Return the result table in any order. The query result format is in the following example.

.Sol) create two empty table ‘Books’ & ‘Orders’ and insert records inside it row by row manually. And then perform the desired query in it.

ORDER table missing in question.

Q49.

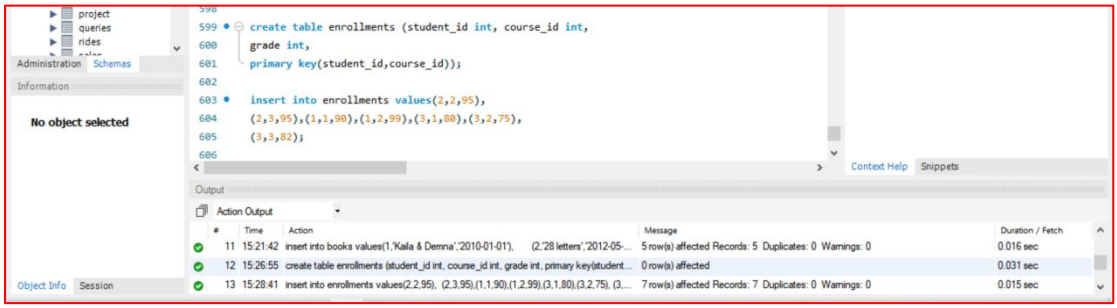
Table: Enrollments

Column Name	Type
student_id	int
course_id	int
grade	int

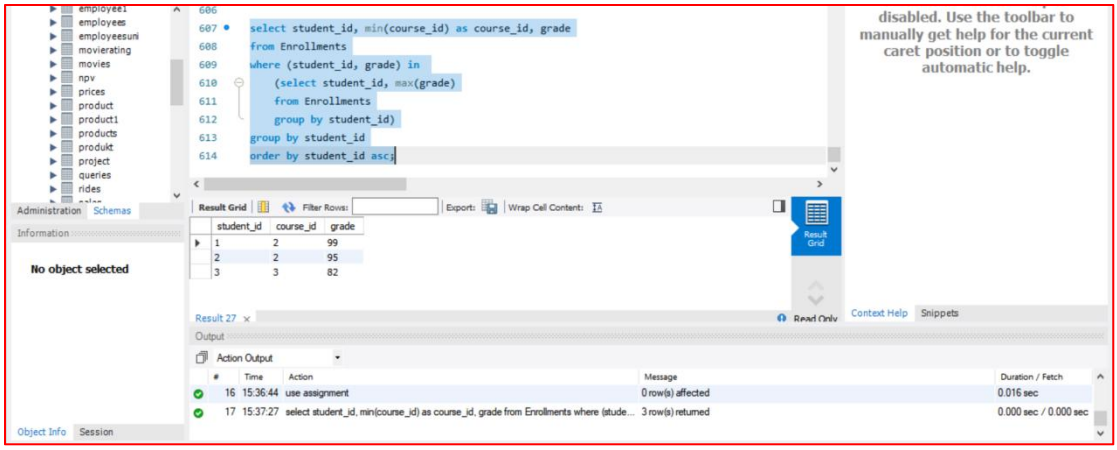
(student_id, course_id) is the primary key of this table.

Write a SQL query to find the highest grade with its corresponding course for each student. In case of a tie, you should find the course with the smallest course_id.Return the result table ordered by student_id in ascending order. The query result format is in the following example.

.Sol) create an empty table 'Enrollments' and insert records inside it row by row manually. And then perform the desired query in it.




```
select student_id, min(course_id) as course_id, grade
from Enrollments
where (student_id, grade) in
    (select student_id, max(grade)
    from Enrollments
    group by student_id)
group by student_id
order by student_id asc;
```



Q50)

Table: Players

Column Name	Type
player_id	int
Group_id	int

team_id is the primary key of this table.Each row of this table represents a single football team.

Table: Matches

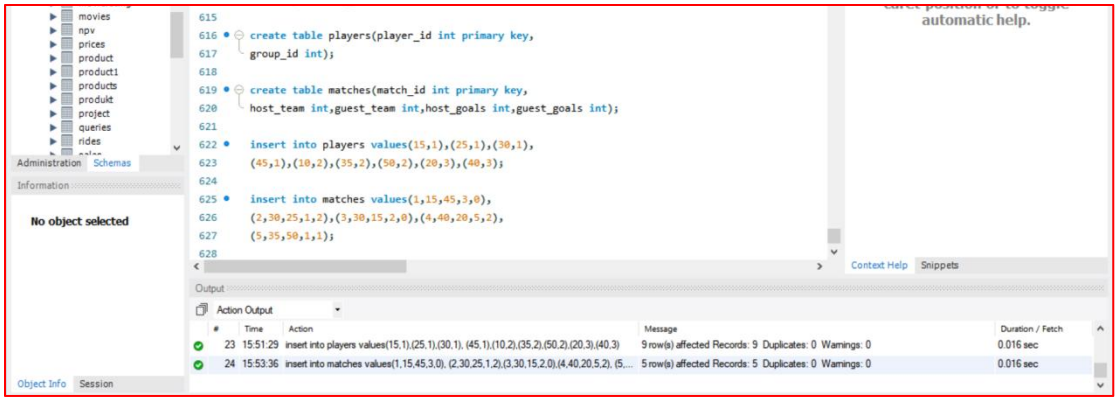
Column Name	Type
match_id	int
host_team	int
guest_team	int
host_goals	int
guest_goals	int

match_id is the primary key of this table.

Each row is a record of a finished match between two different teams. Teams host_team and guest_team are represented by their IDs in the Teams table (team_id), and they scored host_goals and guest_goals goals, respectively.The winner in each group is the player who scored the maximum total points within the group. In the case of a tie, the lowest player_id wins.

Write an SQL query to find the winner in each group. Return the result table in any order. The query result format is in the following example.

.Sol) create two empty table ‘Players’ & Matches and insert records inside it row by row manually. And then perform the desired query in it.




```
select group_id, player_id from
   (select p.group_id, ps.player_id,
           sum(ps.score) as score
                   from   players p inner join
              (select  first_player  as  player_id,  first_score   as
score from  matches
                        union all
              select  second_player  as  player_id,second_score
as score
              from   matches) ps
       on  p.player_id = ps.player_id
       group  by ps.player_id
       order  by group_id, score desc, player_id) top_scores
group  by group_id;
```