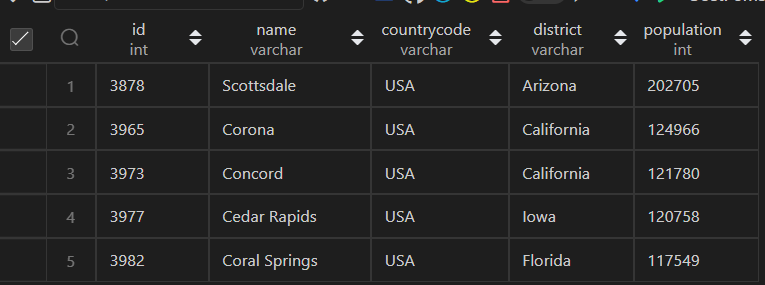
**Q1. Query all columns for all American cities in the CITY table with populations larger than 100000.**

**The CountryCode for America is USA.**

**The CITY table is described as follows:**

**Answer:**

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

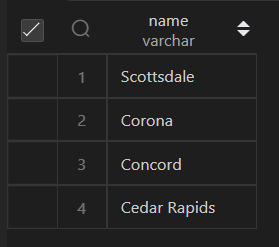


**Q2**. **Query the NAME field for all American cities in the CITY table with populations larger than 120000. The CountryCode for America is USA.**

**The CITY table is described as follows:**

**Answer:**

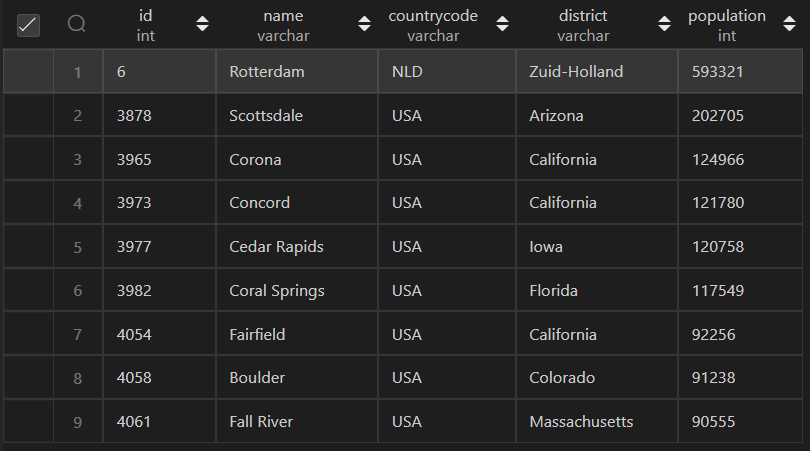
select name from city where CountryCode ='USA' and population>120000;



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

**Answer:**

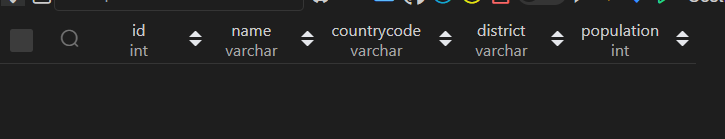
select \* from city;



**Q4**. Query all columns for a city in CITY with the ID 1661. The CITY table is described as follows:

**Answer:**

select \* from city where id=1661;

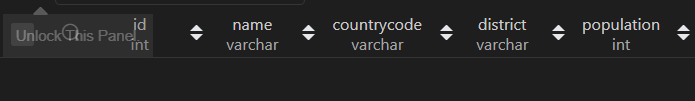


**Q5**. Query all attributes of every Japanese city in the CITY table. The COUNTRYCODE for Japan is JPN.

The CITY table is described as follows:

**Answer:**

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

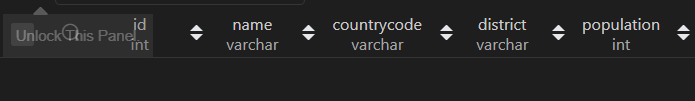


**Q6.** Query the names of all the Japanese cities in the CITY table. The COUNTRYCODE for Japan is JPN.

The CITY table is described as follows:

**Answer:**

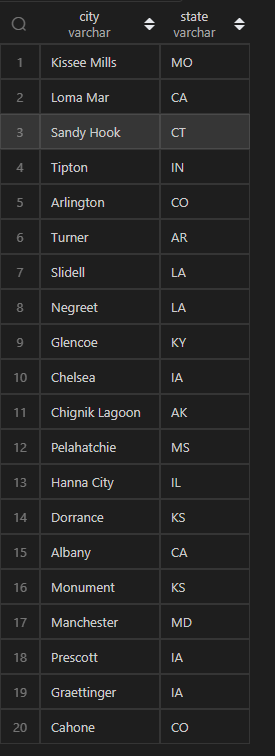
Select name  from city where countrycode='JPN';



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

**Anser:**

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. The STATION table is described as follows:

**Answer:**

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

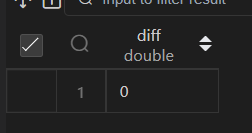


**Q9**. Find the difference between the total number of CITY entries in the table and the number of distinct CITY entries in the table.

The STATION table is described as follows:

**Answer:**

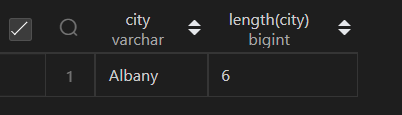
select distinct(city)-city as diff from station;



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

**Answer:**

select distinct city,length(city) from station order by length(city) asc,city ASC limit 1;



select  distinct city,length(city) from station order by length(city) DESC,city DESC 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.

Input Format

The STATION table is described as follows:

**Answer:**

select distinct(city) from station where REGEXP\_LIKE(city,'^[aeiouAEIOU]');



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

Input Format

The STATION table is described as follows:

**Answer:**

select distinct(city) from  station where REGEXP\_LIKE(CITY,'[aeiou]$','i');



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

Input Format

The STATION table is described as follows:

select city from station where REGEXP\_LIKE(city,'^[^[aeiouAEIOU]]');



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

**Answer:**

select distinct(city) from  station where NOT REGEXP\_LIKE(CITY,'[aeiouAEIOU]$');



**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.

Input Format

The STATION table is described as follows:

Answer:

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%')

or (CITY NOT LIKE '%a' AND

    CITY  NOT LIKE '%e' AND

    CITY NOT LIKE '%i' AND

    CITY NOT LIKE '%o' AND

    CITY NOT LIKE '%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.

**Answer:**

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%')

and (CITY NOT LIKE '%a' AND

    CITY  NOT LIKE '%e' AND

    CITY NOT LIKE '%i' AND

    CITY NOT LIKE '%o' AND

    CITY NOT LIKE '%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.

The query result format is in the following example.

**Answer:**

SELECT product\_id,

       product\_name

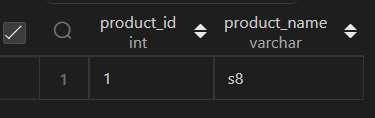
FROM   product

WHERE  product\_id NOT IN (SELECT product\_id

                          FROM   sales

                          WHERE  sale\_date NOT 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. The query result format is in the following example.

**Answer:**

select distinct(t1.author\_id)

from views t1 inner join views t2

on t1.author\_id=t2.viewer\_id order by t1.author\_id asc;



**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.

The query result format is in the following example.

**Answer:**

select round(100\* t2.immediate\_c/count(t1.delivery\_id),2) immediate\_percentage

from delivery t1,(

select count(\*) as immediate\_c from delivery where order\_date = customer\_pref\_delivery\_date order by order\_date) t2;

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

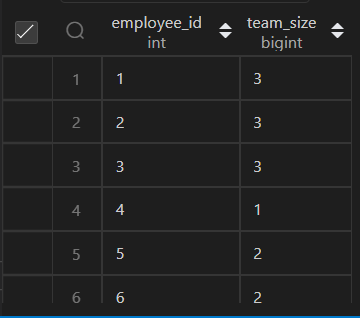
Return result table in any order.

The query result format is in the following example.

**Answer:**

select employee\_id,

count(team\_id) over(partition by team\_id) as team\_size from employee order by employee\_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.

**Answer:**

SELECT c.country\_name,

       CASE

           WHEN AVG(w.weather\_state ) <= 15.0 THEN 'Cold'

           WHEN AVG(w.weather\_state ) >= 25.0 THEN 'Hot'

           ELSE 'Warm'

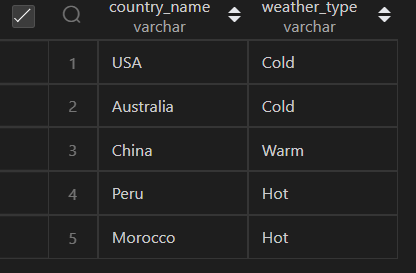
       END AS weather\_type

FROM countries AS c

INNER JOIN weather AS w ON c.country\_id = w.country\_id

WHERE w.day like '\_\_\_\_\_11\_\_\_'

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.

**Answer:**

select p.product\_id,

    round(sum(p.price \* u.units)/sum(u.units), 2) as average\_price

from prices p

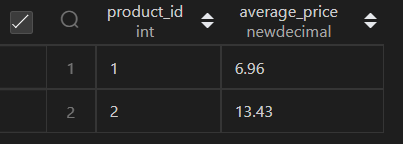
inner join unitssold u

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

    datediff(u.purchase\_date, p.start\_date) >= 0 and

    datediff(p.end\_date, u.purchase\_date) >= 0

group by p.product\_id



**Q24.**

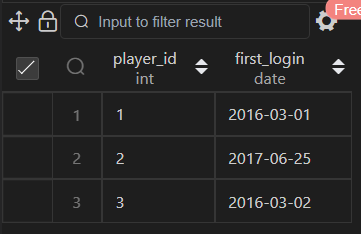
Write an SQL query to report the first login date for each player.

Return the result table in any order.

The query result format is in the following example.

**Answer:**

select player\_id,min(event\_date) as first\_login from activity group by player\_id;



**Q25.**

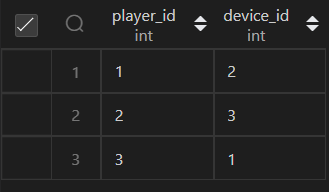
Write an SQL query to report the device that is first logged in for each player.

Return the result table in any order.

The query result format is in the following example.

**Answer:**

select player\_id,device\_id from activity where event\_date in (select min(event\_date) from activity group by player\_id);



**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.

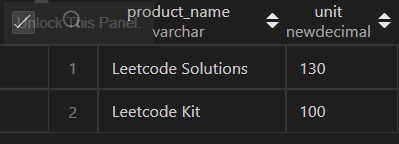
The query result format is in the following example.

**Answer:**

select p.product\_name,sum(o.unit) as unit

from products p inner join orders o on p.product\_id=o.product\_id where o.order\_date like '%\_\_\_\_\_02\_\_\_%'

group by p.product\_id having sum(o.unit)>=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:

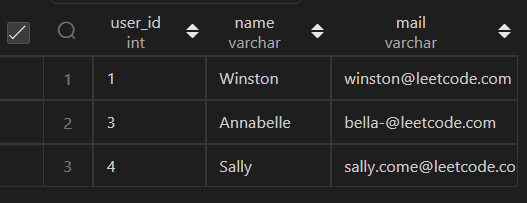
* The prefix name is a string that may contain letters (upper or lower case), digits, underscore '\_', period '.', and/or dash '-'. The prefix name must start with a letter.
* The domain is '@leetcode.com'.

**Answer:**

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.

**Answer:**

select o.customer\_id, c.name

from customers c, product p, orders o

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

group by o.customer\_id

having

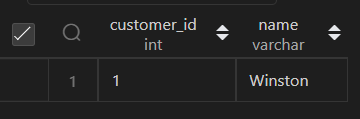
(

    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**.

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.

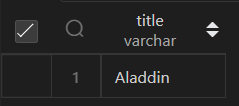
**Answer:**

select c.title from

tvprogram t inner join content c

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

 where c.kids\_content='Y' and t.program\_date between '2020-06-01' and '2020-06-30';



**Q30.**

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.

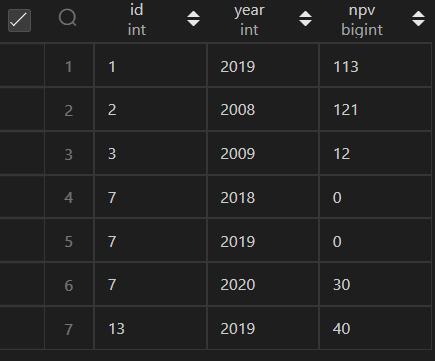
**Answer:**

select q.id, q.year, ifnull(n.npv,0) as npv

from queries as q

left join npv as n

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



**Q31.**

**Same question as 30th**

**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.

**Answer:**

select ifnull(e2.unique\_id,'null') as unique\_id,e1.name

from employees e1 left join employeeuni e2

on e1.id=e2.id order by e1.name;



**Q33.**

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

Return the result table ordered by travelled\_distance in descending order, if two or more users travelled the same distance, order them by their name in ascending order.

**Answer:**

select name, sum(ifnull(distance, 0)) as travelled\_distance

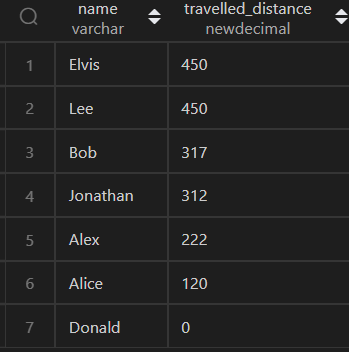
from rides r

right join users u

on r.user\_id = u.id

group by name

order by 2 desc,1 asc;



**Q34.**

**This question is incomplete**

**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.

**Answer:**

SELECT user\_name AS results FROM

(

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

    JOIN users AS a

    on a.user\_id = b.user\_id

    GROUP BY b.user\_id

    ORDER BY counts DESC, user\_name ASC LIMIT 1

) first\_query

UNION

SELECT movie\_name AS results FROM

(

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

    JOIN movies AS c

    on c.movie\_id = d.movie\_id

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

    GROUP BY d.movie\_id

    ORDER BY rate DESC, movie\_name ASC LIMIT 1

) second\_query;



**Q36.**

**Same as Q.33**

**Q.37**

**Same as Q.32**

**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.

**Answer:**

SELECT id, name

FROM students

WHERE department\_id not in (SELECT id from departments);

**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.

**Answer:**

    with caller as (select from\_id as person1, to\_id as person2, duration

    from calls

    UNION ALL

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

    from calls),

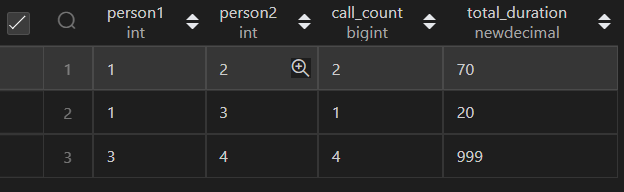
    unique\_caller as (

    select person1, person2, duration

    from caller

    where person1 < person2)

select person1,person2,count(1) as call\_count,sum(duration) as total\_duration from unique\_caller group by person1,person2;



**Q.40 is same as Q.23**

**Q41.** 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.

**Answer:**

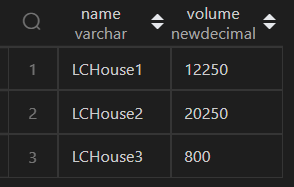
select w.name,

sum(p.width\*p.length\*p.height\*w.units) as volume

from warehouse w inner 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.

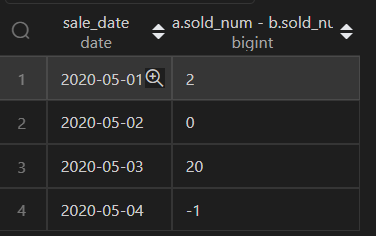
**Answer:**

select a.sale\_date, a.sold\_num - b.sold\_num

from sales a inner join sales b

on a.sale\_date = b.sale\_date

where a.fruit = 'apples' and b.fruit = 'orranges';



**Q43.**

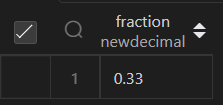
Write an SQL query to report the fraction of players that logged in again on the day after the day they first logged in, rounded to 2 decimal places. In other words, you need to count the number of players that logged in for at least two consecutive days starting from their first login date, then divide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

**Answer;**

select round((select a.player\_id

from activity a inner join activity b

on a.player\_id=b.player\_id where datediff(a.event\_date,b.event\_date)=1 group by a.player\_id)/3,2) as fraction;



**Q44.**

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

Return the result table in any order.

**Answer:**

select a.name from Employee a inner join Employee b

on (a.id = b.managerid) group by a.name having count(a.Id)>=5;



**Q45.**

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

**Answer:**

select d.dept\_name,count(s.dept\_id) as student\_number

from department d left join student s

on d.dept\_id=s.dept\_id group by d.dept\_id order by student\_number desc,d.dept\_name asc;

