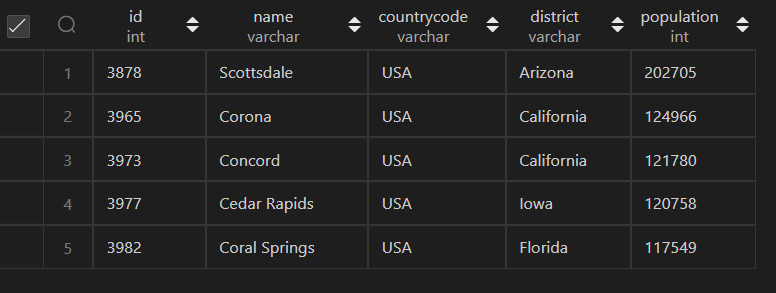
MySQL Challenge

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

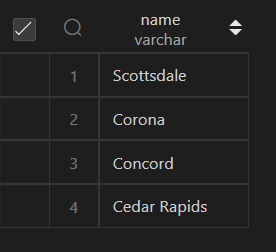
where countrycode = 'USA' and `population` > 100000;



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

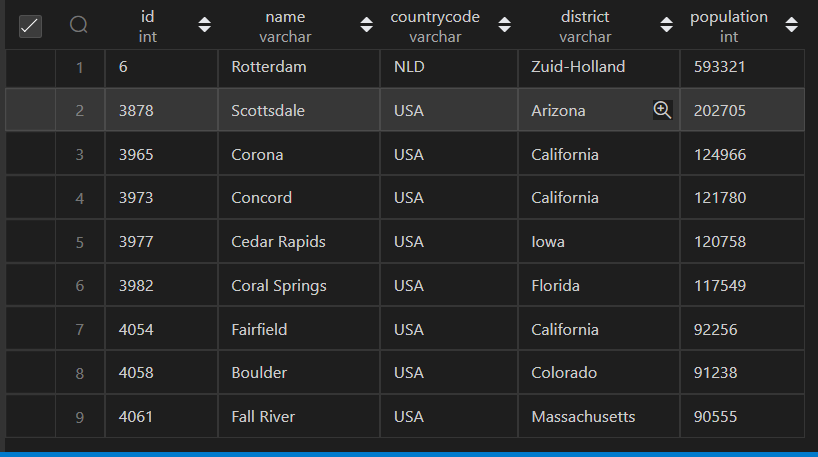
select `name` from city

where countrycode = 'USA' and `population` > 120000;



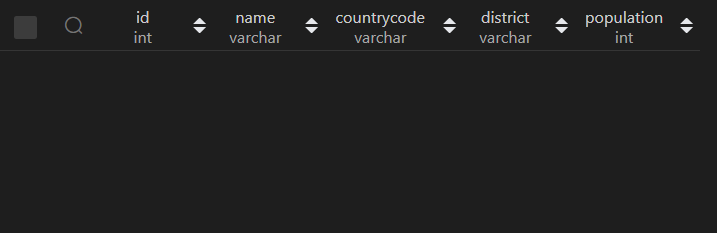
Q3. Query all columns (attributes) for every row in the CITY table.

select \* from city;



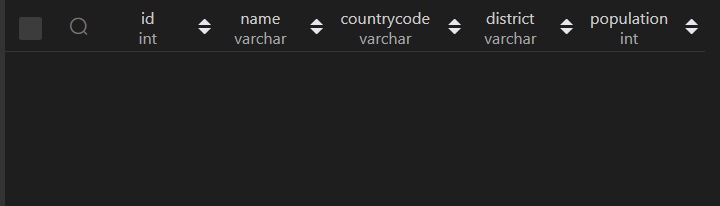
Q4. Query all columns for a city in CITY with the ID 1661

select \* from city where id = 1661;



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

select \* from city where countrycode = 'JPN';



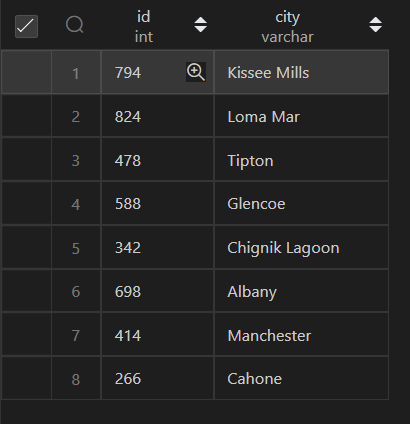
Q7. Query a list of CITY and STATE from the STATION table.

select DISTINCT 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 id,city from station where (id % 2) = 0;

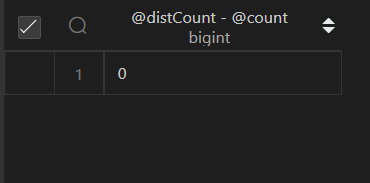


Q9. Find the difference between the total number of CITY entries in the table and the number of distinct CITY entries in the table.

select @distCount := count(DISTINCT city) from station;

select @count := count(city) from station;

select @distCount - @count;



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) as name\_length\_max

from station

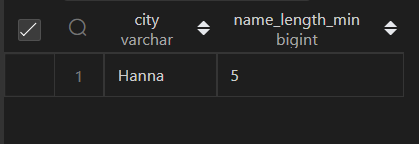
where LENGTH(city) = (select max(LENGTH(city)) from station) order by city;

select city, LENGTH(city) as name\_length\_min

from station

where LENGTH(city) = (select min(LENGTH(city)) from station) order by city;

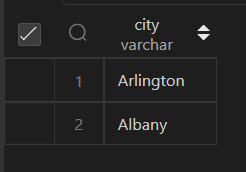




Q11. Query the list of CITY names starting with vowels (i.e., a, e, i, o, or u) from STATION. Your result cannot contain duplicates.

select distinct city from station

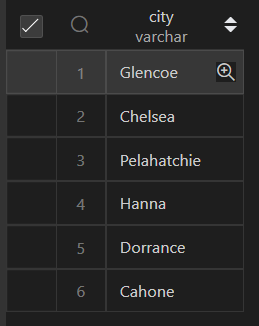
where substr(city,1,1) in ('a','e','i','o','u');



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

select distinct city from station

where substr(city,-1,1) in ('a','e','i','o','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 substr(city,1,1) not in ('a','e','i','o','u');



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

select distinct city from station

where substr(city,-1,1) not in ('a','e','i','o','u');



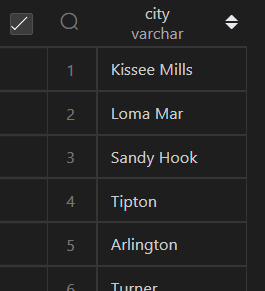
Q15. Query the list of CITY names from STATION that either do not start with vowels or do not end with vowels. Your result cannot contain duplicates.

select distinct city from station

where (substr(city,-1,1) not in ('a','e','i','o','u'))

or

(substr(city,1,1) not in ('a','e','i','o','u'));



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 (substr(city,-1,1) not in ('a','e','i','o','u'))

and

(substr(city,1,1) not in ('a','e','i','o','u'));



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. Return the result table in any order.

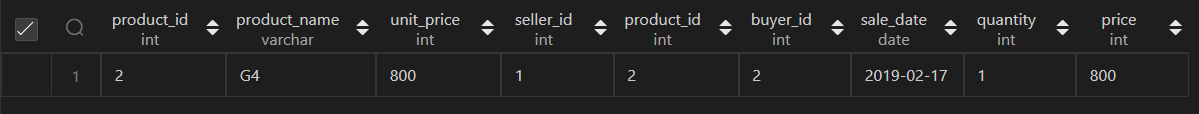
select p.\*, s.\* from product p

inner join sales s on p.product\_id = s.product\_id

where (s.sale\_date BETWEEN  "2019-01-01" and "2019-03-31")

and

(s.sale\_date < (select sale\_date from sales where product\_id = p.product\_id and sale\_date > "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.

select author\_id from views

where viewer\_id = author\_id

group by author\_id,article\_id

order by author\_id;



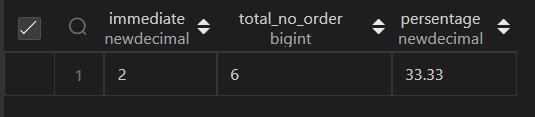
Q19. If the customer's preferred delivery date is the same as the order date, then the order is called immediately; otherwise, it is called scheduled. Write an SQL query to find the percentage of immediate orders in the table, rounded to 2 decimal places.

select sum(case when order\_date = customer\_pref\_delivery\_date then 1 else 0 end) as `immediate`,

count(delivery\_id) as total\_no\_order,

round(((sum(case when order\_date = customer\_pref\_delivery\_date then 1 else 0 end)/count(delivery\_id))\*100),2) as persentage

from delivery;



Q20. 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.

select

    cal.ad\_id,

    case

        when (cal.total\_clicks + cal.total\_views) = 0 then 0

        else round((total\_clicks/(cal.total\_clicks + cal.total\_views))\*100,2)

    end as ctr

from (select

            ad\_id,

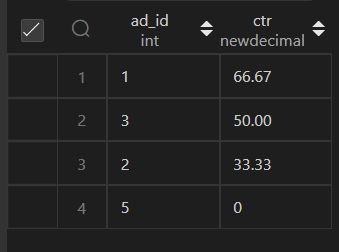
            sum(case when `action` = 'Clicked' then 1 else 0 end) as total\_clicks,

            sum(case when `action` = 'Viewed' then 1 else 0 end) as total\_views

        from ads GROUP BY ad\_id

     ) cal

     ORDER BY ctr desc;



Q21. Write an SQL query to find the team size of each of the employees. Return result table in any order.

select e.employee\_id, em.team\_size from employee e join

(select team\_id, count(\*) as team\_size from employee group by team\_id) em

on e.team\_id = em.team\_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.

select c.country\_name,final.\* from countries c join

(select

    cal.country\_id,

    case

        when cal.avg\_weather <= 15 then "Cold"

        when cal.avg\_weather >= 25 then "Hot"

        else "Warm"

    end as weather\_type

from (select

        country\_id,

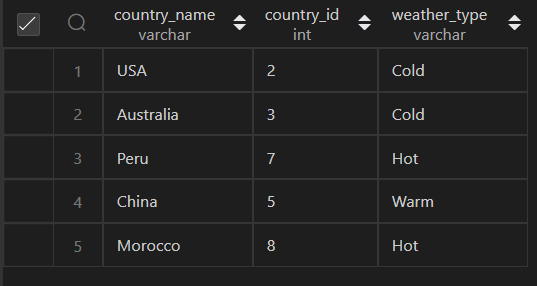
        avg(weather\_state) as avg\_weather from weather

        where day like "2019-11-%" GROUP BY country\_id

     ) cal

) final

on c.country\_id = final.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.

Average selling price = Total Price of Product / Number of products sold.

Average selling price for product 1 = ((100 \* 5) + (15 \* 20)) / 115 = 6.96

Average selling price for product 2 = ((200 \* 15) + (30 \* 30)) / 230 = 16.96

select

    filt.product\_id,

    round((sum(filt.total\_price)/sum(units)),2) as average\_price

    from (select

                p.product\_id,

                (p.price \* u.units) as total\_price,

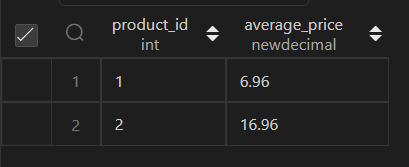
                u.units

                from prices p join unitsSold u

                on p.product\_id = u.product\_id and

                u.purchase\_date BETWEEN p.`start\_date` and p.end\_date) filt

group by filt.product\_id;



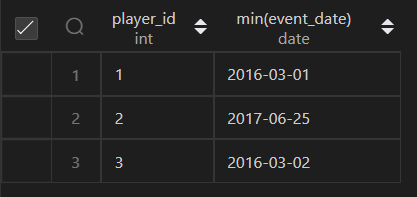
Q24. Write an SQL query to report the first login date for each player. Return the result table in any order.

select

    player\_id,

    min(event\_date) 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.

select

    tmp.player\_id,

    a.device\_id

    from (select

            player\_id,

            min(event\_date) as first\_log\_in

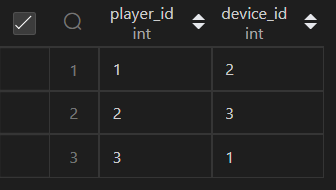
        from activity

        group by player\_id) tmp

    join activity a

    on tmp.player\_id = a.player\_id

    and a.event\_date = tmp.first\_log\_in;



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.

select

    p.product\_name,

    tmp.total\_unit\_sold

    FROM

    (select

        product\_id,

        sum(unit) as total\_unit\_sold

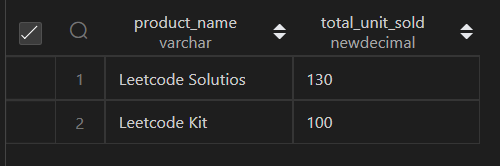
        from orders

    where order\_date like "2020-02-%"

    group by product\_id) tmp

join products p on p.product\_id = tmp.product\_id

where total\_unit\_sold >= 100;



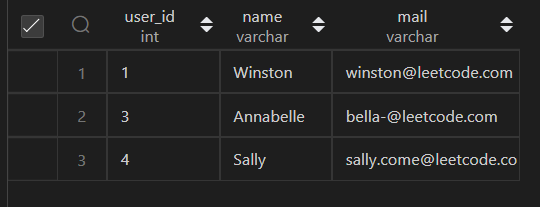
Q27. Write an SQL query to find the users who have valid emails. A valid e-mail has a prefix name and a domain where:

* 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 users

where mail REGEXP '^[a-zA-Z]+[a-zA-Z0-9\_\.\-][\*@leetcode\.com](mailto:*@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.

select

    customer\_id,

    `name`

from Customers

where customer\_id =

    (select s.customer\_id from

        (select

            final.customer\_id,

            final.month\_total as 5th\_mnth\_total,

            lead(final.month\_total,1,0)

            over(partition by final.customer\_id) as 6th\_mnth\_total

            from

            (select

            tmp.customer\_id,

            sum(tmp.price) as month\_total,

            month(tmp.order\_date) as mnth

        from (select

                ord.customer\_id,

                (ord.quantity \* p.price) as price,

                ord.order\_date

            from Product p

            join (select

                    customer\_id,

                    product\_id,

                    quantity,

                    order\_date

                from Orders

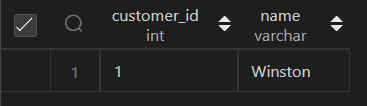
                where month(order\_date) in (6,7)) ord

            on p.product\_id = ord.product\_id) tmp

        group by month(tmp.order\_date),tmp.customer\_id

        order by tmp.customer\_id) final) s

    where s.5th\_mnth\_total >= 100 and s.6th\_mnth\_total >= 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.

select

    distinct c.title

from TVProgram t join

Content c

on c.content\_id = t.content\_id

where c.Kids\_content = 'Y'

and c.content\_type = "Movies"

and t.program\_date like "2020-06-%";



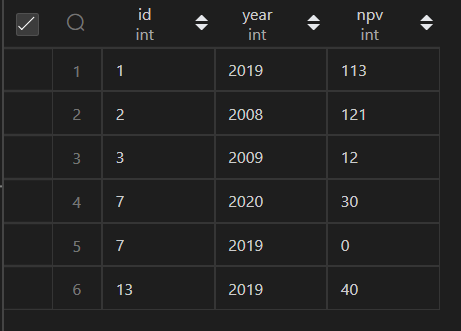
Q30. Write an SQL query to find the npv of each query of the Queries table. Return the result table in any order.

select n.id,n.year,n.npv

from NPV n join  Queries q

on n.id = q.id and n.year = q.year

order by n.id;



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. Return the result table in any order.

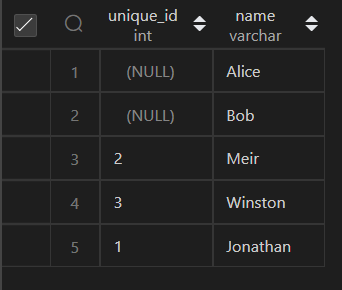
select

    eu.unique\_id,

    e.name from Employees e

    left join EmployeeUNI eu

on e.id = eu.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.

select

    u.name,

    COALESCE(s.travelled\_distance,0) as travelled\_distance

from Users u left join

    (select

        user\_id,

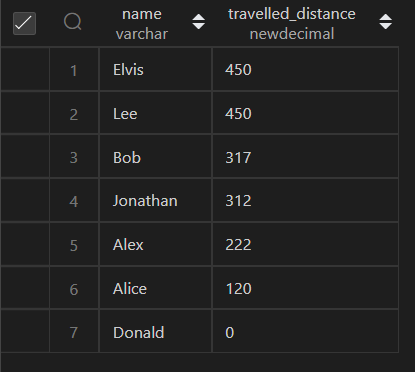
        sum(distance) as travelled\_distance

    from Rides

    group by user\_id) s

on u.id = s.user\_id

order by s.travelled\_distance desc, name;



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.

The query result format is in the following example.

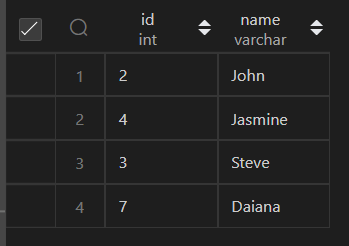
Q38. Write an SQL query to find the id and the name of all students who are enrolled in departments that no longer exist. Return the result table in any order.

select s.id, s.name from Departments d right join Students s

on

d.id = s.department\_id

where d.id 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. Return the result table in any order.

with test as (select

    from\_id,to\_id,duration,

    case

        when  from\_id < to\_id then concat(from\_id,'-',to\_id)

        else reverse(concat(from\_id,'-',to\_id))

    end as ar

from Calls)

select

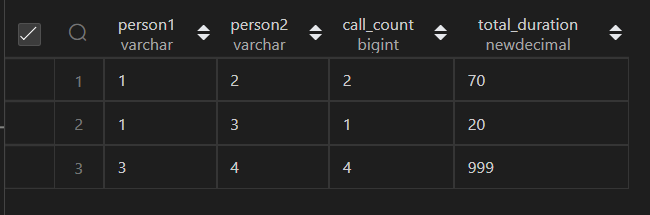
    substr(ar,1,1) as person1,

    substr(ar,-1,1) as person2,

    count(\*) as call\_count,

    sum(duration) as total\_duration

     from test group by ar;



Q41. 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.

select

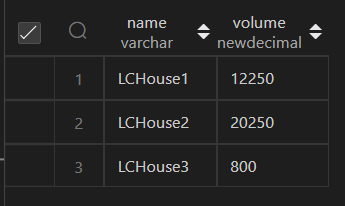
    w.name,

    sum((p.Width \* p.Length \* p.Height)\*w.units) as volume

from Warehouse w join Products p

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

GROUP BY w.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.

select

    s.sale\_date,

    (s.total\_salse\_ap - s.total\_salse\_or) as diff from

    (select

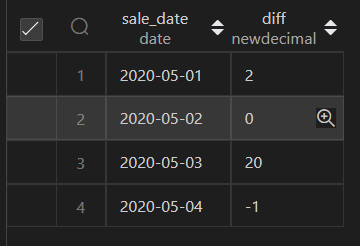
    sale\_date,

    sum(case when fruit = 'apples' then sold\_num else 0 end) as total\_salse\_ap,

    sum(case when fruit = 'oranges' then sold\_num else 0 end) as total\_salse\_or

from Sales

GROUP BY sale\_date) s order 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.

select

    (sum(case when s.day\_count > 1 then 1 else 0 end) / count(DISTINCT s.player\_id)) as diff

from

    (select \*,

        count(\*) over(partition by player\_id order by event\_date

                        range BETWEEN interval '1' day preceding

                        and current row) as day\_count

    from Activity) s;



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

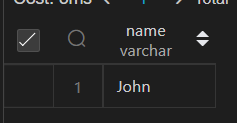
with test as (select \*,

    count(\*) over(partition by managerId) as reports\_count from Employee)

select e.name from Employee e join

(select managerId from test where reports\_count >= 5 group by managerId) s

on e.id = s.managerId;



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

with stu\_count as (select d.dept\_name, s.student\_id from Department d left join

Student s on d.dept\_id = s.dept\_id)

select

    dept\_name,

    COALESCE(count(student\_id),0) as student\_count

from stu\_count group by dept\_name order by student\_count desc, dept\_name;

