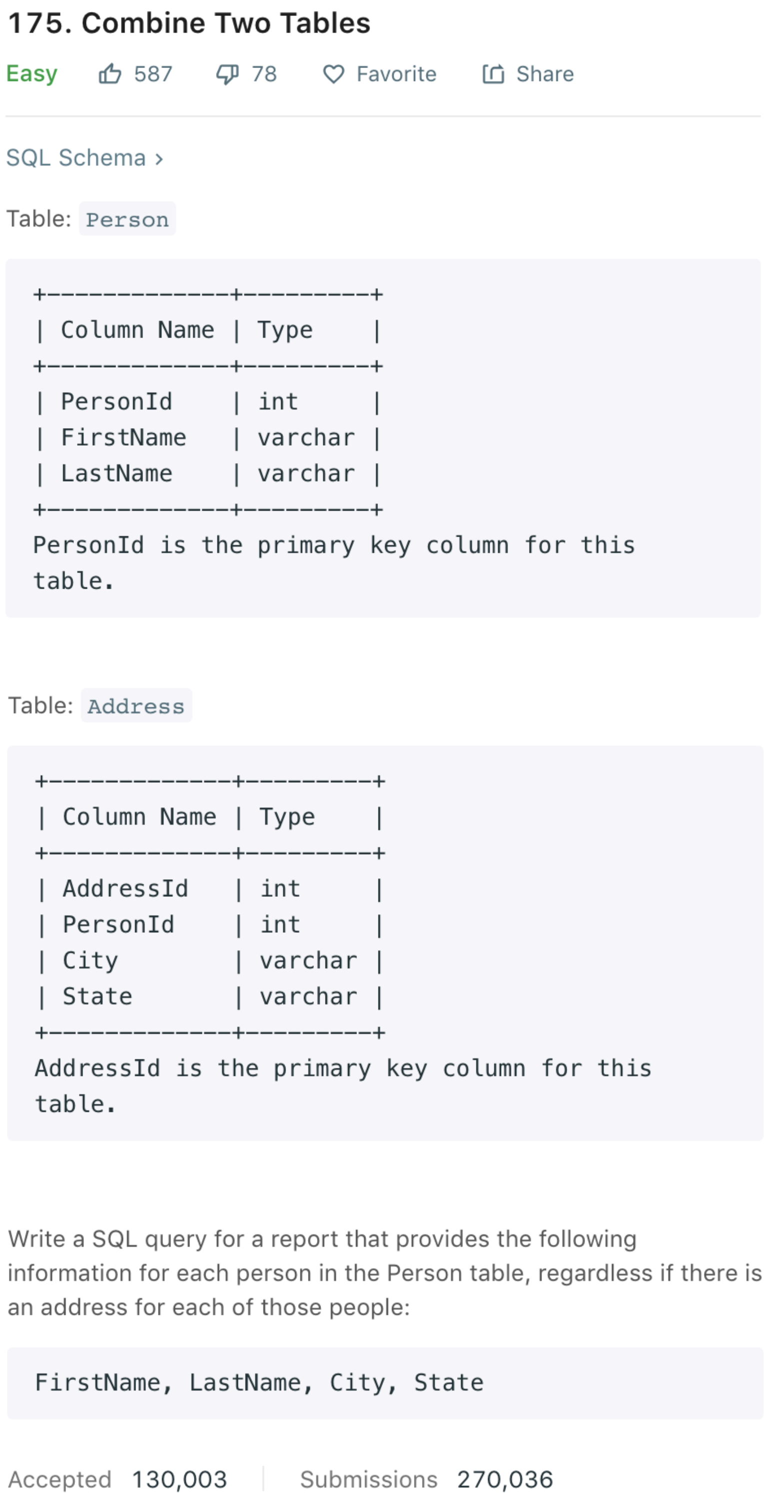
Leetcode SQL



175. Combine Two Tables

SQL Schema ›

Table: `Person`

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

Table: `Address`

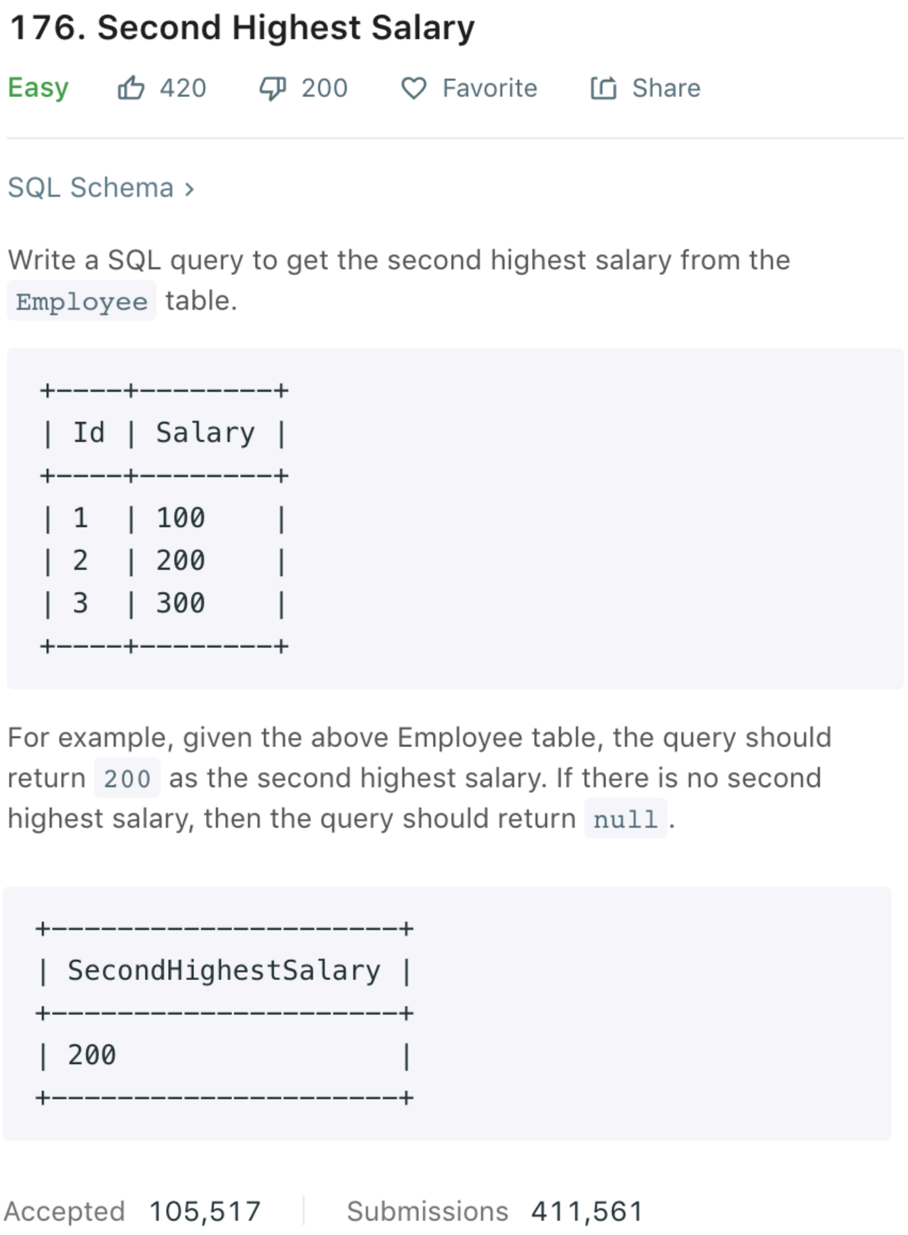
```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| AddressId   | int     |
| PersonId    | int     |
| City        | varchar |
| State       | varchar |
+-------------+---------+
AddressId is the primary key column for this
table.
```

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

```sql
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.

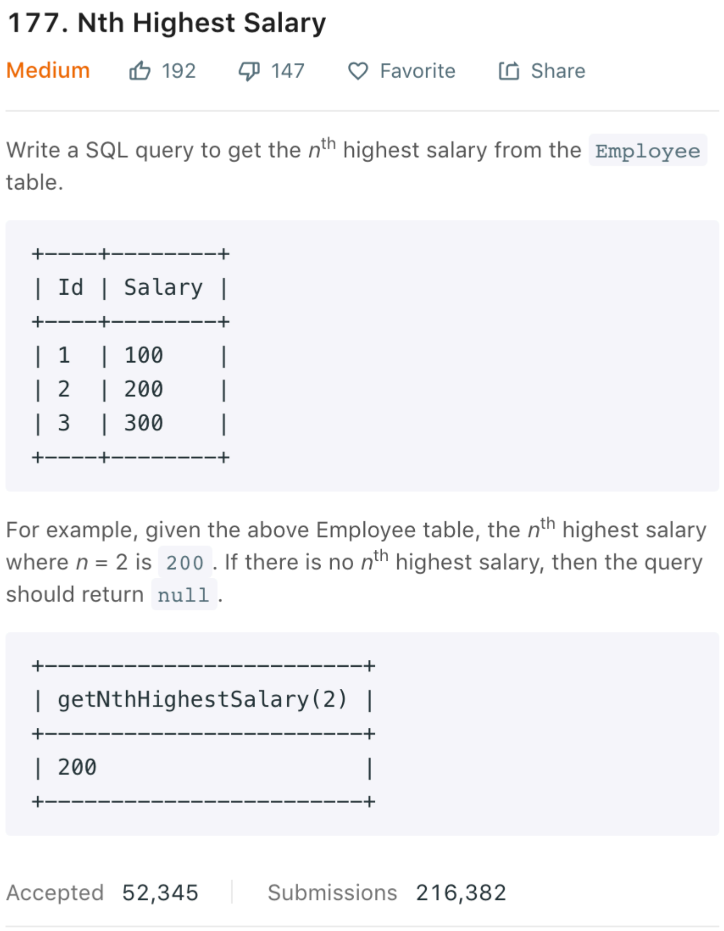
```
+----+--------+
| Id | Salary |
+----+--------+
| 1  | 100    |
| 2  | 200    |
| 3  | 300    |
+----+--------+
```

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

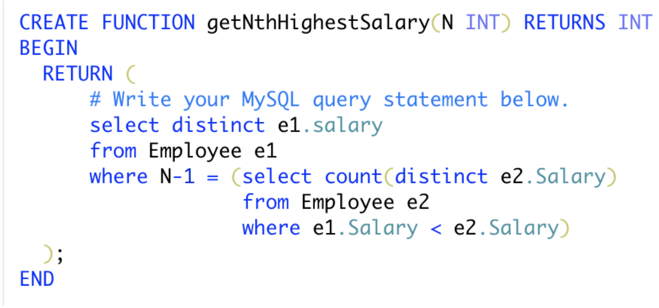
Write a SQL query to get the n^{th} highest salary from the `Employee` table.

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

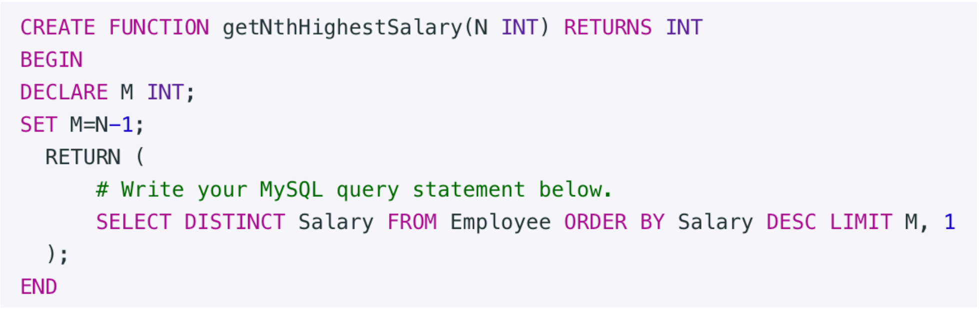
For example, given the above Employee table, the n^{th} highest salary where $n = 2$ is `200`. If there is no n^{th} highest salary, then the query should return `null`.

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

Accepted 52,345 | Submissions 216,382



```
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
```



```
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.

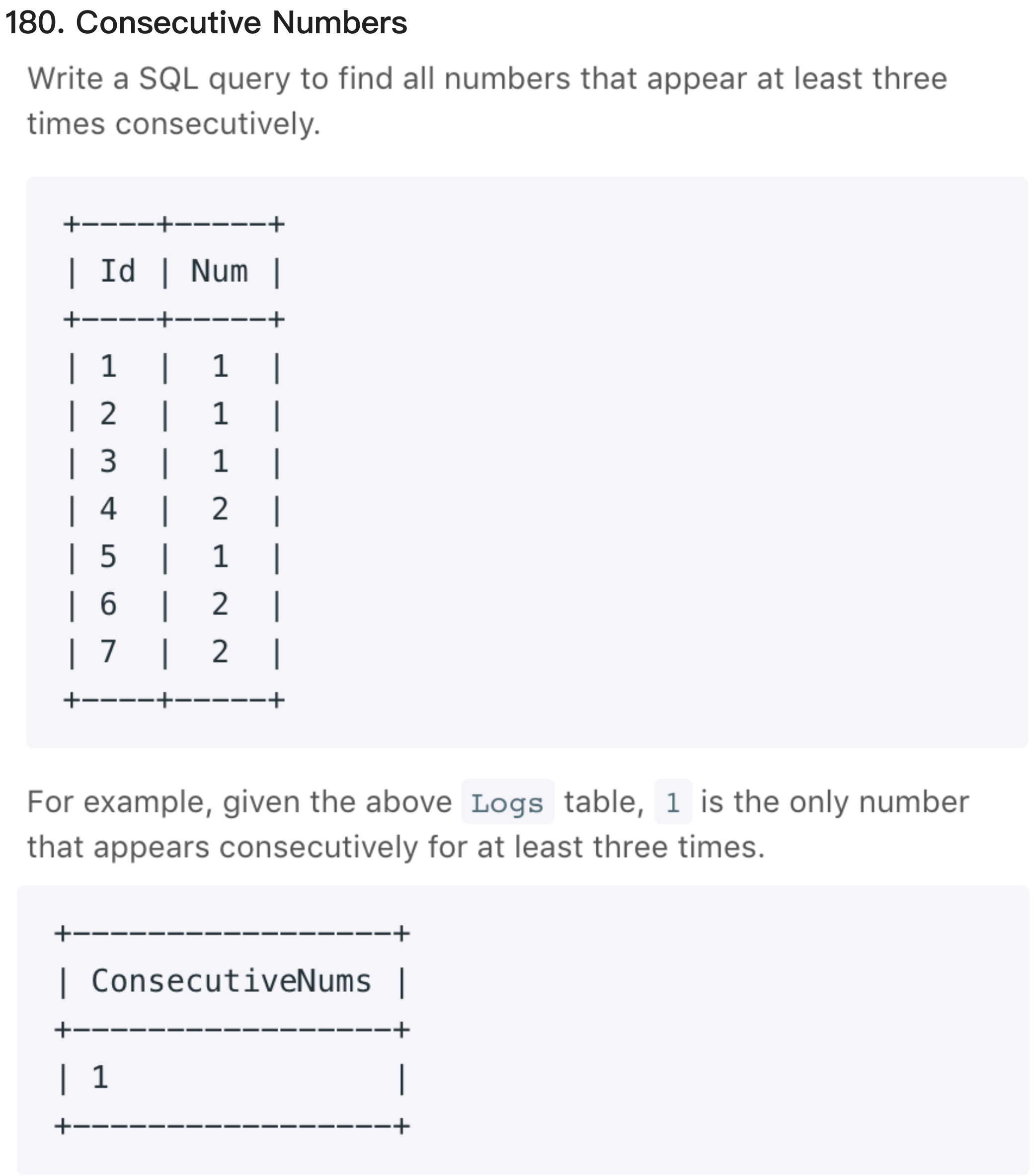
```
| seat_id |
|---------|
| 3       |
| 4       |
| 5       |
```

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.

```
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;
```

```
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.

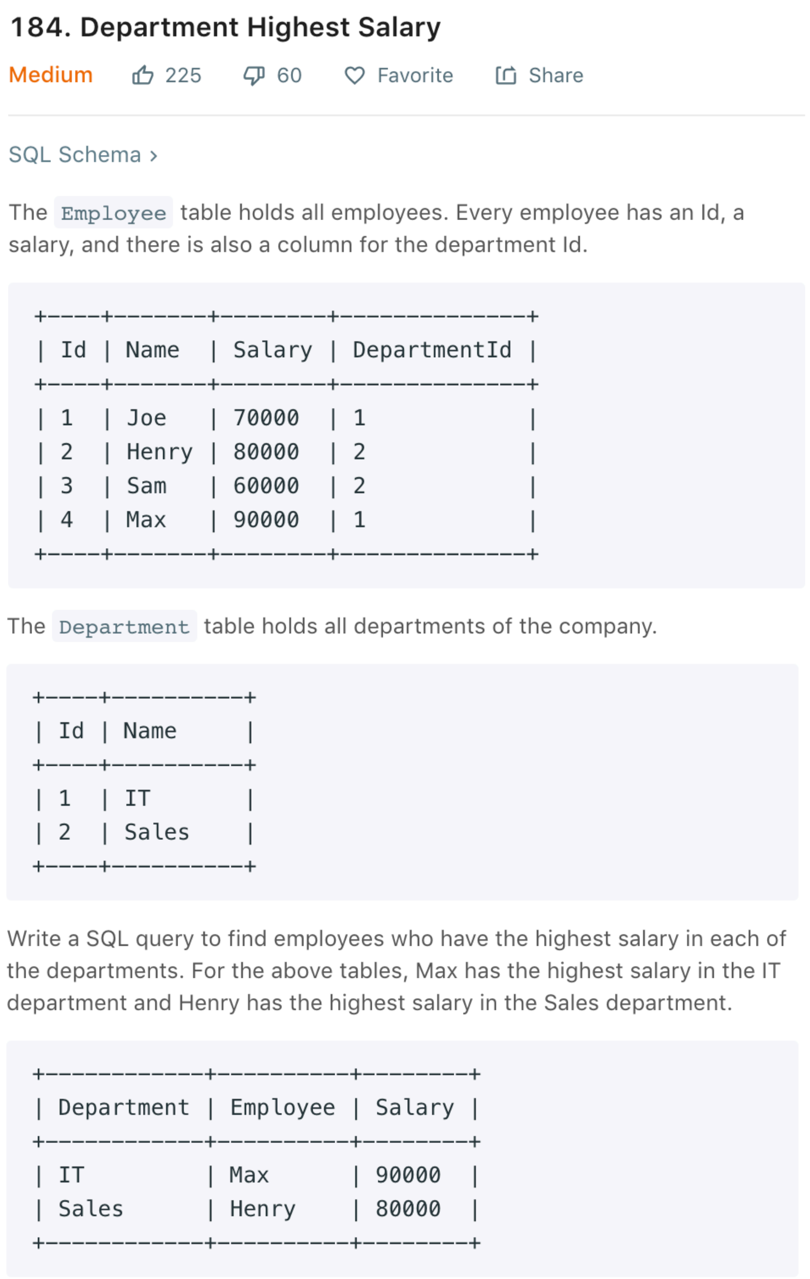
```
+----+-----+
| Id | Num |
+----+-----+
| 1  |  1  |
| 2  |  1  |
| 3  |  1  |
| 4  |  2  |
| 5  |  1  |
| 6  |  2  |
| 7  |  2  |
+----+-----+
```

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

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.

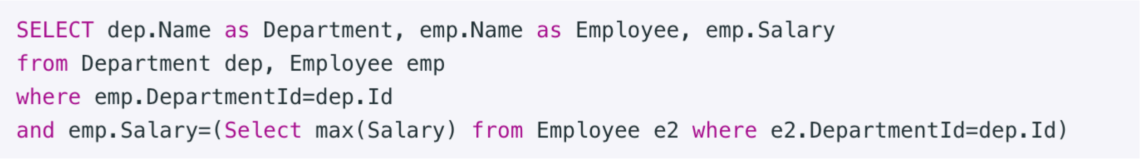
```
+----+-------+--------+--------------+
| Id | Name  | Salary | DepartmentId |
+----+-------+--------+--------------+
| 1  | Joe   | 70000  | 1            |
| 2  | Henry | 80000  | 2            |
| 3  | Sam   | 60000  | 2            |
| 4  | Max   | 90000  | 1            |
+----+-------+--------+--------------+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

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

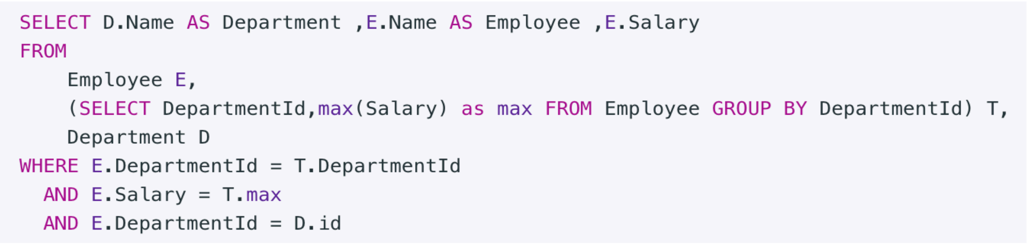
```
+----+----------+
| Id | Name     |
+----+----------+
| 1  | IT       |
| 2  | Sales    |
+----+----------+
```

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

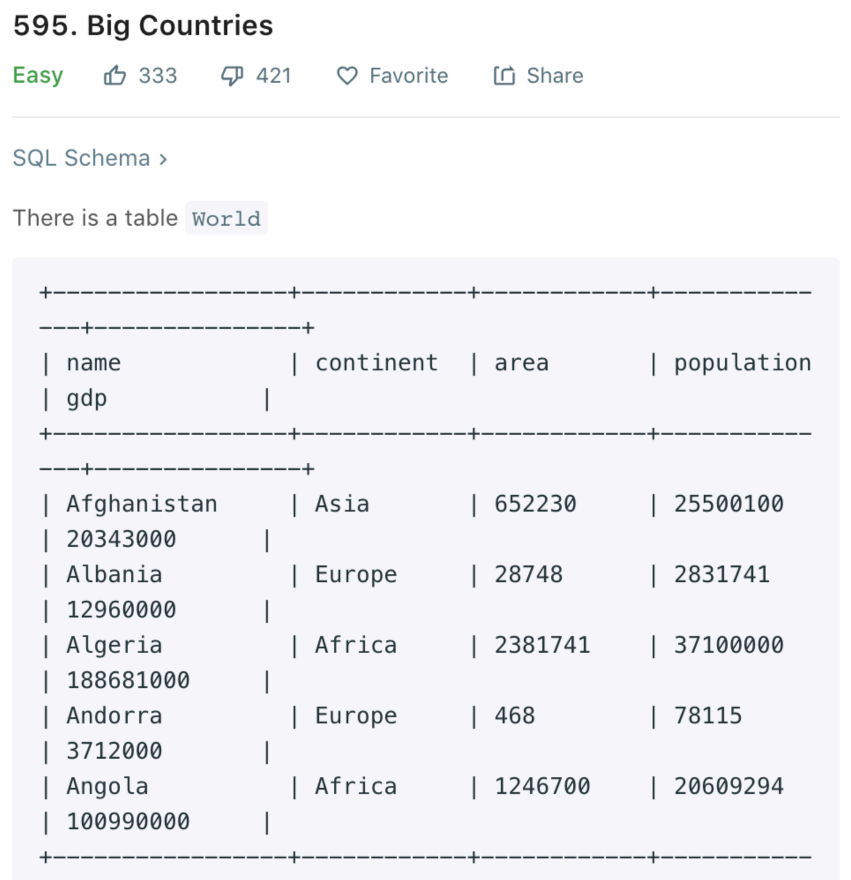
The `Department` table holds all departments of the company.

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

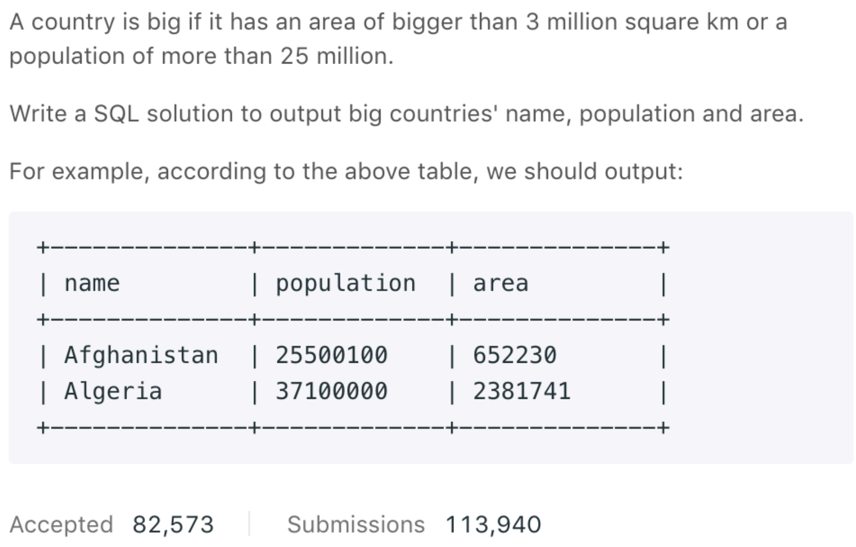


595. Big Countries

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.

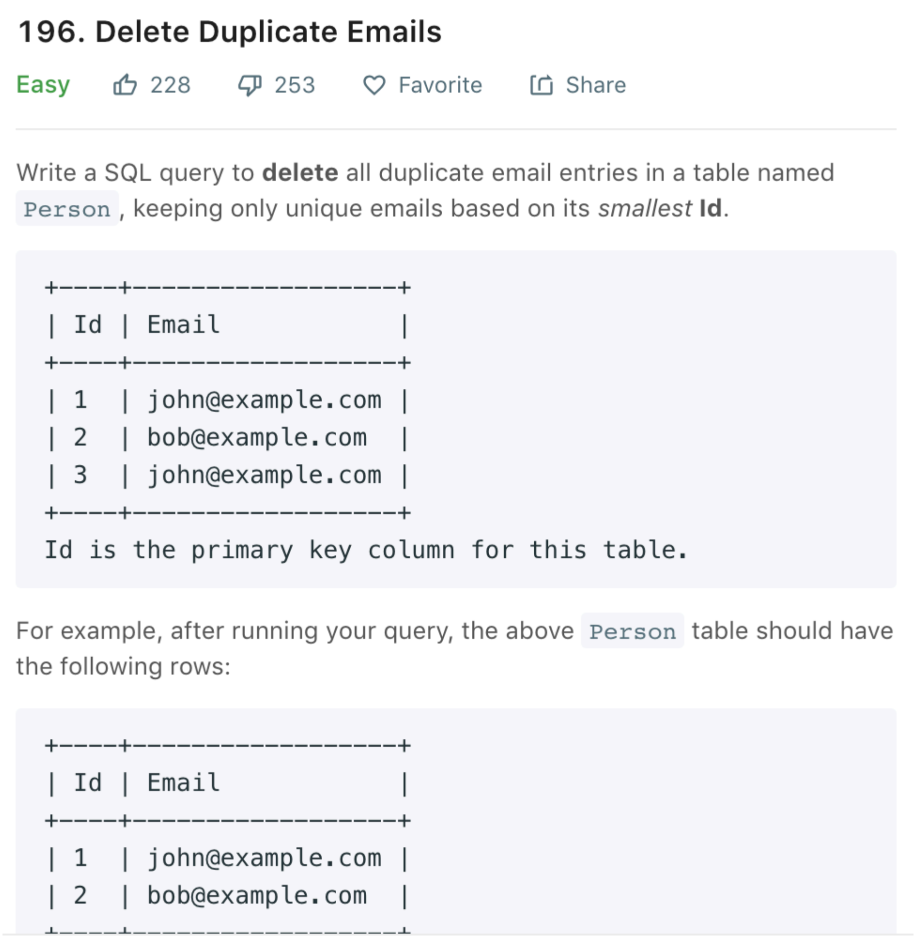
Write a SQL solution to output big countries' name, population and area.

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

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



196. Delete Duplicate Emails

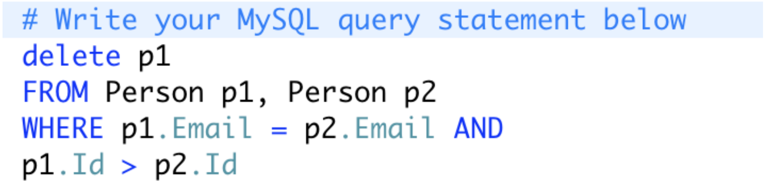
Easy 228 253 Favorite Share

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



```
# 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...

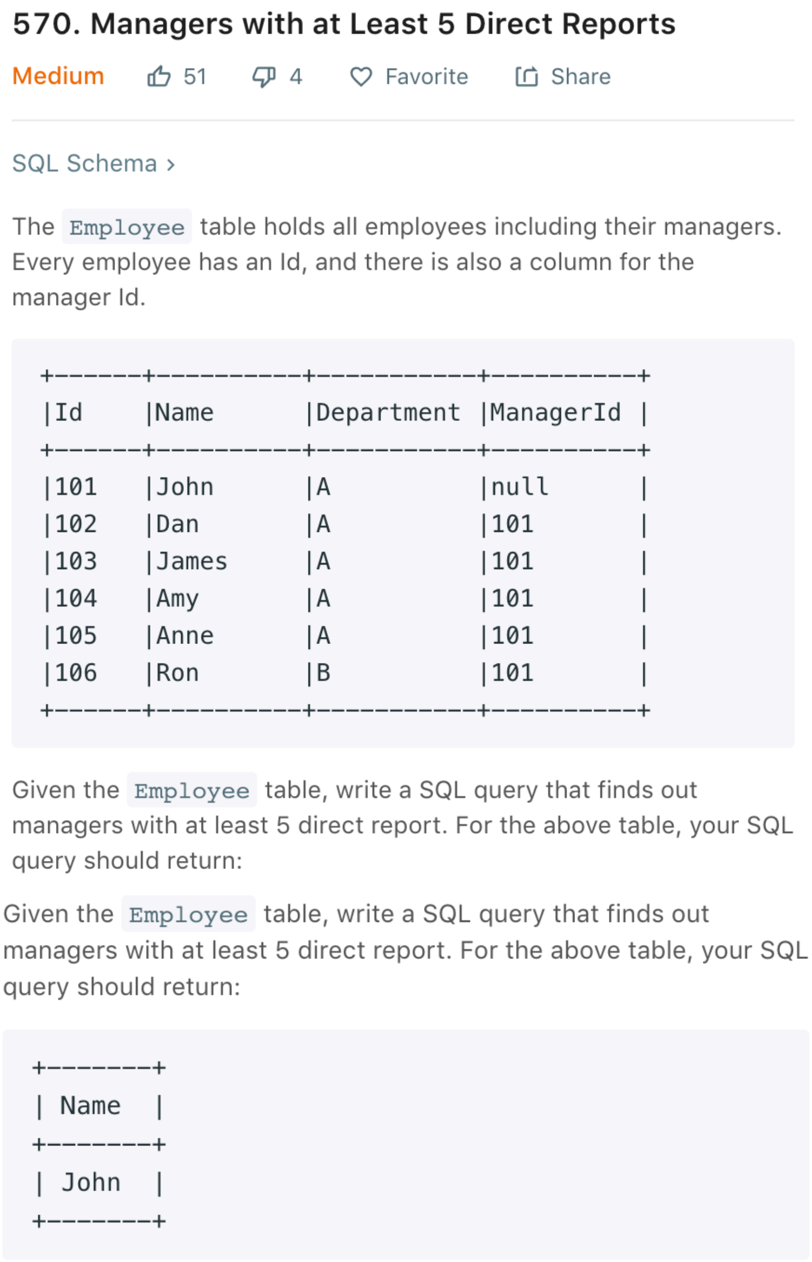
The average salary for department '1' is 9000, which is the salary of **employee_id** '1' since there is only one employee in this department. So the comparison result is 'higher' since 9000 > 8333.33 obviously.

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 ›

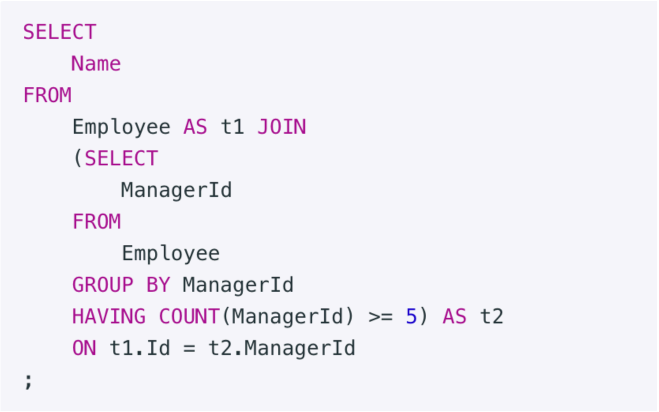
The `Employee` table holds all employees including their managers. Every employee has an Id, and there is also a column for the manager Id.

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