Leetcode SQL

175. Combine Two Tables

Easy 587 78 Favorite Share

SQL Schema ›

Table: `Person`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| PersonId    | int     |
| FirstName   | varchar |
| LastName    | varchar |
+-------------+---------+
PersonId is the primary key column for this
table.
```

Table: `Address`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| AddressId   | int     |
| PersonId    | int     |
| City        | varchar |
| State       | varchar |
+-------------+---------+
AddressId is the primary key column for this
table.
```

Write a SQL query for a report that provides the following information for each person in the Person table, regardless if there is an address for each of those people:

```
FirstName, LastName, City, State
```

Accepted 130,003 | Submissions 270,036

```
SELECT Person.FirstName, Person.LastName, Address.City, Address.State from Person LEFT JOIN Address on Person.PersonId = Address.PersonId;
```

176. Second Highest Salary

Easy 420 200 Favorite Share

SQL Schema ›

Write a SQL query to get the second highest salary from the `Employee` table.

```
+----+--------+
| Id | Salary |
+----+--------+
| 1  | 100    |
| 2  | 200    |
| 3  | 300    |
+----+--------+
```

For example, given the above Employee table, the query should return `200` as the second highest salary. If there is no second highest salary, then the query should return `null`.

```
+---------------------+
| SecondHighestSalary |
+---------------------+
| 200                 |
+---------------------+
```

Accepted 105,517 | Submissions 411,561

```
SELECT max(Salary)
FROM Employee
WHERE Salary < (SELECT max(Salary) FROM Employee)
```

177. Nth Highest Salary

Medium 192 147 Favorite Share

Write a SQL query to get the n^{th} highest salary from the `Employee` table.

```
+----+--------+
| Id | Salary |
+----+--------+
| 1  | 100    |
| 2  | 200    |
| 3  | 300    |
+----+--------+
```

For example, given the above Employee table, the n^{th} highest salary where $n = 2$ is `200`. If there is no n^{th} highest salary, then the query should return `null`.

```
+------------------------+
| getNthHighestSalary(2) |
+------------------------+
| 200                    |
+------------------------+
```

Accepted 52,345 | Submissions 216,382

```mysql
CREATE FUNCTION getNthHighestSalary(N INT) RETURNS INT
BEGIN
  RETURN (
      # Write your MySQL query statement below.
      select distinct e1.salary
      from Employee e1
      where N-1 = (select count(distinct e2.Salary)
                   from Employee e2
                   where e1.Salary < e2.Salary)
  );
END
```

```mysql
CREATE FUNCTION getNthHighestSalary(N INT) RETURNS INT
BEGIN
DECLARE M INT;
SET M=N-1;
  RETURN (
      # Write your MySQL query statement below.
      SELECT DISTINCT Salary FROM Employee ORDER BY Salary DESC LIMIT M, 1
  );
END
```

603. Consecutive Available Seats

SQL Schema ›

Several friends at a cinema ticket office would like to reserve consecutive available seats.
Can you help to query all the consecutive available seats order by the seat_id using the following `cinema` table?

```
| seat_id | free |
|---------|------|
| 1       | 1    |
| 2       | 0    |
| 3       | 1    |
| 4       | 1    |
| 5       | 1    |
```

Your query should return the following result for the sample case above.

```
| seat_id |
|---------|
| 3       |
| 4       |
| 5       |
```

Note:

- The seat_id is an auto increment int, and free is bool ('1' means free, and '0' means occupied.).
- Consecutive available seats are more than 2(inclusive) seats consecutively available.

```sql
select distinct a.seat_id
from cinema a
join cinema b
on abs(a.seat_id - b.seat_id) = 1
and a.free=true and b.free=true
order by a.seat_id;
```

```sql
select C1.seat_id from cinema C1  where
C1.free=1
and
(
    C1.seat_id+1 in (select seat_id from cinema where free=1)
    or
    C1.seat_id-1 in (select seat_id from cinema where free=1)
)
order by C1.seat_id
```

180. Consecutive Numbers

Write a SQL query to find all numbers that appear at least three times consecutively.

```
+----+-----+
| Id | Num |
+----+-----+
| 1  |  1  |
| 2  |  1  |
| 3  |  1  |
| 4  |  2  |
| 5  |  1  |
| 6  |  2  |
| 7  |  2  |
+----+-----+
```

For example, given the above `Logs` table, `1` is the only number that appears consecutively for at least three times.

```
+-----------------+
| ConsecutiveNums |
+-----------------+
| 1               |
+-----------------+
```

Solution:

```
Select distinct l1.Num as ConsecutiveNums from Logs l1, logs l2, logs l3
where l1.Id = l2.Id-1 and l2.Id=l3.Id-1
and l1.Num = l2.Num and l2.Num=l3.Num
```

184. Department Highest Salary

Medium 225 60 Favorite Share

SQL Schema ›

The `Employee` table holds all employees. Every employee has an Id, a salary, and there is also a column for the department Id.

```
+----+-------+--------+--------------+
| Id | Name  | Salary | DepartmentId |
+----+-------+--------+--------------+
| 1  | Joe   | 70000  | 1            |
| 2  | Henry | 80000  | 2            |
| 3  | Sam   | 60000  | 2            |
| 4  | Max   | 90000  | 1            |
+----+-------+--------+--------------+
```

The `Department` table holds all departments of the company.

```
+----+----------+
| Id | Name     |
+----+----------+
| 1  | IT       |
| 2  | Sales    |
+----+----------+
```

Write a SQL query to find employees who have the highest salary in each of the departments. For the above tables, Max has the highest salary in the IT department and Henry has the highest salary in the Sales department.

```
+------------+----------+--------+
| Department | Employee | Salary |
+------------+----------+--------+
| IT         | Max      | 90000  |
| Sales      | Henry    | 80000  |
+------------+----------+--------+
```

```sql
SELECT dep.Name as Department, emp.Name as Employee, emp.Salary
from Department dep, Employee emp
where emp.DepartmentId=dep.Id
and emp.Salary=(Select max(Salary) from Employee e2 where e2.DepartmentId=dep.Id)
```

```sql
SELECT D.Name AS Department ,E.Name AS Employee ,E.Salary
FROM
    Employee E,
    (SELECT DepartmentId,max(Salary) as max FROM Employee GROUP BY DepartmentId) T,
    Department D
WHERE E.DepartmentId = T.DepartmentId
  AND E.Salary = T.max
  AND E.DepartmentId = D.id
```

185. Department Top Three Salaries

Hard 272 61 Favorite Share

SQL Schema ›

The `Employee` table holds all employees. Every employee has an Id, and there is also a column for the department Id.

```
+----+-------+--------+--------------+
| Id | Name  | Salary | DepartmentId |
+----+-------+--------+--------------+
| 1  | Joe   | 70000  | 1            |
| 2  | Henry | 80000  | 2            |
| 3  | Sam   | 60000  | 2            |
| 4  | Max   | 90000  | 1            |
| 5  | Janet | 69000  | 1            |
| 6  | Randy | 85000  | 1            |
+----+-------+--------+--------------+
```

The `Department` table holds all departments of the company.

```
+----+----------+
| Id | Name     |
+----+----------+
| 1  | IT       |
| 2  | Sales    |
+----+----------+
```

Write a SQL query to find employees who earn the top three salaries in each of the department. For the above tables, your SQL query should return the following rows.

```
+------------+----------+--------+
| Department | Employee | Salary |
+------------+----------+--------+
| IT         | Max      | 90000  |
| IT         | Randy    | 85000  |
| IT         | Joe      | 70000  |
| Sales      | Henry    | 80000  |
| Sales      | Sam      | 60000  |
+------------+----------+--------+
```

```
SELECT D.Name as Department, E.Name as Employee, E.Salary
FROM Department D, Employee E, Employee E2
WHERE D.ID = E.DepartmentId and E.DepartmentId = E2.DepartmentId and
E.Salary <= E2.Salary
group by D.ID,E.Name having count(distinct E2.Salary) <= 3
order by D.Name, E.Salary desc
```

595. Big Countries

Easy 333 421 Favorite Share

SQL Schema ›

There is a table `World`

```
+-----------------+------------+------------+-----------
---+---------------+
| name            | continent  | area       | population
| gdp           |
+-----------------+------------+------------+-----------
---+---------------+
| Afghanistan     | Asia       | 652230     | 25500100
| 20343000      |
| Albania         | Europe     | 28748      | 2831741
| 12960000      |
| Algeria         | Africa     | 2381741    | 37100000
| 188681000     |
| Andorra         | Europe     | 468        | 78115
| 3712000       |
| Angola          | Africa     | 1246700    | 20609294
| 100990000     |
+-----------------+------------+------------+-----------
```

A country is big if it has an area of bigger than 3 million square km or a population of more than 25 million.

Write a SQL solution to output big countries' name, population and area.

For example, according to the above table, we should output:

```
+--------------+-------------+--------------+
| name         | population  | area         |
+--------------+-------------+--------------+
| Afghanistan  | 25500100    | 652230       |
| Algeria      | 37100000    | 2381741      |
+--------------+-------------+--------------+
```

Accepted 82,573 | Submissions 113,940

```sql
SELECT
    name, population, area
FROM
    world
WHERE
    area > 3000000 OR population > 25000000
;
```

196. Delete Duplicate Emails

Easy 228 253 Favorite Share

Write a SQL query to **delete** all duplicate email entries in a table named `Person`, keeping only unique emails based on its *smallest* **Id**.

```
+----+------------------+
| Id | Email            |
+----+------------------+
| 1  | john@example.com |
| 2  | bob@example.com  |
| 3  | john@example.com |
+----+------------------+
Id is the primary key column for this table.
```

For example, after running your query, the above `Person` table should have the following rows:

```
+----+------------------+
| Id | Email            |
+----+------------------+
| 1  | john@example.com |
| 2  | bob@example.com  |
+----+------------------+
```

```mysql
# Write your MySQL query statement below
delete p1
FROM Person p1, Person p2
WHERE p1.Email = p2.Email AND
p1.Id > p2.Id
```

626. Exchange Seats

Medium 130 120 Favorite Share

SQL Schema ›

Mary is a teacher in a middle school and she has a table `seat` storing students' names and their corresponding seat ids.

The column **id** is continuous increment.

Mary wants to change seats for the adjacent students.

Can you write a SQL query to output the result for Mary?

```
+---------+---------+
|    id   | student |
+---------+---------+
|    1    | Abbot   |
|    2    | Doris   |
|    3    | Emerson |
|    4    | Green   |
|    5    | Jeames  |
+---------+---------+
```

For the sample input, the output is:

```
+---------+---------+
|    id   | student |
+---------+---------+
|    1    | Doris   |
|    2    | Abbot   |
|    3    | Green   |
|    4    | Emerson |
|    5    | Jeames  |
+---------+---------+
```

Note:
If the number of students is odd, there is no need to change the last one's seat.

```
# Write your MySQL query statement below
SELECT
    CASE
        WHEN seat.id % 2 <> 0 AND seat.id = (SELECT COUNT(*) FROM seat) THEN seat.id
        WHEN seat.id % 2 = 0 THEN seat.id - 1
        ELSE
            seat.id + 1
    END as id,
    student
FROM seat
ORDER BY id
;
```

569. Median Employee Salary

Hard 31 17 Favorite Share

SQL Schema ›

The `Employee` table holds all employees. The employee table has three columns: Employee Id, Company Name, and Salary.

```
+-----+------------+--------+
|Id   | Company    | Salary |
+-----+------------+--------+
|1    | A          | 2341   |
|2    | A          | 341    |
|3    | A          | 15     |
|4    | A          | 15314  |
|5    | A          | 451    |
|6    | A          | 513    |
|7    | B          | 15     |
|8    | B          | 13     |
|9    | B          | 1154   |
|10   | B          | 1345   |
|11   | B          | 1221   |
|12   | B          | 234    |
|13   | C          | 2345   |
|14   | C          | 2645   |
|15   | C          | 2645   |
|16   | C          | 2652   |
|17   | C          | 65     |
+-----+------------+--------+
```

Write a SQL query to find the median salary of each company. Bonus points if you can solve it without using any built-in SQL functions.

```
+-----+------------+--------+
|Id   | Company    | Salary |
+-----+------------+--------+
|5    | A          | 451    |
|6    | A          | 513    |
|12   | B          | 234    |
|9    | B          | 1154   |
|14   | C          | 2645   |
+-----+------------+--------+
```

```sql
SELECT
  id,
  Company,
  Salary
FROM Employee e
WHERE 1 >= ABS((SELECT COUNT(*) FROM Employee e1 WHERE e.company = e1.company AND e.Salary >= e1.Salary) -
           (SELECT COUNT(*) FROM Employee e2 WHERE e.company = e2.company AND e.Salary <= e2.Salary))
GROUP BY Company, Salary
```

615. Average Salary: Departments VS Company

Hard 28 6 Favorite Share

SQL Schema ›

Given two tables as below, write a query to display the comparison result (higher/lower/same) of the average salary of employees in a department to the company's average salary.

Table: `salary`

```
| id | employee_id | amount | pay_date   |
|----|-------------|--------|------------|
| 1  | 1           | 9000   | 2017-03-31 |
| 2  | 2           | 6000   | 2017-03-31 |
| 3  | 3           | 10000  | 2017-03-31 |
| 4  | 1           | 7000   | 2017-02-28 |
| 5  | 2           | 6000   | 2017-02-28 |
| 6  | 3           | 8000   | 2017-02-28 |
```

The **employee_id** column refers to the **employee_id** in the following table `employee`.

```
| employee_id | department_id |
|-------------|---------------|
| 1           | 1             |
| 2           | 2             |
| 3           | 2             |
```

So for the sample data above, the result is:

```
| pay_month | department_id | comparison  |
|-----------|---------------|-------------|
| 2017-03   | 1             | higher      |
| 2017-03   | 2             | lower       |
| 2017-02   | 1             | same        |
| 2017-02   | 2             | same        |
```

Explanation

In March, the company's average salary is (9000+6000+10000)/3 = 8333.33...

The average salary for department '1' is 9000, which is the salary of **employee_id** '1' since there is only one employee in this department. So the comparison result is 'higher' since 9000 > 8333.33 obviously.

The average salary of department '2' is (6000 + 10000)/2 = 8000, which is the average of **employee_id** '2' and '3'. So the comparison result is 'lower' since 8000 < 8333.33.

With he same formula for the average salary comparison in February, the result is 'same' since both the department '1' and '2' have the same average salary with the company, which is 7000.

Accepted 2,707 | Submissions 7,883

```
SELECT d1.pay_month, d1.department_id,
CASE WHEN d1.department_avg > c1.company_avg THEN 'higher'
     WHEN d1.department_avg < c1.company_avg THEN 'lower'
     ELSE 'same'
END AS 'comparison'
FROM ((SELECT LEFT(s1.pay_date, 7) pay_month, e1.department_id, AVG(s1.amount) department_avg
FROM salary s1
JOIN employee e1 ON s1.employee_id = e1.employee_id
GROUP BY pay_month, e1.department_id) d1
LEFT JOIN (SELECT LEFT(pay_date, 7) pay_month, AVG(amount) company_avg
FROM salary
GROUP BY pay_month) c1 ON d1.pay_month = c1.pay_month)
ORDER BY pay_month DESC, department_id;
```

570. Managers with at Least 5 Direct Reports

Medium 51 4 Favorite Share

SQL Schema ›

The `Employee` table holds all employees including their managers. Every employee has an Id, and there is also a column for the manager Id.

```
+------+----------+-----------+----------+
|Id    |Name      |Department |ManagerId |
+------+----------+-----------+----------+
|101   |John      |A          |null      |
|102   |Dan       |A          |101       |
|103   |James     |A          |101       |
|104   |Amy       |A          |101       |
|105   |Anne      |A          |101       |
|106   |Ron       |B          |101       |
+------+----------+-----------+----------+
```

Given the `Employee` table, write a SQL query that finds out managers with at least 5 direct report. For the above table, your SQL query should return:

Given the `Employee` table, write a SQL query that finds out managers with at least 5 direct report. For the above table, your SQL query should return:

```
+-------+
| Name  |
+-------+
| John  |
+-------+
```

```sql
SELECT
    Name
FROM
    Employee AS t1 JOIN
    (SELECT
        ManagerId
    FROM
        Employee
    GROUP BY ManagerId
    HAVING COUNT(ManagerId) >= 5) AS t2
    ON t1.Id = t2.ManagerId
;
```

597. Friend Requests I: Overall Acceptance Rate

Easy 74 78 Favorite Share

SQL Schema ›

In social network like Facebook or Twitter, people send friend requests and accept others' requests as well. Now given two tables as below:

Table: `friend_request`

```
| sender_id | send_to_id |request_date|
|-----------|------------|------------|
| 1         | 2          | 2016_06-01 |
| 1         | 3          | 2016_06-01 |
| 1         | 4          | 2016_06-01 |
| 2         | 3          | 2016_06-02 |
| 3         | 4          | 2016-06-09 |
```

Table: `request_accepted`

```
| requester_id | accepter_id |accept_date |
|--------------|-------------|------------|
| 1            | 2           | 2016_06-03 |
| 1            | 3           | 2016-06-08 |
| 2            | 3           | 2016-06-08 |
| 3            | 4           | 2016-06-09 |
| 3            | 4           | 2016-06-10 |
```

Write a query to find the overall acceptance rate of requests rounded to 2 decimals, which is the number of acceptance divide the number of requests.

For the sample data above, your query should return the following result.

```
|accept_rate|
|-----------|
|       0.80|
```

Note:

- The accepted requests are not necessarily from the table `friend_request`. In this case, you just need to simply count the total accepted requests (no matter whether they are in the original requests), and divide it by the number of requests to get the acceptance rate.
- It is possible that a sender sends multiple requests to the same receiver, and a request could be accepted more than once. In this case, the 'duplicated' requests or acceptances are only counted once.
- If there is no requests at all, you should return 0.00 as the accept_rate.

Explanation: There are 4 unique accepted requests, and there are 5 requests in total. So the rate is 0.80.

Follow-up:

- Can you write a query to return the accept rate but for every month?
- How about the cumulative accept rate for every day?

```
# Write your MySQL query statement below
SELECT ifnull(Round(count(distinct requester_id, accepter_id) / count(distinct
sender_id, send_to_id), 2),0) as accept_rate
FROM request_accepted, friend_request
```

574. Winning Candidate

Medium 15 125 Favorite Share

SQL Schema ›

Table: `Candidate`

```
+-----+---------+
| id  | Name    |
+-----+---------+
| 1   | A       |
| 2   | B       |
| 3   | C       |
| 4   | D       |
| 5   | E       |
+-----+---------+
```

Table: `Vote`

Table: `Vote`

```
+-----+--------------+
| id  | CandidateId  |
+-----+--------------+
| 1   |     2        |
| 2   |     4        |
| 3   |     3        |
| 4   |     2        |
| 5   |     5        |
+-----+--------------+
id is the auto-increment primary key,
CandidateId is the id appeared in Candidate
table.
```

Write a sql to find the name of the winning candidate, the above example will return the winner `B`.

```sql
SELECT
    name AS 'Name'
FROM
    Candidate
        JOIN
    (SELECT
        Candidateid
    FROM
        Vote
    GROUP BY Candidateid
    ORDER BY COUNT(*) DESC
    LIMIT 1) AS winner
WHERE
    Candidate.id = winner.Candidateid
;
```

178. Rank Scores

Medium 396 38 Favorite Share

SQL Schema ›

Write a SQL query to rank scores. If there is a tie between two scores, both should have the same ranking. Note that after a tie, the next ranking number should be the next consecutive integer value. In other words, there should be no "holes" between ranks.

```
+----+-------+
| Id | Score |
+----+-------+
| 1  | 3.50  |
| 2  | 3.65  |
| 3  | 4.00  |
| 4  | 3.85  |
| 5  | 4.00  |
| 6  | 3.65  |
+----+-------+
```

For example, given the above `Scores` table, your query should generate the following report (order by highest score):

```
+-------+------+
| Score | Rank |
+-------+------+
| 4.00  | 1    |
| 4.00  | 1    |
| 3.85  | 2    |
| 3.65  | 3    |
| 3.65  | 3    |
| 3.50  | 4    |
+-------+------+
```

```
SELECT
  Score,
  (SELECT count(distinct Score) FROM Scores WHERE Score >= s.Score) Rank
FROM Scores s
ORDER BY Score desc
```

579. Find Cumulative Salary of an Employee

Hard 32 35 Favorite Share

SQL Schema ›

The **Employee** table holds the salary information in a year.

Write a SQL to get the cumulative sum of an employee's salary over a period of 3 months but exclude the most recent month.

The result should be displayed by 'Id' ascending, and then by 'Month' descending.

Example
Input

```
| Id | Month | Salary |
|----|-------|--------|
| 1  | 1     | 20     |
| 2  | 1     | 20     |
| 1  | 2     | 30     |
| 2  | 2     | 30     |
| 3  | 2     | 40     |
| 1  | 3     | 40     |
| 3  | 3     | 60     |
| 1  | 4     | 60     |
| 3  | 4     | 70     |
```

Output

```
| Id | Month | Salary |
|----|-------|--------|
| 1  | 3     | 90     |
| 1  | 2     | 50     |
| 1  | 1     | 20     |
| 2  | 1     | 20     |
| 3  | 3     | 100    |
| 3  | 2     | 40     |
```

```sql
SELECT   A.Id, MAX(B.Month) as Month, SUM(B.Salary) as Salary
FROM     Employee A, Employee B
WHERE    A.Id = B.Id AND B.Month BETWEEN (A.Month-3) AND (A.Month-1)
GROUP BY A.Id, A.Month
ORDER BY Id, Month DESC
```

608. Tree Node

Medium 82 4 Favorite Share

SQL Schema ›

Given a table `tree`, **id** is identifier of the tree node and **p_id** is its parent node's **id**.

```
+----+------+
| id | p_id |
+----+------+
| 1  | null |
| 2  | 1    |
| 3  | 1    |
| 4  | 2    |
| 5  | 2    |
+----+------+
```

Each node in the tree can be one of three types:

- Leaf: if the node is a leaf node.
- Root: if the node is the root of the tree.
- Inner: If the node is neither a leaf node nor a root node.

Write a query to print the node id and the type of the node. Sort your output by the node id. The result for the above sample is:

```
+----+------+
| id | Type |
+----+------+
| 1  | Root |
| 2  | Inner|
| 3  | Leaf |
| 4  | Leaf |
| 5  | Leaf |
+----+------+
```

```mysql
# Write your MySQL query statement below
select T.id,
IF(isnull(T.p_id), 'Root', IF(T.id in (select p_id from tree), 'Inner', 'Leaf')) Type
from tree T
```

610. Triangle Judgement

Easy 40 8 Favorite Share

SQL Schema ›

A pupil Tim gets homework to identify whether three line segments could possibly form a triangle.

However, this assignment is very heavy because there are hundreds of records to calculate.

Could you help Tim by writing a query to judge whether these three sides can form a triangle, assuming table `triangle` holds the length of the three sides x, y and z.

```
| x  | y  | z  |
|----|----|----|
| 13 | 15 | 30 |
| 10 | 20 | 15 |
```

For the sample data above, your query should return the follow result:

```
| x  | y  | z  | triangle |
|----|----|----|----------|
| 13 | 15 | 30 | No       |
| 10 | 20 | 15 | Yes      |
```

Accepted 7,763 | Submissions 12,818

```mysql
# Write your MySQL query statement below
SELECT x, y, z,
CASE WHEN x+y<=z OR
          x+z<=y OR
          y+z<=x
     THEN 'No'
     ELSE 'Yes'
END AS 'triangle'
FROM triangle;
```

180. Consecutive Numbers

Medium 189 49 Favorite Share

SQL Schema ›

Write a SQL query to find all numbers that appear at least three times consecutively.

```
+----+-----+
| Id | Num |
+----+-----+
| 1  |  1  |
| 2  |  1  |
| 3  |  1  |
| 4  |  2  |
| 5  |  1  |
| 6  |  2  |
| 7  |  2  |
+----+-----+
```

For example, given the above `Logs` table, `1` is the only number that appears consecutively for at least three times.

```
+-----------------+
| ConsecutiveNums |
+-----------------+
| 1               |
+-----------------+
```

Accepted 43,962 | Submissions 140,109

```
# Write your MySQL query statement below
Select distinct l1.Num as ConsecutiveNums from Logs l1, logs l2, logs l3
where l1.Id = l2.Id-1 and l2.Id=l3.Id-1
and l1.Num = l2.Num and l2.Num=l3.Num
```

```
SELECT T.Num as ConsecutiveNums
FROM
(SELECT DISTINCT A.Num FROM
Logs A
LEFT JOIN Logs B on A.Id = B.Id-1
LEFT JOIN Logs C on A.Id = C.Id-2
WHERE A.Num = B.Num AND A.Num = C.Num) T
```

181. Employees Earning More Than Their Managers

Easy 287 28 Favorite Share

SQL Schema ›

The `Employee` table holds all employees including their managers. Every employee has an Id, and there is also a column for the manager Id.

```
+----+-------+--------+-----------+
| Id | Name  | Salary | ManagerId |
+----+-------+--------+-----------+
| 1  | Joe   | 70000  | 3         |
| 2  | Henry | 80000  | 4         |
| 3  | Sam   | 60000  | NULL      |
| 4  | Max   | 90000  | NULL      |
+----+-------+--------+-----------+
```

Given the `Employee` table, write a SQL query that finds out employees who earn more than their managers. For the above table, Joe is the only employee who earns more than his manager.

```
+----------+
| Employee |
+----------+
| Joe      |
+----------+
```

```mysql
# Write your MySQL query statement below
select e.Name as Employee
from Employee e, Employee m
where e.ManagerId is not NULL and
e.ManagerId = m.ID and e.Salary > m.Salary
```

182. Duplicate Emails

Easy 228 14 Favorite Share

SQL Schema ›

Write a SQL query to find all duplicate emails in a table named `Person`.

```
+----+---------+
| Id | Email   |
+----+---------+
| 1  | a@b.com |
| 2  | c@d.com |
| 3  | a@b.com |
+----+---------+
```

For example, your query should return the following for the above table:

```
+---------+
| Email   |
+---------+
| a@b.com |
+---------+
```

Note: All emails are in lowercase.

Accepted 94,031 | Submissions 182,717

```mysql
# Write your MySQL query statement below
select Email
from Person
group by Email
having count(*) > 1
```

183. Customers Who Never Order

Easy 195 21 Favorite Share

SQL Schema ›

Suppose that a website contains two tables, the `Customers` table and the `Orders` table. Write a SQL query to find all customers who never order anything.

Table: `Customers`.

```
+----+-------+
| Id | Name  |
+----+-------+
| 1  | Joe   |
| 2  | Henry |
| 3  | Sam   |
| 4  | Max   |
+----+-------+
```

Table: `Orders`.

```
+----+------------+
| Id | CustomerId |
+----+------------+
| 1  | 3          |
| 2  | 1          |
+----+------------+
```

Using the above tables as example, return the following:

```
+-----------+
| Customers |
+-----------+
| Henry     |
| Max       |
+-----------+
```

```sql
SELECT A.Name from Customers A
WHERE NOT EXISTS (SELECT 1 FROM Orders B WHERE A.Id = B.CustomerId)

SELECT A.Name from Customers A
LEFT JOIN Orders B on  a.Id = B.CustomerId
WHERE b.CustomerId is NULL

SELECT A.Name from Customers A
WHERE A.Id NOT IN (SELECT B.CustomerId from Orders B)
```

196. Delete Duplicate Emails

Easy 231 255 Favorite Share

Write a SQL query to **delete** all duplicate email entries in a table named `Person`, keeping only unique emails based on its *smallest* **Id**.

```
+----+------------------+
| Id | Email            |
+----+------------------+
| 1  | john@example.com |
| 2  | bob@example.com  |
| 3  | john@example.com |
+----+------------------+
Id is the primary key column for this table.
```

For example, after running your query, the above `Person` table should have the following rows:

```
+----+------------------+
| Id | Email            |
+----+------------------+
| 1  | john@example.com |
| 2  | bob@example.com  |
+----+------------------+
```

Note:

Your output is the whole `Person` table after executing your sql. Use `delete` statement.

Accepted 59,830 | Submissions 200,696

```sql
# Write your MySQL query statement below
DELETE p1
FROM Person p1, Person p2
WHERE p1.Email = p2.Email AND
p1.Id > p2.Id
```

197. Rising Temperature

Easy 194 72 Favorite Share

SQL Schema ›

Given a `Weather` table, write a SQL query to find all dates' Ids with higher temperature compared to its previous (yesterday's) dates.

```
+---------+------------------+------------------+
| Id(INT) | RecordDate(DATE) | Temperature(INT) |
+---------+------------------+------------------+
|       1 |       2015-01-01 |               10 |
|       2 |       2015-01-02 |               25 |
|       3 |       2015-01-03 |               20 |
|       4 |       2015-01-04 |               30 |
+---------+------------------+------------------+
```

For example, return the following Ids for the above `Weather` table:

```
+----+
| Id |
+----+
|  2 |
|  4 |
+----+
```

Accepted 64,462 | Submissions 194,514

```sql
# Write your MySQL query statement below
SELECT wt1.Id
FROM Weather wt1, Weather wt2
WHERE wt1.Temperature > wt2.Temperature AND
      TO_DAYS(wt1.RecordDate)-TO_DAYS(wt2.RecordDate)=1;
```

262. Trips and Users

Hard 158 116 Favorite Share

SQL Schema ›

The `Trips` table holds all taxi trips. Each trip has a unique Id, while Client_Id and Driver_Id are both foreign keys to the Users_Id at the `Users` table. Status is an ENUM type of (‘completed’, ‘cancelled_by_driver’, ‘cancelled_by_client’).

```
+----+-----------+-----------+---------+---------
-----------+----------+
| Id | Client_Id | Driver_Id | City_Id |
Status      |Request_at|
+----+-----------+-----------+---------+---------
-----------+----------+
| 1  |     1     |    10     |    1    |
completed      |2013-10-01|
| 2  |     2     |    11     |    1    |
cancelled_by_driver|2013-10-01|
| 3  |     3     |    12     |    6    |
completed      |2013-10-01|
| 4  |     4     |    13     |    6    |
cancelled_by_client|2013-10-01|
| 5  |     1     |    10     |    1    |
completed      |2013-10-02|
| 6  |     2     |    11     |    6    |
completed      |2013-10-02|
| 7  |     3     |    12     |    6    |
completed      |2013-10-02|
| 8  |     2     |    12     |    12   |
completed      |2013-10-03|
| 9  |     3     |    10     |    12   |
completed      |2013-10-03|
| 10 |     4     |    13     |    12   |
cancelled_by_driver|2013-10-03|
+----+-----------+-----------+---------+---------
-----------+----------+
```

The `Users` table holds all users. Each user has an unique Users_Id, and Role is an ENUM type of ('client', 'driver', 'partner').

```
+----------+--------+--------+
| Users_Id | Banned |  Role  |
+----------+--------+--------+
|    1     |   No   | client |
|    2     |   Yes  | client |
|    3     |   No   | client |
|    4     |   No   | client |
|    10    |   No   | driver |
|    11    |   No   | driver |
|    12    |   No   | driver |
|    13    |   No   | driver |
+----------+--------+--------+
```

Write a SQL query to find the cancellation rate of requests made by unbanned users between **Oct 1, 2013** and **Oct 3, 2013**. For the above tables, your SQL query should return the following rows with the cancellation rate being rounded to *two* decimal places.

```
+------------+-------------------+
|     Day    | Cancellation Rate |
+------------+-------------------+
| 2013-10-01 |       0.33        |
| 2013-10-02 |       0.00        |
| 2013-10-03 |       0.50        |
+------------+-------------------+
```

Credits:
Special thanks to @cak1erlizhou for contributing this question, writing the problem description and adding part of the test cases.

Accepted 27,129 | Submissions 119,476

```sql
SELECT Request_at as Day,
       ROUND(SUM(CASE WHEN Status LIKE 'cancelled%' THEN 1 ELSE 0 END) / COUNT(*), 2) as "Cancellation Rate"
FROM(
    SELECT * FROM Trips t
    WHERE
        t.Client_Id not in (select Users_Id from Users where Banned = 'Yes') AND
        t.Driver_Id not in (select Users_Id from Users where Banned = 'Yes') AND
        t.Request_at between '2013-10-01' and '2013-10-03'
    ) AS newT
GROUP BY Request_at
```

571. Find Median Given Frequency of Numbers

Hard 39 14 Favorite Share

SQL Schema ›

The `Numbers` table keeps the value of number and its frequency.

```
+----------+-------------+
|  Number  |  Frequency  |
+----------+-------------|
|  0       |  7          |
|  1       |  1          |
|  2       |  3          |
|  3       |  1          |
+----------+-------------+
```

In this table, the numbers are `0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 2, 3`, so the median is `(0 + 0) / 2 = 0`.

```
+--------+
| median |
+--------|
| 0.0000 |
+--------+
```

Write a query to find the median of all numbers and name the result as `median`.

```
select  avg(n.Number) median
from Numbers n
where n.Frequency >= abs((select sum(Frequency) from Numbers where Number<=n.Number) -
                         (select sum(Frequency) from Numbers where Number>=n.Number))
```

Explanation:
Let's take all numbers from left including current number and then do same for right.
(select sum(Frequency) from Numbers where Number<=n.Number) as left
(select sum(Frequency) from Numbers where Number<=n.Number) as right
Now if difference between Left and Right less or equal to Frequency of the current number that means this number is median.
Ok, what if we get two numbers satisfied this condition? Easy peasy - take AVG(). Ta-da!

577. Employee Bonus

Easy 31 14 Favorite Share

SQL Schema ›

Select all employee's name and bonus whose bonus is < 1000.

Table: `Employee`

```
+-------+--------+-----------+--------+
| empId |  name  | supervisor| salary |
+-------+--------+-----------+--------+
|   1   | John   |  3        | 1000   |
|   2   | Dan    |  3        | 2000   |
|   3   | Brad   |  null     | 4000   |
|   4   | Thomas |  3        | 4000   |
+-------+--------+-----------+--------+
empId is the primary key column for this table.
```

Table: `Bonus`

```
+-------+-------+
| empId | bonus |
+-------+-------+
| 2     | 500   |
| 4     | 2000  |
+-------+-------+
empId is the primary key column for this table.
```

Example ouput:

```
+-------+-------+
| name  | bonus |
+-------+-------+
| John  | null  |
| Dan   | 500   |
| Brad  | null  |
+-------+-------+
```

Accepted 9,507 | Submissions 16,991

```mysql
# Write your MySQL query statement below
SELECT name, bonus
FROM Employee LEFT JOIN Bonus
ON Employee.empID=Bonus.empID
WHERE bonus<1000 OR bonus IS NULL
```

578. Get Highest Answer Rate Question

Medium 18 160 Favorite Share

SQL Schema ›

Get the highest answer rate question from a table `survey_log` with these columns: **uid**, **action**, **question_id**, **answer_id**, **q_num**, **timestamp**.

uid means user id; action has these kind of values: "show", "answer", "skip"; answer_id is not null when action column is "answer", while is null for "show" and "skip"; q_num is the numeral order of the question in current session.

Write a sql query to identify the question which has the highest answer rate.

Example:

Input:

```
+------+-----------+--------------+------------+-
----------+------------+
| uid  | action    | question_id  | answer_id  |
q_num     | timestamp  |
+------+-----------+--------------+------------+-
----------+------------+
| 5    | show      | 285          | null       |
1         | 123        |
| 5    | answer    | 285          | 124124     |
1         | 124        |
| 5    | show      | 369          | null       |
2         | 125        |
| 5    | skip      | 369          | null       |
2         | 126        |
+------+-----------+--------------+------------+-
----------+------------+
```

Output:

```
+-------------+
| survey_log  |
+-------------+
|    285      |
+-------------+
```

Explanation:

question 285 has answer rate 1/1, while question 369 has 0/1 answer rate, so output 285.

Note: The highest answer rate meaning is: answer number's ratio in show number in the same question.

Accepted 6,584 | Submissions 19,002

```mysql
# Write your MySQL query statement below
SELECT question_id as survey_log
FROM
(
    SELECT question_id, SUM(case when action="show" THEN 1 ELSE 0 END) as num_show,
 SUM(case when action="answer" THEN 1 ELSE 0 END) as num_answer
    FROM survey_log
    GROUP BY question_id
) as tbl
ORDER BY (num_answer / num_show) DESC LIMIT 1
```

580. Count Student Number in Departments

Medium 46 3 Favorite Share

SQL Schema ›

A university uses 2 data tables, ***student*** and ***department***, to store data about its students and the departments associated with each major.

Write a query to print 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).

Sort your results by descending number of students; if two or more departments have the same number of students, then sort those departments alphabetically by department name.

The ***student*** is described as follow:

```
| Column Name  | Type      |
|--------------|-----------|
| student_id   | Integer   |
| student_name | String    |
| gender       | Character |
| dept_id      | Integer   |
```

where student_id is the student's ID number, student_name is the student's name, gender is their gender, and dept_id is the department ID associated with their declared major.

And the ***department*** table is described as below:

```
| Column Name | Type    |
|-------------|---------|
| dept_id     | Integer |
| dept_name   | String  |
```

where dept_id is the department's ID number and dept_name is the department name.

Here is an example **input**:
student table:

```
| student_id | student_name | gender | dept_id |
|------------|--------------|--------|---------|
| 1          | Jack         | M      | 1       |
| 2          | Jane         | F      | 1       |
| 3          | Mark         | M      | 2       |
```

department table:

```
| dept_id | dept_name   |
|---------|-------------|
| 1       | Engineering |
| 2       | Science     |
| 3       | Law         |
```

The **Output** should be:

```
| dept_name   | student_number |
|-------------|----------------|
| Engineering | 2              |
| Science     | 1              |
| Law         | 0              |
```

```mysql
# Write your MySQL query statement below
SELECT d.dept_name, COUNT(s.student_id) AS student_number
FROM student s RIGHT JOIN department d ON s.dept_id = d.dept_id
GROUP BY d.dept_name
ORDER BY student_number DESC, d.dept_name;
```

584. Find Customer Referee

Easy 34 15 Favorite Share

SQL Schema ›

Given a table `customer` holding customers information and the referee.

```
+------+------+-----------+
| id   | name | referee_id|
+------+------+-----------+
|    1 | Will |      NULL |
|    2 | Jane |      NULL |
|    3 | Alex |         2 |
|    4 | Bill |      NULL |
|    5 | Zack |         1 |
|    6 | Mark |         2 |
+------+------+-----------+
```

Write a query to return the list of customers **NOT** referred by the person with id '2'.

For the sample data above, the result is:

```
+------+
| name |
+------+
| Will |
| Jane |
| Bill |
| Zack |
+------+
```

Accepted 9,314 | Submissions 14,229

```sql
SELECT name
FROM customer
WHERE referee_id <> 2 OR referee_id IS NULL
```

585. Investments in 2016

Medium 39 31 Favorite Share

SQL Schema ›

Write a query to print the sum of all total investment values in 2016 (**TIV_2016**), to a scale of 2 decimal places, for all policy holders who meet the following criteria:

1. Have the same **TIV_2015** value as one or more other policyholders.
2. Are not located in the same city as any other policyholder (i.e.: the (latitude, longitude) attribute pairs must be unique).

Input Format:
The ***insurance*** table is described as follows:

Column Name	Type
PID	INTEGER(11)
TIV_2015	NUMERIC(15,2)
TIV_2016	NUMERIC(15,2)
LAT	NUMERIC(5,2)
LON	NUMERIC(5,2)

where **PID** is the policyholder's policy ID, **TIV_2015** is the total investment value in 2015, **TIV_2016** is the total investment value in 2016, **LAT** is the latitude of the policy holder's city, and **LON** is the longitude of the policy holder's city.

Sample Input

PID	TIV_2015	TIV_2016	LAT	LON
1	10	5	10	10
2	20	20	20	20
3	10	30	20	20
4	10	40	40	40

Sample Output

TIV_2016
45.00

Explanation

The first record in the table, like the last record, meets both of the two criteria.
The **TIV_2015** value '10' is as the same as the third and forth record, and its location unique.

The second record does not meet any of the two criteria. Its **TIV_2015** is not like any other policyholders.

And its location is the same with the third record, which makes the third record fail, too.

So, the result is the sum of **TIV_2016** of the first and last record, which is 45.

```
# Write your MySQL query statement below
select sum(TIV_2016) TIV_2016
from insurance a
where 1 = (select count(*) from insurance b where a.LAT=b.LAT and a.LON=b.LON)
and 1 < (select count(*) from insurance c where a.TIV_2015=c.TIV_2015);
```

586. Customer Placing the Largest Number of Orders

Easy 52 3 Favorite Share

SQL Schema ›

Query the **customer_number** from the ***orders*** table for the customer who has placed the largest number of orders.

It is guaranteed that exactly one customer will have placed more orders than any other customer.

The ***orders*** table is defined as follows:

```
| Column            | Type      |
|-------------------|-----------|
| order_number (PK) | int       |
| customer_number   | int       |
| order_date        | date      |
| required_date     | date      |
| shipped_date      | date      |
| status            | char(15)  |
| comment           | char(200) |
```

Sample Input

```
| order_number | customer_number | order_date |
required_date | shipped_date | status | comment |
|--------------|-----------------|------------|--
-------------|--------------|--------|---------|
| 1            | 1               | 2017-04-09 |
2017-04-13    | 2017-04-12   | Closed |         |
| 2            | 2               | 2017-04-15 |
2017-04-20    | 2017-04-18   | Closed |         |
| 3            | 3               | 2017-04-16 |
2017-04-25    | 2017-04-20   | Closed |         |
| 4            | 3               | 2017-04-18 |
2017-04-28    | 2017-04-25   | Closed |         |
```

Sample Output

```
| customer_number |
|-----------------|
| 3               |
```

Explanation

```
The customer with number '3' has two orders,
which is greater than either customer '1' or '2'
because each of them  only has one order.
So the result is customer_number '3'.
```

Follow up: *What if more than one customer have the largest number of orders, can you find all the customer_number in this case?*

Accepted 10,299 | Submissions 16,024

```mysql
# Write your MySQL query statement below
select customer_number from orders
group by customer_number
order by count(*) desc limit 1;
```

596. Classes More Than 5 Students

Easy 132 345 Favorite Share

SQL Schema ›

There is a table `courses` with columns: **student** and **class**

Please list out all classes which have more than or equal to 5 students.

For example, the table:

```
+---------+------------+
| student | class      |
+---------+------------+
| A       | Math       |
| B       | English    |
| C       | Math       |
| D       | Biology    |
| E       | Math       |
| F       | Computer   |
| G       | Math       |
| H       | Math       |
| I       | Math       |
+---------+------------+
```

Should output:

```
+---------+
| class   |
+---------+
| Math    |
+---------+
```

Note:
The students should not be counted duplicate in each course.

Accepted 33,285 | Submissions 97,098

```mysql
# Write your MySQL query statement below
select class from courses group by class having count(distinct student) >= 5;
```

601. Human Traffic of Stadium

Hard 65 146 Favorite Share

SQL Schema ›

X city built a new stadium, each day many people visit it and the stats are saved as these columns: **id**, **date**, **people**

Please write a query to display the records which have 3 or more consecutive rows and the amount of people more than 100(inclusive).

For example, the table `stadium`:

```
+------+------------+-----------+
| id   | date       | people    |
+------+------------+-----------+
| 1    | 2017-01-01 | 10        |
| 2    | 2017-01-02 | 109       |
| 3    | 2017-01-03 | 150       |
| 4    | 2017-01-04 | 99        |
| 5    | 2017-01-05 | 145       |
| 6    | 2017-01-06 | 1455      |
| 7    | 2017-01-07 | 199       |
| 8    | 2017-01-08 | 188       |
+------+------------+-----------+
```

For the sample data above, the output is:

```
+------+------------+-----------+
| id   | date       | people    |
+------+------------+-----------+
| 5    | 2017-01-05 | 145       |
| 6    | 2017-01-06 | 1455      |
| 7    | 2017-01-07 | 199       |
| 8    | 2017-01-08 | 188       |
+------+------------+-----------+
```

Note:
Each day only have one row record, and the dates are increasing with id increasing.

Accepted 10,631 | Submissions 30,404

```sql
SELECT s1.* FROM stadium AS s1, stadium AS s2, stadium as s3
    WHERE
    ((s1.id + 1 = s2.id
    AND s1.id + 2 = s3.id)
    OR
    (s1.id - 1 = s2.id
    AND s1.id + 1 = s3.id)
    OR
    (s1.id - 2 = s2.id
    AND s1.id - 1 = s3.id)
    )
    AND s1.people>=100
    AND s2.people>=100
    AND s3.people>=100

    GROUP BY s1.id
```

602. Friend Requests II: Who Has the Most Friends

Medium 64 26 Favorite Share

SQL Schema ›

In social network like Facebook or Twitter, people send friend requests and accept others' requests as well.

Table `request_accepted` holds the data of friend acceptance, while **requester_id** and **accepter_id** both are the id of a person.

```
| requester_id | accepter_id | accept_date|
|--------------|-------------|------------|
| 1            | 2           | 2016_06-03 |
| 1            | 3           | 2016-06-08 |
| 2            | 3           | 2016-06-08 |
| 3            | 4           | 2016-06-09 |
```

Write a query to find the the people who has most friends and the most friends number. For the sample data above, the result is:

```
| id | num |
|----|-----|
| 3  | 3   |
```

Note:

- It is guaranteed there is only 1 people having the most friends.
- The friend request could only been accepted once, which mean there is no multiple records with the same **requester_id** and **accepter_id** value.

Explanation:
The person with id '3' is a friend of people '1', '2' and '4', so he has 3 friends in total, which is the most number than any others.

Follow-up:
In the real world, multiple people could have the same most number of friends, can you find all these people in this case?

Accepted 7,880 | Submissions 18,203

```
select id1 as id, count(id2) as num
from
(select requester_id as id1, accepter_id as id2
from request_accepted
union
select accepter_id as id1, requester_id as id2
from request_accepted) tmp1
group by id1
order by num desc limit 1
```

607. Sales Person

Easy 49 7 Favorite Share

SQL Schema ›

Description

Given three tables: `salesperson`, `company`, `orders`.
Output all the **names** in the table `salesperson`, who didn't have sales to company 'RED'.

Example
Input

Table: `salesperson`

```
+----------+------+--------+-----------------+---
--------+
| sales_id | name | salary | commission_rate |
hire_date |
+----------+------+--------+-----------------+---
--------+
|   1      | John | 100000 |     6           |
4/1/2006  |
|   2      | Amy  | 120000 |     5           |
5/1/2010  |
|   3      | Mark | 65000  |     12          |
12/25/2008|
|   4      | Pam  | 25000  |     25          |
1/1/2005  |
|   5      | Alex | 50000  |     10          |
2/3/2007  |
+----------+------+--------+-----------------+---
--------+
```

The table `salesperson` holds the salesperson information. Every salesperson has a **sales_id** and a **name**.
Table: `company`

```
+---------+--------+------------+
| com_id  |  name  |    city    |
+---------+--------+------------+
|   1     |  RED   |   Boston   |
|   2     | ORANGE |   New York |
|   3     | YELLOW |   Boston   |
|   4     | GREEN  |   Austin   |
+---------+--------+------------+
```

The table `company` holds the company information. Every company has a **com_id** and a **name**.
Table: `orders`

```
+----------+----------+---------+----------+-----
---+
| order_id |  date    | com_id  | sales_id |
amount |
+----------+----------+---------+----------+-----
---+
| 1        | 1/1/2014 |    3    |    4     |
100000 |
| 2        | 2/1/2014 |    4    |    5     | 5000
|
| 3        | 3/1/2014 |    1    |    1     |
50000  |
| 4        | 4/1/2014 |    1    |    4     |
25000  |
+----------+----------+---------+----------+-----
---+
```

The table `orders` holds the sales record information, salesperson and customer company are represented by **sales_id** and **com_id**.
output

```
+------+
| name |
+------+
| Amy  |
| Mark |
| Alex |
+------+
```

Explanation

According to order '3' and '4' in table `orders`, it is easy to tell only salesperson 'John' and 'Alex' have sales to company 'RED',
so we need to output all the other **names** in table `salesperson`.

Accepted 7,995 | Submissions 14,891

```
# Write your MySQL query statement below
select salesperson.name
from orders o join company c on (o.com_id = c.com_id and c.name = 'RED')
right join salesperson on salesperson.sales_id = o.sales_id
where o.sales_id is null
```

612. Shortest Distance in a Plane

Medium 38 6 Favorite Share

SQL Schema ›

Table `point_2d` holds the coordinates (x,y) of some unique points (more than two) in a plane.

Write a query to find the shortest distance between these points rounded to 2 decimals.

```
| x  | y  |
|----|----|
| -1 | -1 |
| 0  | 0  |
| -1 | -2 |
```

The shortest distance is 1.00 from point (-1,-1) to (-1,2). So the output should be:

```
| shortest |
|----------|
| 1.00     |
```

Note: The longest distance among all the points are less than 10000.

Accepted 5,313 | Submissions 10,155

```
select round(sqrt(min(pow(a.x-b.x,2)+pow(a.y-b.y,2))),2) shortest
from point_2d a, point_2d b
where (a.x,a.y)!=(b.x,b.y)
```

613. Shortest Distance in a Line

Easy 71 10 Favorite Share

SQL Schema ›

Table `point` holds the x coordinate of some points on x-axis in a plane, which are all integers.

Write a query to find the shortest distance between two points in these points.

```
| x   |
|-----|
| -1  |
| 0   |
| 2   |
```

The shortest distance is '1' obviously, which is from point '-1' to '0'. So the output is as below:

```
| shortest|
|---------|
| 1       |
```

Note: Every point is unique, which means there is no duplicates in table `point`.

Follow-up: What if all these points have an id and are arranged from the left most to the right most of x axis?

Accepted 10,264 | Submissions 14,298

```
# Write your MySQL query statement below
SELECT MIN(ABS(P1.x - P2.x)) AS shortest FROM point AS P1
JOIN point AS P2 ON P1.x <> P2.x
```

614. Second Degree Follower

Medium 20 186 Favorite Share

SQL Schema ›

In facebook, there is a `follow` table with two columns: **followee**, **follower**.

Please write a sql query to get the amount of each follower's follower if he/she has one.

For example:

```
+-------------+------------+
| followee    | follower   |
+-------------+------------+
|     A       |     B      |
|     B       |     C      |
|     B       |     D      |
|     D       |     E      |
+-------------+------------+
```

should output:

```
+-------------+------------+
| follower    | num        |
+-------------+------------+
|     B       |  2         |
|     D       |  1         |
+-------------+------------+
```

Explaination:
Both B and D exist in the follower list, when as a followee, B's follower is C and D, and D's follower is E. A does not exist in follower list.

Note:
Followee would not follow himself/herself in all cases.
Please display the result in follower's alphabet order.

Accepted 4,699 | Submissions 20,611

```sql
select distinct follower, num
from follow,
(select followee, count(distinct follower) as num from follow
group by followee) as t
where follower = t.followee
order by follower;
```

```sql
select f1.follower, count(distinct f2.follower) as num
from follow f1
join follow f2 on f1.follower = f2.followee
group by f1.follower
order by f1.follower;
```

618. Students Report By Geography

Hard 12 40 Favorite Share

SQL Schema ›

A U.S graduate school has students from Asia, Europe and America. The students' location information are stored in table `student` as below.

```
| name   | continent |
|--------|-----------|
| Jack   | America   |
| Pascal | Europe    |
| Xi     | Asia      |
| Jane   | America   |
```

Pivot the continent column in this table so that each name is sorted alphabetically and displayed underneath its corresponding continent. The output headers should be America, Asia and Europe respectively. It is guaranteed that the student number from America is no less than either Asia or Europe.

For the sample input, the output is:

For the sample input, the output is:

```
| America | Asia | Europe |
|---------|------|--------|
| Jack    | Xi   | Pascal |
| Jane    |      |        |
```

Follow-up: If it is unknown which continent has the most students, can you write a query to generate the student report?

Accepted 1,889 | Submissions 4,560

```mysql
# Write your MySQL query statement below
SELECT MAX(America) AS America, MAX(Asia) as Asia, MAX(Europe) AS Europe
FROM (
    SELECT
        CASE WHEN continent = 'America' THEN @r1 := @r1 + 1
             WHEN continent = 'Asia'    THEN @r2 := @r2 + 1
             WHEN continent = 'Europe'  THEN @r3 := @r3 + 1 END id,
        CASE WHEN continent = 'America' THEN name ELSE NULL END America,
        CASE WHEN continent = 'Asia'    THEN name ELSE NULL END Asia,
        CASE WHEN continent = 'Europe'  THEN name ELSE NULL END Europe
    FROM student, (SELECT @r1 := 0, @r2 := 0, @r3 := 0) AS ids
    ORDER BY name
) AS tempTable
GROUP BY id;
```

619. Biggest Single Number

Easy 28 27 Favorite Share

SQL Schema ›

Table `number` contains many numbers in column **num** including duplicated ones.
Can you write a SQL query to find the biggest number, which only appears once.

```
+---+
|num|
+---+
| 8 |
| 8 |
| 3 |
| 3 |
| 1 |
| 4 |
| 5 |
| 6 |
```

For the sample data above, your query should return the following result:

```
+---+
|num|
+---+
| 6 |
```

Note:
If there is no such number, just output **null**.

Accepted 8,178 | Submissions 21,610

```sql
select(
  select num
  from number
  group by num
  having count(*) = 1
  order by num desc limit 1
) as num;
```

620. Not Boring Movies

Easy 163 179 Favorite Share

SQL Schema ›

X city opened a new cinema, many people would like to go to this cinema. The cinema also gives out a poster indicating the movies' ratings and descriptions.
Please write a SQL query to output movies with an odd numbered ID and a description that is not 'boring'. Order the result by rating.

For example, table `cinema`:

```
+---------+-----------+--------------+-----------
+
|   id    | movie     |  description |  rating
|
+---------+-----------+--------------+-----------
+
|   1     | War       |   great 3D   |   8.9
|
|   2     | Science   |   fiction    |   8.5
|
|   3     | irish     |   boring     |   6.2
|
|   4     | Ice song  |   Fantacy    |   8.6
|
|   5     | House card|   Interesting|   9.1
|
+---------+-----------+--------------+-----------
+
```

For the example above, the output should be:

```
+---------+-----------+-------------+-----------
+
|   id    | movie     |  description |  rating
|
+---------+-----------+-------------+-----------
+
|   5     | House card|   Interesting|   9.1
|
|   1     | War       |   great 3D   |   8.9
|
+---------+-----------+-------------+-----------
+
```

Accepted 47,501 | Submissions 78,166

```sql
# Write your MySQL query statement below
SELECT *
FROM cinema
WHERE (id % 2 = 1) AND (description <> 'boring')
ORDER BY rating DESC
```

627. Swap Salary

Easy 254 176 Favorite Share

SQL Schema ›

Given a table `salary`, such as the one below, that has m=male and f=female values. Swap all f and m values (i.e., change all f values to m and vice versa) with a single update query and no intermediate temp table.

For example:

```
| id | name | sex | salary |
|----|------|-----|--------|
| 1  | A    | m   | 2500   |
| 2  | B    | f   | 1500   |
| 3  | C    | m   | 5500   |
| 4  | D    | f   | 500    |
```

For the sample data above, the output is:

```
+------+------------+-----------+
| id   | date       | people    |
+------+------------+-----------+
| 5    | 2017-01-05 | 145       |
| 6    | 2017-01-06 | 1455      |
| 7    | 2017-01-07 | 199       |
| 8    | 2017-01-08 | 188       |
+------+------------+-----------+
```

Note:
Each day only have one row record, and the dates are increasing with id increasing.

Accepted 10,631 | Submissions 30,404

```
update salary set sex = CHAR(ASCII('f') ^ ASCII('m') ^ ASCII(sex));
```

```
update salary set sex= CHAR(ASCII('f') + ASCII('m') - ASCII(sex));
```