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     |
```

```
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 |
+----+------+
```

```
# 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

```
# 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      |
+----------+
```

```sql
# 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

```
# 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       |
+-----------+
```

```
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

```
# 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

```
# 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|
+----+-----------+-----------+---------+---------
----------+----------+
```

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

```
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

```
# 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

```
# 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              |
```

```
# 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;
```