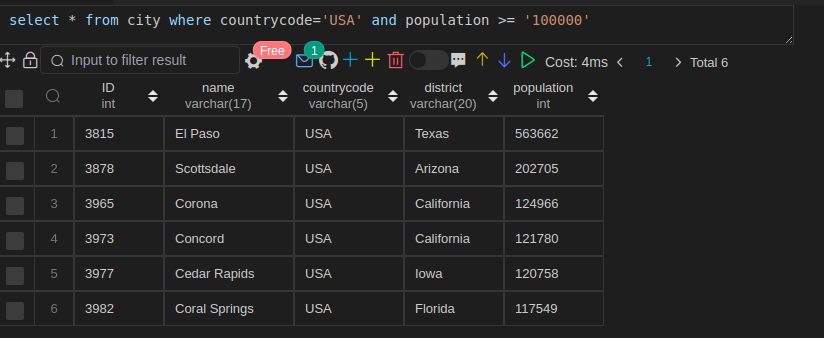
Q1. Query all columns for all American cities in the CITY table with populations larger than 100000. The CountryCode for America is USA.

Query :

| select \* from city where countrycode='USA' and population >= '100000' |
| --- |

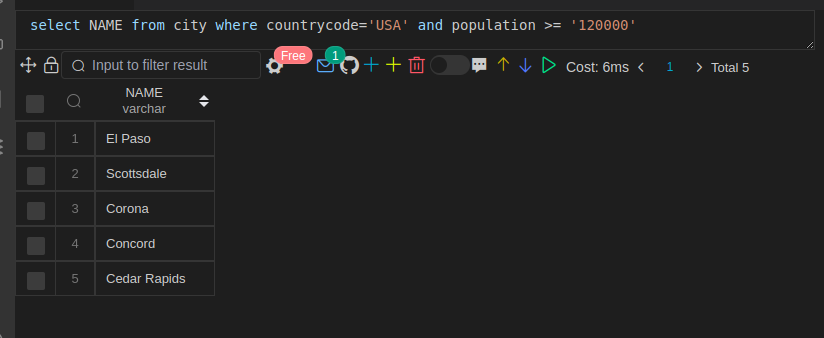
Output : 

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

Query:

| select NAME from city where countrycode='USA' and population >= '120000' |
| --- |

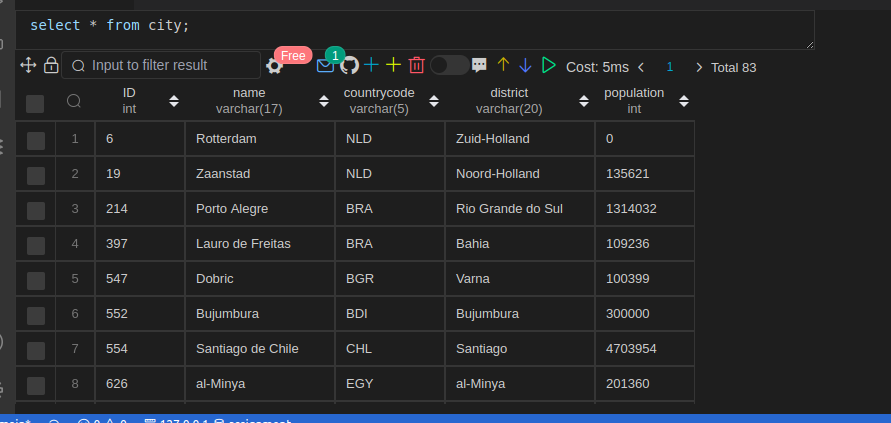
Output:



Q3. Query all columns (attributes) for every row in the CITY table.

Query:

| Select \* from city; |
| --- |

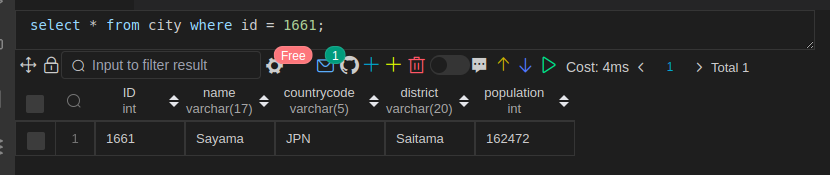


Q4. Query all columns for a city in CITY with the ID 1661.

Query:

| select \* from city where id = 1661; |
| --- |

Output:

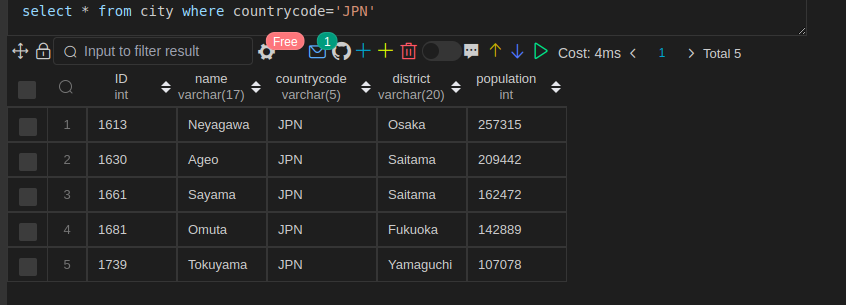


Q5. Query all attributes of every Japanese city in the CITY table. The COUNTRYCODE for Japan is JPN.

Query:

| select \* from city where countrycode='JPN' |
| --- |

Output:

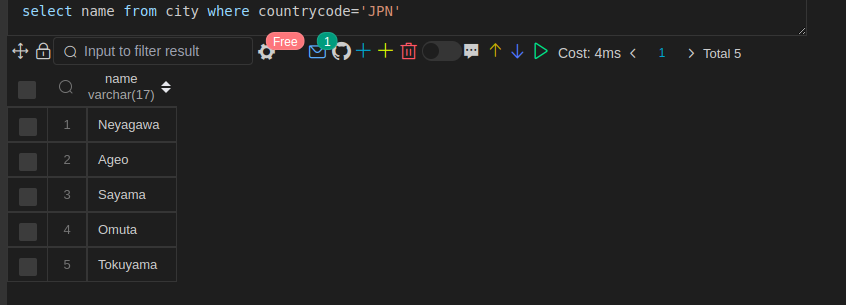


Q6. Query the names of all the Japanese cities in the CITY table. The COUNTRYCODE for Japan is JPN.

Query:

| select nam from city where countrycode='JPN' |
| --- |

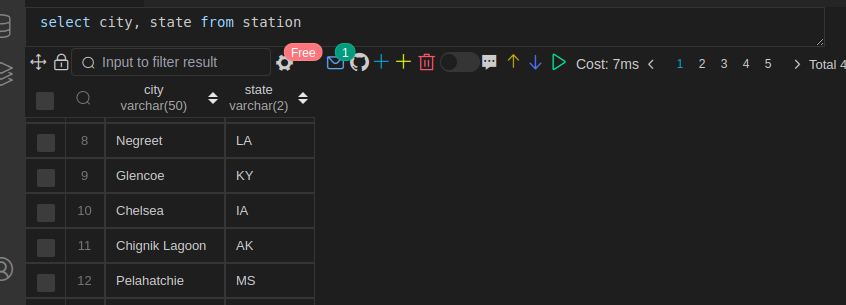
Output:



Q7. Query a list of CITY and STATE from the STATION table.

Query:

| select city, state from station |
| --- |

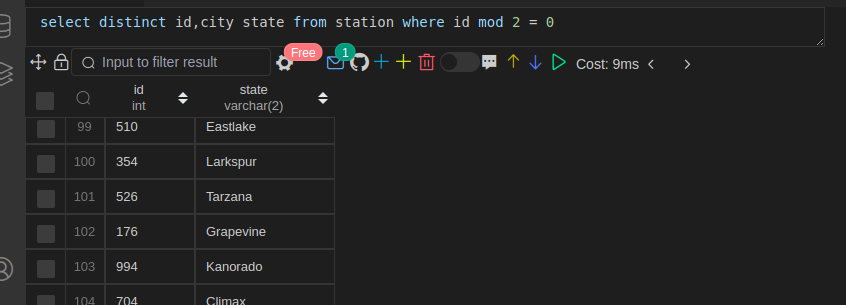


Q8. Query a list of CITY names from STATION for cities that have an even ID number. Print the results in any order, but exclude duplicates from the answer.

Query:

| select distinct id,city state from station where id mod 2 = 0 |
| --- |

Output:

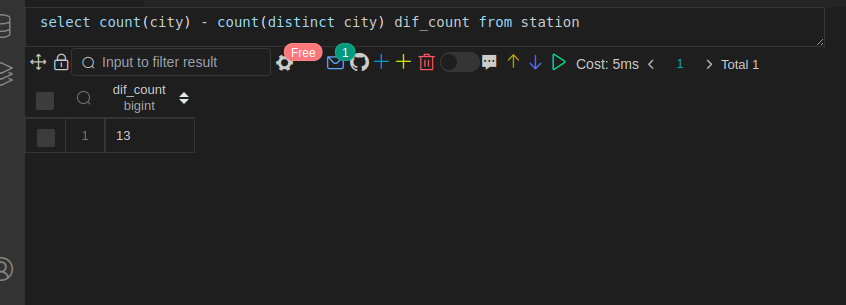


Q9. Find the difference between the total number of CITY entries in the table and the number of distinct CITY entries in the table.

Query:

| select count(city) - count(distinct city) dif\_count from station |
| --- |

Output

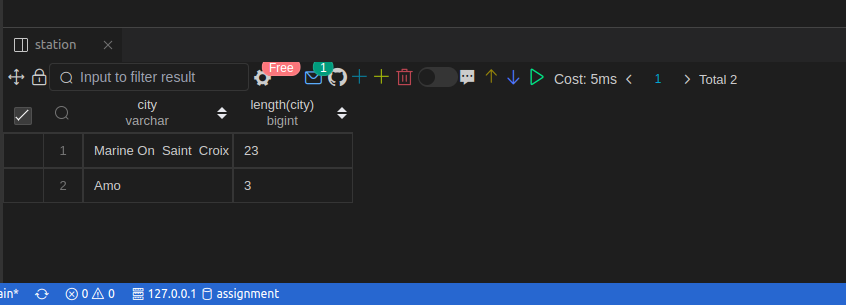


Q10. Query the two cities in STATION with the shortest and longest CITY names, as well as their respective lengths (i.e.: number of characters in the name). If there is more than one smallest or largest city, choose the one that comes first when ordered alphabetically.

Query:

| select city,length(city) from (  select city, ROW\_NUMBER() over(order by city) as rnk   from station   WHERE LENGTH(city) in (  select max(LENGTH(city)) from station  )  ) a where a.rnk = 1 union ALL select city,length(city) from (  select city, ROW\_NUMBER() over(order by city) as rnk   from station   WHERE LENGTH(city) in (  select min(LENGTH(city)) from station  )  ) a where a.rnk = 1 |
| --- |

Output:

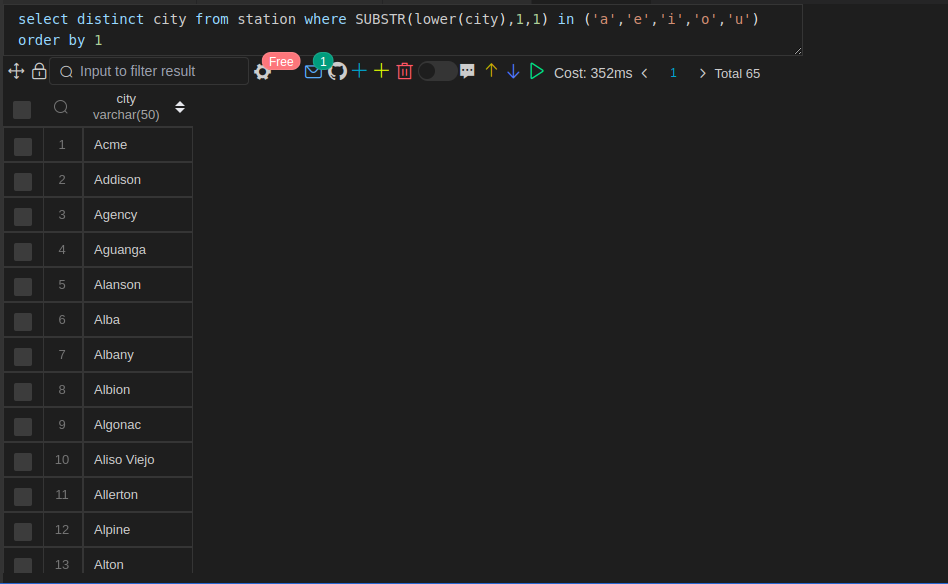


Q11. Query the list of CITY names starting with vowels (i.e., a, e, i, o, or u) from STATION. Your result cannot contain duplicates.

Query:

| select distinct city from station where SUBSTR(lower(city),1,1) in ('a','e','i','o','u') order by 1 |
| --- |

Output:

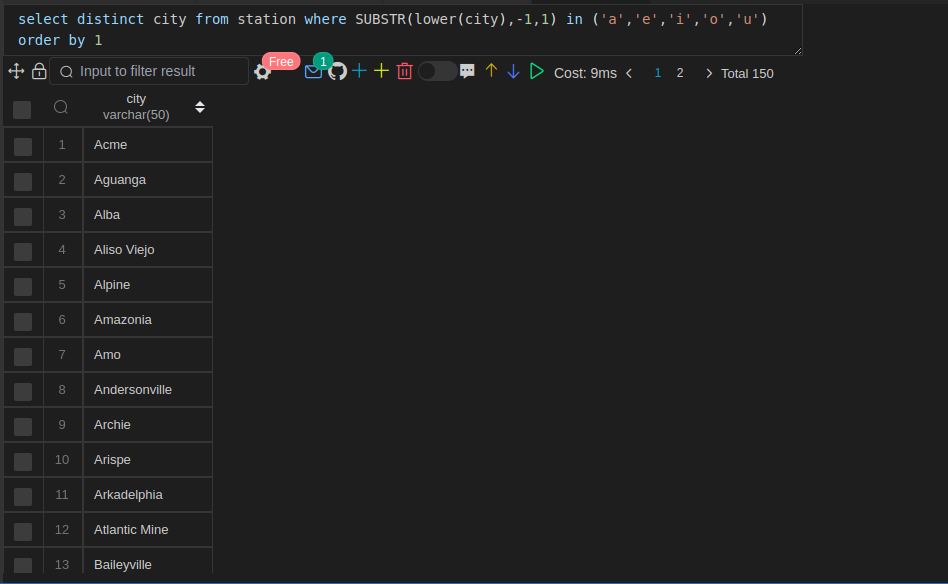


Q12. Query the list of CITY names ending with vowels (a, e, i, o, u) from STATION. Your result cannot contain duplicates.

Query:

| select distinct city from station where SUBSTR(lower(city),-1,1) in ('a','e','i','o','u') order by 1 |
| --- |

Output:

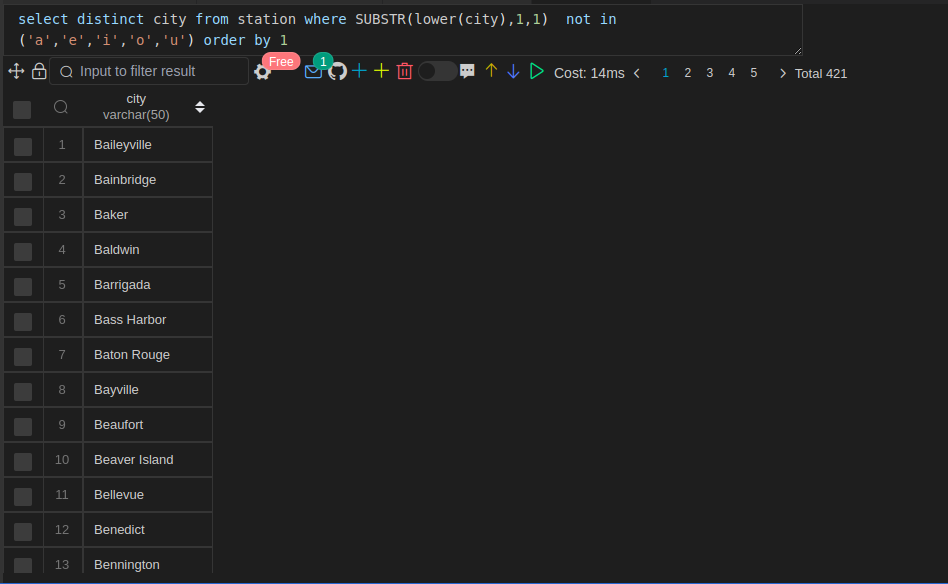


Q13. Query the list of CITY names from STATION that do not start with vowels. Your result cannot contain duplicates.

Query:

| select distinct city from station where SUBSTR(lower(city),1,1) not in ('a','e','i','o','u') order by 1 |
| --- |

Output:

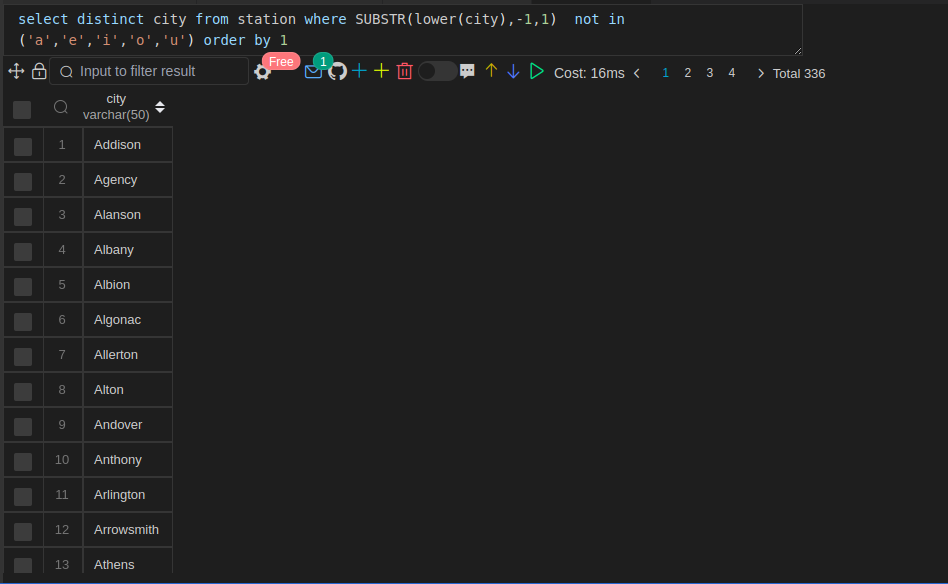


Q14. Query the list of CITY names from STATION that do not end with vowels. Your result cannot contain duplicates.

Query:

| select distinct city from station where SUBSTR(lower(city),-1,1) not in ('a','e','i','o','u') order by 1 |
| --- |

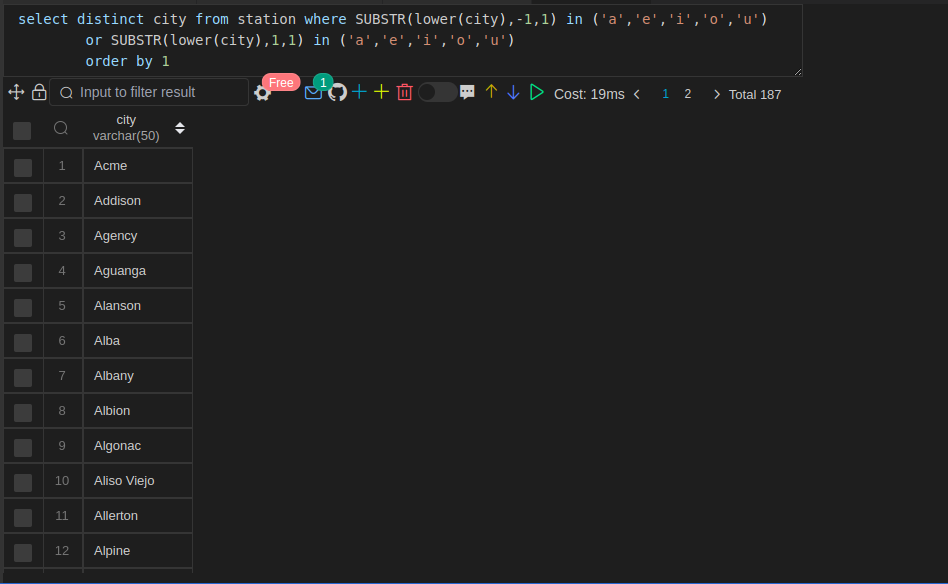
Output:



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 SUBSTR(lower(city),-1,1) in ('a','e','i','o','u')   or SUBSTR(lower(city),1,1) in ('a','e','i','o','u')   order by 1 |
| --- |

Output:

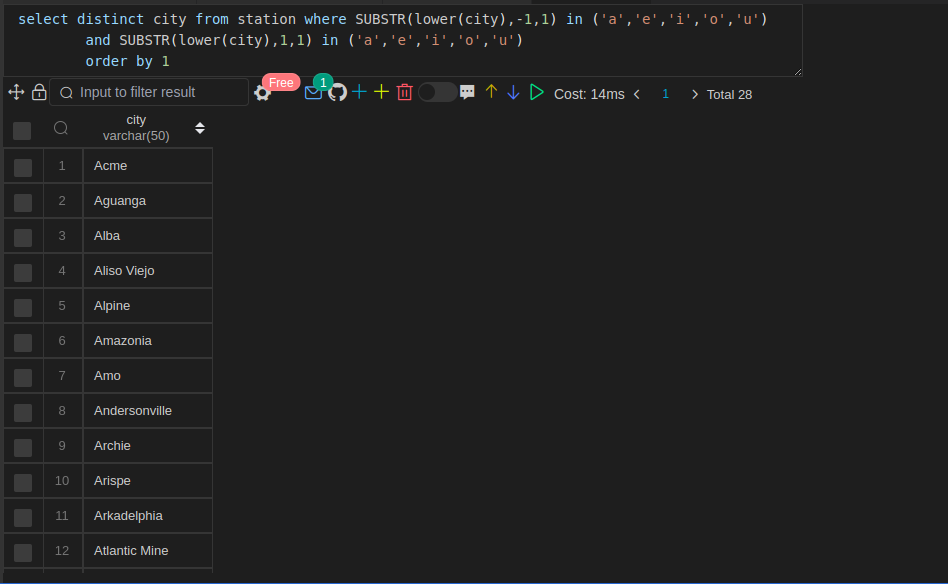


Q16. Query the list of CITY names from STATION that do not start with vowels and do not end with vowels. Your result cannot contain duplicates.

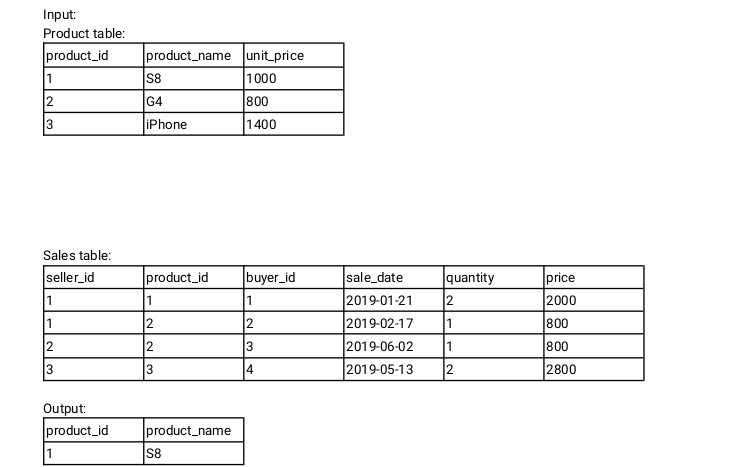
Query:

| select distinct city from station where SUBSTR(lower(city),-1,1) in ('a','e','i','o','u')   and SUBSTR(lower(city),1,1) in ('a','e','i','o','u')   order by 1 |
| --- |

Output:



Q17: 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.

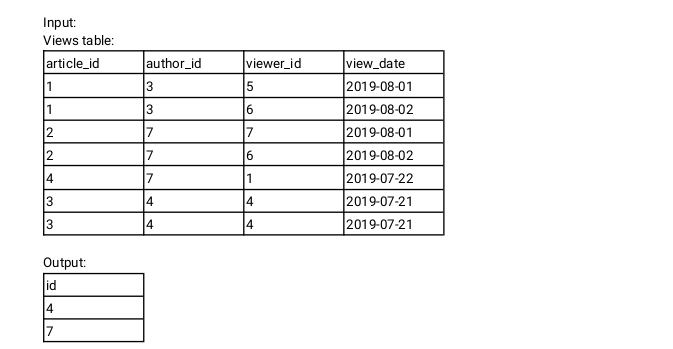


Query:

| Select s.product\_id, p.product\_name From sales s left join product p on s.product\_id = p.product\_id where s.sale\_date between 2019-01-01 and 2019-03-31 |
| --- |

Q18: 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.



Query:

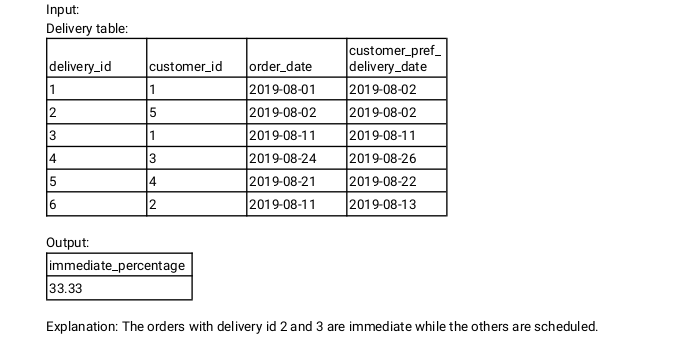
| Select viewer\_id as id from viewer where viewer\_id in (select distinct author\_id from viewer) order by 1 asce |
| --- |

Q19:

delivery\_id is the primary key of this table.

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).

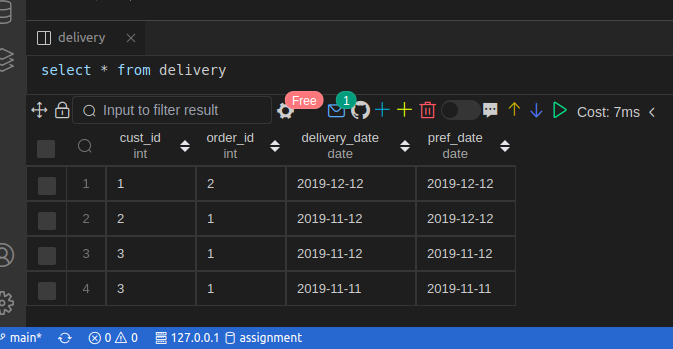
If the customer's preferred delivery date is the same as the order date, then the order is called immediately; otherwise, it is called scheduled. Write an SQL query to find the percentage of immediate orders in the table, rounded to 2 decimal places.

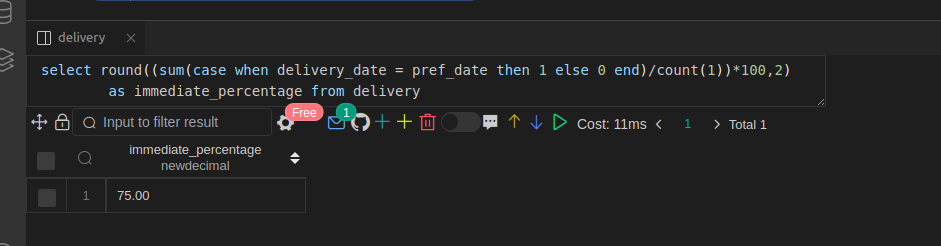


Query:

| select round((sum(case when delivery\_date = pref\_date then 1 else 0 end)/count(1))\*100,2) from delivery |
| --- |

Output:





Q20: A company is running Ads and wants to calculate the performance of each Ad.

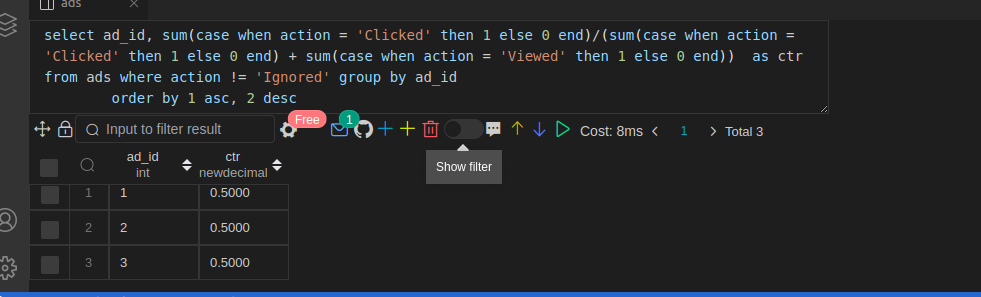
Performance of the Ad is measured using Click-Through Rate (CTR). Write an SQL query to find the ctr of each Ad. Round ctr to two decimal points.

Return the result table ordered by ctr in descending order and by ad\_id in ascending order in case of a tie.

Query:

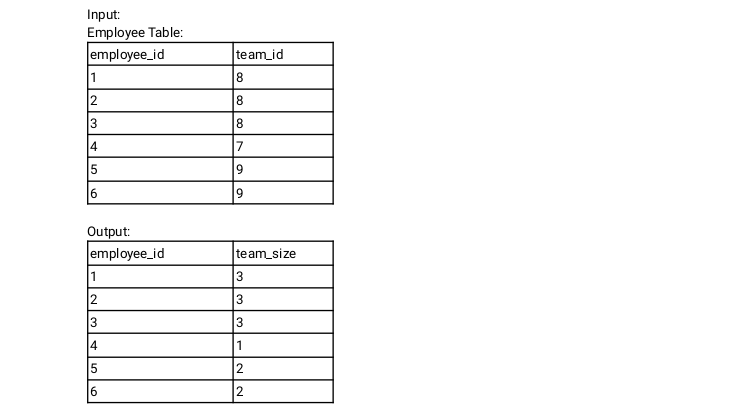
| select ad\_id, sum(case when action = 'Clicked' then 1 else 0 end)/(sum(case when action = 'Clicked' then 1 else 0 end) + sum(case when action = 'Viewed' then 1 else 0 end)) as ctr from ads where action != 'Ignored' group by ad\_id  order by 1 asc, 2 desc |
| --- |

Output:



Q21: Write an SQL query to find the team size of each of the employees.

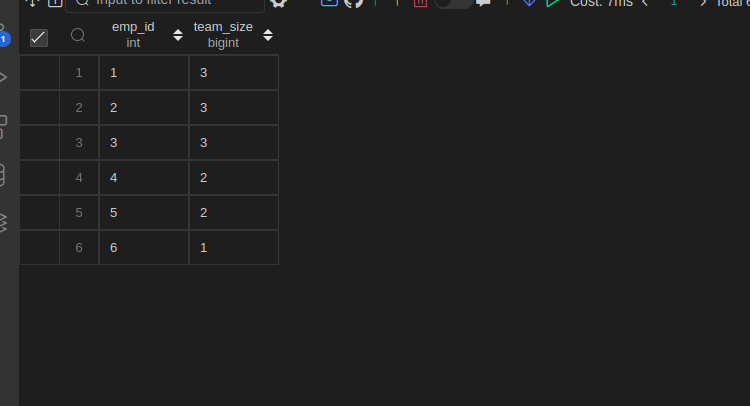
Return result table in any order.



Query:

| with cte as (select team\_id ,count(team\_id) as tm\_size from emp group by team\_id)  select a.emp\_id, b.tm\_size as team\_size from emp a left join cte b on a.team\_id = b.team\_id |
| --- |

Output:



Q22. Write an SQL query to find the type of weather in each country for November 2019.

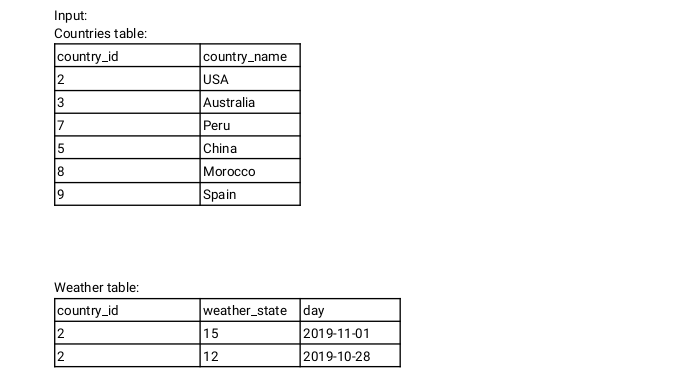
The type of weather is:

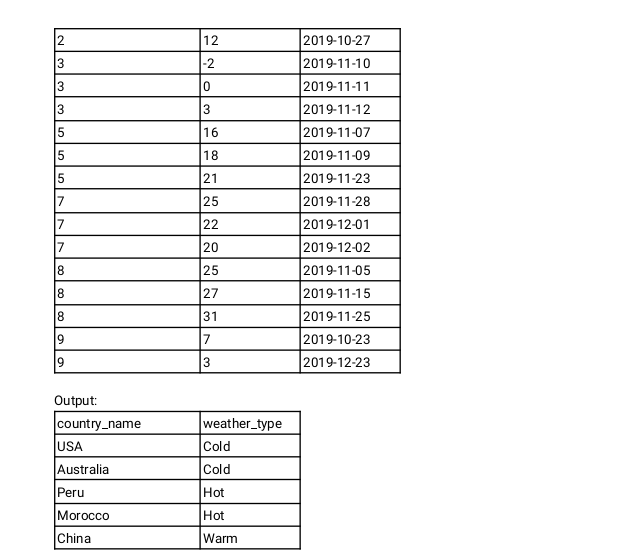
● Cold if the average weather\_state is less than or equal 15,

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

● Warm otherwise.

Return result table in any order.

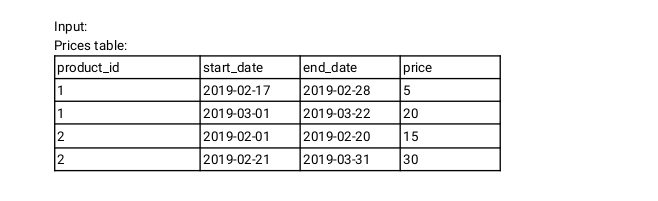


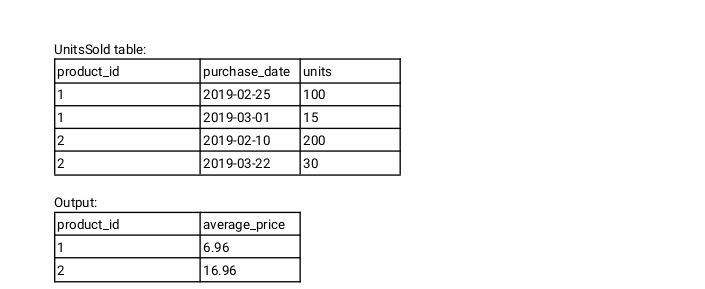


Query:

| with cte as (select country\_id, avg(weather\_state) avg\_weather from weather group by country\_id)  select b.country\_name, case when a.avg\_weather <=15 then 'Cold'   when a.avg\_weather >15 and a.avg\_weather >=25 then 'Hot'  else 'Warm' end as weather\_type   from cte a left join country b on a.country\_id = b.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.



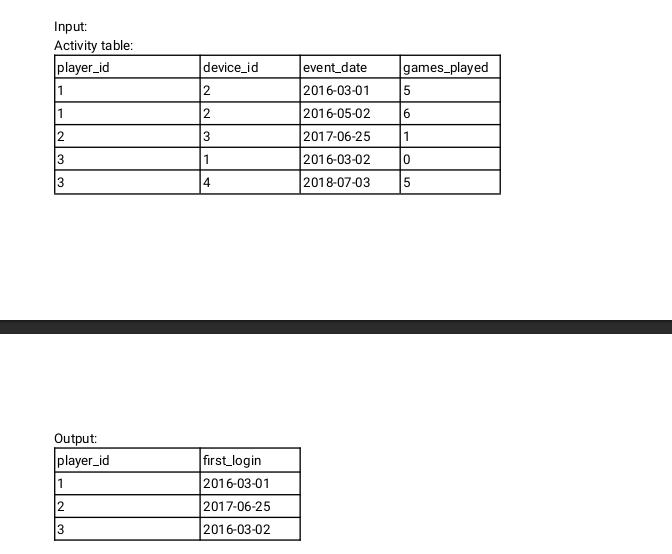


Query:

| Select u.product\_id, avg(p.price) as average\_price from unit\_sold u left join price p on u.product\_id = p.product\_id and u.purchase\_date between p.start\_date and p.end\_date group by u.product\_id |
| --- |

Q24. Write an SQL query to report the first login date for each player.

Return the result table in any order.

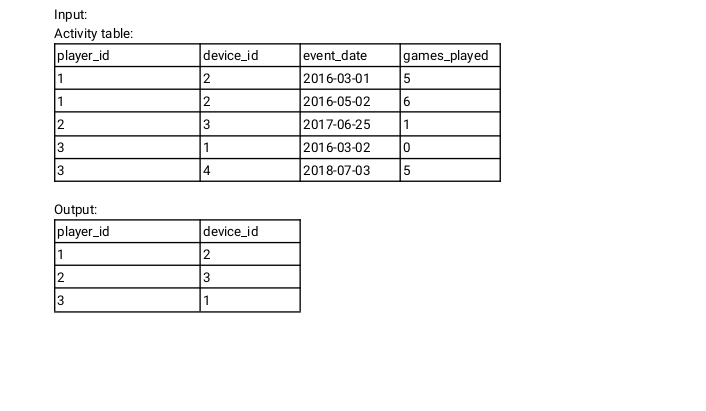


Query:

| Select player\_id, min(event\_date) as first\_login from activity group by player\_id |
| --- |

Q25. Write an SQL query to report the device that is first logged in for each player.

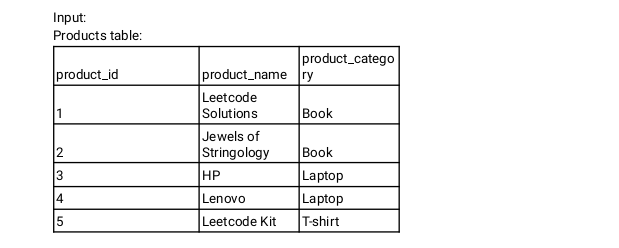
Return the result table in any order.

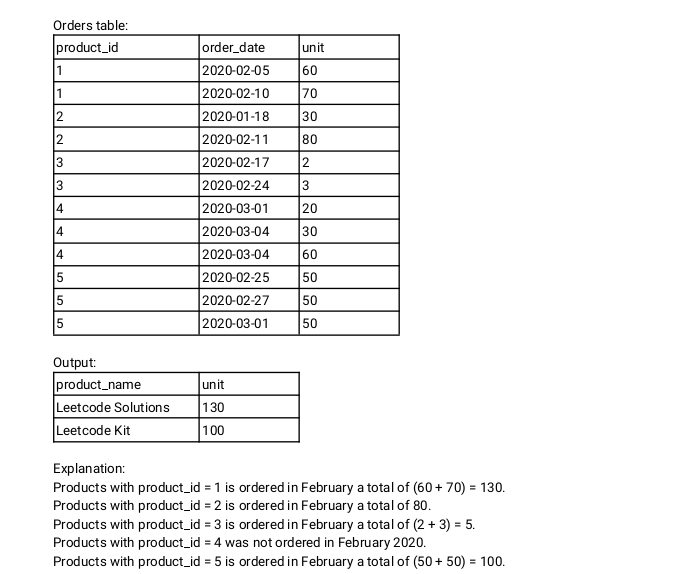


Query:

| Select player\_id, device\_id from activity b where exists (select player\_id, first\_login from (Select player\_id, min(event\_date) as first\_login from activity group by player\_id) a where a.player\_id = b.player\_id and a.first\_login = b.event\_date) |
| --- |

Q26. 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.





Query:

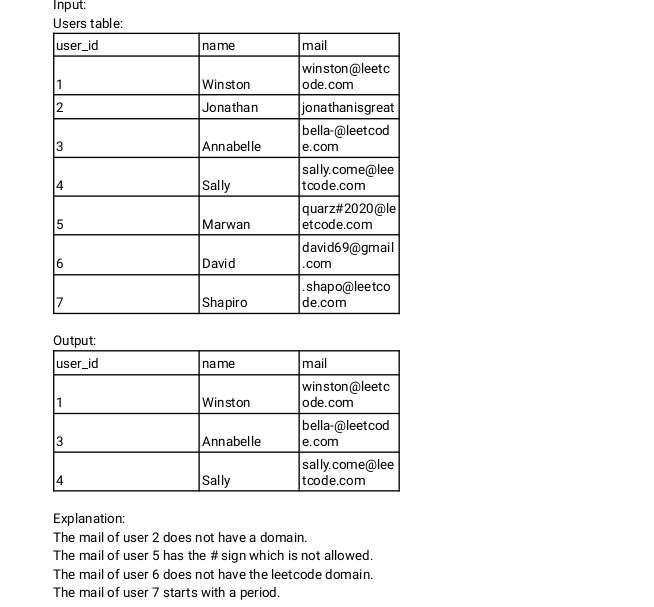
| With o as (Select product\_id, sum(unit) as units from orders order\_date between '2020-02-01' and '2020-02-28' group by product\_id)  Select p.product\_name, o.units from o left join product p on o.product\_id = p.product\_id where o.units>100 |
| --- |

Q27. 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.

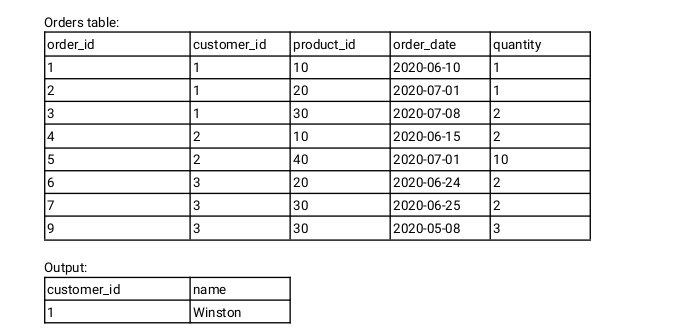
● The domain is '@leetcode.com'. Return the result table in any order.



| select \* from user where mail like '%@%leetcode.com' |
| --- |

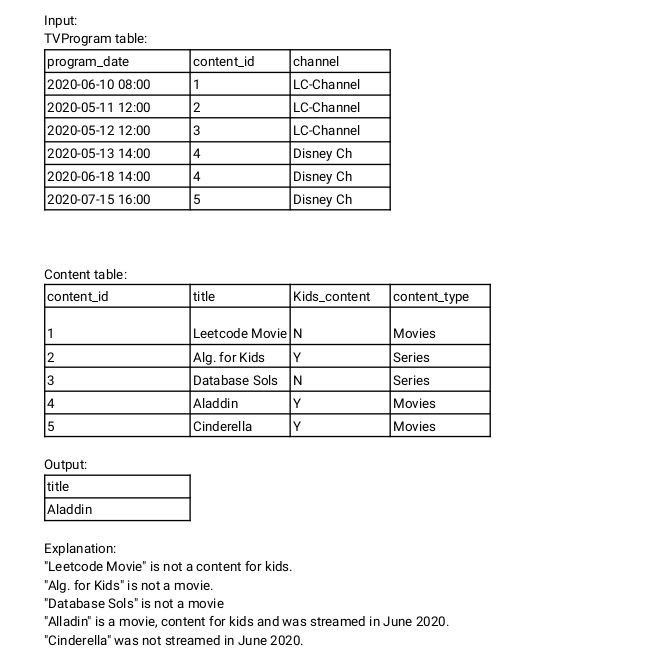
Q28. Write an SQL query to report the customer\_id and customer\_name of customers who have spent at least $100 in each month of June and July 2020. Return the result table in any order.





| With o as (Select customer\_id, product\_id, substr(replace(order\_date,'-',''),1,6) order\_month, sum(quantity) as sum\_quantity from order where order\_date between '2020-06-01' and '2020-07-31' group by customer\_id, product\_id, substr(replace(order\_date,'-',''),1,6)) Select  Select c.customer\_id, c.name From o left join product p on o.product\_id p.product\_id=o.product\_id left join customer c on c.customer\_id = o.customer\_id where o.sum\_quantity \* p.price > 100 |
| --- |

Q29. 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

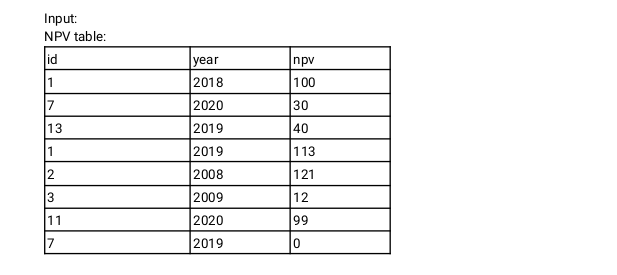


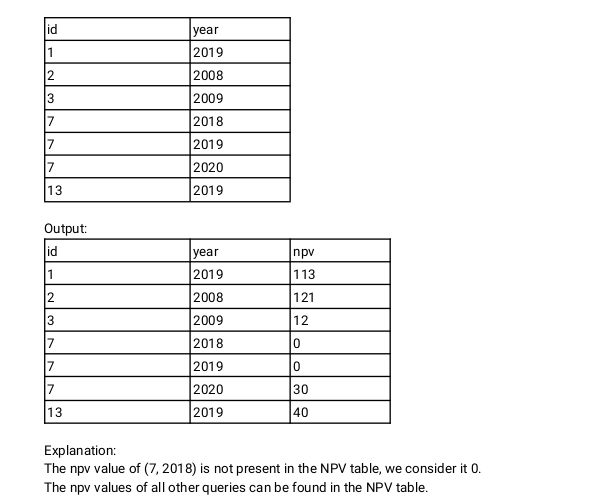
Query:

| Select c.title From tv\_program tv left join content c on tv.content\_id = c.content\_id where c.kids\_content = 'Y' and substr(replace(tv.program\_date,'-',''),1,6) = '202006' |
| --- |

Q30:Write an SQL query to find the npv of each query of the Queries table.

Return the result table in any order.



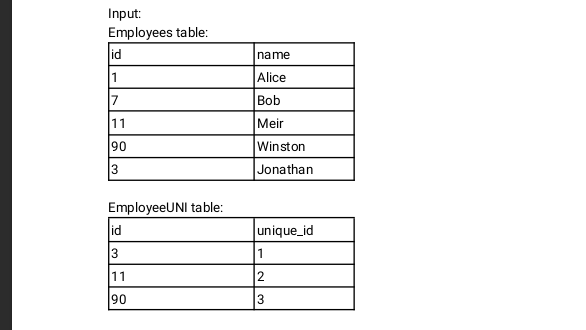


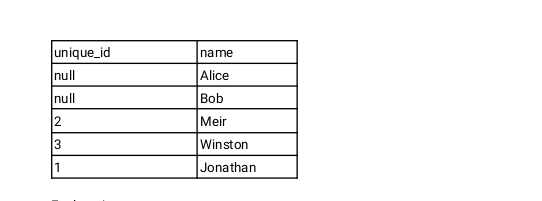
Query:

| Select q.id, q.year, coalesce(n.nvp, 0) From query q left join nvp n on q.id = n.id |
| --- |

Q31. its is same as Q30

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.





| Select emp\_u.unique\_id, emp.name from employee emp left join employee\_uni emp\_u on emp.id = emp\_u.id |
| --- |

Q33.

Write an SQL query to report the distance travelled by each user.

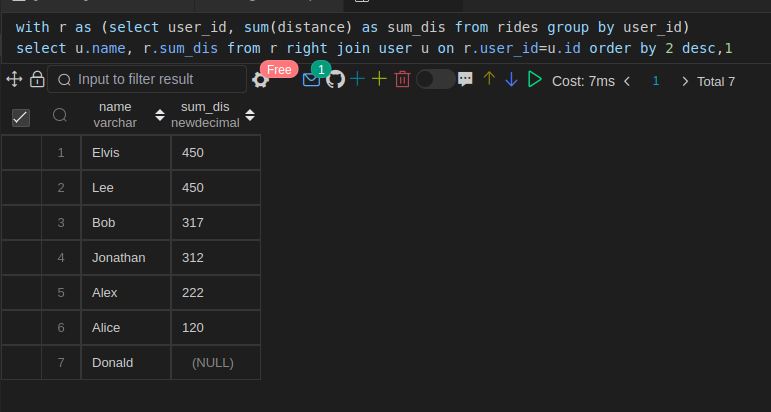
Return the result table ordered by travelled\_distance in descending order, if two or more users

travelled the same distance, order them by their name in ascending order.

Query:

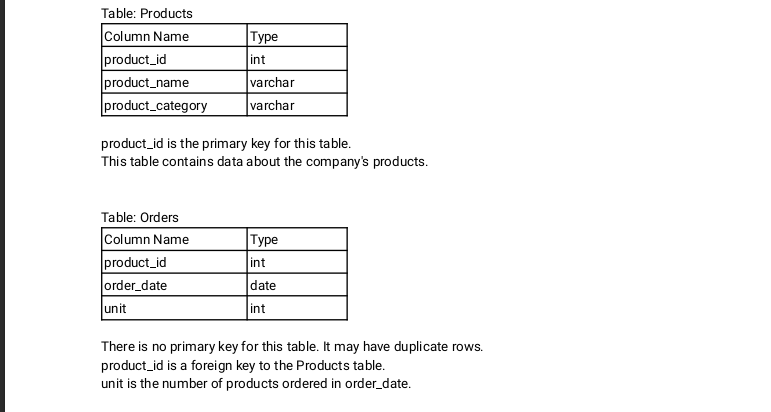
| with r as (select user\_id, sum(distance) as sum\_dis from rides group by user\_id) select u.name, r.sum\_dis from r right join user u on r.user\_id=u.id order by 2 desc,1 |
| --- |

Output:



Q34.

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



| With o as (Select producst\_id, sum(unit) unit\_sold From orders where order\_date between '2020-02-21' and '2020-02-29') select distinct p.product\_name from o left join product p on p.product\_id = o.product\_id where o.unit\_sold> 100 |
| --- |

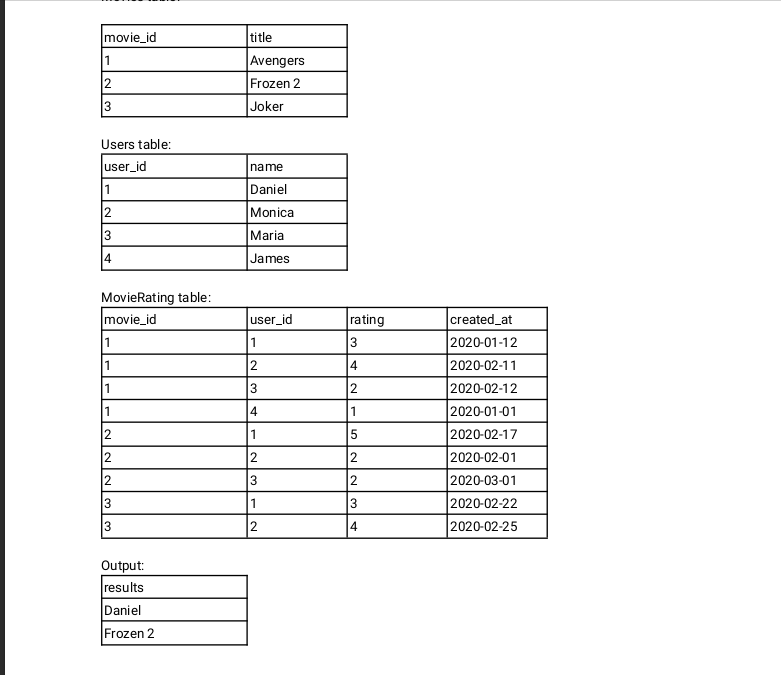
Q35. Write an SQL query to:

● 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.

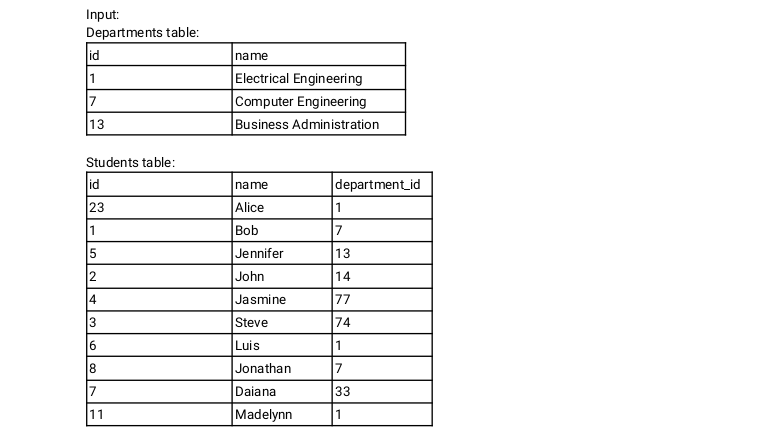


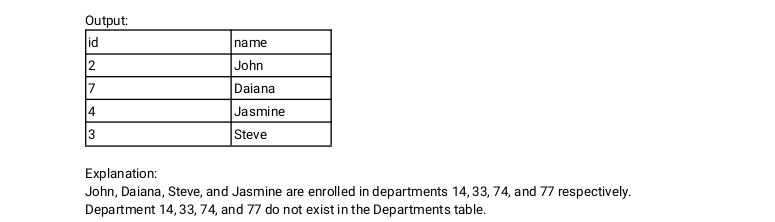
| Select name from (Select name,row\_number() over(order by name ) as rnk from (Select user\_id, count(rating) From movie\_rating group by user\_id) a left join user u on a.user\_id=u.user\_id) b where b.rnk=1 Union Select title (Select m.title,row\_number() over(order by title ) as rnk from (Select movie\_id, avg(rating) as avg\_rating From movie\_rating where created\_at between '2020-02-01' and between '2020-02-29' group by movie\_id) a left join movie m on a.movie\_id = m.movie\_id where a.avg\_rating = max(a.avg\_rating)) b where b.rnk=1 |
| --- |

Q36. same as Q30

Q37. same as Q32

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





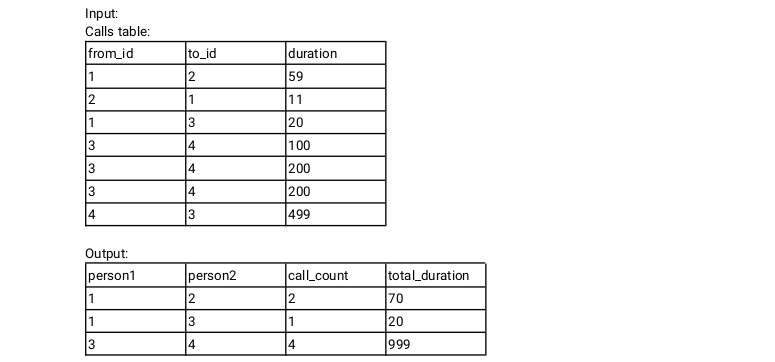
Query:

| Select id, name From student where department\_id not in (select distinct department\_id from department) |
| --- |

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.

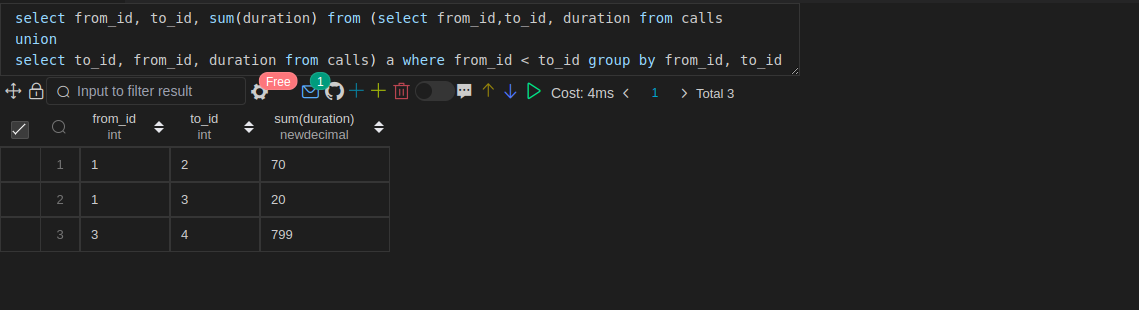
Return the result table in any order.



Query:

| select from\_id, to\_id, sum(duration) from (select from\_id,to\_id, duration from calls union  select to\_id, from\_id, duration from calls) a where from\_id < to\_id group by from\_id, to\_id |
| --- |

Output:



Q40. It is same as Q23

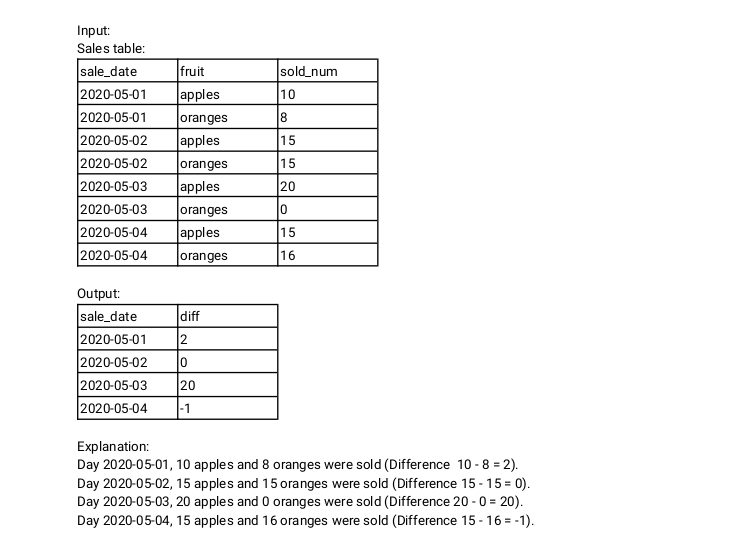
Q41. Write an SQL query to report the number of cubic feet of volume the inventory occupies in each warehouse.

| select w.name, sum(w.units \* p.width \* p.length \* p.height) from warehouse w left join products p on w.product\_id = p.product\_id group by 1 |
| --- |

Output:



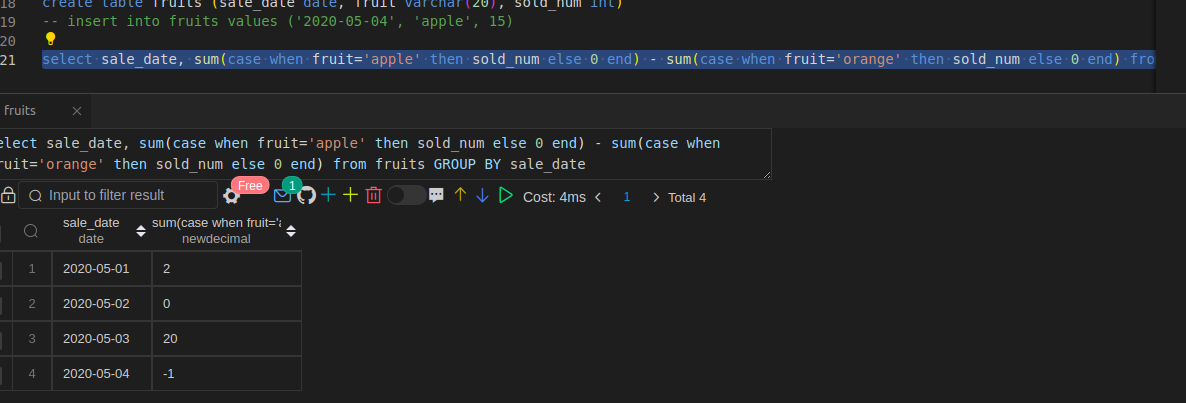
Q42. Write an SQL query to report the difference between the number of apples and oranges sold each day. Return the result table ordered by sale\_date.



Query:

| select sale\_date, sum(case when fruit='apple' then sold\_num else 0 end) - sum(case when fruit='orange' then sold\_num else 0 end) from fruits GROUP BY sale\_date |
| --- |

Output:



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

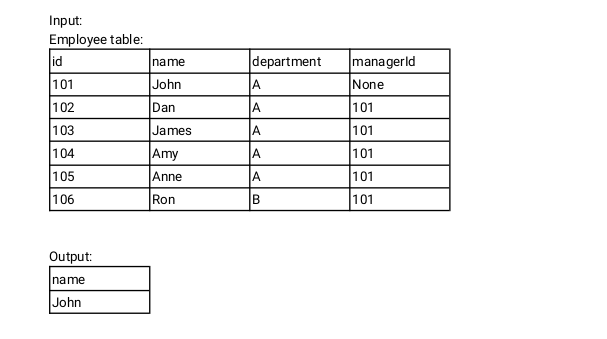
number by the total number of players.

Query:

| WITH CTE AS ( SELECT player\_id, min(event\_date) as event\_start\_date from Activity group by player\_id )  SELECT round((count(distinct c.player\_id) / (select count(distinct player\_id) from activity)),2)as fraction FROM CTE c JOIN Activity a on c.player\_id = a.player\_id and datediff(c.event\_start\_date, a.event\_date) = -1 |
| --- |

Q44. Write an SQL query to report the managers with at least five direct reports.

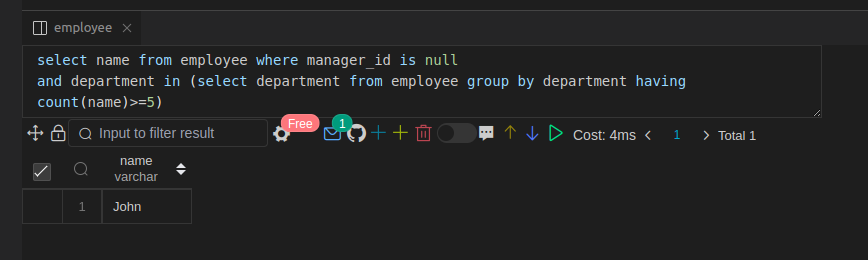
Return the result table in any order



Query:

| select name from employee where manager\_id is null and department in (select department from employee group by department having count(name)>=5) |
| --- |

Output:

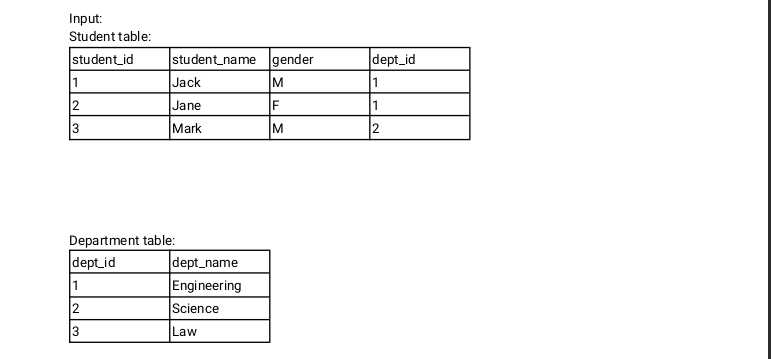


Q45. Write an SQL query to report the respective department name and number of students majoring in

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.

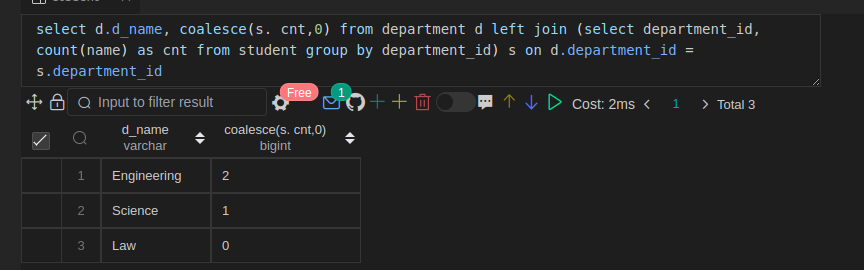




Query:

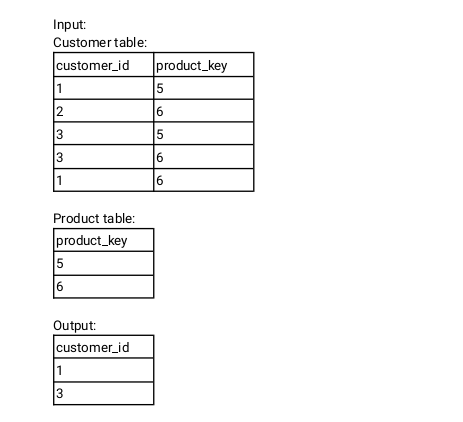
| select d.d\_name, coalesce(s. cnt,0) from department d left join (select department\_id, count(name) as cnt from student group by department\_id) s on d.department\_id = s.department\_id |
| --- |

Output:



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

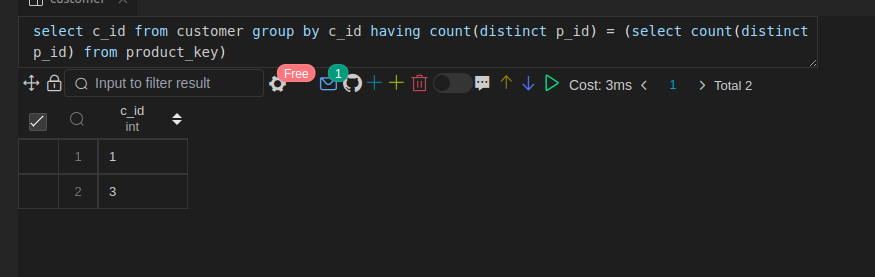
the Product table.



Query:

| select c\_id from customer group by c\_id having count(distinct p\_id) = (select count(distinct p\_id) from product\_key) |
| --- |

Output



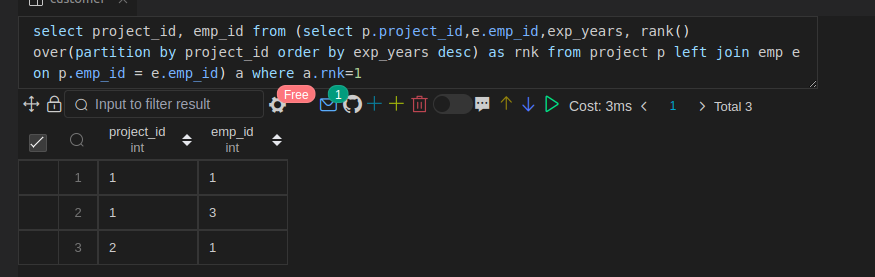
Q47. Write an SQL query that reports the most experienced employees in each project. In case of a tie,

report all employees with the maximum number of experience years.

Query:

| select project\_id, emp\_id from (select p.project\_id,e.emp\_id,exp\_years, rank() over(partition by project\_id order by exp\_years desc) as rnk from project p left join emp e on p.emp\_id = e.emp\_id) a where a.rnk=1 |
| --- |

Output:



Q48. 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

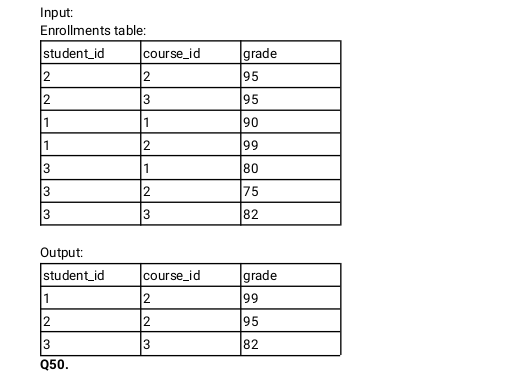
2019-06-23.

Query:

| Select b.book\_name, From orders o left join (Select book\_id From books where available\_from not between '2019-06-01' and '2019-06-30') b on o.book\_id = b.book\_id where o.dispatch\_date between '2018-06-01' and '2019-05-30'  group by o.book\_id having sum(quantiy)<10; |
| --- |

Q49. 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.



Query:

| Select student\_id, course\_id, grade from (Select student\_id, course\_id, grade, rank() over(partition by student\_id,course\_id order by grade desc) as rnk From enrollment ) a where a.rnk = 1 |
| --- |

Q50. Write an SQL query to find the winner in each group.

Query:

| select group\_id, player\_id from (  select p.group\_id, ps.player\_id, sum(ps.score) as score  from Players p,  (  select first\_player as player\_id, first\_score as score  from Matches  union all  select second\_player, second\_score  from Matches  ) ps  where 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; |
| --- |