LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

578. Get Highest Answer Rate Question

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.

```sql
# Write your MySQL query statement below

WITH cte AS
(SELECT question_id, SUM(CASE WHEN action
= 'answer' THEN 1 ELSE 0 END)/ SUM(CASE
WHEN action = 'show' THEN 1 ELSE 0 END) AS
answer_rate
FROM SurveyLog
GROUP BY question_id)

SELECT question_id AS survey_log
FROM cte
ORDER BY answer_rate DESC, question_id
LIMIT 1
```

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

577. Employee Bonus

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  |
```

```sql
# Write your MySQL query statement below

SELECT e.name, b.bonus
FROM Employee e
LEFT JOIN Bonus b
ON e.empId = b.empId
WHERE b.bonus < 1000
OR b.bonus IS NULL
```


574. Winning Candidate

Table: `Candidate`

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

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`.

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

```
+------+
| Name |
+------+
| B    |
+------+
```

Notes:

1. You may assume **there is no tie**, in other words there will be **at most one** winning candidate.

```mysql
# Write your MySQL query statement below

WITH cte AS
(SELECT candidateId
FROM Vote
GROUP BY candidateId
ORDER BY COUNT(id) DESC
LIMIT 1)

SELECT name
FROM Candidate
WHERE id IN (SELECT candidateId FROM cte)
```

570. Managers with at Least 5 Direct Reports

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:

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

Note:
No one would report to himself.

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

MySQL • Auto

```mysql
# Write your MySQL query statement below
SELECT e2.name
FROM Employee e1
INNER JOIN Employee e2
ON e1.managerId = e2.id
GROUP BY e2.id, e2.name
HAVING COUNT(e1.id) >= 5;
```

569. Median Employee Salary

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.

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

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```sql
# Write your MySQL query statement below

WITH cte AS
(SELECT *, ROW_NUMBER() OVER(PARTITION BY company
ORDER BY salary) AS rnk, COUNT(id) OVER(PARTITION BY
company) AS n
FROM Employee)

SELECT id, company, salary
FROM cte
WHERE rnk BETWEEN n/2 AND (n/2)+1
```

550. Game Play Analysis IV

Table: `Activity`

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| player_id    | int     |
| device_id    | int     |
| event_date   | date    |
| games_played | int     |
+--------------+---------+
(player_id, event_date) is the primary key of this table.
This table shows the activity of players of some game.
Each row is a record of a player who logged in and played a number of games (possibly 0) before logging out on some day using some device.
```

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

The query result format is in the following example:

```
Activity table:
+-----------+-----------+------------+--------------+
| player_id | device_id | event_date | games_played |
+-----------+-----------+------------+--------------+
| 1         | 2         | 2016-03-01 | 5            |
| 1         | 2         | 2016-03-02 | 6            |
| 2         | 3         | 2017-06-25 | 1            |
| 3         | 1         | 2016-03-02 | 0            |
| 3         | 4         | 2018-07-03 | 5            |
+-----------+-----------+------------+--------------+

Result table:
+-----------+
| fraction  |
+-----------+
| 0.33      |
+-----------+
Only the player with id 1 logged back in after the first day he had logged in so the answer is 1/3 = 0.33
```

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

```sql
# Write your MySQL query statement below
SELECT ROUND(COUNT(DISTINCT T.player_id)/COUNT(DISTINCT a.player_id),2)
AS fraction
FROM Activity a
LEFT JOIN
(SELECT player_id, MIN(event_date) AS FLD
FROM Activity
GROUP BY player_id) T
ON a.player_id = T.player_id
```

534. Game Play Analysis III

Table: `Activity`

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| player_id    | int     |
| device_id    | int     |
| event_date   | date    |
| games_played | int     |
+--------------+---------+
(player_id, event_date) is the primary key of this table.
This table shows the activity of players of some game.
Each row is a record of a player who logged in and played a number of games (possibly 0) before logging out on some day using some device.
```

Write an SQL query that reports for each player and date, how many games played **so far** by the player. That is, the total number of games played by the player until that date. Check the example for clarity.

The query result format is in the following example:

```
Activity table:
+-----------+-----------+------------+--------------+
| player_id | device_id | event_date | games_played |
+-----------+-----------+------------+--------------+
| 1         | 2         | 2016-03-01 | 5            |
| 1         | 2         | 2016-05-02 | 6            |
| 1         | 3         | 2017-06-25 | 1            |
| 3         | 1         | 2016-03-02 | 0            |
| 3         | 4         | 2018-07-03 | 5            |
+-----------+-----------+------------+--------------+

Result table:
+-----------+------------+---------------------+
| player_id | event_date | games_played_so_far |
+-----------+------------+---------------------+
| 1         | 2016-03-01 | 5                   |
| 1         | 2016-05-02 | 11                  |
| 1         | 2017-06-25 | 12                  |
| 3         | 2016-03-02 | 0                   |
| 3         | 2018-07-03 | 5                   |
+-----------+------------+---------------------+
For the player with id 1, 5 + 6 = 11 games played by 2016-05-02, and 5 + 6 + 1 = 12 games played by 2017-06-25.
For the player with id 3, 0 + 5 = 5 games played by 2018-07-03.
Note that for each player we only care about the days when the player logged in.
```


LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```sql
# Write your MySQL query statement below

SELECT player_id, event_date,SUM(games_played)
OVER(PARTITION BY player_id ORDER BY event_date
ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
AS games_played_so_far
FROM Activity
```

512. Game Play Analysis II

Table: `Activity`

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| player_id    | int     |
| device_id    | int     |
| event_date   | date    |
| games_played | int     |
+--------------+---------+
(player_id, event_date) is the primary key of this table.
This table shows the activity of players of some game.
Each row is a record of a player who logged in and played a number of games (possibly 0) before logging out on some day using some device.
```

Write a SQL query that reports the **device** that is first logged in for each player.

Write a SQL query that reports the **device** that is first logged in for each player.

The query result format is in the following example:

```
Activity table:
+-----------+-----------+------------+--------------+
| player_id | device_id | event_date | games_played |
+-----------+-----------+------------+--------------+
| 1         | 2         | 2016-03-01 | 5            |
| 1         | 2         | 2016-05-02 | 6            |
| 2         | 3         | 2017-06-25 | 1            |
| 3         | 1         | 2016-03-02 | 0            |
| 3         | 4         | 2018-07-03 | 5            |
+-----------+-----------+------------+--------------+

Result table:
+-----------+-----------+
| player_id | device_id |
+-----------+-----------+
| 1         | 2         |
| 2         | 3         |
| 3         | 1         |
+-----------+-----------+
```

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```sql
# Write your MySQL query statement below

WITH cte AS
(SELECT *, ROW_NUMBER() OVER(PARTITION BY
player_id ORDER BY event_date) AS rnk
FROM Activity)

SELECT player_id, device_id
FROM cte
WHERE rnk = 1
```

511. Game Play Analysis I

Table: `Activity`

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| player_id    | int     |
| device_id    | int     |
| event_date   | date    |
| games_played | int     |
+--------------+---------+
(player_id, event_date) is the primary key of this table.
This table shows the activity of players of some game.
Each row is a record of a player who logged in and played a number of games (possibly 0) before logging out on some day using some device.
```

Write an SQL query that reports the **first login date** for each player.

Write an SQL query that reports the **first login date** for each player.

The query result format is in the following example:

```
Activity table:
+-----------+-----------+------------+--------------+
| player_id | device_id | event_date | games_played |
+-----------+-----------+------------+--------------+
| 1         | 2         | 2016-03-01 | 5            |
| 1         | 2         | 2016-05-02 | 6            |
| 2         | 3         | 2017-06-25 | 1            |
| 3         | 1         | 2016-03-02 | 0            |
| 3         | 4         | 2018-07-03 | 5            |
+-----------+-----------+------------+--------------+

Result table:
+-----------+-------------+
| player_id | first_login |
+-----------+-------------+
| 1         | 2016-03-01  |
| 2         | 2017-06-25  |
| 3         | 2016-03-02  |
+-----------+-------------+
```

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

```sql
# Write your MySQL query statement below

SELECT player_id, MIN(event_date) AS first_login
FROM Activity
GROUP BY player_id
```

197. Rising Temperature

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 |
+----+
```

```sql
# Write your MySQL query statement below
SELECT w1.id
FROM Weather w1
LEFT JOIN Weather w2
ON w1.recordDate - INTERVAL 1 DAY = w2.recordDate
WHERE w1.temperature > w2.temperature;
```


LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

196. Delete Duplicate Emails

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.

```sql
# Write your MySQL query statement below
DELETE p1.*
FROM Person p1
CROSS JOIN Person p2
WHERE p1.email = p2.email
AND p1.id > p2.id
```

185. Department Top Three Salaries

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   | 85000  | 1            |
| 2  | Henry | 80000  | 2            |
| 3  | Sam   | 60000  | 2            |
| 4  | Max   | 90000  | 1            |
| 5  | Janet | 69000  | 1            |
| 6  | Randy | 85000  | 1            |
| 7  | Will  | 70000  | 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 (order of rows does not matter).

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

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 (order of rows does not matter).

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

Explanation:

In IT department, Max earns the highest salary, both Randy and Joe earn the second highest salary, and Will earns the third highest salary. There are only two employees in the Sales department, Henry earns the highest salary while Sam earns the second highest salary.

```sql
With T as (Select name,
salary,
departmentId,
Dense_rank() over (partition by departmentId order by salary desc) as ranking
from Employee)

Select D.name as Department,
    T.name as Employee,
    Salary
from T join Department D on T.departmentId = D.id
where ranking <4
```

184. Department Highest Salary

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  | Jim   | 90000  | 1            |
| 3  | Henry | 80000  | 2            |
| 4  | Sam   | 60000  | 2            |
| 5  | 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, your SQL query should return the following rows (order of rows does not matter).

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

Write a SQL query to find employees who have the highest salary in each of the departments. For the above tables, your SQL query should return the following rows (order of rows does not matter).

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

Explanation:

Max and Jim both have the highest salary in the IT department and Henry has the highest salary in the Sales department.

```sql
# Write your MySQL query statement below

WITH cte AS
(SELECT e.name AS Employee, e.salary, d.name AS
department,MAX(e.salary) OVER(PARTITION BY
e.departmentId) AS max_salary
FROM Employee e
LEFT JOIN Department d
ON e.departmentId = d.id)

SELECT department, employee, salary
FROM cte
WHERE salary = max_salary
```

183. Customers Who Never Order

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

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

Using the above tables as example, return the following:

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

```sql
# Write your MySQL query statement below
SELECT c.name AS Customers
FROM Customers c
LEFT JOIN Orders o
ON c.id = o.customerId
WHERE o.id IS NULL
```

182. Duplicate Emails

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.

```sql
# Write your MySQL query statement below
SELECT email
FROM Person
GROUP BY email
HAVING COUNT(DISTINCT id) > 1
```

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

181. Employees Earning More Than Their Managers

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 e1.name AS Employee
FROM Employee e1
LEFT JOIN Employee e2
ON e1.managerId = e2.id
WHERE e1.salary > e2.salary
```

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

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```sql
# Write your MySQL query statement below

WITH cte AS
(SELECT *, LEAD(num,1) OVER() AS next_1,
LEAD(num,2) OVER() AS next_2
FROM logs)

SELECT DISTINCT num AS ConsecutiveNums
FROM cte
WHERE num = next_1
AND num = next_2
```

178. Rank Scores

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

```sql
# Write your MySQL query statement below

SELECT score, DENSE_RANK() OVER(ORDER BY Score
DESC) AS 'rank'
FROM Scores
ORDER BY score DESC
```

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

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

```sql
CREATE FUNCTION getNthHighestSalary(N INT)
RETURNS INT
BEGIN
  RETURN (
      # Write your MySQL query statement below.
      WITH cte AS
      (SELECT *, DENSE_RANK() OVER(ORDER BY Salary DESC) AS rnk
      FROM Employee)

      SELECT DISTINCT IFNULL(salary,null)
      FROM cte
      WHERE rnk = N
  );
END
```

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

176. Second Highest Salary

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

```mysql
# Write your MySQL query statement below

WITH cte AS
(SELECT *, DENSE_RANK() OVER(ORDER BY salary DESC)
AS r
FROM Employee)

SELECT IFNULL((SELECT salary FROM cte WHERE r = 2
LIMIT 1), null) AS SecondHighestSalary
```

175. Combine Two Tables

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.
```

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

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
```

```sql
# Write your MySQL query statement below
SELECT p.firstName, p.lastName, a.city, a.state
FROM Person p
LEFT JOIN Address a
ON p.personId = a.personId
```

580. Count Student Number in Departments

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.

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              |
```

```sql
# Write your MySQL query statement below

WITH cte AS
(SELECT d.*, s.student_id
FROM Department d
LEFT JOIN Student s
ON d.dept_id = s.dept_id)

SELECT dept_name, CASE WHEN student_id IS NOT NULL
THEN COUNT(*) ELSE 0 END AS student_number
FROM cte
GROUP BY dept_name
ORDER BY student_number DESC, dept_name
```


LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

584. Find Customer Referee

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

```mysql
# Write your MySQL query statement below

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

585. Investments in 2016

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.

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

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.

```sql
# Write your MySQL query statement below

WITH cte AS
(SELECT CONCAT(lat,',',lon) AS location
FROM Insurance
GROUP BY lat, lon
HAVING COUNT(pid) > 1),

cte2 AS
(SELECT DISTINCT I1.*
FROM Insurance I1
LEFT JOIN Insurance I2
ON I1.tiv_2015 = I2.tiv_2015
WHERE I1.pid <> I2.pid
AND CONCAT(I1.lat,',',I1.lon) NOT IN (SELECT location
FROM cte))

SELECT ROUND(SUM(tiv_2016),2) AS tiv_2016
FROM cte2
```

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

586. Customer Placing the Largest Number of Orders

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?*

```sql
# Write your MySQL query statement below

WITH cte AS
(SELECT customer_number, COUNT(order_number) AS NumOrd
FROM Orders
GROUP BY customer_number)

SELECT customer_number
FROM cte
WHERE NumOrd = (SELECT Max(NumOrd) FROM cte)
```

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

595. Big Countries

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

```mysql
# Write your MySQL query statement below

SELECT name, population, area
FROM World
WHERE area >= 3000000
OR population >= 25000000
```

596. Classes More Than 5 Students

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.

```sql
# Write your MySQL query statement below

SELECT class
FROM Courses
GROUP BY class
HAVING COUNT(student) >= 5
```

597. Friend Requests I: Overall Acceptance Rate

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

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

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.

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?

```
88  Select ISNULL(Round(
89  (Select Cast (Count(*) as float)
90  from (Select requester_id,accepter_id
91  from request_accepted_597
92  group by requester_id,accepter_id) R2)
93  /
94  (Select Cast (Count(*) as float)
95  from (Select sender_id,send_to_id
96  from friend_request_597
97  group by sender_id,send_to_id) R1)
98  ,2),0.00)
99  as accept_rate
100
```

100 %

Results | Messages

	accept_rate
1	0.8

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

602. Friend Requests II: Who Has the Most Friends

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.

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?

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```sql
# Write your MySQL query statement below

WITH cte AS
(SELECT requester_id AS id, accepter_id
FROM RequestAccepted
UNION
SELECT accepter_id AS id, requester_id
FROM RequestAccepted)

SELECT id, COUNT(DISTINCT accepter_id) AS num
FROM cte
GROUP BY id
ORDER BY num DESC
LIMIT 1;
```

```sql
Select top(1)accepter_id as id, Sum(friends) as 'num'
from
(Select accepter_id , Count(*) as friends
from request_accepted_602
group by accepter_id
Union all
Select requester_id as accepter_id, Count(*) as friends
from request_accepted_602
group by requester_id) Result

group by accepter_id
order by num desc
```

Results | Messages

id	num
3	3

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

603. Consecutive Available Seats

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 W.seat_id from (select *,
lead(free) over (order by seat_id) as nextseat,
lag(free) over (order by seat_id) as prevseat
from cinema_603) as W
where W.free=1 AND W.nextseat=1
OR W.free=1 and W.prevseat=1
order by seat_id;
```

100 %

Results | Messages

	seat_id
1	3
2	4
3	5

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

607. Sales Person

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`

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 | order_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 |
```


```
106
107  Select name
108  from salesperson_607
109  where sales_id NOT IN (select Distinct s.sales_id
110  from orders_607 o
111  left join company_607 c
112  on o.com_id = c.com_id
113  left join salesperson_607 s
114  on o.sales_id = s.sales_id
115  where c.name='RED')
116
```

100 %

Results | Messages

	name
1	Amy
2	Mark
3	Alex

608. Tree Node

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.

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

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

Explanation

- Node '1' is root node, because its parent node is NULL and it has child node '2' and '3'.
- Node '2' is inner node, because it has parent node '1' and child node '4' and '5'.
- Node '3', '4' and '5' is Leaf node, because they have parent node and they don't have child node.
- And here is the image of the sample tree as below:

- And here is the image of the sample tree as below:


```
          1
        /   \
      2       3
    /   \
  4       5
```

Note

If there is only one node on the tree, you only need to output its root attributes.

```sql
Select id, case when p_id is null then 'Root'
when p_id  not in (select p_id from tree_608 where p_id is not null) then 'leaf'
else 'Inner' end as type
from tree_608
order by id;
```

100 %

Results | Messages

	id	type
1	1	Root
2	2	Inner
3	3	Inner
4	4	Inner
5	5	Inner

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

610. Triangle Judgement

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      |
```

```
34  select *,case when x+y>z and y+z>x and z+x>y then 'Yes' else 'No' end as triangle
35  from triangle_610
36
```

100 %

Results | Messages

	x	y	z	triangle
1	13	15	30	No
2	10	20	15	Yes

612. Shortest Distance in a Plane

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.

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

```
with cte as(
select *,ROW_NUMBER() over (order by x,y) as rnk
from point_2d_612)
select round(min(sqrt(power((c1.x-c2.x),2) + power((c1.y-c2.y),2))),2) as shortest
from cte c1
inner join cte c2
on c1.rnk< c2.rnk;
```

100 %

Results Messages

	shortest
1	1

613. Shortest Distance in a Line

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`.

```
select min(ABS(p1.x-p2.x)) as shortest
from point_613 as p1
cross join point_613 as p2
where p1.x<>p2.x;

select min(ABS(p1.x-p2.x)) as shortest
```

100 %

Results Messages

	shortest
1	1

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

614. Second Degree Follower

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.

Difficulty:

Medium

```
47
48      select followee as follower ,count(*) as num
49      from follow_614
50      group by followee
51      having followee in (select distinct follower from follow_614)
52
```

100 %

Results | Messages

	follower	num
1	B	2
2	D	1

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

615. Average Salary: Departments VS Company

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.

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```sql
Select SUBSTRING(Cast(S.pay_date as varchar),1,7) as pay_month,
    S.department_id as department_id,
    CASE WHEN S.dept_avg > T.company_avg THEN 'higher'
        WHEN S.dept_avg = T.company_avg THEN 'same'
        ELSE 'lower'
        END as comparison
from
(select temp.department_id,temp.pay_date,avg(amount) as dept_avg from (select s.*,e.department_id
from salary_615 s
left join employee_615 e
on s.employee_id = e.employee_id) temp
group by temp.department_id,temp.pay_date) S left join (
Select pay_date,avg(amount) as company_avg
from salary_615
group by pay_date) T
on S.pay_date = T.pay_date
order by S.pay_date Desc;
```

00 %

Results | Messages

	pay_month	department_id	comparison
1	2017-03	1	higher
2	2017-03	2	lower
3	2017-02	1	same
4	2017-02	2	same

619. Biggest Single Number

Table `my_numbers` 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**.

Difficulty:

Easy

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

```
45  with cte as (select max(num) as num
46  from my_numbers_619
47  group by num
48  having count(*)=1)
49  select max(num) as num from cte;
50
```

100 %

Results | Messages

	num
1	6

```
1   # Write your MySQL query statement below
2
3   WITH cte AS
4   (SELECT num
5   FROM MyNumbers
6   GROUP BY num
7   HAVING COUNT(num) = 1)
8
9   SELECT CASE WHEN COUNT(*) > 0 THEN MAX(num)
10  ELSE NULL END AS num
11  FROM cte
```


LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

620. Not Boring Movies

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

```
45
46  select id,movie,description,rating
47  from cinema_620
48  where id%2<>0 and description<>'boring'
49  order by rating desc;
50
51
```

100 %

Results | Messages

	id	movie	description	rating
1	5	House card	Interesting	9.1
2	1	War	great 3D	8.9

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

626. Exchange Seats

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.

Difficulty:

Medium

```
54  WITH cte as (SELECT *,
55        lead(id) OVER (ORDER BY  id) as Next,
56        lag(id) OVER (ORDER BY  id) as Prev
57  from seat_626)
58  SELECT CASE WHEN ((id%2=1) and Next is not null)
59  THEN Next
60  WHEN ((id%2=0)) THEN prev
61  ELSE id END AS id, student
62  FROM cte
63  ORDER BY id;
64
```

100 %

Results | Messages

	id	student
1	1	Doris
2	2	Abbot
3	3	Green
4	4	Emerson
5	5	Jeames

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

627. Swap Salary

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 statement** and no intermediate temp table.

Note that you must write a single update statement, **DO NOT** write any select statement for this problem.

Example:

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

After running your **update** statement, the above salary table should have the following rows:

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

Difficulty:

```
39  UPDATE salary_627
40  SET sex = CASE
41              WHEN sex = 'm' THEN 'f'
42              WHEN sex = 'f' THEN 'm'
43            END;
44  select * from salary_627
```

100 %

Results | Messages

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

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1045. Customers Who Bought All Products

Table: `Customer`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| customer_id | int     |
| product_key | int     |
+-------------+---------+
product_key is a foreign key to Product table.
```

Table: `Product`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| product_key | int     |
+-------------+---------+
product_key is the primary key column for this table.
```

Write an SQL query for a report that provides the customer ids from the `Customer` table that bought all the products in the `Product` table.

For example:

```
Customer table:
+-------------+-------------+
| customer_id | product_key |
+-------------+-------------+
| 1           | 5           |
| 2           | 6           |
| 3           | 5           |
| 3           | 6           |
| 1           | 6           |
+-------------+-------------+

Product table:
+-------------+
| product_key |
+-------------+
| 5           |
| 6           |
+-------------+

Result table:
+-------------+
| customer_id |
+-------------+
| 1           |
| 3           |
+-------------+
The customers who bought all the products (5 and 6) are customers with id 1 and 3.
```

```
80
81  Select customer_id
82  from Customer_1045
83  group by customer_id
84  having Count(distinct product_key)= (select count(distinct product_key) from  Product_1045)
85
86
87
88
89
```

100 %

Results | Messages

	customer_id
1	1
2	3

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1050. Actors and Directors Who Cooperated At Least Three Times

Table: `ActorDirector`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| actor_id    | int     |
| director_id | int     |
| timestamp   | int     |
+-------------+---------+
timestamp is the primary key column for this table.
```

Write a SQL query for a report that provides the pairs `(actor_id, director_id)` where the actor have cooperated with the director at least 3 times.

Example:

```
ActorDirector table:
+-------------+-------------+-------------+
| actor_id    | director_id | timestamp   |
+-------------+-------------+-------------+
| 1           | 1           | 0           |
| 1           | 1           | 1           |
| 1           | 1           | 2           |
| 1           | 2           | 3           |
| 1           | 2           | 4           |
| 2           | 1           | 5           |
| 2           | 1           | 6           |
+-------------+-------------+-------------+
```

```
Result table:
+-------------+-------------+
| actor_id    | director_id |
+-------------+-------------+
| 1           | 1           |
+-------------+-------------+
The only pair is (1, 1) where they cooperated exactly 3 times.
```

Difficulty:

Easy

```sql
select actor_id,
       director_id
from ActorDirector_1050
group by actor_id,director_id
having count(timestamp)>=3;
```

100 %

Results | Messages

	actor_id	director_id
1	1	1

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1068. Product Sales Analysis I

Table: `Sales`

```
+-------------+-------+
| Column Name | Type  |
+-------------+-------+
| sale_id     | int   |
| product_id  | int   |
| year        | int   |
| quantity    | int   |
| price       | int   |
+-------------+-------+
(sale_id, year) is the primary key of this table.
product_id is a foreign key to Product table.
Note that the price is per unit.
```

Table: `Product`

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| product_id   | int     |
| product_name | varchar |
+--------------+---------+
product_id is the primary key of this table.
```

Write an SQL query that reports all **product names** of the products in the `Sales` table along with their selling **year** and **price**.

For example:

```
Sales table:
+---------+------------+------+----------+-------+
| sale_id | product_id | year | quantity | price |
+---------+------------+------+----------+-------+
| 1       | 100        | 2008 | 10       | 5000  |
| 2       | 100        | 2009 | 12       | 5000  |
| 7       | 200        | 2011 | 15       | 9000  |
+---------+------------+------+----------+-------+

Product table:
+------------+--------------+
| product_id | product_name |
+------------+--------------+
| 100        | Nokia        |
| 200        | Apple        |
| 300        | Samsung      |
+------------+--------------+

Result table:
+--------------+-------+-------+
| product_name | year  | price |
+--------------+-------+-------+
| Nokia        | 2008  | 5000  |
| Nokia        | 2009  | 5000  |
| Apple        | 2011  | 9000  |
+--------------+-------+-------+
```

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

```
87  select distinct p.product_name,
88          s.year,
89          s.price
90  from Sales_1068 s
91  left join Product_1068 p
92  on s.product_id = p.product_id;
93
94
95
```

00 %

Results | Messages

	product_name	year	price
1	Apple	2011	9000
2	Nokia	2008	5000
3	Nokia	2009	5000

1069. Product Sales Analysis II

Table: `Sales`

```
+-------------+-------+
| Column Name | Type  |
+-------------+-------+
| sale_id     | int   |
| product_id  | int   |
| year        | int   |
| quantity    | int   |
| price       | int   |
+-------------+-------+
sale_id is the primary key of this table.
product_id is a foreign key to Product table.
Note that the price is per unit.
```

Table: `Product`

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| product_id   | int     |
| product_name | varchar |
+--------------+---------+
product_id is the primary key of this table.
```


LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

Write an SQL query that reports the total quantity sold for every product id.

The query result format is in the following example:

```
Sales table:
+---------+------------+------+----------+-------+
| sale_id | product_id | year | quantity | price |
+---------+------------+------+----------+-------+
| 1       | 100        | 2008 | 10       | 5000  |
| 2       | 100        | 2009 | 12       | 5000  |
| 7       | 200        | 2011 | 15       | 9000  |
+---------+------------+------+----------+-------+

Product table:
+------------+--------------+
| product_id | product_name |
+------------+--------------+
| 100        | Nokia        |
| 200        | Apple        |
| 300        | Samsung      |
+------------+--------------+

Result table:
+--------------+----------------+
| product_id   | total_quantity |
+--------------+----------------+
| 100          | 22             |
| 200          | 15             |
+--------------+----------------+
```

Difficulty:

Easy

```
85
86  select product_id,
87         sum(quantity) as total_quantity
88  from Sales_1069
89  group by product_id;
90
```

100 %

Results | Messages

	product_id	total_quantity
1	100	22
2	200	15

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1070. Product Sales Analysis III

Table: `Sales`

```
+-------------+-------+
| Column Name | Type  |
+-------------+-------+
| sale_id     | int   |
| product_id  | int   |
| year        | int   |
| quantity    | int   |
| price       | int   |
+-------------+-------+
sale_id is the primary key of this table.
product_id is a foreign key to Product table.
Note that the price is per unit.
```

Table: `Product`

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| product_id   | int     |
| product_name | varchar |
+--------------+---------+
product_id is the primary key of this table.
```

Write an SQL query that selects the **product id**, **year**, **quantity**, and **price** for the **first year** of every product sold.

The query result format is in the following example:

```
Sales table:
+---------+------------+------+----------+-------+
| sale_id | product_id | year | quantity | price |
+---------+------------+------+----------+-------+
| 1       | 100        | 2008 | 10       | 5000  |
| 2       | 100        | 2009 | 12       | 5000  |
| 7       | 200        | 2011 | 15       | 9000  |
+---------+------------+------+----------+-------+

Product table:
+------------+--------------+
| product_id | product_name |
+------------+--------------+
| 100        | Nokia        |
| 200        | Apple        |
| 300        | Samsung      |
+------------+--------------+

Result table:
+------------+------------+----------+-------+
| product_id | first_year | quantity | price |
+------------+------------+----------+-------+
| 100        | 2008       | 10       | 5000  |
| 200        | 2011       | 15       | 9000  |
+------------+------------+----------+-------+
```

Difficulty:

Medium

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

```sql
select s.product_id,
       s.year,
       s.quantity,
       s.price
       from Sales_1070 s
join (select product_id,
       min(year) as first_year
from Sales_1070
group by product_id) first_sales

on s.product_id = first_sales.product_id
and s.year = first_sales.first_year
;
```

100 %

Results | Messages

	product_id	year	quantity	price
1	100	2008	10	5000
2	200	2011	15	9000

1075. Project Employees I

Table: `Project`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| project_id  | int     |
| employee_id | int     |
+-------------+---------+
(project_id, employee_id) is the primary key of this table.
employee_id is a foreign key to Employee table.
```

Table: `Employee`

```
+------------------+---------+
| Column Name      | Type    |
+------------------+---------+
| employee_id      | int     |
| name             | varchar |
| experience_years | int     |
+------------------+---------+
employee_id is the primary key of this table.
```

Write an SQL query that reports the **average** experience years of all the employees for each project, **rounded to 2 digits**.

The query result format is in the following example:

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

Write an SQL query that reports the **average** experience years of all the employees for each project, **rounded to 2 digits**.

The query result format is in the following example:

```
Project table:
+-------------+-------------+
| project_id  | employee_id |
+-------------+-------------+
| 1           | 1           |
| 1           | 2           |
| 1           | 3           |
| 2           | 1           |
| 2           | 4           |
+-------------+-------------+

Employee table:
+-------------+--------+------------------+
| employee_id | name   | experience_years |
+-------------+--------+------------------+
| 1           | Khaled | 3                |
| 2           | Ali    | 2                |
| 3           | John   | 1                |
| 4           | Doe    | 2                |
+-------------+--------+------------------+

Result table:
+-------------+---------------+
| project_id  | average_years |
+-------------+---------------+
| 1           | 2.00          |
| 2           | 2.50          |
+-------------+---------------+
The average experience years for the first project is (3 + 2 + 1) / 3 = 2.00 and for the second project is (3 + 2) / 2 = 2.50
```

```
88  select p.project_id,round(avg(cast(e.experience_years as float)),2) as average_years
89  from Employee_1075 e
90  left join Project_1075 p
91  on e.employee_id = p.employee_id
92  group by p.project_id
93
```

100 %

Results | Messages

	project_id	average_years
1	1	2
2	2	2.5

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1076. Project Employees II

Table: `Project`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| project_id  | int     |
| employee_id | int     |
+-------------+---------+
(project_id, employee_id) is the primary key of this table.
employee_id is a foreign key to Employee table.
```

Table: `Employee`

```
+------------------+---------+
| Column Name      | Type    |
+------------------+---------+
| employee_id      | int     |
| name             | varchar |
| experience_years | int     |
+------------------+---------+
employee_id is the primary key of this table.
```

Write an SQL query that reports all the **projects** that have the most employees.

The query result format is in the following example:

```
Project table:
+-------------+-------------+
| project_id  | employee_id |
+-------------+-------------+
| 1           | 1           |
| 1           | 2           |
| 1           | 3           |
| 2           | 1           |
| 2           | 4           |
+-------------+-------------+

Employee table:
+-------------+--------+------------------+
| employee_id | name   | experience_years |
+-------------+--------+------------------+
| 1           | Khaled | 3                |
| 2           | Ali    | 2                |
| 3           | John   | 1                |
| 4           | Doe    | 2                |
+-------------+--------+------------------+

Result table:
+-------------+
| project_id  |
+-------------+
| 1           |
+-------------+
The first project has 3 employees while the second one has 2.
```

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

```
89  with cte as (select project_id,count(employee_id) as employee_no
90  from Project_1076
91  group by project_id)
92   select project_id from cte
93   where employee_no=(select max(employee_no) from cte)
94
```

100 %

Results | Messages

	project_id
1	1

1077. Project Employees III

Table: `Project`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| project_id  | int     |
| employee_id | int     |
+-------------+---------+
(project_id, employee_id) is the primary key of this table.
employee_id is a foreign key to Employee table.
```

Table: `Employee`

```
+------------------+---------+
| Column Name      | Type    |
+------------------+---------+
| employee_id      | int     |
| name             | varchar |
| experience_years | int     |
+------------------+---------+
employee_id is the primary key of this table.
```

Write an SQL query that reports the **most experienced** employees in each project. In case of a tie, report all employees with the maximum number of experience years.

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

The query result format is in the following example:

```
Project table:
+-------------+-------------+
| project_id  | employee_id |
+-------------+-------------+
| 1           | 1           |
| 1           | 2           |
| 1           | 3           |
| 2           | 1           |
| 2           | 4           |
+-------------+-------------+

Employee table:
+-------------+--------+------------------+
| employee_id | name   | experience_years |
+-------------+--------+------------------+
| 1           | Khaled | 3                |
| 2           | Ali    | 2                |
| 3           | John   | 3                |
| 4           | Doe    | 2                |
+-------------+--------+------------------+

Result table:
+-------------+---------------+
| project_id  | employee_id   |
+-------------+---------------+
| 1           | 1             |
| 1           | 3             |
| 2           | 1             |
+-------------+---------------+
Both employees with id 1 and 3 have the most experience among the employees of the first project. For the second project, the employee with
```

```
90  with cte as (select p.*,
91  e.experience_years,
92  max(e.experience_years) over (partition by p.project_id) as max_exp
93  from Project_1077 as p
94  left join Employee_1077 as e
95  on p.employee_id=e.employee_id)
96  select project_id,employee_id
97  from cte
98  where experience_years= max_exp
```

100 %

Results | Messages

	project_id	employee_id
1	1	1
2	1	3
3	2	1

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1082. Sales Analysis I

Table: `Product`

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| product_id   | int     |
| product_name | varchar |
| unit_price   | int     |
+--------------+---------+
product_id is the primary key of this table.
```

Table: `Sales`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| seller_id   | int     |
| product_id  | int     |
| buyer_id    | int     |
| sale_date   | date    |
| quantity    | int     |
| price       | int     |
+------ ------+---------+
This table has no primary key, it can have repeated rows.
product_id is a foreign key to Product table.
```

Write an SQL query that reports the best **seller** by total sales price, If there is a tie, report them all.

The query result format is in the following example:

```
Product table:
+------------+--------------+------------+
| product_id | product_name | unit_price |
+------------+--------------+------------+
| 1          | S8           | 1000       |
| 2          | G4           | 800        |
| 3          | iPhone       | 1400       |
+------------+--------------+------------+

Sales table:
+-----------+------------+----------+------------+----------+-------+
| seller_id | product_id | buyer_id | sale_date  | quantity | price |
+-----------+------------+----------+------------+----------+-------+
| 1         | 1          | 1        | 2019-01-21 | 2        | 2000  |
| 1         | 2          | 2        | 2019-02-17 | 1        | 800   |
| 2         | 2          | 3        | 2019-06-02 | 1        | 800   |
| 3         | 3          | 4        | 2019-05-13 | 2        | 2800  |
+-----------+------------+----------+------------+----------+-------+

Result table:
+-------------+
| seller_id   |
+-------------+
| 1           |
| 3           |
+-------------+
Both sellers with id 1 and 3 sold products with the most total price of 2800.
```

Difficulty:

Easy

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```
--using subquery
select seller_id
from Sales_1082
group by seller_id
having sum(price) =(select top 1 sum(price) from Sales_1082 group by seller_id order by sum(price) desc);

----using cte & sub query
with cte as
(select seller_id, sum(price) as tot_sale
from Sales_1082
group by seller_id)

select seller_id
from cte
where tot_sale = (select max(tot_sale) from cte)
```

100 %

Results Messages

	seller_id
1	1
2	3

	seller_id
1	1
2	3

1083. Sales Analysis II

Table: `Product`

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| product_id   | int     |
| product_name | varchar |
| unit_price   | int     |
+--------------+---------+
product_id is the primary key of this table.
```

Table: `Sales`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| seller_id   | int     |
| product_id  | int     |
| buyer_id    | int     |
| sale_date   | date    |
| quantity    | int     |
| price       | int     |
+------ ------+---------+
This table has no primary key, it can have repeated rows.
product_id is a foreign key to Product table.
```

Write an SQL query that reports the **buyers** who have bought *S8* but not *iPhone*. Note that *S8* and *iPhone* are products present in the `Product`

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

Write an SQL query that reports the **buyers** who have bought *S8* but not *iPhone*. Note that *S8* and *iPhone* are products present in the `Product` table.

The query result format is in the following example:

```
Product table:
+------------+--------------+------------+
| product_id | product_name | unit_price |
+------------+--------------+------------+
| 1          | S8           | 1000       |
| 2          | G4           | 800        |
| 3          | iPhone       | 1400       |
+------------+--------------+------------+

Sales table:
+-----------+------------+----------+------------+----------+-------+
| seller_id | product_id | buyer_id | sale_date  | quantity | price |
+-----------+------------+----------+------------+----------+-------+
| 1         | 1          | 1        | 2019-01-21 | 2        | 2000  |
| 1         | 2          | 2        | 2019-02-17 | 1        | 800   |
| 2         | 1          | 3        | 2019-06-02 | 1        | 800   |
| 3         | 3          | 3        | 2019-05-13 | 2        | 2800  |
+-----------+------------+----------+------------+----------+-------+

Result table:
+-------------+
| buyer_id    |
+-------------+
| 1           |
+-------------+
The buyer with id 1 bought an S8 but didn't buy an iPhone. The buyer with id 3 bought both.
```

```sql
(select distinct s.buyer_id from Sales_1083 s join Product_1083 p on s.product_id=p.product_id where p.product_name='S8')
except
(select distinct buyer_id from Sales_1083 join Product_1083 on Sales_1083.product_id=Product_1083.product_id where product_name='iPhone');


SELECT *--DISTINCT buyer_id
FROM Sales_1083 s
JOIN Product_1083 p ON s.product_id = p.product_id
WHERE p.product_name = 'S8'
AND buyer_id NOT IN (
SELECT buyer_id
    FROM Sales_1083 s
    JOIN Product_1083 p ON s.product_id = p.product_id
    WHERE p.product_name = 'iPhone'
);
```

100 %

Results | Messages

	buyer_id
1	1

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1084. Sales Analysis III

Table: `Product`

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| product_id   | int     |
| product_name | varchar |
| unit_price   | int     |
+--------------+---------+
product_id is the primary key of this table.
```

Table: `Sales`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| seller_id   | int     |
| product_id  | int     |
| buyer_id    | int     |
| sale_date   | date    |
| quantity    | int     |
| price       | int     |
+------ ------+---------+
This table has no primary key, it can have repeated rows.
product_id is a foreign key to Product table.
```

Write an SQL query that reports the **products** that were **only** sold in spring 2019. That is, between **2019-01-01** and **2019-03-31** inclusive.

The query result format is in the following example:

```
Product table:
+------------+--------------+------------+
| product_id | product_name | unit_price |
+------------+--------------+------------+
| 1          | S8           | 1000       |
| 2          | G4           | 800        |
| 3          | iPhone       | 1400       |
+------------+--------------+------------+

Sales table:
+-----------+------------+----------+------------+----------+-------+
| seller_id | product_id | buyer_id | sale_date  | quantity | price |
+-----------+------------+----------+------------+----------+-------+
| 1         | 1          | 1        | 2019-01-21 | 2        | 2000  |
| 1         | 2          | 2        | 2019-02-17 | 1        | 800   |
| 2         | 2          | 3        | 2019-06-02 | 1        | 800   |
| 3         | 3          | 4        | 2019-05-13 | 2        | 2800  |
+-----------+------------+----------+------------+----------+-------+

Result table:
+-------------+--------------+
| product_id  | product_name |
+-------------+--------------+
| 1           | S8           |
+-------------+--------------+
The product with id 1 was only sold in spring 2019 while the other two were sold after.
```

Difficulty:

Easy

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```sql
select distinct p.product_id,p.product_name
from
((select distinct s.product_id
from Sales_1084 s
where s.sale_date between '2019-01-01' and '2019-03-31')
except
(select distinct s.product_id
from Sales_1084 s
where s.sale_date not between '2019-01-01' and '2019-03-31')) R
left join Product_1084 p
on R.product_id = p.product_id
;

select distinct p.product_id,p.product_name
from Sales_1084 s
join Product_1084 p
on s.product_id = p.product_id

SELECT product_id, product_name
```

100 %

Results | Messages

	product_id	product_name
1	1	S8

1098. Unpopular Books

Table: `Books`

```
+----------------+---------+
| Column Name    | Type    |
+----------------+---------+
| book_id        | int     |
| name           | varchar |
| available_from | date    |
+----------------+---------+
book_id is the primary key of this table.
```

Table: `Orders`

```
+----------------+---------+
| Column Name    | Type    |
+----------------+---------+
| order_id       | int     |
| book_id        | int     |
| quantity       | int     |
| dispatch_date  | date    |
+----------------+---------+
order_id is the primary key of this table.
book_id is a foreign key to the Books table.
```

Write an SQL query that reports the **books** that have sold **less than 10** copies in the last year, excluding books that have been available for less than 1 month from today. **Assume today is 2019-06-23**.

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

The query result format is in the following example:

```
Books table:
+---------+--------------------+----------------+
| book_id | name               | available_from |
+---------+--------------------+----------------+
| 1       | "Kalila And Demna" | 2010-01-01     |
| 2       | "28 Letters"       | 2012-05-12     |
| 3       | "The Hobbit"       | 2019-06-10     |
| 4       | "13 Reasons Why"   | 2019-06-01     |
| 5       | "The Hunger Games" | 2008-09-21     |
+---------+--------------------+----------------+

Orders table:
+----------+---------+----------+---------------+
| order_id | book_id | quantity | dispatch_date |
+----------+---------+----------+---------------+
| 1        | 1       | 2        | 2018-07-26    |
| 2        | 1       | 1        | 2018-11-05    |
| 3        | 3       | 8        | 2019-06-11    |
| 4        | 4       | 6        | 2019-06-05    |
| 5        | 4       | 5        | 2019-06-20    |
| 6        | 5       | 9        | 2009-02-02    |
| 7        | 5       | 8        | 2010-04-13    |
+----------+---------+----------+---------------+

Result table:
+-----------+--------------------+
| book_id   | name               |
+-----------+--------------------+
| 1         | "Kalila And Demna" |
| 2         | "28 Letters"       |
| 5         | "The Hunger Games" |
+-----------+--------------------+
```

```sql
with cte as
(select book_id,sum(quantity) as total_sold
from Orders_1098
where dispatch_date > (DATEADD(YEAR, -1, '2019-06-23'))
group by book_id)

select b.book_id,b.name
from Books_1098 b
left join cte
on b.book_id=cte.book_id
where isnull(cte.total_sold,0)<10
and b.available_from <= DATEADD(MONTH, -1, '2019-06-23');
```

100 %

Results | Messages

	book_id	name
1	1	Kalila And Demna
2	2	28 Letters
3	5	The Hunger Games

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

1107. New Users Daily Count

Table: `Traffic`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| user_id       | int     |
| activity      | enum    |
| activity_date | date    |
+---------------+---------+
There is no primary key for this table, it may have duplicate rows.
The activity column is an ENUM type of ('login', 'logout', 'jobs', 'groups', 'homepage').
```

Write an SQL query that reports for every date within at most **90 days** from today, the number of users that logged in for the first time on that date. Assume today is **2019-06-30**.

The query result format is in the following example:

The query result format is in the following example:

```
Traffic table:
+---------+----------+---------------+
| user_id | activity | activity_date |
+---------+----------+---------------+
| 1       | login    | 2019-05-01    |
| 1       | homepage | 2019-05-01    |
| 1       | logout   | 2019-05-01    |
| 2       | login    | 2019-06-21    |
| 2       | logout   | 2019-06-21    |
| 3       | login    | 2019-01-01    |
| 3       | jobs     | 2019-01-01    |
| 3       | logout   | 2019-01-01    |
| 4       | login    | 2019-06-21    |
| 4       | groups   | 2019-06-21    |
| 4       | logout   | 2019-06-21    |
| 5       | login    | 2019-03-01    |
| 5       | logout   | 2019-03-01    |
| 5       | login    | 2019-06-21    |
| 5       | logout   | 2019-06-21    |
+---------+----------+---------------+

Result table:
+------------+-------------+
| login_date | user_count  |
+------------+-------------+
| 2019-05-01 | 1           |
| 2019-06-21 | 2           |
+------------+-------------+
Note that we only care about dates with non zero user count.
The user with id 5 first logged in on 2019-03-01 so he's not counted on 2019-06-21.
```


LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```sql
WITH FirstLogin AS (
    SELECT user_id,
           MIN(activity_date) AS first_login_date
     FROM Traffic_1107
     WHERE  activity = 'login'
     GROUP BY user_id
        ),
FilteredLogin AS (
    SELECT first_login_date,
           COUNT(DISTINCT user_id) AS user_count
    FROM FirstLogin
    WHERE first_login_date BETWEEN DATEADD(DAY, -90, '2019-06-30') AND '2019-06-30'
    GROUP BY first_login_date)
SELECT first_login_date AS login_date,
       user_count
FROM FilteredLogin
 ORDER BY first_login_date;
```

100 %

Results | Messages

	login_date	user_count
1	2019-05-01	1
2	2019-06-21	2

```sql
WITH cte AS
(SELECT user_id, activity_date, ROW_NUMBER() OVER
(PARTITION BY user_id ORDER BY activity_date) AS
rnk
FROM Traffic
WHERE activity = 'login')

SELECT activity_date AS login_date, COUNT(user_id)
AS user_count
FROM cte
WHERE rnk = 1
AND activity_date BETWEEN DATE_SUB('2019-06-30',
INTERVAL 90 DAY) AND '2019-06-30'
GROUP BY activity_date;
```

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1112. Highest Grade For Each Student

Table: `Enrollments`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| student_id    | int     |
| course_id     | int     |
| grade         | int     |
+---------------+---------+
(student_id, course_id) is the primary key of this table.
```

Write a SQL query to find the highest grade with its corresponding course for each student. In case of a tie, you should find the course with the smallest `course_id`. The output must be sorted by increasing `student_id`.

The query result format is in the following example:

Write a SQL query to find the highest grade with its corresponding course for each student. In case of a tie, you should find the course with the smallest `course_id`. The output must be sorted by increasing `student_id`.

The query result format is in the following example:

```
Enrollments table:
+------------+-------------------+
| student_id | course_id | grade |
+------------+-----------+-------+
| 2          | 2         | 95    |
| 2          | 3         | 95    |
| 1          | 1         | 90    |
| 1          | 2         | 99    |
| 3          | 1         | 80    |
| 3          | 2         | 75    |
| 3          | 3         | 82    |
+------------+-----------+-------+

Result table:
+------------+-------------------+
| student_id | course_id | grade |
+------------+-----------+-------+
| 1          | 2         | 99    |
| 2          | 2         | 95    |
| 3          | 3         | 82    |
+------------+-----------+-------+
```

Difficulty:

Medium

```sql
select e.student_id,min(e.course_id) as course_id,max(e.grade) as grade
from Enrollments_1112 e
left join (select student_id,max(grade) as max_grade from Enrollments_1112 group by student_id) T
on e.student_id = T.student_id
where e.grade=T.max_grade
group by e.student_id
order by e.student_id;

--using window function
select tab.student_id,tab.course_id,tab.grade
from (select *,row_number() over (partition by student_id order by grade desc,course_id) as rnk from Enrollments_1112) tab
where tab.rnk=1
order by tab.student_id;
```

100 %

Results | Messages

	student_id	course_id	grade
1	1	2	99
2	2	2	95
3	3	3	82

	student_id	course_id	grade
1	1	2	99
2	2	2	95
3	3	3	82

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1113. Reported Posts

Table: `Actions`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| user_id       | int     |
| post_id       | int     |
| action_date   | date    |
| action        | enum    |
| extra         | varchar |
+---------------+---------+
There is no primary key for this table, it may have duplicate rows.
The action column is an ENUM type of ('view', 'like', 'reaction', 'comment', 'report', 'share').
The extra column has optional information about the action such as a reason for report or a type of reaction.
```

Write an SQL query that reports the number of posts reported yesterday for each report reason. Assume today is **2019-07-05**.

The query result format is in the following example:

```
Actions table:
+---------+---------+-------------+--------+--------+
| user_id | post_id | action_date | action | extra  |
+---------+---------+-------------+--------+--------+
| 1       | 1       | 2019-07-01  | view   | null   |
| 1       | 1       | 2019-07-01  | like   | null   |
| 1       | 1       | 2019-07-01  | share  | null   |
| 2       | 4       | 2019-07-04  | view   | null   |
| 2       | 4       | 2019-07-04  | report | spam   |
| 3       | 4       | 2019-07-04  | view   | null   |
| 3       | 4       | 2019-07-04  | report | spam   |
| 4       | 3       | 2019-07-02  | view   | null   |
| 4       | 3       | 2019-07-02  | report | spam   |
| 5       | 2       | 2019-07-04  | view   | null   |
| 5       | 2       | 2019-07-04  | report | racism |
| 5       | 5       | 2019-07-04  | view   | null   |
| 5       | 5       | 2019-07-04  | report | racism |
+---------+---------+-------------+--------+--------+

Result table:
+---------------+--------------+
| report_reason | report_count |
+---------------+--------------+
| spam          | 1            |
| racism        | 2            |
+---------------+--------------+
Note that we only care about report reasons with non zero number of reports.
```

Difficulty:

Easy

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```
79  select extra as report_reason,count(distinct post_id) as report_count
80  from Actions_1113
81  where action_date=dateadd(day,-1,'2019-07-05') and action='report'and extra is not null
82  group by extra
83  order by report_count
84
85
86
87
88
```

100 %

Results | Messages

	report_reason	report_count
1	spam	1
2	racism	2

```
2
3   SELECT extra AS report_reason, COUNT(DISTINCT
    post_id) AS report_count
4   FROM Actions
5   WHERE action_date = '2019-07-04'
6   AND action = 'report'
7   AND extra IS NOT NULL
8   GROUP BY extra
```

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

1126. Active Businesses

Table: `Events`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| business_id   | int     |
| event_type    | varchar |
| occurences    | int     |
+---------------+---------+
(business_id, event_type) is the primary key of this table.
Each row in the table logs the info that an event of some type occured at some business for a number of times.
```

Write an SQL query to find all *active businesses*.

An active business is a business that has more than one event type with occurences greater than the average occurences of that event type among all businesses.

The query result format is in the following example:

```
Events table:
+-------------+------------+------------+
| business_id | event_type | occurences |
+-------------+------------+------------+
| 1           | reviews    | 7          |
| 3           | reviews    | 3          |
| 1           | ads        | 11         |
| 2           | ads        | 7          |
| 3           | ads        | 6          |
| 1           | page views | 3          |
| 2           | page views | 12         |
+-------------+------------+------------+

Result table:
+-------------+
| business_id |
+-------------+
| 1           |
+-------------+
Average for 'reviews', 'ads' and 'page views' are (7+3)/2=5, (11+7+6)/3=8, (3+12)/2=7.5 respectively.
Business with id 1 has 7 'reviews' events (more than 5) and 11 'ads' events (more than 8) so it is an active business.
```

Difficulty:

Medium

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```sql
With cte as(select event_type,avg(occurences) as avg_occurences
from Events_1126
group by event_type)

select business_id
from Events_1126 e
left join cte
on e.event_type=cte.event_type
where e.occurences>cte.avg_occurences
group by business_id
having count(e.event_type)>1;

---using window analytic fn

with cte1 as (
select *,avg(occurences) over (partition by event_type) as avg_occur
from Events_1126)

select business_id
from cte1 where occurences>avg_occur
group by business_id
having count(event_type)>1;
```

100 %

Results Messages

	business_id
1	1

	business_id
1	1

1132. Reported Posts II

Table: `Actions`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| user_id       | int     |
| post_id       | int     |
| action_date   | date    |
| action        | enum    |
| extra         | varchar |
+---------------+---------+
There is no primary key for this table, it may have duplicate rows.
The action column is an ENUM type of ('view', 'like', 'reaction', 'comment', 'report', 'share').
The extra column has optional information about the action such as a reason for report or a type of reaction.
```

Table: `Removals`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| post_id       | int     |
| remove_date   | date    |
+---------------+---------+
post_id is the primary key of this table.
Each row in this table indicates that some post was removed as a result of being reported or as a result of an admin review.
```

Write an SQL query to find the average for daily percentage of posts that got removed after being reported as spam, **rounded to 2 decimal places**.

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

The query result format is in the following example:

```
Actions table:
+---------+---------+-------------+--------+--------+
| user_id | post_id | action_date | action | extra  |
+---------+---------+-------------+--------+--------+
| 1       | 1       | 2019-07-01  | view   | null   |
| 1       | 1       | 2019-07-01  | like   | null   |
| 1       | 1       | 2019-07-01  | share  | null   |
| 2       | 2       | 2019-07-04  | view   | null   |
| 2       | 2       | 2019-07-04  | report | spam   |
| 3       | 4       | 2019-07-04  | view   | null   |
| 3       | 4       | 2019-07-04  | report | spam   |
| 4       | 3       | 2019-07-02  | view   | null   |
| 4       | 3       | 2019-07-02  | report | spam   |
| 5       | 2       | 2019-07-03  | view   | null   |
| 5       | 2       | 2019-07-03  | report | racism |
| 5       | 5       | 2019-07-03  | view   | null   |
| 5       | 5       | 2019-07-03  | report | racism |
+---------+---------+-------------+--------+--------+

Removals table:
+---------+-------------+
| post_id | remove_date |
+---------+-------------+
| 2       | 2019-07-20  |
| 3       | 2019-07-18  |
+---------+-------------+

Result table:
+----------------------+
| average_daily_percent |
+----------------------+
| 75.00                |
+----------------------+
The percentage for 2019-07-04 is 50% because only one post of two spam reported posts was removed.
```

```
Removals table:
+---------+-------------+
| post_id | remove_date |
+---------+-------------+
| 2       | 2019-07-20  |
| 3       | 2019-07-18  |
+---------+-------------+

Result table:
+----------------------+
| average_daily_percent |
+----------------------+
| 75.00                |
+----------------------+
The percentage for 2019-07-04 is 50% because only one post of two spam reported posts was removed.
The percentage for 2019-07-02 is 100% because one post was reported as spam and it was removed.
The other days had no spam reports so the average is (50 + 100) / 2 = 75%
Note that the output is only one number and that we do not care about the remove dates.
```

Difficulty:

Medium

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

```sql
With cte as (Select distinct a.post_id,a.action_date from Actions_1132 a
where a.action = 'report' and a.extra = 'spam'
),
Result as (Select cte.action_date,Cast(Count(R.remove_date) as float)/Count(*) * 100 as daily_removal_percentage
from cte left join Removals_1132 R on cte.post_id = R.post_id
group by cte.action_date)

Select Round(Avg(result.daily_removal_percentage),2) as average_daily_percent
from
result;
```

100 %

Results | Messages

	average_daily_percent
1	75

1141. User Activity for the Past 30 Days I

Table: `Activity`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| user_id       | int     |
| session_id    | int     |
| activity_date | date    |
| activity_type | enum    |
+---------------+---------+
There is no primary key for this table, it may have duplicate rows.
The activity_type column is an ENUM of type ('open_session', 'end_session', 'scroll_down', 'send_message').
The table shows the user activities for a social media website.
Note that each session belongs to exactly one user.
```

Write an SQL query to find the daily active user count for a period of 30 days ending **2019-07-27** inclusively. A user was active on some day if he/she made at least one activity on that day.

The query result format is in the following example:

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

Write an SQL query to find the daily active user count for a period of 30 days ending **2019-07-27** inclusively. A user was active on some day if he/she made at least one activity on that day.

The query result format is in the following example:

```
Activity table:
+---------+------------+---------------+---------------+
| user_id | session_id | activity_date | activity_type |
+---------+------------+---------------+---------------+
| 1       | 1          | 2019-07-20    | open_session  |
| 1       | 1          | 2019-07-20    | scroll_down   |
| 1       | 1          | 2019-07-20    | end_session   |
| 2       | 4          | 2019-07-20    | open_session  |
| 2       | 4          | 2019-07-21    | send_message  |
| 2       | 4          | 2019-07-21    | end_session   |
| 3       | 2          | 2019-07-21    | open_session  |
| 3       | 2          | 2019-07-21    | send_message  |
| 3       | 2          | 2019-07-21    | end_session   |
| 4       | 3          | 2019-06-25    | open_session  |
| 4       | 3          | 2019-06-25    | end_session   |
+---------+------------+---------------+---------------+

Result table:
+------------+--------------+
| day        | active_users |
+------------+--------------+
| 2019-07-20 | 2            |
| 2019-07-21 | 2            |
+------------+--------------+
Note that we do not care about days with zero active users.
```

Difficulty:

Easy

```sql
71  SELECT activity_date AS day,COUNT(DISTINCT user_id) AS active_users
72  FROM Activity_1141
73  WHERE activity_date between dateadd(day,-29,'2019-07-27') and '2019-07-27'
74  GROUP BY activity_date
```

100 %

Results | Messages

	day	active_users
1	2019-07-20	2
2	2019-07-21	2

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1142. User Activity for the Past 30 Days II

Table: `Activity`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| user_id       | int     |
| session_id    | int     |
| activity_date | date    |
| activity_type | enum    |
+---------------+---------+
There is no primary key for this table, it may have duplicate rows.
The activity_type column is an ENUM of type ('open_session', 'end_session', 'scroll_down', 'send_message').
The table shows the user activities for a social media website.
Note that each session belongs to exactly one user.
```

Write an SQL query to find the average number of sessions per user for a period of 30 days ending **2019-07-27** inclusively, **rounded to 2 decimal places**. The sessions we want to count for a user are those with at least one activity in that time period.

The query result format is in the following example:

The query result format is in the following example:

```
Activity table:
+---------+------------+---------------+---------------+
| user_id | session_id | activity_date | activity_type |
+---------+------------+---------------+---------------+
| 1       | 1          | 2019-07-20    | open_session  |
| 1       | 1          | 2019-07-20    | scroll_down   |
| 1       | 1          | 2019-07-20    | end_session   |
| 2       | 4          | 2019-07-20    | open_session  |
| 2       | 4          | 2019-07-21    | send_message  |
| 2       | 4          | 2019-07-21    | end_session   |
| 3       | 2          | 2019-07-21    | open_session  |
| 3       | 2          | 2019-07-21    | send_message  |
| 3       | 2          | 2019-07-21    | end_session   |
| 3       | 5          | 2019-07-21    | open_session  |
| 3       | 5          | 2019-07-21    | scroll_down   |
| 3       | 5          | 2019-07-21    | end_session   |
| 4       | 3          | 2019-06-25    | open_session  |
| 4       | 3          | 2019-06-25    | end_session   |
+---------+------------+---------------+---------------+

Result table:
+---------------------------+
| average_sessions_per_user |
+---------------------------+
| 1.33                      |
+---------------------------+
User 1 and 2 each had 1 session in the past 30 days while user 3 had 2 sessions so the average is (1 + 1 + 2) / 3 = 1.33.
```

Difficulty:

Easy

```sql
select isnull(round(cast(count(distinct session_id)as float)/count(distinct user_id),2),0.00) as average_sessions_per_user
from Activity_1142
where activity_date between DATEADD(day,-29,'2019-07-27') and '2019-07-27';
```

100 %

Results | Messages

	average_sessions_per_user
1	1.33

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1148. Article Views I

Table: `Views`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| article_id    | int     |
| author_id     | int     |
| viewer_id     | int     |
| view_date     | date    |
+---------------+---------+
There is no primary key for this table, it may have duplicate rows.
Each row of this table indicates that some viewer viewed an article (written by some author) on some date.
Note that equal author_id and viewer_id indicate the same person.
```

Write an SQL query to find all the authors that viewed at least one of their own articles, sorted in ascending order by their id.

The query result format is in the following example:

```
Views table:
+------------+-----------+-----------+------------+
| article_id | author_id | viewer_id | view_date  |
+------------+-----------+-----------+------------+
| 1          | 3         | 5         | 2019-08-01 |
| 1          | 3         | 6         | 2019-08-02 |
| 2          | 7         | 7         | 2019-08-01 |
| 2          | 7         | 6         | 2019-08-02 |
| 4          | 7         | 1         | 2019-07-22 |
| 3          | 4         | 4         | 2019-07-21 |
| 3          | 4         | 4         | 2019-07-21 |
+------------+-----------+-----------+------------+

Result table:
+------+
| id   |
+------+
| 4    |
| 7    |
+------+
```

Difficulty:

Easy

```
62  select distinct author_id as id
63  from Views_1148
64  where author_id=viewer_id
65  order by author_id;
```

100 %

Results | Messages

	id
1	4
2	7

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1149. Article Views II

Table: `Views`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| article_id    | int     |
| author_id     | int     |
| viewer_id     | int     |
| view_date     | date    |
+---------------+---------+
There is no primary key for this table, it may have duplicate rows.
Each row of this table indicates that some viewer viewed an article (written by some author) on some date.
Note that equal author_id and viewer_id indicate the same person.
```

Write an SQL query to find all the people who viewed more than one article on the same date, sorted in ascending order by their id.

Write an SQL query to find all the people who viewed more than one article on the same date, sorted in ascending order by their id.

The query result format is in the following example:

```
Views table:
+------------+-----------+-----------+------------+
| article_id | author_id | viewer_id | view_date  |
+------------+-----------+-----------+------------+
| 1          | 3         | 5         | 2019-08-01 |
| 3          | 4         | 5         | 2019-08-01 |
| 1          | 3         | 6         | 2019-08-02 |
| 2          | 7         | 7         | 2019-08-01 |
| 2          | 7         | 6         | 2019-08-02 |
| 4          | 7         | 1         | 2019-07-22 |
| 3          | 4         | 4         | 2019-07-21 |
| 3          | 4         | 4         | 2019-07-21 |
+------------+-----------+-----------+------------+

Result table:
+------+
| id   |
+------+
| 5    |
| 6    |
+------+
```

Difficulty:

Medium

```
63
64  select distinct viewer_id as id
65  from Views_1149
66  group by view_date,viewer_id
67  having count(distinct article_id)>1
68  order by id;
```

100 %

Results | Messages

	id
1	5
2	6

1158. Market Analysis I

Table: `Users`

```
+----------------+---------+
| Column Name    | Type    |
+----------------+---------+
| user_id        | int     |
| join_date      | date    |
| favorite_brand | varchar |
+----------------+---------+
user_id is the primary key of this table.
This table has the info of the users of an online shopping website where users can sell and buy items.
```

Table: `Orders`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| order_id      | int     |
| order_date    | date    |
| item_id       | int     |
| buyer_id      | int     |
| seller_id     | int     |
+---------------+---------+
order_id is the primary key of this table.
item_id is a foreign key to the Items table.
buyer_id and seller_id are foreign keys to the Users table.
```


LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

Table: `Items`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| item_id       | int     |
| item_brand    | varchar |
+---------------+---------+
item_id is the primary key of this table.
```

Write an SQL query to find for each user, the join date and the number of orders they made as a buyer in **2019**.

The query result format is in the following example:

```
Users table:
+---------+------------+----------------+
| user_id | join_date  | favorite_brand |
+---------+------------+----------------+
| 1       | 2018-01-01 | Lenovo         |
| 2       | 2018-02-09 | Samsung        |
| 3       | 2018-01-19 | LG             |
| 4       | 2018-05-21 | HP             |
+---------+------------+----------------+
```

```
Orders table:
+----------+------------+---------+----------+-----------+
| order_id | order_date | item_id | buyer_id | seller_id |
+----------+------------+---------+----------+-----------+
| 1        | 2019-08-01 | 4       | 1        | 2         |
| 2        | 2018-08-02 | 2       | 1        | 3         |
| 3        | 2019-08-03 | 3       | 2        | 3         |
| 4        | 2018-08-04 | 1       | 4        | 2         |
| 5        | 2018-08-04 | 1       | 3        | 4         |
| 6        | 2019-08-05 | 2       | 2        | 4         |
+----------+------------+---------+----------+-----------+

Items table:
+---------+------------+
| item_id | item_brand |
+---------+------------+
| 1       | Samsung    |
| 2       | Lenovo     |
| 3       | LG         |
| 4       | HP         |
+---------+------------+

Result table:
+-----------+------------+----------------+
| buyer_id  | join_date  | orders_in_2019 |
+-----------+------------+----------------+
| 1         | 2018-01-01 | 1              |
| 2         | 2018-02-09 | 2              |
| 3         | 2018-01-19 | 0              |
| 4         | 2018-05-21 | 0              |
+-----------+------------+----------------+
```

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```sql
with cte as (select u.user_id, sum(case when year(o.order_date) = 2019 then 1 else 0 end) as orders_in_2019
from Users_1158 u
left join Orders_1158 o
on u.user_id = o.buyer_id
group by u.user_id)

select u.user_id,u.join_date,orders_in_2019
from cte c
left join Users_1158 u
on c.user_id = u.user_id;
```

100 %

Results | Messages

	user_id	join_date	orders_in_2019
1	1	2018-01-01	1
2	2	2018-02-09	2
3	3	2018-01-19	0
4	4	2018-05-21	0

```sql
select u.user_id,u.join_date,isnull(o.orders_in_2019,0) as orders_in_2019
from
Users_1158 u left join
(select buyer_id,count(order_id) as orders_in_2019
from Orders_1158
where year(order_date) = 2019
group by buyer_id) as o
on o.buyer_id=u.user_id
where year(u.join_date) <= '2019'
```

100 %

Results | Messages

	user_id	join_date	orders_in_2019
1	1	2018-01-01	1
2	2	2018-02-09	2
3	3	2018-01-19	0
4	4	2018-05-21	0

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

1159. Market Analysis II

Table: `Users`

```
+----------------+---------+
| Column Name    | Type    |
+----------------+---------+
| user_id        | int     |
| join_date      | date    |
| favorite_brand | varchar |
+----------------+---------+
user_id is the primary key of this table.
This table has the info of the users of an online shopping website where users can sell and buy items.
```

Table: `Orders`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| order_id      | int     |
| order_date    | date    |
| item_id       | int     |
| buyer_id      | int     |
| seller_id     | int     |
+---------------+---------+
order_id is the primary key of this table.
item_id is a foreign key to the Items table.
buyer_id and seller_id are foreign keys to the Users table.
```

Table: `Items`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| item_id       | int     |
| item_brand    | varchar |
+---------------+---------+
item_id is the primary key of this table.
```

Write an SQL query to find for each user, whether the brand of the second item (by date) they sold is their favorite brand. If a user sold less than two items, report the answer for that user as no.

It is guaranteed that no seller sold more than one item on a day.

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

The query result format is in the following example:

```
Users table:
+---------+------------+----------------+
| user_id | join_date  | favorite_brand |
+---------+------------+----------------+
| 1       | 2019-01-01 | Lenovo         |
| 2       | 2019-02-09 | Samsung        |
| 3       | 2019-01-19 | LG             |
| 4       | 2019-05-21 | HP             |
+---------+------------+----------------+

Orders table:
+----------+------------+---------+----------+-----------+
| order_id | order_date | item_id | buyer_id | seller_id |
+----------+------------+---------+----------+-----------+
| 1        | 2019-08-01 | 4       | 1        | 2         |
| 2        | 2019-08-02 | 2       | 1        | 3         |
| 3        | 2019-08-03 | 3       | 2        | 3         |
| 4        | 2019-08-04 | 1       | 4        | 2         |
| 5        | 2019-08-04 | 1       | 3        | 4         |
| 6        | 2019-08-05 | 2       | 2        | 4         |
+----------+------------+---------+----------+-----------+

Items table:
+---------+------------+
| item_id | item_brand |
+---------+------------+
| 1       | Samsung    |
| 2       | Lenovo     |
| 3       | LG         |
| 4       | HP         |
+---------+------------+
```

```
Result table:
+-----------+--------------------+
| seller_id | 2nd_item_fav_brand |
+-----------+--------------------+
| 1         | no                 |
| 2         | yes                |
| 3         | yes                |
| 4         | no                 |
+-----------+--------------------+

The answer for the user with id 1 is no because they sold nothing.
The answer for the users with id 2 and 3 is yes because the brands of their second sold items are their favorite brands.
The answer for the user with id 4 is no because the brand of their second sold item is not their favorite brand.
```

Difficulty:

Hard

```sql
with cte as (Select seller_id,item_id,row_number() over(partition by seller_id order by order_date) as days from Orders_1159),

day2seller as (Select cte.seller_id as seller_id,cte.item_id as item_id from cte where cte.days=2),

AllInfo as (Select day2seller.*,Items_1159.item_brand as item_brand,Users_1159.favorite_brand as favorite_brand from day2seller
left join Items_1159 on day2seller.item_id=Items_1159.item_id
left join Users_1159 on day2seller.seller_id=Users_1159.user_id),

finalInfo as (Select seller_id,Case when (item_brand=favorite_brand) then 'yes' else 'no' end as [2nd_item_fav_brand]
from AllInfo)

(select user_id, Case When f.[2nd_item_fav_brand] is null then 'no' else f.[2nd_item_fav_brand] end as [2nd_item_fav_brand]
from Users_1159 u
left join finalInfo f on u.user_id=f.seller_id)
```

100 %

Results | Messages

	user_id	2nd_item_fav_brand
1	1	no
2	2	yes
3	3	yes
4	4	no

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```sql
with cte as (select u.user_id,u.favorite_brand,o.order_date,i.item_brand,
dense_rank() over (partition by u.user_id order by
o.order_date) as rnk,
count(*) over (partition by u.user_id) as count_items
from Users_1159 as u
left join Orders_1159 as o
on u.user_id = o.seller_id
left join items_1159 as i
on o.item_id = i.item_id),

cte2 as (select user_id as seller_id,case when count_items<2 then 'no'
 when count_items>=2 and rnk =2 and favorite_brand = item_brand then 'yes'
 when count_items>=2 and rnk =2 and favorite_brand <> item_brand then 'no'
 else null end as [2nd_item_fav_brand]
 from cte)

 select *
 from cte2
 where [2nd_item_fav_brand] is not null;
```

100 %

Results | Messages

	seller_id	2nd_item_fav_brand
1	1	no
2	2	yes
3	3	yes
4	4	no

1164. Product Price at a Given Date

Table: `Products`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| product_id    | int     |
| new_price     | int     |
| change_date   | date    |
+---------------+---------+
(product_id, change_date) is the primary key of this table.
Each row of this table indicates that the price of some product was changed to a new price at some date.
```

Write an SQL query to find the prices of all products on **2019-08-16**. Assume the price of all products before any change is **10**.

The query result format is in the following example:

```
Products table:
+------------+-----------+-------------+
| product_id | new_price | change_date |
+------------+-----------+-------------+
| 1          | 20        | 2019-08-14  |
| 2          | 50        | 2019-08-14  |
| 1          | 30        | 2019-08-15  |
| 1          | 35        | 2019-08-16  |
| 2          | 65        | 2019-08-17  |
| 3          | 20        | 2019-08-18  |
+------------+-----------+-------------+
```

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

```
Result table:
+------------+-------+
| product_id | price |
+------------+-------+
| 2          | 50    |
| 1          | 35    |
| 3          | 10    |
+------------+-------+
```

Difficulty:

Medium

```sql
with data as (select *, DATEDIFF(day, change_date, '2019-08-16') AS diff
from Products_1164)

select T.product_id, isnull(sum(T.price),10) as price  from
(select data.*,d2.min_diff,case when data.diff=d2.min_diff then new_price
                                when data.diff<>d2.min_diff then 0
                                else null end as price
from data left join
(select product_id,min(diff) as min_diff from data
where diff>0
group by product_id) d2
on data.product_id=d2.product_id) T
group by T.product_id;
```

100 %

Results | Messages

	product_id	price
1	1	30
2	2	50
3	3	10

1173. Immediate Food Delivery I

Table: `Delivery`

```
+-----------------------------+---------+
| Column Name                 | Type    |
+-----------------------------+---------+
| delivery_id                 | int     |
| customer_id                 | int     |
| order_date                  | date    |
| customer_pref_delivery_date | date    |
+-----------------------------+---------+
delivery_id is the primary key of this table.
The table holds information about food delivery to customers that make orders at some date and specify a preferred delivery date (on the sam
```

If the preferred delivery date of the customer is the same as the order date then the order is called *immediate* otherwise it's called *scheduled*.

Write an SQL query to find the percentage of immediate orders in the table, **rounded to 2 decimal places**.

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

The query result format is in the following example:

```
Delivery table:
+-------------+-------------+------------+-----------------------------+
| delivery_id | customer_id | order_date | customer_pref_delivery_date |
+-------------+-------------+------------+-----------------------------+
| 1           | 1           | 2019-08-01 | 2019-08-02                  |
| 2           | 5           | 2019-08-02 | 2019-08-02                  |
| 3           | 1           | 2019-08-11 | 2019-08-11                  |
| 4           | 3           | 2019-08-24 | 2019-08-26                  |
| 5           | 4           | 2019-08-21 | 2019-08-22                  |
| 6           | 2           | 2019-08-11 | 2019-08-13                  |
+-------------+-------------+------------+-----------------------------+

Result table:
+----------------------+
| immediate_percentage |
+----------------------+
| 33.33                |
+----------------------+
The orders with delivery id 2 and 3 are immediate while the others are scheduled.
```

Difficulty:

Easy

```
61
62  select
63        round(sum(case when order_date=customer_pref_delivery_date then 1 else 0 end)*100.0 / COUNT(*),2) as  immediate_percentage
64  from Delivery_1173;
65
66
67
68
```

100 %

Results | Messages

	immediate_percentage
1	33.330000000000

1174. Immediate Food Delivery II

Table: `Delivery`

```
+-----------------------------+---------+
| Column Name                 | Type    |
+-----------------------------+---------+
| delivery_id                 | int     |
| customer_id                 | int     |
| order_date                  | date    |
| customer_pref_delivery_date | date    |
+-----------------------------+---------+
delivery_id is the primary key of this table.
The table holds information about food delivery to customers that make orders at some date and specify a preferred delivery date (on the sam
```

If the preferred delivery date of the customer is the same as the order date then the order is called *immediate* otherwise it's called *scheduled*.

The *first order* of a customer is the order with the earliest order date that customer made. It is guaranteed that a customer has exactly one first order.

Write an SQL query to find the percentage of immediate orders in the first orders of all customers, **rounded to 2 decimal places**.

The query result format is in the following example:

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

Write an SQL query to find the percentage of immediate orders in the first orders of all customers, **rounded to 2 decimal places**.

The query result format is in the following example:

```
Delivery table:
+-------------+-------------+------------+-----------------------------+
| delivery_id | customer_id | order_date | customer_pref_delivery_date |
+-------------+-------------+------------+-----------------------------+
| 1           | 1           | 2019-08-01 | 2019-08-02                  |
| 2           | 2           | 2019-08-02 | 2019-08-02                  |
| 3           | 1           | 2019-08-11 | 2019-08-12                  |
| 4           | 3           | 2019-08-24 | 2019-08-24                  |
| 5           | 3           | 2019-08-21 | 2019-08-22                  |
| 6           | 2           | 2019-08-11 | 2019-08-13                  |
| 7           | 4           | 2019-08-09 | 2019-08-09                  |
+-------------+-------------+------------+-----------------------------+

Result table:
+----------------------+
| immediate_percentage |
+----------------------+
| 50.00                |
+----------------------+
The customer id 1 has a first order with delivery id 1 and it is scheduled.
The customer id 2 has a first order with delivery id 2 and it is immediate.
The customer id 3 has a first order with delivery id 5 and it is scheduled.
The customer id 4 has a first order with delivery id 7 and it is immediate.
Hence, half the customers have immediate first orders.
```

Difficulty:

Medium

```
with cte as (select Case When s.order_date=s.customer_pref_delivery_date then 1 else 0 end as is_Immediate from
(select *, row_number() over (partition by customer_id order by order_date) as rnk from Delivery_1174) s
where s.rnk=1)

Select Round(Cast(Sum(is_Immediate) as float)*100/Count(*),2) as immediate_percentage
from cte
```

100 %

Results | Messages

	immediate_percentage
1	50

1179. Reformat Department Table

Table: `Department`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| revenue       | int     |
| month         | varchar |
+---------------+---------+
(id, month) is the primary key of this table.
The table has information about the revenue of each department per month.
The month has values in ["Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"].
```

Write an SQL query to reformat the table such that there is a department id column and a revenue column **for each month**.

The query result format is in the following example:

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

Write an SQL query to reformat the table such that there is a department id column and a revenue column **for each month**.

The query result format is in the following example:

```
Department table:
+------+---------+-------+
| id   | revenue | month |
+------+---------+-------+
| 1    | 8000    | Jan   |
| 2    | 9000    | Jan   |
| 3    | 10000   | Feb   |
| 1    | 7000    | Feb   |
| 1    | 6000    | Mar   |
+------+---------+-------+

Result table:
+------+-------------+-------------+-------------+-----+-------------+
| id   | Jan_Revenue | Feb_Revenue | Mar_Revenue | ... | Dec_Revenue |
+------+-------------+-------------+-------------+-----+-------------+
| 1    | 8000        | 7000        | 6000        | ... | null        |
| 2    | 9000        | null        | null        | ... | null        |
| 3    | null        | 10000       | null        | ... | null        |
+------+-------------+-------------+-------------+-----+-------------+

Note that the result table has 13 columns (1 for the department id + 12 for the months).
```

Difficulty:

Easy

```
select id, max(case when month ='Jan' then revenue else null end) as Jan_Revenue,
                   max(case when month ='Feb' then revenue else null end) as Feb_Revenue,
                   max(case when month ='Mar' then revenue else null end) as Mar_Revenue,
                   max(case when month ='Apr' then revenue else null end) as Apr_Revenue,
                   max(case when month ='May' then revenue else null end) as May_Revenue,
                   max(case when month ='Jun' then revenue else null end) as Jun_Revenue,
                   max(case when month ='Jul' then revenue else null end) as Jul_Revenue,
                   max(case when month ='Aug' then revenue else null end) as Aug_Revenue,
                   max(case when month ='Sep' then revenue else null end) as Sep_Revenue,
                   max(case when month ='Oct' then revenue else null end) as Oct_Revenue,
                   max(case when month ='Nov' then revenue else null end) as Nov_Revenue,
                   max(case when month ='Dec' then revenue else null end) as Dec_Revenue
from Department_1179
group by id
```

100 %

Results | Messages

	id	Jan_Revenue	Feb_Revenue	Mar_Revenue	Apr_Revenue	May_Revenue	Jun_Revenue	Jul_Revenue	Aug_Revenue	Sep_Revenue	Oct_Revenue	Nov_Revenue	Dec_Revenue
1	1	8000	7000	6000	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
2	2	9000	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
3	3	NULL	10000	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

1193. Monthly Transactions I

Table: `Transactions`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| country       | varchar |
| state         | enum    |
| amount        | int     |
| trans_date    | date    |
+---------------+---------+
id is the primary key of this table.
The table has information about incoming transactions.
The state column is an enum of type ["approved", "declined"].
```

Write an SQL query to find for each month and country, the number of transactions and their total amount, the number of approved transactions and their total amount.

The query result format is in the following example:

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

The query result format is in the following example:

```
Transactions table:
+------+---------+----------+--------+------------+
| id   | country | state    | amount | trans_date |
+------+---------+----------+--------+------------+
| 121  | US      | approved | 1000   | 2018-12-18 |
| 122  | US      | declined | 2000   | 2018-12-19 |
| 123  | US      | approved | 2000   | 2019-01-01 |
| 124  | DE      | approved | 2000   | 2019-01-07 |
+------+---------+----------+--------+------------+

Result table:
+----------+---------+-------------+----------------+--------------------+-----------------------+
| month    | country | trans_count | approved_count | trans_total_amount | approved_total_amount |
+----------+---------+-------------+----------------+--------------------+-----------------------+
| 2018-12  | US      | 2           | 1              | 3000               | 1000                  |
| 2019-01  | US      | 1           | 1              | 2000               | 2000                  |
| 2019-01  | DE      | 1           | 1              | 2000               | 2000                  |
+----------+---------+-------------+----------------+--------------------+-----------------------+
```

Difficulty:

Medium

```
60 with cte as(Select *,substring(cast(trans_date as varchar),1,7) as month
61 from Transactions_1193)
62 select [month],
63        country,
64        count(*) as trans_count,
65        sum(case when [state]='approved' then 1 else 0 end) as approved_count,
66        sum(amount) as trans_total_amount,
67        sum(case when [state]='approved' then amount else 0 end )as approved_total_amount from cte
68 group by [month],country
69 order by [month],country desc;
```

100 %

Results | Messages

	month	country	trans_count	approved_count	trans_total_amount	approved_total_amount
1	2018-12	US	2	1	3000	1000
2	2019-01	US	1	1	2000	2000
3	2019-01	DE	1	1	2000	2000

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1194. Tournament Winners

Table: `Players`

```
+-------------+-------+
| Column Name | Type  |
+-------------+-------+
| player_id   | int   |
| group_id    | int   |
+-------------+-------+
player_id is the primary key of this table.
Each row of this table indicates the group of each player.
```

Table: `Matches`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| match_id      | int     |
| first_player  | int     |
| second_player | int     |
| first_score   | int     |
| second_score  | int     |
+---------------+---------+
match_id is the primary key of this table.
Each row is a record of a match, first_player and second_player contain the player_id of each match.
first_score and second_score contain the number of points of the first_player and second_player respectively.
You may assume that, in each match, players belongs to the same group.
```

The winner in each group is the player who scored the maximum total points within the group. In the case of a tie, the **lowest** player_id wins.

Write an SQL query to find the winner in each group.

The query result format is in the following example:

```
Players table:
+-----------+------------+
| player_id | group_id   |
+-----------+------------+
| 15        | 1          |
| 25        | 1          |
| 30        | 1          |
| 45        | 1          |
| 10        | 2          |
| 35        | 2          |
| 50        | 2          |
| 20        | 3          |
| 40        | 3          |
+-----------+------------+
```


LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```
Matches table:
+------------+--------------+---------------+-------------+--------------+
| match_id   | first_player | second_player | first_score | second_score |
+------------+--------------+---------------+-------------+--------------+
| 1          | 15           | 45            | 3           | 0            |
| 2          | 30           | 25            | 1           | 2            |
| 3          | 30           | 15            | 2           | 0            |
| 4          | 40           | 20            | 5           | 2            |
| 5          | 35           | 50            | 1           | 1            |
+------------+--------------+---------------+-------------+--------------+

Result table:
+-----------+------------+
| group_id  | player_id  |
+-----------+------------+
| 1         | 15         |
| 2         | 35         |
| 3         | 40         |
+-----------+------------+
```

Difficulty:

Hard

```sql
With T as (Select first_player as player_id,first_score as score
from Matches_1194
union
Select second_player as player_id,second_score as score
from Matches_1194),

T1 as (Select T.player_id,SUM(T.score) as score
from T
group by T.player_id),

T2 as (Select Players_1194.*,isnull(T1.score,0) as score
from Players_1194 left join T1 on Players_1194.player_id = T1.player_id),

T3 as (select T2.*, Row_number() over (partition by group_id order by score desc,player_id asc) as rnk from T2)

Select player_id,group_id
from T3
where rnk=1
order by player_id
```

100 %

Results | Messages

	player_id	group_id
1	15	1
2	35	2
3	40	3

```sql
with cte as (select p.group_id,
      p.player_id,
      sum(case when p.player_id=m.first_player then m.first_score
      when p.player_id=m.second_player then m.second_score else 0 end) as total_score
from Players_1194 p
left join Matches_1194 m
on p.player_id =m.first_player
or p.player_id =m.second_player
group by p.group_id,player_id),
cte2 as(select *, dense_rank() over (partition by group_id order by total_score desc,player_id ) as rnk
from cte)
select player_id,group_id from cte2 where rnk=1;
```

100 %

Results | Messages

	player_id	group_id
1	15	1
2	35	2
3	40	3

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1204. Last Person to Fit in the Elevator

Table: `Queue`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| person_id   | int     |
| person_name | varchar |
| weight      | int     |
| turn        | int     |
+-------------+---------+
person_id is the primary key column for this table.
This table has the information about all people waiting for an elevator.
The person_id and turn columns will contain all numbers from 1 to n, where n is the number of rows in the table.
```

The maximum weight the elevator can hold is **1000**.

Write an SQL query to find the `person_name` of the last person who will fit in the elevator without exceeding the weight limit. It is guaranteed that the person who is first in the queue can fit in the elevator.

The query result format is in the following example:

```
Queue table
+-----------+-------------------+--------+------+
| person_id | person_name       | weight | turn |
+-----------+-------------------+--------+------+
| 5         | George Washington | 250    | 1    |
| 3         | John Adams        | 350    | 2    |
| 6         | Thomas Jefferson  | 400    | 3    |
| 2         | Will Johnliams    | 200    | 4    |
| 4         | Thomas Jefferson  | 175    | 5    |
| 1         | James Elephant    | 500    | 6    |
+-----------+-------------------+--------+------+

Result table
+-------------------+
| person_name       |
+-------------------+
| Thomas Jefferson  |
+-------------------+

Queue table is ordered by turn in the example for simplicity.
In the example George Washington(id 5), John Adams(id 3) and Thomas Jefferson(id 6) will enter the elevator as their weight sum is 250 + 350
Thomas Jefferson(id 6) is the last person to fit in the elevator because he has the last turn in these three people.
```

Difficulty:

Medium

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```sql
with cte as (select *,SUM([weight]) OVER (ORDER BY turn) AS cumulative_sum
from Queue_1204),

cte1 as (select person_name,cumulative_sum,ROW_NUMBER() over(order by cumulative_sum desc) as rnk from cte where cumulative_sum <=1000)

select person_name
from cte1
where rnk=1;

---------------------

with cte3 as (select *,SUM([weight]) OVER (ORDER BY turn) AS cumulative_sum
from Queue_1204),

cte4 as (select *,isnull(lead(cumulative_sum) over (order by turn),cumulative_sum) as lead_sum from cte3)

Select person_name
from cte4
where cumulative_sum<=1000 and lead_sum >1000;
```

100 %

Results | Messages

	person_name
1	Thomas Jefferson

	person_name
1	Thomas Jefferson

1211. Queries Quality and Percentage

Table: `Queries`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| query_name  | varchar |
| result      | varchar |
| position    | int     |
| rating      | int     |
+-------------+---------+
There is no primary key for this table, it may have duplicate rows.
This table contains information collected from some queries on a database.
The position column has a value from 1 to 500.
The rating column has a value from 1 to 5. Query with rating less than 3 is a poor query.
```

We define query `quality` as:

> The average of the ratio between query rating and its position.

We also define `poor query percentage` as:

> The percentage of all queries with rating less than 3.

Write an SQL query to find each `query_name`, the `quality` and `poor_query_percentage`.

Both `quality` and `poor_query_percentage` should be **rounded to 2 decimal places**.

The query result format is in the following example:

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

Write an SQL query to find each `query_name`, the `quality` and `poor_query_percentage`.

Both `quality` and `poor_query_percentage` should be **rounded to 2 decimal places**.

The query result format is in the following example:

```
Queries table:
+------------+-------------------+----------+--------+
| query_name | result            | position | rating |
+------------+-------------------+----------+--------+
| Dog        | Golden Retriever  | 1        | 5      |
| Dog        | German Shepherd   | 2        | 5      |
| Dog        | Mule              | 200      | 1      |
| Cat        | Shirazi           | 5        | 2      |
| Cat        | Siamese           | 3        | 3      |
| Cat        | Sphynx            | 7        | 4      |
+------------+-------------------+----------+--------+

Result table:
+------------+---------+-----------------------+
| query_name | quality | poor_query_percentage |
+------------+---------+-----------------------+
| Dog        | 2.50    | 33.33                 |
| Cat        | 0.66    | 33.33                 |
+------------+---------+-----------------------+

Dog queries quality is ((5 / 1) + (5 / 2) + (1 / 200)) / 3 = 2.50
Dog queries poor_ query_percentage is (1 / 3) * 100 = 33.33

Cat queries quality equals ((2 / 5) + (3 / 3) + (4 / 7)) / 3 = 0.66
Cat queries poor_ query_percentage is (1 / 3) * 100 = 33.33
```

```sql
select query_name,
       round(avg(cast (rating as float)/position),2) as quality,
       round(avg(case when rating<3 then 1.0 else 0.0 end)*100,2) as poor_query_percentage
from Queries_1211
group by query_name
```

100 %

Results | Messages

	query_name	quality	poor_query_percentage
1	Cat	0.66	33.330000
2	Dog	2.5	33.330000

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1212. Team Scores in Football Tournament

Table: `Teams`

```
+---------------+----------+
| Column Name   | Type     |
+---------------+----------+
| team_id       | int      |
| team_name     | varchar  |
+---------------+----------+
team_id is the primary key of this table.
Each row of this table represents a single football team.
```

Table: `Matches`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| match_id      | int     |
| host_team     | int     |
| guest_team    | int     |
| host_goals    | int     |
| guest_goals   | int     |
+---------------+---------+
match_id is the primary key of this table.
Each row is a record of a finished match between two different teams.
Teams host_team and guest_team are represented by their IDs in the teams table (team_id) and they scored host_goals and guest_goals goals re
```

You would like to compute the scores of all teams after all matches. Points are awarded as follows:

- A team receives three points if they win a match (Score strictly more goals than the opponent team).
- A team receives one point if they draw a match (Same number of goals as the opponent team).
- A team receives no points if they lose a match (Score less goals than the opponent team).

Write an SQL query that selects the **team_id**, **team_name** and **num_points** of each team in the tournament after all described matches. Result table should be ordered by **num_points** (decreasing order). In case of a tie, order the records by **team_id** (increasing order).

The query result format is in the following example:

```
Teams table:
+-----------+--------------+
| team_id   | team_name    |
+-----------+--------------+
| 10        | Leetcode FC  |
| 20        | NewYork FC   |
| 30        | Atlanta FC   |
| 40        | Chicago FC   |
| 50        | Toronto FC   |
+-----------+--------------+

Matches table:
+------------+--------------+---------------+-------------+--------------+
| match_id   | host_team    | guest_team    | host_goals  | guest_goals  |
+------------+--------------+---------------+-------------+--------------+
| 1          | 10           | 20            | 3           | 0            |
| 2          | 30           | 10            | 2           | 2            |
| 3          | 10           | 50            | 5           | 1            |
| 4          | 20           | 30            | 1           | 0            |
| 5          | 50           | 30            | 1           | 0            |
+------------+--------------+---------------+-------------+--------------+
```


LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```
Matches table:
+------------+--------------+--------------+-------------+--------------+
| match_id   | host_team    | guest_team   | host_goals  | guest_goals  |
+------------+--------------+--------------+-------------+--------------+
| 1          | 10           | 20           | 3           | 0            |
| 2          | 30           | 10           | 2           | 2            |
| 3          | 10           | 50           | 5           | 1            |
| 4          | 20           | 30           | 1           | 0            |
| 5          | 50           | 30           | 1           | 0            |
+------------+--------------+--------------+-------------+--------------+

Result table:
+------------+--------------+---------------+
| team_id    | team_name    | num_points    |
+------------+--------------+---------------+
| 10         | Leetcode FC  | 7             |
| 20         | NewYork FC   | 3             |
| 50         | Toronto FC   | 3             |
| 30         | Atlanta FC   | 1             |
| 40         | Chicago FC   | 0             |
+------------+--------------+---------------+
```

Difficulty:

Medium

```sql
select t.team_id,t.team_name, Sum(case when t.team_id = m.host_team and m.host_goals>m.guest_goals then 3
                when t.team_id = m.guest_team and m.guest_goals>m.host_goals then 3
                when (((t.team_id = m.guest_team) or (t.team_id = m.host_team)) and (m.host_goals=m.guest_goals)) then 1
                else 0 end) as points
from Teams_1212 t
left join Matches_1212 m
on t.team_id=m.host_team
or t.team_id=m.guest_team
group by t.team_id,t.team_name
order by points desc,t.team_id;
```

100 %

Results | Messages

	team_id	team_name	points
1	10	Leetcode FC	7
2	20	NewYork FC	3
3	50	Toronto FC	3
4	30	Atlanta FC	1
5	40	Chicago FC	0

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1225. Report Contiguous Dates

Table: `Failed`

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| fail_date    | date    |
+--------------+---------+
Primary key for this table is fail_date.
Failed table contains the days of failed tasks.
```

Table: `Succeeded`

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| success_date | date    |
+--------------+---------+
Primary key for this table is success_date.
Succeeded table contains the days of succeeded tasks.
```

A system is running one task **every day**. Every task is independent of the previous tasks. The tasks can fail or succeed.

Write an SQL query to generate a report of `period_state` for each continuous interval of days in the period from **2019-01-01** to **2019-12-31**.

`period_state` is *'failed'* if tasks in this interval failed or *'succeeded'* if tasks in this interval succeeded. Interval of days are retrieved as `start_date` and `end_date`.

Order result by `start_date`.

The query result format is in the following example:

```
Failed table:
+-------------------+
| fail_date         |
+-------------------+
| 2018-12-28        |
| 2018-12-29        |
| 2019-01-04        |
| 2019-01-05        |
+-------------------+

Succeeded table:
+-------------------+
| success_date      |
+-------------------+
| 2018-12-30        |
| 2018-12-31        |
| 2019-01-01        |
| 2019-01-02        |
| 2019-01-03        |
| 2019-01-06        |
+-------------------+


Result table:
+--------------+--------------+--------------+
| period_state | start_date   | end_date     |
+--------------+--------------+--------------+
| succeeded    | 2019-01-01   | 2019-01-03   |
| failed       | 2019-01-04   | 2019-01-05   |
| succeeded    | 2019-01-06   | 2019-01-06   |
+--------------+--------------+--------------+
```

```
The report ignored the system state in 2018 as we care about the system in the period 2019-01-01 to 2019-12-31.
From 2019-01-01 to 2019-01-03 all tasks succeeded and the system state was "succeeded".
From 2019-01-04 to 2019-01-05 all tasks failed and system state was "failed".
From 2019-01-06 to 2019-01-06 all tasks succeeded and system state was "succeeded".
```

Difficulty:

Hard

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```sql
With cte as (select fail_date as [date], 'failed' as state
from Failed_1225 where year(fail_date)=2019
union all
select success_date as [date], 'succeeded' as state
from Succeeded_1225 where year(success_date)=2019),

cte1 as (Select *,isnull(lag(state) over (order by [date]),[state]) as prev_state
from
cte),

cte2 as (select [date],[state],case when [state]=prev_state then 0 else 1 end as has_changed
from cte1),

cte3 as (select [date],[state] ,sum(has_changed) over (order by [date]) as grp
from cte2)

select [state] as period_state,min([date]) as start_date,max([date]) as end_date
from
cte3
group by grp,[state]
order by start_date;
```

Results | Messages

	period_state	start_date	end_date
1	succeeded	2019-01-01	2019-01-03
2	failed	2019-01-04	2019-01-05
3	succeeded	2019-01-06	2019-01-06

```sql
SELECT [status], MIN([date]) AS start_date, MAX([date]) AS end_date
FROM (
    SELECT *, SUM(has_status_changed) OVER (ORDER BY date) AS grp
    FROM (
        SELECT *, CASE WHEN status = prev_state THEN 0 ELSE 1 END AS has_status_changed
        FROM (
            SELECT *, LAG(status) OVER (ORDER BY date) AS prev_state
            FROM (
                SELECT fail_date AS date, 'failed' AS [status] FROM Failed_1225 WHERE YEAR(fail_date) = 2019
                UNION ALL
                SELECT success_date AS date, 'succeeded' AS [status] FROM Succeeded_1225 WHERE YEAR(success_date) = 2019
            ) a
        ) a
    ) a
) a
GROUP BY [status], grp;


-----
With cte as (select fail_date as [date], 'failed' as state
from Failed_1225 where year(fail_date)=2019
```

Results | Messages

	status	start_date	end_date
1	succeeded	2019-01-01	2019-01-03
2	failed	2019-01-04	2019-01-05
3	succeeded	2019-01-06	2019-01-06

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1241. Number of Comments per Post

Table: `Submissions`

```
+---------------+----------+
| Column Name   | Type     |
+---------------+----------+
| sub_id        | int      |
| parent_id     | int      |
+---------------+----------+
There is no primary key for this table, it may have duplicate rows.
Each row can be a post or comment on the post.
parent_id is null for posts.
parent_id for comments is sub_id for another post in the table.
```

Write an SQL query to find number of comments per each post.

Result table should contain `post_id` and its corresponding `number_of_comments`, and must be sorted by `post_id` in ascending order.

`Submissions` may contain duplicate comments. You should count the number of **unique comments** per post.

`Submissions` may contain duplicate posts. You should treat them as one post.

The query result format is in the following example:

```
Submissions table:
+---------+------------+
| sub_id  | parent_id  |
+---------+------------+
| 1       | Null       |
| 2       | Null       |
| 1       | Null       |
| 12      | Null       |
| 3       | 1          |
| 5       | 2          |
| 3       | 1          |
| 4       | 1          |
| 9       | 1          |
| 10      | 2          |
| 6       | 7          |
+---------+------------+

Result table:
+---------+--------------------+
| post_id | number_of_comments |
+---------+--------------------+
| 1       | 3                  |
| 2       | 2                  |
| 12      | 0                  |
+---------+--------------------+

The post with id 1 has three comments in the table with id 3, 4 and 9. The comment with id 3 is repeated in the table, we counted it only on
The post with id 2 has two comments in the table with id 5 and 10.
The post with id 12 has no comments in the table.
The comment with id 6 is a comment on a deleted post with id 7 so we ignored it.
```

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

```
79
80  Select distinct sub_id as post_id,isnull(number_of_comments,0) as number_of_comments
81  from Submissions_1241 s
82  LEFT JOIN(Select T.parent_id,count(*) as number_of_comments
83  from (select distinct sub_id, parent_id
84  from Submissions_1241) T
85  where parent_id is not null
86  group by T.parent_id) T1
87  on s.sub_id = T1.parent_id
88  where s.parent_id is null
89  order by post_id;
90
```

100 %

Results | Messages

	post_id	number_of_comments
1	1	3
2	2	2
3	12	0

1251. Average Selling Price

Table: `Prices`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| product_id    | int     |
| start_date    | date    |
| end_date      | date    |
| price         | int     |
+---------------+---------+
(product_id, start_date, end_date) is the primary key for this table.
Each row of this table indicates the price of the product_id in the period from start_date to end_date.
For each product_id there will be no two overlapping periods. That means there will be no two intersecting periods for the same product_id.
```

Table: `UnitsSold`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| product_id    | int     |
| purchase_date | date    |
| units         | int     |
+---------------+---------+
There is no primary key for this table, it may contain duplicates.
Each row of this table indicates the date, units and product_id of each product sold.
```


LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

Write an SQL query to find the average selling price for each product.

`average_price` should be **rounded to 2 decimal places**.

The query result format is in the following example:

```
Prices table:
+------------+------------+------------+--------+
| product_id | start_date | end_date   | price  |
+------------+------------+------------+--------+
| 1          | 2019-02-17 | 2019-02-28 | 5      |
| 1          | 2019-03-01 | 2019-03-22 | 20     |
| 2          | 2019-02-01 | 2019-02-20 | 15     |
| 2          | 2019-02-21 | 2019-03-31 | 30     |
+------------+------------+------------+--------+

UnitsSold table:
+------------+---------------+-------+
| product_id | purchase_date | units |
+------------+---------------+-------+
| 1          | 2019-02-25    | 100   |
| 1          | 2019-03-01    | 15    |
| 2          | 2019-02-10    | 200   |
| 2          | 2019-03-22    | 30    |
+------------+---------------+-------+

Result table:
+------------+---------------+
| product_id | average_price |
+------------+---------------+
| 1          | 6.96          |
| 2          | 16.96         |
+------------+---------------+
Average selling price = Total Price of Product / Number of products sold.
Average selling price for product 1 = ((100 * 5) + (15 * 20)) / 115 = 6.96
Average selling price for product 2 = ((200 * 15) + (30 * 30)) / 230 = 16.96
```

```sql
select p.product_id,round(cast(sum(p.price*u.units)as float)/sum(units),2) as average_price
from UnitsSold_1251 u
left join Prices_1251 p
on p.product_id = u.product_id
and
u.purchase_date between p.start_date and p.end_date
group by p.product_id;
```

100 %

Results | Messages

	product_id	average_price
1	1	6.96
2	2	16.96

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1264. Page Recommendations

SQL Schema ›
Table: `Friendship`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| user1_id      | int     |
| user2_id      | int     |
+---------------+---------+
(user1_id, user2_id) is the primary key for this table.
Each row of this table indicates that there is a friendship relation between user1_id and user2_id.
```

Table: `Likes`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| user_id     | int     |
| page_id     | int     |
+-------------+---------+
(user_id, page_id) is the primary key for this table.
Each row of this table indicates that user_id likes page_id.
```

Write an SQL query to recommend pages to the user with `user_id` = 1 using the pages that your friends liked. It should not recommend pages you already liked.

Return result table in any order without duplicates.

The query result format is in the following example:

```
Friendship table:
+----------+----------+
| user1_id | user2_id |
+----------+----------+
| 1        | 2        |
| 1        | 3        |
| 1        | 4        |
| 2        | 3        |
| 2        | 4        |
| 2        | 5        |
| 6        | 1        |
+----------+----------+

Likes table:
+---------+---------+
| user_id | page_id |
+---------+---------+
| 1       | 88      |
| 2       | 23      |
| 3       | 24      |
| 4       | 56      |
| 5       | 11      |
| 6       | 33      |
| 2       | 77      |
| 3       | 77      |
| 6       | 88      |
+---------+---------+
```

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```
Result table:
+------------------+
| recommended_page |
+------------------+
| 23               |
| 24               |
| 56               |
| 33               |
| 77               |
+------------------+
User one is friend with users 2, 3, 4 and 6.
Suggested pages are 23 from user 2, 24 from user 3, 56 from user 3 and 33 from user 6.
Page 77 is suggested from both user 2 and user 3.
Page 88 is not suggested because user 1 already likes it.
```

```sql
select distinct l.page_id as recommended_page from(select user1_friends from(select case when user1_id=1 then user2_id
            when user2_id=1 then user1_id
            end as user1_friends
from Friendship_1264) T
where T.user1_friends is not null) T1
left join Likes_1264 l
on T1.user1_friends=l.[user_id]

except

select page_id as recommended_page from Likes_1264 where [user_id] = 1;
```

100 %

Results | Messages

	recommended_page
1	23
2	24
3	33
4	56
5	77

1270. All People Report to the Given Manager

SQL Schema ›

Table: `Employees`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| employee_id   | int     |
| employee_name | varchar |
| manager_id    | int     |
+---------------+---------+
employee_id is the primary key for this table.
Each row of this table indicates that the employee with ID employee_id and name employee_name reports his work to his/her direct manager wit
The head of the company is the employee with employee_id = 1.
```

Write an SQL query to find `employee_id` of all employees that directly or indirectly report their work to the head of the company.

The indirect relation between managers will not exceed 3 managers as the company is small.

Return result table in any order without duplicates.

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

The query result format is in the following example:

```
Employees table:
+-------------+---------------+------------+
| employee_id | employee_name | manager_id |
+-------------+---------------+------------+
| 1           | Boss          | 1          |
| 3           | Alice         | 3          |
| 2           | Bob           | 1          |
| 4           | Daniel        | 2          |
| 7           | Luis          | 4          |
| 8           | Jhon          | 3          |
| 9           | Angela        | 8          |
| 77          | Robert        | 1          |
+-------------+---------------+------------+

Result table:
+-------------+
| employee_id |
+-------------+
| 2           |
| 77          |
| 4           |
| 7           |
+-------------+

The head of the company is the employee with employee_id 1.
The employees with employee_id 2 and 77 report their work directly to the head of the company.
The employee with employee_id 4 report his work indirectly to the head of the company 4 --> 2 --> 1.
The employee with employee_id 7 report his work indirectly to the head of the company 7 --> 4 --> 2 --> 1.
The employees with employee_id 3, 8 and 9 don't report their work to head of company directly or indirectly.
```

```sql
with cte as (select employee_id
from Employees_1270
where manager_id=1 and employee_id<>manager_id),

cte1 as (select employee_id from Employees_1270 where manager_id in (select * from cte)),

cte2 as (select employee_id,employee_name from Employees_1270 where manager_id in (select * from cte1))

select employee_id from cte
union
select employee_id from cte1
union
select employee_id from cte2
```

100 %

Results | Messages

	employee_id
1	2
2	4
3	7
4	77

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```sql
WITH EmployeeHierarchy AS (
    -- Anchor member: Start with employees who report directly to the head (employee_id = 1)
    SELECT employee_id, manager_id
    FROM Employees_1270
    WHERE manager_id = 1 AND employee_id != 1

    UNION ALL

    -- Recursive member: Find employees who report to the employees found in the previous step
    SELECT e.employee_id, e.manager_id
    FROM Employees_1270 e
    INNER JOIN EmployeeHierarchy eh ON e.manager_id = eh.employee_id
)
-- Select the distinct employee_ids from the CTE
SELECT DISTINCT employee_id
FROM EmployeeHierarchy
ORDER BY employee_id;
```

100 %

Results | Messages

	employee_id
1	2
2	4
3	7
4	77

1280. Students and Examinations

SQL Schema

Table: `Students`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| student_id    | int     |
| student_name  | varchar |
+---------------+---------+
student_id is the primary key for this table.
Each row of this table contains the ID and the name of one student in the school.
```

Table: `Subjects`

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| subject_name | varchar |
+--------------+---------+
subject_name is the primary key for this table.
Each row of this table contains a name of one subject in the school.
```

Table: `Examinations`

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

Table: `Examinations`

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| student_id   | int     |
| subject_name | varchar |
+--------------+---------+
There is no primary key for this table. It may contain duplicates.
Each student from Students table takes every course from Subjects table.
Each row of this table indicates that a student with ID student_id attended the exam of subject_name.
```

Write an SQL query to find the number of times each student attended each exam.

Order the result table by `student_id` and `subject_name`.

The query result format is in the following example:

```
Students table:
+------------+--------------+
| student_id | student_name |
+------------+--------------+
| 1          | Alice        |
| 2          | Bob          |
| 13         | John         |
| 6          | Alex         |
+------------+--------------+
```

```
Subjects table:
+--------------+
| subject_name |
+--------------+
| Math         |
| Physics      |
| Programming  |
+--------------+
Examinations table:
+------------+--------------+
| student_id | subject_name |
+------------+--------------+
| 1          | Math         |
| 1          | Physics      |
| 1          | Programming  |
| 2          | Programming  |
| 1          | Physics      |
| 1          | Math         |
| 13         | Math         |
| 13         | Programming  |
| 13         | Physics      |
| 2          | Math         |
| 1          | Math         |
+------------+--------------+
```


LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```
Result table:
+------------+--------------+--------------+----------------+
| student_id | student_name | subject_name | attended_exams |
+------------+--------------+--------------+----------------+
| 1          | Alice        | Math         | 3              |
| 1          | Alice        | Physics      | 2              |
| 1          | Alice        | Programming  | 1              |
| 2          | Bob          | Math         | 1              |
| 2          | Bob          | Physics      | 0              |
| 2          | Bob          | Programming  | 1              |
| 6          | Alex         | Math         | 0              |
| 6          | Alex         | Physics      | 0              |
| 6          | Alex         | Programming  | 0              |
| 13         | John         | Math         | 1              |
| 13         | John         | Physics      | 1              |
| 13         | John         | Programming  | 1              |
+------------+--------------+--------------+----------------+
The result table should contain all students and all subjects.
Alice attended Math exam 3 times, Physics exam 2 times and Programming exam 1 time.
Bob attended Math exam 1 time, Programming exam 1 time and didn't attend the Physics exam.
Alex didn't attend any exam.
John attended Math exam 1 time, Physics exam 1 time and Programming exam 1 time.
```

Difficulty:

Easy

```
147 Select T1.student_id,T1.student_name,T1.subject_name,isnull(T.attended_exams,0)
148 from (select student_id,subject_name,count(student_id) as attended_exams
149 from Examinations_1280
150 group by student_id,subject_name) as T
151 right join
152 (select *
153 from Students_1280
154 cross join Subjects_1280) as T1
155 on T.student_id=T1.student_id and T.subject_name=T1.subject_name
156 order by T1.student_id,T1.student_name;
157
158
```

100 %

Results | Messages

	student_id	student_name	subject_name	(No column name)
1	1	Alice	Math	3
2	1	Alice	Physics	2
3	1	Alice	Programming	1
4	2	Bob	Math	1
5	2	Bob	Physics	0
6	2	Bob	Programming	1
7	6	Alex	Math	0
8	6	Alex	Physics	0
9	6	Alex	Programming	0
10	13	John	Math	1
11	13	John	Physics	1
12	13	John	Programming	1

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1285. Find the Start and End Number of Continuous Ranges

SQL Schema ›
Table: `Logs`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| log_id        | int     |
+---------------+---------+
id is the primary key for this table.
Each row of this table contains the ID in a log Table.
```

Since some IDs have been removed from `Logs`. Write an SQL query to find the start and end number of continuous ranges in table `Logs`.

Order the result table by `start_id`.

The query result format is in the following example:

```
Logs table:
+------------+
| log_id     |
+------------+
| 1          |
| 2          |
| 3          |
| 7          |
| 8          |
| 10         |
+------------+
```

```
Result table:
+------------+--------------+
| start_id   | end_id       |
+------------+--------------+
| 1          | 3            |
| 7          | 8            |
| 10         | 10           |
+------------+--------------+
The result table should contain all ranges in table Logs.
From 1 to 3 is contained in the table.
From 4 to 6 is missing in the table
From 7 to 8 is contained in the table.
Number 9 is missing in the table.
Number 10 is contained in the table.
```

Difficulty:

Medium

```
select min(log_id) as start_id,max(log_id) as end_id
from (select T2.*,sum(grp) over (order by log_id) as cum_sum
from (select T1.*,case when log_id-1= prev_id then 0 else 1 end as grp
from (select log_id,lag(log_id) over (order by log_id) as prev_id
from Logs_1285) T1) T2 ) T3
group by T3.cum_sum
```

100 %

Results | Messages

	start_id	end_id
1	1	3
2	7	8
3	10	10

1294. Weather Type in Each Country

SQL Schema ›

Table: `Countries`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| country_id    | int     |
| country_name  | varchar |
+---------------+---------+
country_id is the primary key for this table.
Each row of this table contains the ID and the name of one country.
```

Table: `Weather`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| country_id    | int     |
| weather_state | varchar |
| day           | date    |
+---------------+---------+
(country_id, day) is the primary key for this table.
Each row of this table indicates the weather state in a country for one day.
```

Write an SQL query to find the type of weather in each country for November 2019.

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

The type of weather is **Cold** if the average `weather_state` is less than or equal 15, **Hot** if the average `weather_state` is greater than or equal 25 and **Warm** otherwise.

Return result table in any order.

The query result format is in the following example:

```
Countries table:
+------------+--------------+
| country_id | country_name |
+------------+--------------+
| 2          | USA          |
| 3          | Australia    |
| 7          | Peru         |
| 5          | China        |
| 8          | Morocco      |
| 9          | Spain        |
+------------+--------------+
Weather table:
+------------+---------------+------------+
| country_id | weather_state | day        |
+------------+---------------+------------+
| 2          | 15            | 2019-11-01 |
| 2          | 12            | 2019-10-28 |
| 2          | 12            | 2019-10-27 |
| 3          | -2            | 2019-11-10 |
| 3          | 0             | 2019-11-11 |
| 3          | 3             | 2019-11-12 |
| 5          | 16            | 2019-11-07 |
| 5          | 18            | 2019-11-09 |
| 5          | 21            | 2019-11-23 |
| 7          | 25            | 2019-11-28 |
| 7          | 22            | 2019-12-01 |
| 7          | 20            | 2019-12-02 |
| 8          | 25            | 2019-11-05 |
| 8          | 27            | 2019-11-15 |
| 8          | 31            | 2019-11-25 |
| 9          | 7             | 2019-10-23 |
| 9          | 3             | 2019-12-23 |
+------------+---------------+------------+
Result table:
+--------------+--------------+
| country_name | weather_type |
+--------------+--------------+
| USA          | Cold         |
| Austraila    | Cold         |
| Peru         | Hot          |
| China        | Warm         |
| Morocco      | Hot          |
+--------------+--------------+
Average weather_state in USA in November is (15) / 1 = 15 so weather type is Cold.
Average weather_state in Austraila in November is (-2 + 0 + 3) / 3 = 0.333 so weather type is Cold.
Average weather_state in Peru in November is (25) / 1 = 25 so weather type is Hot.
Average weather_state in China in November is (16 + 18 + 21) / 3 = 18.333 so weather type is Warm.
Average weather_state in Morocco in November is (25 + 27 + 31) / 3 = 27.667 so weather type is Hot.
```


```
Average weather_state in USA in November is (15) / 1 = 15 so weather type is Cold.
Average weather_state in Austraila in November is (-2 + 0 + 3) / 3 = 0.333 so weather type is Cold.
Average weather_state in Peru in November is (25) / 1 = 25 so weather type is Hot.
Average weather_state in China in November is (16 + 18 + 21) / 3 = 18.333 so weather type is Warm.
Average weather_state in Morocco in November is (25 + 27 + 31) / 3 = 27.667 so weather type is Hot.
We know nothing about average weather_state in Spain in November so we don't include it in the result table.
```


Difficulty:

Easy

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

```sql
select c1.country_name,weather_type
from
(select country_id,case when avg(weather_state)<=15 then 'cold'
                        when avg(weather_state)>=25 then 'hot'
                        else 'warm' end as weather_type
from Weather_1294
where month([day])=11 and year([day])=2019
group by country_id) c
left join Countries_1294 c1
on c.country_id=c1.country_id;
```

100 %

Results | Messages

	country_name	weather_type
1	USA	cold
2	Australia	cold
3	China	warm
4	Peru	hot
5	Morocco	hot

1303. Find the Team Size

SQL Schema ›
Table: `Employee`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| employee_id   | int     |
| team_id       | int     |
+---------------+---------+
employee_id is the primary key for this table.
Each row of this table contains the ID of each employee and their respective team.
```

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

Return result table in any order.

The query result format is in the following example:

```
Employee Table:
+-------------+------------+
| employee_id | team_id    |
+-------------+------------+
|     1       |     8      |
|     2       |     8      |
|     3       |     8      |
|     4       |     7      |
|     5       |     9      |
|     6       |     9      |
+-------------+------------+
```


```
Result table:
+-------------+------------+
| employee_id | team_size  |
+-------------+------------+
|     1       |     3      |
|     2       |     3      |
|     3       |     3      |
|     4       |     1      |
|     5       |     2      |
|     6       |     2      |
+-------------+------------+
Employees with Id 1,2,3 are part of a team with team_id = 8.
Employees with Id 4 is part of a team with team_id = 7.
Employees with Id 5,6 are part of a team with team_id = 9.
```

Difficulty:

Easy

```sql
select e.employee_id,t.team_size from Employee_1303 e
left join (select team_id,count(employee_id) as team_size
from Employee_1303
group by team_id) t
on e.team_id=t.team_id;
```

100 %

Results | Messages

	employee_id	team_size
1	1	3
2	2	3
3	3	3
4	4	1
5	5	2
6	6	2

1308. Running Total for Different Genders

SQL Schema ›
Table: `Scores`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| player_name   | varchar |
| gender        | varchar |
| day           | date    |
| score_points  | int     |
+---------------+---------+
(gender, day) is the primary key for this table.
A competition is held between females team and males team.
Each row of this table indicates that a player_name and with gender has scored score_point in someday.
Gender is 'F' if the player is in females team and 'M' if the player is in males team.
```

Write an SQL query to find the total score for each gender at each day.

Order the result table by gender and day

The query result format is in the following example:

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```
Scores table:
+-------------+--------+------------+--------------+
| player_name | gender | day        | score_points |
+-------------+--------+------------+--------------+
| Aron        | F      | 2020-01-01 | 17           |
| Alice       | F      | 2020-01-07 | 23           |
| Bajrang     | M      | 2020-01-07 | 7            |
| Khali       | M      | 2019-12-25 | 11           |
| Slaman      | M      | 2019-12-30 | 13           |
| Joe         | M      | 2019-12-31 | 3            |
| Jose        | M      | 2019-12-18 | 2            |
| Priya       | F      | 2019-12-31 | 23           |
| Priyanka    | F      | 2019-12-30 | 17           |
+-------------+--------+------------+--------------+
Result table:
+--------+------------+-------+
| gender | day        | total |
+--------+------------+-------+
| F      | 2019-12-30 | 17    |
| F      | 2019-12-31 | 40    |
| F      | 2020-01-01 | 57    |
| F      | 2020-01-07 | 80    |
| M      | 2019-12-18 | 2     |
| M      | 2019-12-25 | 13    |
| M      | 2019-12-30 | 26    |
| M      | 2019-12-31 | 29    |
| M      | 2020-01-07 | 36    |
+--------+------------+-------+
```

```
For females team:
First day is 2019-12-30, Priyanka scored 17 points and the total score for the team is 17.
Second day is 2019-12-31, Priya scored 23 points and the total score for the team is 40.
Third day is 2020-01-01, Aron scored 17 points and the total score for the team is 57.
Fourth day is 2020-01-07, Alice scored 23 points and the total score for the team is 80.
For males team:
First day is 2019-12-18, Jose scored 2 points and the total score for the team is 2.
Second day is 2019-12-25, Khali scored 11 points and the total score for the team is 13.
Third day is 2019-12-30, Slaman scored 13 points and the total score for the team is 26.
Fourth day is 2019-12-31, Joe scored 3 points and the total score for the team is 29.
Fifth day is 2020-01-07, Bajrang scored 7 points and the total score for the team is 36.
```

Difficulty:

Medium

```
select gender,[day],sum(score_points) over (partition by gender order by [day]) as total
from Scores_1308
order by gender,[day]
```

100 %

Results | Messages

	gender	day	total
1	F	2019-12-30	17
2	F	2019-12-31	40
3	F	2020-01-01	57
4	F	2020-01-07	80
5	M	2019-12-18	2
6	M	2019-12-25	13
7	M	2019-12-30	26
8	M	2019-12-31	29
9	M	2020-01-07	36

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1321. Restaurant Growth

SQL Schema ›
Table: `Customer`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| customer_id   | int     |
| name          | varchar |
| visited_on    | date    |
| amount        | int     |
+---------------+---------+
(customer_id, visited_on) is the primary key for this table.
This table contains data about customer transactions in a restaurant.
visited_on is the date on which the customer with ID (customer_id) have visited the restaurant.
amount is the total paid by a customer.
```

You are the restaurant owner and you want to analyze a possible expansion (there will be at least one customer every day).

Write an SQL query to compute moving average of how much customer paid in a 7 days window (current day + 6 days before) .

The query result format is in the following example:

Return result table ordered by visited_on.

`average_amount` should be **rounded to 2 decimal places**, all dates are in the format ('YYYY-MM-DD').

```
Customer table:
+-------------+--------------+--------------+-------------+
| customer_id | name         | visited_on   | amount      |
+-------------+--------------+--------------+-------------+
| 1           | Jhon         | 2019-01-01   | 100         |
| 2           | Daniel       | 2019-01-02   | 110         |
| 3           | Jade         | 2019-01-03   | 120         |
| 4           | Khaled       | 2019-01-04   | 130         |
| 5           | Winston      | 2019-01-05   | 110         |
| 6           | Elvis        | 2019-01-06   | 140         |
| 7           | Anna         | 2019-01-07   | 150         |
| 8           | Maria        | 2019-01-08   | 80          |
| 9           | Jaze         | 2019-01-09   | 110         |
| 1           | Jhon         | 2019-01-10   | 130         |
| 3           | Jade         | 2019-01-10   | 150         |
+-------------+--------------+--------------+-------------+

Result table:
+--------------+--------------+----------------+
| visited_on   | amount       | average_amount |
+--------------+--------------+----------------+
| 2019-01-07   | 860          | 122.86         |
| 2019-01-08   | 840          | 120            |
| 2019-01-09   | 840          | 120            |
| 2019-01-10   | 1000         | 142.86         |
+--------------+--------------+----------------+

1st moving average from 2019-01-01 to 2019-01-07 has an average_amount of (100 + 110 + 120 + 130 + 110 + 140 + 150)/7 = 122.86
2nd moving average from 2019-01-02 to 2019-01-08 has an average_amount of (110 + 120 + 130 + 110 + 140 + 150 + 80)/7 = 120
3rd moving average from 2019-01-03 to 2019-01-09 has an average_amount of (120 + 130 + 110 + 140 + 150 + 80 + 110)/7 = 120
4th moving average from 2019-01-04 to 2019-01-10 has an average_amount of (130 + 110 + 140 + 150 + 80 + 110 + 130 + 150)/7 = 142.86
```


LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```sql
--window analytic functions
WITH MovingAverage AS (
    SELECT
        visited_on,
        SUM(amount) OVER (
            ORDER BY visited_on
            ROWS BETWEEN 6 PRECEDING AND CURRENT ROW
        ) AS total_amount,
        ROUND(AVG(CAST(amount AS FLOAT)) OVER (
            ORDER BY visited_on
            ROWS BETWEEN 6 PRECEDING AND CURRENT ROW
        ), 2) AS average_amount
    FROM (Select visited_on,Sum(amount) as amount from Customer_1321 group by visited_on) S
)
SELECT
    visited_on,
    total_amount AS amount,
    average_amount
FROM MovingAverage
where visited_on>=(select dateadd(day,6,min(visited_on)) from Customer_1321)
ORDER BY visited_on;

-----
--using join
```

Results Messages

	visited_on	amount	average_amount
1	2019-01-07	860	122.86
2	2019-01-08	840	120
3	2019-01-09	840	120
4	2019-01-10	1000	142.86

```sql
-----
--using join
with cstmr as (select visited_on,sum(amount) as amount
from Customer_1321
group by visited_on)
select c1.visited_on,sum(c2.amount) as amount,round(avg(cast (c2.amount as float)),2) as average_amount
from
(Select visited_on from cstmr where visited_on>=(select dateadd(day,6,min(visited_on)) from Customer_1321)) c1
left join cstmr as c2
on c2.visited_on>=dateadd(day,-6,c1.visited_on)
and c1.visited_on>=c2.visited_on
group by c1.visited_on


--using sub query/corelated query
Select distinct visited_on,
(Select sum(amount) as amount from Customer 1321 c2 where c2.visited on>=dateadd(day,-6,c1.visited on)and c1.visited
```

Results Messages

	visited_on	amount	average_amount
1	2019-01-07	860	122.86
2	2019-01-08	840	120
3	2019-01-09	840	120
4	2019-01-10	1000	142.86

```sql
--using sub query/corelated query
Select distinct visited_on,
(Select sum(amount) as amount from Customer_1321 c2 where c2.visited_on>=dateadd(day,-6,c1.visited_on)and c1.visited_on>=c2.visited_on)-- this is inner query
as amount,
Round(Cast ((Select sum(amount) as amount from Customer_1321 c2 where c2.visited_on>=dateadd(day,-6,c1.visited_on)and c1.visited_on>=c2.visited_on) as float)/7,2)
as average_amount
from Customer_1321 c1
where c1.visited_on>=(select dateadd(day,6,min(visited_on))from Customer_1321)
group by visited_on
order by visited_on
```

Results Messages

	visited_on	amount	average_amount
1	2019-01-07	860	122.86
2	2019-01-08	840	120
3	2019-01-09	840	120
4	2019-01-10	1000	142.86

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1322. 1322. Ads Performance

SQL Schema ›
Table: `Ads`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| ad_id         | int     |
| user_id       | int     |
| action        | enum    |
+---------------+---------+
(ad_id, user_id) is the primary key for this table.
Each row of this table contains the ID of an Ad, the ID of a user and the action taken by this user regarding this Ad.
The action column is an ENUM type of ('Clicked', 'Viewed', 'Ignored').
```

A company is running Ads and wants to calculate the performance of each Ad.

Performance of the Ad is measured using Click-Through Rate (CTR) where:

$$CTR = \begin{cases} 0, & \text{if Ad total clicks} + \text{Ad total views} = 0 \\ \frac{\text{Ad total clicks}}{\text{Ad total clicks} + \text{Ad total views}} \times 100, & \text{otherwise} \end{cases}$$

Write an SQL query to find the `ctr` of each Ad.

Round `ctr` to 2 decimal points. **Order** the result table by `ctr` in descending order and by `ad_id` in ascending order in case of a tie.

The query result format is in the following example:

```
Ads table:
+-------+---------+---------+
| ad_id | user_id | action  |
+-------+---------+---------+
| 1     | 1       | Clicked |
| 2     | 2       | Clicked |
| 3     | 3       | Viewed  |
| 5     | 5       | Ignored |
| 1     | 7       | Ignored |
| 2     | 7       | Viewed  |
| 3     | 5       | Clicked |
| 1     | 4       | Viewed  |
| 2     | 11      | Viewed  |
| 1     | 2       | Clicked |
+-------+---------+---------+
Result table:
+-------+-------+
| ad_id | ctr   |
+-------+-------+
| 1     | 66.67 |
| 3     | 50.00 |
| 2     | 33.33 |
| 5     | 0.00  |
+-------+-------+
for ad_id = 1, ctr = (2/(2+1)) * 100 = 66.67
for ad_id = 2, ctr = (1/(1+2)) * 100 = 33.33
for ad_id = 3, ctr = (1/(1+1)) * 100 = 50.00
for ad_id = 5, ctr = 0.00, Note that ad_id = 5 has no clicks or views.
Note that we don't care about Ignored Ads.
Result table is ordered by the ctr. in case of a tie we order them by ad_id
```

Difficulty:

Easy

```sql
with cte as(select ad_id,
       sum(case when action ='Clicked' then 1.0 else 0.0 end) as Ad_Totalclicks,
       sum(case when action ='Viewed' then 1.0 else 0.0 end) as Ad_Totalviews
from Ads_1322
group by ad_id)
select ad_id,
       case when (Ad_Totalclicks+Ad_Totalviews = 0.0) then 0.0 else round(Ad_Totalclicks/(Ad_Totalclicks+Ad_Totalviews)*100,2) end as ctr
 from cte order by ctr desc,ad_id
```

100 %

Results | Messages

	ad_id	ctr
1	1	66.670000
2	3	50.000000
3	2	33.330000
4	5	0.000000

1327. List the Products Ordered in a Period

SQL Schema ›

Table: `Products`

```
+------------------+---------+
| Column Name      | Type    |
+------------------+---------+
| product_id       | int     |
| product_name     | varchar |
| product_category | varchar |
+------------------+---------+
product_id is the primary key for this table.
This table contains data about the company's products.
```

Table: `Orders`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| product_id    | int     |
| order_date    | date    |
| unit          | int     |
+---------------+---------+
There is no primary key for this table. It may have duplicate rows.
product_id is a foreign key to Products table.
unit is the number of products ordered in order_date.
```

Write an SQL query to get the names of products with greater than or equal to 100 units ordered in February 2020 and their amount.

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

Write an SQL query to get the names of products with greater than or equal to 100 units ordered in February 2020 and their amount.

Return result table in any order.

The query result format is in the following example:

```
Products table:
+-------------+-----------------------+------------------+
| product_id  | product_name          | product_category |
+-------------+-----------------------+------------------+
| 1           | Leetcode Solutions Java Python C++    | Book             |
| 2           | Jewels of Stringology | Book             |
| 3           | HP                    | Laptop           |
| 4           | Lenovo                | Laptop           |
| 5           | Leetcode Kit          | T-shirt          |
+-------------+-----------------------+------------------+

Orders table:
+--------------+--------------+----------+
| product_id   | order_date   | unit     |
+--------------+--------------+----------+
| 1            | 2020-02-05   | 60       |
| 1            | 2020-02-10   | 70       |
| 2            | 2020-01-18   | 30       |
| 2            | 2020-02-11   | 80       |
| 3            | 2020-02-17   | 2        |
| 3            | 2020-02-24   | 3        |
| 4            | 2020-03-01   | 20       |
| 4            | 2020-03-04   | 30       |
| 4            | 2020-03-04   | 60       |
| 5            | 2020-02-25   | 50       |
| 5            | 2020-02-27   | 50       |
| 5            | 2020-03-01   | 50       |
+--------------+--------------+----------+
```

```
Result table:
+--------------------+---------+
| product_name       | unit    |
+--------------------+---------+
| Leetcode Solutions Java Python C++ | 130     |
| Leetcode Kit       | 100     |
+--------------------+---------+

Products with product_id = 1 is ordered in February a total of (60 + 70) = 130.
Products with product_id = 2 is ordered in February a total of 80.
Products with product_id = 3 is ordered in February a total of (2 + 3) = 5.
Products with product_id = 4 was not ordered in February 2020.
Products with product_id = 5 is ordered in February a total of (50 + 50) = 100.
```

Difficulty:

Easy

```
121 with cte as(select product_id,sum(unit) as unit
122 from Orders_1327
123 where month(order_date)=2
124 group by product_id
125 having sum(unit)>=100)
126 select p.product_name,c.unit
127 from cte c
128 left join Products_1327 p
129 on c.product_id=p.product_id;
```

100 %

Results | Messages

	product_name	unit
1	Leetcode Solutions Java Python C++	130
2	Leetcode Kit	100

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

1341. Movie Rating

SQL Schema ›
Table: `Movies`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| movie_id      | int     |
| title         | varchar |
+---------------+---------+
movie_id is the primary key for this table.
title is the name of the movie.
```

Table: `Users`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| user_id       | int     |
| name          | varchar |
+---------------+---------+
user_id is the primary key for this table.
```

Table: `Movie_Rating`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| movie_id      | int     |
| user_id       | int     |
| rating        | int     |
| created_at    | date    |
+---------------+---------+
(movie_id, user_id) is the primary key for this table.
This table contains the rating of a movie by a user in their review.
created_at is the user's review date.
```

Write the following SQL query:

- Find the name of the user who has rated the greatest number of the movies.

  In case of a tie, return lexicographically smaller user name.

- Find the movie name with the ***highest average*** rating in **February 2020**.

  In case of a tie, return lexicographically smaller movie name..

Query is returned in 2 rows, the query result format is in the following example:

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

Query is returned in 2 rows, the query result format is in the following example:

```
Movies table:
+-------------+--------------+
| movie_id    |  title       |
+-------------+--------------+
| 1           | Avengers     |
| 2           | Frozen 2     |
| 3           | Joker        |
+-------------+--------------+

Users table:
+-------------+--------------+
| user_id     |  name        |
+-------------+--------------+
| 1           | Daniel       |
| 2           | Monica       |
| 3           | Maria        |
| 4           | James        |
+-------------+--------------+

Movie_Rating table:
+-------------+--------------+--------------+-------------+
| movie_id    | user_id      | rating       | created_at  |
+-------------+--------------+--------------+-------------+
| 1           | 1            | 3            | 2020-01-12  |
| 1           | 2            | 4            | 2020-02-11  |
| 1           | 3            | 2            | 2020-02-12  |
| 1           | 4            | 1            | 2020-01-01  |
| 2           | 1            | 5            | 2020-02-17  |
| 2           | 2            | 2            | 2020-02-01  |
| 2           | 3            | 2            | 2020-03-01  |
| 3           | 1            | 3            | 2020-02-22  |
| 3           | 2            | 4            | 2020-02-25  |
+-------------+--------------+--------------+-------------+
```

```
Result table:
+--------------+
| results      |
+--------------+
| Daniel       |
| Frozen 2     |
+--------------+

Daniel and Maria have rated 3 movies ("Avengers", "Frozen 2" and "Joker") but Daniel is smaller lexicographically.
Frozen 2 and Joker have a rating average of 3.5 in February but Frozen 2 is smaller lexicographically.
```

Difficulty:

Medium

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

```
147  with cte as(select top 1 m.user_id,count(m.user_id) as count_no,u.name
148  from Movie_Rating_1341 m
149  left join Users_1341 u
150  on m.user_id=u.user_id
151  group by m.user_id,u.name
152  order by count_no desc,u.name),
153
154  cte1 as(select top 1 m.movie_id,sum(rating)/count(*) as avg_no,m1.title
155  from Movie_Rating_1341 m
156  left join Movies_1341 m1
157  on m.movie_id=m1.movie_id
158  group by m.movie_id,m1.title
159  order by avg_no desc)
160
161  select name as results from cte
162  union
163  select title as results from cte1;
```

100 %

Results Messages

	results
1	Daniel
2	Frozen 2

1350. Students With Invalid Departments

SQL Schema ›

Table: `Departments`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| name          | varchar |
+---------------+---------+
id is the primary key of this table.
The table has information about the id of each department of a university.
```

Table: `Students`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| name          | varchar |
| department_id | int     |
+---------------+---------+
id is the primary key of this table.
The table has information about the id of each student at a university and the id of the department he/she studies at.
```


LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

Write an SQL query to find the id and the name of all students who are enrolled in departments that no longer exists.

Return the result table in any order.

The query result format is in the following example:

```
Departments table:
+------+--------------------------+
| id   | name                     |
+------+--------------------------+
| 1    | Electrical Engineering   |
| 7    | Computer Engineering     |
| 13   | Bussiness Administration |
+------+--------------------------+

Students table:
+------+----------+---------------+
| id   | name     | department_id |
+------+----------+---------------+
| 23   | Alice    | 1             |
| 1    | Bob      | 7             |
| 5    | Jennifer | 13            |
| 2    | John     | 14            |
| 4    | Jasmine  | 77            |
| 3    | Steve    | 74            |
| 6    | Luis     | 1             |
| 8    | Jonathan | 7             |
| 7    | Daiana   | 33            |
| 11   | Madelynn | 1             |
+------+----------+---------------+
```

```
Result table:
+------+----------+
| id   | name     |
+------+----------+
| 2    | John     |
| 7    | Daiana   |
| 4    | Jasmine  |
| 3    | Steve    |
+------+----------+

John, Daiana, Steve and Jasmine are enrolled in departments 14, 33, 74 and 77 respectively. department 14, 33, 74 and 77 doesn't exist in th
```

Difficulty:

Easy

```
105  select s.id,s.name from Students_1350 s
106  left join Departments_1350 d
107  on s.department_id=d.id
108  where d.id is null
109
```

100 %

Results | Messages

	id	name
1	2	John
2	3	Steve
3	4	Jasmine
4	7	Daiana

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1355. Activity Participants

SQL Schema ›
Table: `Friends`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| name          | varchar |
| activity      | varchar |
+---------------+---------+
id is the id of the friend and primary key for this table.
name is the name of the friend.
activity is the name of the activity which the friend takes part in.
```

Table: `Activities`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| name          | varchar |
+---------------+---------+
id is the primary key for this table.
name is the name of the activity.
```

Write an SQL query to find the names of all the activities with neither maximum, nor minimum number of participants.

Return the result table in any order. Each activity in table Activities is performed by any person in the table Friends.

The query result format is in the following example:

```
Friends table:
+------+--------------+---------------+
| id   | name         | activity      |
+------+--------------+---------------+
| 1    | Jonathan D.  | Eating        |
| 2    | Jade W.      | Singing       |
| 3    | Victor J.    | Singing       |
| 4    | Elvis Q.     | Eating        |
| 5    | Daniel A.    | Eating        |
| 6    | Bob B.       | Horse Riding  |
+------+--------------+---------------+

Activities table:
+------+--------------+
| id   | name         |
+------+--------------+
| 1    | Eating       |
| 2    | Singing      |
| 3    | Horse Riding |
+------+--------------+

Result table:
+--------------+
| results      |
+--------------+
| Singing      |
+--------------+

Eating activity is performed by 3 friends, maximum number of participants, (Jonathan D. , Elvis Q. and Daniel A.)
Horse Riding activity is performed by 1 friend, minimum number of participants, (Bob B.)
Singing is performed by 2 friends (Victor J. and Jade W.)
```

Difficulty:

Medium

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```sql
with cte as(select activity,count(activity) as activity_count
from Friends_1355
group by activity)
select activity as results from cte
where activity_count <>(select max(activity_count) from cte ) and activity_count <>(select min(activity_count) from cte )
```

100 %

Results | Messages

	results
1	Singing

1369. Get the Second Most Recent Activity

SQL Schema ›

Table: `UserActivity`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| username      | varchar |
| activity      | varchar |
| startDate     | Date    |
| endDate       | Date    |
+---------------+---------+
This table does not contain primary key.
This table contain information about the activity performed of each user in a period of time.
A person with username performed a activity from startDate to endDate.
```

Write an SQL query to show the **second most recent activity** of each user.

If the user only has one activity, return that one.

A user can't perform more than one activity at the same time. Return the result table in **any** order.

The query result format is in the following example:

```
UserActivity table:
+------------+--------------+-------------+-------------+
| username   | activity     | startDate   | endDate     |
+------------+--------------+-------------+-------------+
| Alice      | Travel       | 2020-02-12  | 2020-02-20  |
| Alice      | Dancing      | 2020-02-21  | 2020-02-23  |
| Alice      | Travel       | 2020-02-24  | 2020-02-28  |
| Bob        | Travel       | 2020-02-11  | 2020-02-18  |
+------------+--------------+-------------+-------------+

Result table:
+------------+--------------+-------------+-------------+
| username   | activity     | startDate   | endDate     |
+------------+--------------+-------------+-------------+
| Alice      | Dancing      | 2020-02-21  | 2020-02-23  |
| Bob        | Travel       | 2020-02-11  | 2020-02-18  |
+------------+--------------+-------------+-------------+

The most recent activity of Alice is Travel from 2020-02-24 to 2020-02-28, before that she was dancing from 2020-02-21 to 2020-02-23.
Bob only has one record, we just take that one.
```

Difficulty:

Hard

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

```
65  with cte as(select distinct * from UserActivity_1369),
66  cte2 as(select *,row_number() over (partition by username order by startdate desc) as rnk,
67        count(activity) over (partition by username) as num_of_activity
68        from cte)
69  select username,activity,startDate,endDate
70  from cte2 where rnk=2 or num_of_activity=1;
71
```

100 %

Results | Messages

	username	activity	startDate	endDate
1	Alice	Dancing	2020-02-21	2020-02-23
2	Bob	Travel	2020-02-11	2020-02-18

1378. Replace Employee ID With The Unique Identifier

SQL Schema ›

Table: `Employees`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| name          | varchar |
+---------------+---------+
id is the primary key for this table.
Each row of this table contains the id and the name of an employee in a company.
```

Table: `EmployeeUNI`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| unique_id     | int     |
+---------------+---------+
(id, unique_id) is the primary key for this table.
Each row of this table contains the id and the corresponding unique id of an employee in the company.
```


LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

Write an SQL query to show the **unique ID** of each user, If a user doesn't have a unique ID replace just show null.

Return the result table in **any** order.

The query result format is in the following example:

```
Employees table:
+----+----------+
| id | name     |
+----+----------+
| 1  | Alice    |
| 7  | Bob      |
| 11 | Meir     |
| 90 | Winston  |
| 3  | Jonathan |
+----+----------+

EmployeeUNI table:
+----+-----------+
| id | unique_id |
+----+-----------+
| 3  | 1         |
| 11 | 2         |
| 90 | 3         |
+----+-----------+

EmployeeUNI table:
+-----------+----------+
| unique_id | name     |
+-----------+----------+
| null      | Alice    |
| null      | Bob      |
| 2         | Meir     |
| 3         | Winston  |
| 1         | Jonathan |
+-----------+----------+
```

```
Alice and Bob don't have a unique ID, We will show null instead.
The unique ID of Meir is 2.
The unique ID of Winston is 3.
The unique ID of Jonathan is 1.
```

Difficulty:

Easy

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

```sql
select u.unique_id,e.name
from Employees_1378 e
left join EmployeeUNI_1378 u
on e.id=u.id
```

100 %

Results | Messages

	unique_id	name
1	NULL	Alice
2	1	Jonathan
3	NULL	Bob
4	2	Meir
5	3	Winston

1393. Capital Gain/Loss

SQL Schema ›
Table: `Stocks`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| stock_name    | varchar |
| operation     | enum    |
| operation_day | int     |
| price         | int     |
+---------------+---------+
(stock_name, day) is the primary key for this table.
The operation column is an ENUM of type ('Sell', 'Buy')
Each row of this table indicates that the stock which has stock_name had an operation on the day operation_day with the price.
It is guaranteed that each 'Sell' operation for a stock has a corresponding 'Buy' operation in a previous day.
```

Write an SQL query to report the Capital gain/loss for each stock.

The capital gain/loss of a stock is total gain or loss after buying and selling the stock one or many times.

Return the result table in any order.

The query result format is in the following example:

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```
Stocks table:
+---------------+-----------+---------------+--------+
| stock_name    | operation | operation_day | price  |
+---------------+-----------+---------------+--------+
| Leetcode      | Buy       | 1             | 1000   |
| Corona Masks  | Buy       | 2             | 10     |
| Leetcode      | Sell      | 5             | 9000   |
| Handbags      | Buy       | 17            | 30000  |
| Corona Masks  | Sell      | 3             | 1010   |
| Corona Masks  | Buy       | 4             | 1000   |
| Corona Masks  | Sell      | 5             | 500    |
| Corona Masks  | Buy       | 6             | 1000   |
| Handbags      | Sell      | 29            | 7000   |
| Corona Masks  | Sell      | 10            | 10000  |
+---------------+-----------+---------------+--------+

Result table:
+---------------+-------------------+
| stock_name    | capital_gain_loss |
+---------------+-------------------+
| Corona Masks  | 9500              |
| Leetcode      | 8000              |
| Handbags      | -23000            |
+---------------+-------------------+
Leetcode stock was bought at day 1 for 1000$ and was sold at day 5 for 9000$. Capital gain = 9000 - 1000 = 8000$.
Handbags stock was bought at day 17 for 30000$ and was sold at day 29 for 7000$. Capital loss = 7000 - 30000 = -23000$.
Corona Masks stock was bought at day 1 for 10$ and was sold at day 3 for 1010$. It was bought again at day 4 for 1000$ and was sold at day 5
```

Difficulty:

Medium

```sql
--using join
select t.stock_name,(t2.total_price-t.total_price) as capital_gain_loss
from (select stock_name,operation,sum(price) as total_price
from Stocks_1393
group by stock_name,operation
having operation='Buy')  t
left join
(select stock_name,operation,sum(price) as total_price
from Stocks_1393
group by stock_name,operation
having operation='Sell') t2
on (t.stock_name=t2.stock_name)

--using case when then
select stock_name,sum(case when operation='Buy' then price*(-1) else price end)as capital_gain_loss
from Stocks_1393
group by stock_name
```

100 %

Results | Messages

	stock_name	capital_gain_loss
1	Corona Masks	9500
2	Handbags	-23000
3	Leetcode	8000

	stock_name	capital_gain_loss
1	Corona Masks	9500
2	Handbags	-23000
3	Leetcode	8000

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1398. Customers Who Bought Products A and B but Not C

SQL Schema ›

Table: `Customers`

```
+---------------------+---------+
| Column Name         | Type    |
+---------------------+---------+
| customer_id         | int     |
| customer_name       | varchar |
+---------------------+---------+
customer_id is the primary key for this table.
customer_name is the name of the customer.
```

Table: `Orders`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| order_id      | int     |
| customer_id   | int     |
| product_name  | varchar |
+---------------+---------+
order_id is the primary key for this table.
customer_id is the id of the customer who bought the product "product_name".
```

Write an SQL query to report the customer_id and customer_name of customers who bought products "A", "B" but did not buy the product "C" since we want to recommend them buy this product.

Return the result table **ordered** by customer_id.

The query result format is in the following example.

```
Customers table:
+-------------+---------------+
| customer_id | customer_name |
+-------------+---------------+
| 1           | Daniel        |
| 2           | Diana         |
| 3           | Elizabeth     |
| 4           | Jhon          |
+-------------+---------------+

Orders table:
+------------+--------------+---------------+
| order_id   | customer_id  | product_name  |
+------------+--------------+---------------+
| 10         |     1        |     A         |
| 20         |     1        |     B         |
| 30         |     1        |     D         |
| 40         |     1        |     C         |
| 50         |     2        |     A         |
| 60         |     3        |     A         |
| 70         |     3        |     B         |
| 80         |     3        |     D         |
| 90         |     4        |     C         |
+------------+--------------+---------------+
```

```
Result table:
+-------------+---------------+
| customer_id | customer_name |
+-------------+---------------+
| 3           | Elizabeth     |
+-------------+---------------+
Only the customer_id with id 3 bought the product A and B but not the product C.
```

Difficulty:

Medium

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```sql
select t.customer_id,c.customer_name from (

(select distinct customer_id from Orders_1398 where product_name='A'
intersect
select distinct customer_id from Orders_1398 where product_name='B')
except
select distinct customer_id from Orders_1398 where product_name='C') t

left join

Customers_1398 c
on t.customer_id=c.customer_id;
```

100 %

Results | Messages

	customer_id	customer_name
1	3	Elizabeth

1407. Top Travellers

SQL Schema ›

Table: `Users`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| name          | varchar |
+---------------+---------+
id is the primary key for this table.
name is the name of the user.
```

Table: `Rides`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| user_id       | int     |
| distance      | int     |
+---------------+---------+
id is the primary key for this table.
city_id is the id of the city who bought the product "product_name".
```

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

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

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

The query result format is in the following example.

```
Users table:
+------+-----------+
| id   | name      |
+------+-----------+
| 1    | Alice     |
| 2    | Bob       |
| 3    | Alex      |
| 4    | Donald    |
| 7    | Lee       |
| 13   | Jonathan  |
| 19   | Elvis     |
+------+-----------+

Rides table:
+------+----------+----------+
| id   | user_id  | distance |
+------+----------+----------+
| 1    | 1        | 120      |
| 2    | 2        | 317      |
| 3    | 3        | 222      |
| 4    | 7        | 100      |
| 5    | 13       | 312      |
| 6    | 19       | 50       |
| 7    | 7        | 120      |
| 8    | 19       | 400      |
| 9    | 7        | 230      |
+------+----------+----------+
```

```
Result table:
+----------+--------------------+
| name     | travelled_distance |
+----------+--------------------+
| Elvis    | 450                |
| Lee      | 450                |
| Bob      | 317                |
| Jonathan | 312                |
| Alex     | 222                |
| Alice    | 120                |
| Donald   | 0                  |
+----------+--------------------+
Elvis and Lee travelled 450 miles, Elvis is the top traveller as his name is alphabetically smaller than Lee.
Bob, Jonathan, Alex and Alice have only one ride and we just order them by the total distances of the ride.
Donald didn't have any rides, the distance travelled by him is 0.
```

Difficulty:

Easy

```
117 with cte as (select user_id,sum(distance) as travelled_distance
118 from Rides_1407 group by user_id)
119 select u.name,isnull(travelled_distance,0)
120 from cte c
121 right join Users_1407 u
122 on c.user_id=u.id
123 order by travelled_distance desc,name;
124
```

100 %

Results | Messages

	name	(No column name)
1	Elvis	450
2	Lee	450
3	Bob	317
4	Jonathan	312
5	Alex	222
6	Alice	120
7	Donald	0

1412. Find the Quiet Students in All Exams

SQL Schema ›

Table: `Student`

```
+---------------------+---------+
| Column Name         | Type    |
+---------------------+---------+
| student_id          | int     |
| student_name        | varchar |
+---------------------+---------+
student_id is the primary key for this table.
student_name is the name of the student.
```

Table: `Exam`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| exam_id       | int     |
| student_id    | int     |
| score         | int     |
+---------------+---------+
(exam_id, student_id) is the primary key for this table.
Student with student_id got score points in exam with id exam_id.
```


LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

A "quite" student is the one who took at least one exam and didn't score neither the high score nor the low score.

Write an SQL query to report the students (student_id, student_name) being "quiet" in **ALL** exams.

Don't return the student who has never taken any exam. Return the result table **ordered** by student_id.

The query result format is in the following example.

```
Student table:
+-------------+---------------+
| student_id  | student_name  |
+-------------+---------------+
| 1           | Daniel        |
| 2           | Jade          |
| 3           | Stella        |
| 4           | Jonathan      |
| 5           | Will          |
+-------------+---------------+

Exam table:
+------------+--------------+-----------+
| exam_id    | student_id   | score     |
+------------+--------------+-----------+
| 10         |     1        |    70     |
| 10         |     2        |    80     |
| 10         |     3        |    90     |
| 20         |     1        |    80     |
| 30         |     1        |    70     |
| 30         |     3        |    80     |
| 30         |     4        |    90     |
| 40         |     1        |    60     |
| 40         |     2        |    70     |
| 40         |     4        |    80     |
+------------+--------------+-----------+
```

```
Result table:
+-------------+---------------+
| student_id  | student_name  |
+-------------+---------------+
| 2           | Jade          |
+-------------+---------------+

For exam 1: Student 1 and 3 hold the lowest and high score respectively.
For exam 2: Student 1 hold both highest and lowest score.
For exam 3 and 4: Studnet 1 and 4 hold the lowest and high score respectively.
Student 2 and 5 have never got the highest or lowest in any of the exam.
Since student 5 is not taking any exam, he is excluded from the result.
So, we only return the information of Student 2.
```

Difficulty:

Hard

```sql
with cte1 as (select *,max(score) over(partition by exam_id) as max_marks,min(score) over(partition by exam_id) as min_marks
from Exam_1412
),

cte2 as(
select r1.*
from
(select distinct student_id from Exam_1412) r1
left join
(select distinct student_id
from cte1
where score=max_marks or score=min_marks)r2
on r1.student_id=r2.student_id

where r2.student_id is null)

select s.*from cte2 left join Student_1412 s
on cte2.student_id=s.student_id order by s.student_id
```

100 %

Results | Messages

	student_id	student_name
1	2	Jade

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1421. NPV Queries

SQL Schema ›
Table: `NPV`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| year          | int     |
| npv           | int     |
+---------------+---------+
(id, year) is the primary key of this table.
The table has information about the id and the year of each inventory and the corresponding net present value.
```

Table: `Queries`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| year          | int     |
+---------------+---------+
(id, year) is the primary key of this table.
The table has information about the id and the year of each inventory query.
```

Write an SQL query to find the npv of all each query of queries table.

Return the result table in any order.

The query result format is in the following example:

```
NPV table:
+------+--------+--------+
| id   | year   | npv    |
+------+--------+--------+
| 1    | 2018   | 100    |
| 7    | 2020   | 30     |
| 13   | 2019   | 40     |
| 1    | 2019   | 113    |
| 2    | 2008   | 121    |
| 3    | 2009   | 12     |
| 11   | 2020   | 99     |
| 7    | 2019   | 0      |
+------+--------+--------+

Queries table:
+------+--------+
| id   | year   |
+------+--------+
| 1    | 2019   |
| 2    | 2008   |
| 3    | 2009   |
| 7    | 2018   |
| 7    | 2019   |
| 7    | 2020   |
| 13   | 2019   |
+------+--------+
```


```
Result table:
+------+--------+--------+
| id   | year   | npv    |
+------+--------+--------+
| 1    | 2019   | 113    |
| 2    | 2008   | 121    |
| 3    | 2009   | 12     |
| 7    | 2018   | 0      |
| 7    | 2019   | 0      |
| 7    | 2020   | 30     |
| 13   | 2019   | 40     |
+------+--------+--------+

The npv value of (7, 2018) is not present in the NPV table, we consider it 0.
The npv values of all other queries can be found in the NPV table.
```

Difficulty:

Medium

```
113 select q.id,q.year,isnull(n.npv,0) as npv
114 from Queries_1421 q
115 left join NPV_1421 n
116 on q.id=n.id and q.year=n.year
117
118
```

100 %

Results | Messages

	id	year	npv
1	1	2019	113
2	2	2008	121
3	3	2009	12
4	7	2018	0
5	7	2019	0
6	7	2020	30
7	13	2019	40

1435. Create a Session Bar Chart

SQL Schema ›
Table: `Sessions`

```
+---------------------+---------+
| Column Name         | Type    |
+---------------------+---------+
| session_id          | int     |
| duration            | int     |
+---------------------+---------+
session_id is the primary key for this table.
duration is the time in seconds that a user has visited the application.
```

You want to know how long a user visits your application. You decided to create bins of "[0-5>", "[5-10>", "[10-15>" and "15 minutes or more" and count the number of sessions on it.

Write an SQL query to report the (bin, total) in **any** order.

The query result format is in the following example.

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```
Sessions table:
+-------------+---------------+
| session_id  | duration      |
+-------------+---------------+
| 1           | 30            |
| 2           | 299           |
| 3           | 340           |
| 4           | 580           |
| 5           | 1000          |
+-------------+---------------+

Result table:
+--------------+--------------+
| bin          | total        |
+--------------+--------------+
| [0-5>        | 3            |
| [5-10>       | 1            |
| [10-15>      | 0            |
| 15 or more   | 1            |
+--------------+--------------+

For session_id 1, 2 and 3 have a duration greater or equal than 0 minutes and less than 5 minutes.
For session_id 4 has a duration greater or equal than 5 minutes and less than 10 minutes.
There are no session with a duration greater or equial than 10 minutes and less than 15 minutes.
For session_id 5 has a duration greater or equal than 15 minutes.
```

Difficulty:

Easy

```
63  select bin,count(*) as total
64  from
65  (select *,
66  case when duration<300 then '[0-5>'
67       when duration>=300 and duration<600 then '[5-10>'
68       when duration>=600 and duration<900 then '[10-15>'
69       when duration>=900 then '15 or more' end as bin
70  from Sessions_1435) r
71  group by bin
```

100 %

Results | Messages

	bin	total
1	[0-5>	2
2	[5-10>	2
3	15 or more	1

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1440. Evaluate Boolean Expression

SQL Schema ›
Table `Variables`:

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| name          | varchar |
| value         | int     |
+---------------+---------+
name is the primary key for this table.
This table contains the stored variables and their values.
```

Table `Expressions`:

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| left_operand  | varchar |
| operator      | enum    |
| right_operand | varchar |
+---------------+---------+
(left_operand, operator, right_operand) is the primary key for this table.
This table contains a boolean expression that should be evaluated.
operator is an enum that takes one of the values ('<', '>', '=')
The values of left_operand and right_operand are guaranteed to be in the Variables table.
```

Write an SQL query to evaluate the boolean expressions in `Expressions` table.

Return the result table in any order.

The query result format is in the following example.

```
Variables table:
+------+-------+
| name | value |
+------+-------+
| x    | 66    |
| y    | 77    |
+------+-------+

Expressions table:
+--------------+----------+---------------+
| left_operand | operator | right_operand |
+--------------+----------+---------------+
| x            | >        | y             |
| x            | <        | y             |
| x            | =        | y             |
| y            | >        | x             |
| y            | <        | x             |
| x            | =        | x             |
+--------------+----------+---------------+
```

```
Result table:
+--------------+----------+---------------+-------+
| left_operand | operator | right_operand | value |
+--------------+----------+---------------+-------+
| x            | >        | y             | false |
| x            | <        | y             | true  |
| x            | =        | y             | false |
| y            | >        | x             | true  |
| y            | <        | x             | false |
| x            | =        | x             | true  |
+--------------+----------+---------------+-------+
As shown, you need find the value of each boolean exprssion in the table using the variables table.
```

Difficulty:

Medium

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

```sql
with cte as(select e.*,v.value as left_operandvalue,v1.value as right_operandvalue
from Expressions_1440 e
left join Variables_1440 v
on e.left_operand=v.name
left join Variables_1440 v1
on e.right_operand=v1.name)

select left_operand,operator,right_operand,
    case when operator ='<' then (Case when (left_operandvalue <  right_operandvalue) then 'true' else 'false' end)
        when operator = '>' then (Case when (left_operandvalue >  right_operandvalue) then 'true' else 'false' end)
        when operator = '=' then (Case when (left_operandvalue =  right_operandvalue) then 'true' else 'false' end)
        end as value
from cte
```

100 %

Results | Messages

	left_operand	operator	right_operand	value
1	x	<	y	true
2	x	=	x	true
3	x	=	y	false
4	x	>	y	false
5	y	<	x	false
6	y	>	x	true

1445. Apples & Oranges

SQL Schema ›

Table: `Sales`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| sale_date     | date    |
| fruit         | enum    |
| sold_num      | int     |
+---------------+---------+
(sale_date,fruit) is the primary key for this table.
This table contains the sales of "apples" and "oranges" sold each day.
```

Write an SQL query to report the difference between number of apples and oranges sold each day.

Return the result table **ordered** by sale_date in format ('YYYY-MM-DD').

The query result format is in the following example:

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```
Sales table:
+------------+------------+-------------+
| sale_date  | fruit      | sold_num    |
+------------+------------+-------------+
| 2020-05-01 | apples     | 10          |
| 2020-05-01 | oranges    | 8           |
| 2020-05-02 | apples     | 15          |
| 2020-05-02 | oranges    | 15          |
| 2020-05-03 | apples     | 20          |
| 2020-05-03 | oranges    | 0           |
| 2020-05-04 | apples     | 15          |
| 2020-05-04 | oranges    | 16          |
+------------+------------+-------------+

Result table:
+------------+--------------+
| sale_date  | diff         |
+------------+--------------+
| 2020-05-01 | 2            |
| 2020-05-02 | 0            |
| 2020-05-03 | 20           |
| 2020-05-04 | -1           |
+------------+--------------+

Day 2020-05-01, 10 apples and 8 oranges were sold (Difference  10 - 8 = 2).
Day 2020-05-02, 15 apples and 15 oranges were sold (Difference 15 - 15 = 0).
Day 2020-05-03, 20 apples and 0 oranges were sold (Difference 20 - 0 = 20).
Day 2020-05-04, 15 apples and 16 oranges were sold (Difference 15 - 16 = -1).
```

Difficulty:

Medium

```sql
--using case when then
select sale_date,sum(case when fruit='oranges' then sold_num*(-1) else sold_num end) as diff
from Sales_1445 group by sale_date
order by sale_date;

--using join
select t.sale_date,t.sold_num-s.sold_num as diff
from Sales_1445  t
inner join
Sales_1445 s
on t.sale_date = s.sale_date
and t.fruit='apples' and s.fruit='oranges'
order by t.sale_date;
```

100 %

Results | Messages

	sale_date	diff
1	2020-05-01	2
2	2020-05-02	0
3	2020-05-03	20
4	2020-05-04	-1

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1454. Active Users

SQL Schema ›
Table `Accounts`:

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| name          | varchar |
+---------------+---------+
the id is the primary key for this table.
This table contains the account id and the user name of each account.
```

Table `Logins`:

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| login_date    | date    |
+---------------+---------+
There is no primary key for this table, it may contain duplicates.
This table contains the account id of the user who logged in and the login date. A user may log in multiple times in the day.
```

Write an SQL query to find the id and the name of active users.

Active users are those who logged in to their accounts for 5 or more consecutive days.

Return the result table **ordered** by the id.

The query result format is in the following example:

```
Accounts table:
+----+----------+
| id | name     |
+----+----------+
| 1  | Winston  |
| 7  | Jonathan |
+----+----------+

Logins table:
+----+------------+
| id | login_date |
+----+------------+
| 7  | 2020-05-30 |
| 1  | 2020-05-30 |
| 7  | 2020-05-31 |
| 7  | 2020-06-01 |
| 7  | 2020-06-02 |
| 7  | 2020-06-02 |
| 7  | 2020-06-03 |
| 1  | 2020-06-07 |
| 7  | 2020-06-10 |
+----+------------+
```

```
Result table:
+----+----------+
| id | name     |
+----+----------+
| 7  | Jonathan |
+----+----------+
User Winston with id = 1 logged in 2 times only in 2 different days, so, Winston is not an active user.
User Jonathan with id = 7 logged in 7 times in 6 different days, five of them were consecutive days, so, Jonathan is an active user.
```

Follow up question: Can you write a general solution if the active users are those who logged in to their accounts for `n` or more consecutive days?

Difficulty:

Medium

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

```
98
99  with cte as(select distinct id,login_date
100 from Logins_1454),
101
102 cte2 as(select *,lag(login_date) over (partition by id order by login_date) as prev_date from cte),
103
104 cte3 as (Select *,Case when DATEDIFF(day,prev_date,login_date)=1 then 0 else 1 end as part from cte2),
105
106 cte4 as (Select *, sum(part) over(order by id,login_date) as grp from cte3),
107
108 cte5 as (Select distinct id from cte4 group by id,grp having count(*)>=5)
109
110 Select a.*
111 from cte5 left join Accounts_1454 a on cte5.id=a.id
```

100 %

Results | Messages

	id	name
1	7	Jonathan

1459. Rectangles Area

SQL Schema ›

Table: `Points`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| x_value       | int     |
| y_value       | int     |
+---------------+---------+
id is the primary key for this table.
Each point is represented as a 2D Dimensional (x_value, y_value).
```

Write an SQL query to report of all possible rectangles which can be formed by any two points of the table.

Each row in the result contains three columns (p1, p2, area) where:

- **p1** and **p2** are the id of two opposite corners of a rectangle and p1 < p2.
- Area of this rectangle is represented by the column **area**.

Report the query in descending order by area in case of tie in ascending order by p1 and p2.

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

Report the query in descending order by area in case of tie in ascending order by p1 and p2.

```
Points table:
+----------+-------------+-------------+
| id       | x_value     | y_value     |
+----------+-------------+-------------+
| 1        | 2           | 8           |
| 2        | 4           | 7           |
| 3        | 2           | 10          |
+----------+-------------+-------------+

Result table:
+----------+-------------+-------------+
| p1       | p2          | area        |
+----------+-------------+-------------+
| 2        | 3           | 6           |
| 1        | 2           | 2           |
+----------+-------------+-------------+

p1 should be less than p2 and area greater than 0.
p1 = 1 and p2 = 2, has an area equal to |2-4| * |8-7| = 2.
p1 = 2 and p2 = 3, has an area equal to |4-2| * |7-10| = 2.
p1 = 1 and p2 = 3 It's not possible because has an area equal to 0.
```

Difficulty:

Medium

```
59
60  with cte as(select p1.id as p1,p2.id as p2,abs(p1.x_value-p2.x_value)*abs(p1.y_value-p2.y_value) as area
61  from Points_1459 p1
62  inner join
63  Points_1459 p2
64  on p1.id<p2.id)
65  select * from cte where area>0 order by area desc,p1,p2
66
```

100 %

Results | Messages

	p1	p2	area
1	2	3	6
2	1	2	2

1468. Calculate Salaries

SQL Schema ›
Table `Salaries` :

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| company_id    | int     |
| employee_id   | int     |
| employee_name | varchar |
| salary        | int     |
+---------------+---------+
(company_id, employee_id) is the primary key for this table.
This table contains the company id, the id, the name and the salary for an employee.
```

Write an SQL query to find the salaries of the employees after applying taxes.

The tax rate is calculated for each company based on the following criteria:

- 0% If the max salary of any employee in the company is less than 1000$.
- 24% If the max salary of any employee in the company is in the range [1000, 10000] inclusive.
- 49% If the max salary of any employee in the company is greater than 10000$.

Return the result table **in any order**. Round the salary to the nearest integer.

The query result format is in the following example:

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

The query result format is in the following example:

```
Salaries table:
+------------+-------------+---------------+--------+
| company_id | employee_id | employee_name | salary |
+------------+-------------+---------------+--------+
| 1          | 1           | Tony          | 2000   |
| 1          | 2           | Pronub        | 21300  |
| 1          | 3           | Tyrrox        | 10800  |
| 2          | 1           | Pam           | 300    |
| 2          | 7           | Bassem        | 450    |
| 2          | 9           | Hermione      | 700    |
| 3          | 7           | Bocaben       | 100    |
| 3          | 2           | Ognjen        | 2200   |
| 3          | 13          | Nyancat       | 3300   |
| 3          | 15          | Morningcat    | 1866   |
+------------+-------------+---------------+--------+

Result table:
+------------+-------------+---------------+--------+
| company_id | employee_id | employee_name | salary |
+------------+-------------+---------------+--------+
| 1          | 1           | Tony          | 1020   |
| 1          | 2           | Pronub        | 10863  |
| 1          | 3           | Tyrrox        | 5508   |
| 2          | 1           | Pam           | 300    |
| 2          | 7           | Bassem        | 450    |
| 2          | 9           | Hermione      | 700    |
| 3          | 7           | Bocaben       | 76     |
| 3          | 2           | Ognjen        | 1672   |
| 3          | 13          | Nyancat       | 2508   |
| 3          | 15          | Morningcat    | 5911   |
+------------+-------------+---------------+--------+
```

```
For company 1, Max salary is 21300. Employees in company 1 have taxes = 49%
For company 2, Max salary is 700. Employees in company 2 have taxes = 0%
For company 3, Max salary is 7777. Employees in company 3 have taxes = 24%
The salary after taxes = salary - (taxes percentage / 100) * salary
For example, Salary for Morningcat (3, 15) after taxes = 7777 - 7777 * (24 / 100) = 7777 - 1866.48 = 5910.52, which is rounded to 5911.
```

Difficulty:

Medium

```sql
with cte as(select *,max(salary) over (partition by company_id)as max_salary
from Salaries_1468)
select company_id,employee_id,employee_name,
        case when (max_salary<1000) then salary
             when (max_salary>=1000) and (max_salary<=10000) then cast (round(salary-(salary*(24.00/100)),0) as int)
             when (max_salary)>10000 then cast(round(salary-(salary*(49.00/100)),0) as int)
             end as salary
from cte
```

100 %

Results | Messages

	company_id	employee_id	employee_name	salary
1	1	1	Tony	1020
2	1	2	Pronub	10863
3	1	3	Tyrrox	5508
4	2	1	Pam	300
5	2	7	Bassem	450
6	2	9	Hermione	700
7	3	2	Ognjen	1672
8	3	7	Bocaben	76
9	3	13	Nyancat	2508
10	3	15	Morningcat	1418

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1479. Sales by Day of the Week

SQL Schema ›
Table: `Orders`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| order_id      | int     |
| customer_id   | int     |
| order_date    | date    |
| item_id       | varchar |
| quantity      | int     |
+---------------+---------+
(ordered_id, item_id) is the primary key for this table.
This table contains information of the orders placed.
order_date is the date when item_id was ordered by the customer with id customer_id.
```

Table: `Items`

```
+---------------------+---------+
| Column Name         | Type    |
+---------------------+---------+
| item_id             | varchar |
| item_name           | varchar |
| item_category       | varchar |
+---------------------+---------+
item_id is the primary key for this table.
item_name is the name of the item.
item_category is the category of the item.
```

You are the business owner and would like to obtain a sales report for category items and day of the week.

Write an SQL query to report how many units in each category have been ordered on each **day of the week**.

Return the result table **ordered** by category.

The query result format is in the following example:

```
Orders table:
+------------+--------------+-------------+--------------+-------------+
| order_id   | customer_id  | order_date  | item_id      | quantity    |
+------------+--------------+-------------+--------------+-------------+
| 1          | 1            | 2020-06-01  | 1            | 10          |
| 2          | 1            | 2020-06-08  | 2            | 10          |
| 3          | 2            | 2020-06-02  | 1            | 5           |
| 4          | 3            | 2020-06-03  | 3            | 5           |
| 5          | 4            | 2020-06-04  | 4            | 1           |
| 6          | 4            | 2020-06-05  | 5            | 5           |
| 7          | 5            | 2020-06-05  | 1            | 10          |
| 8          | 5            | 2020-06-14  | 4            | 5           |
| 9          | 5            | 2020-06-21  | 3            | 5           |
+------------+--------------+-------------+--------------+-------------+
```


LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```
Items table:
+------------+---------------+---------------+
| item_id    | item_name     | item_category |
+------------+---------------+---------------+
| 1          | LC Alg. Book  | Book          |
| 2          | LC DB. Book   | Book          |
| 3          | LC SmarthPhone| Phone         |
| 4          | LC Phone 2020 | Phone         |
| 5          | LC SmartGlass | Glasses       |
| 6          | LC T-Shirt XL | T-Shirt       |
+------------+---------------+---------------+

Result table:
+------------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
| Category   | Monday    | Tuesday   | Wednesday | Thursday  | Friday    | Saturday  | Sunday    |
+------------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
| Book       | 20        | 5         | 0         | 0         | 10        | 0         | 0         |
| Glasses    | 0         | 0         | 0         | 0         | 5         | 0         | 0         |
| Phone      | 0         | 0         | 5         | 1         | 0         | 0         | 10        |
| T-Shirt    | 0         | 0         | 0         | 0         | 0         | 0         | 0         |
+------------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
On Monday (2020-06-01, 2020-06-08) were sold a total of 20 units (10 + 10) in the category Book (ids: 1, 2).
On Tuesday (2020-06-02) were sold a total of 5 units  in the category Book (ids: 1, 2).
On Wednesday (2020-06-03) were sold a total of 5 units in the category Phone (ids: 3, 4).
On Thursday (2020-06-04) were sold a total of 1 unit in the category Phone (ids: 3, 4).
On Friday (2020-06-05) were sold 10 units in the category Book (ids: 1, 2) and 5 units in Glasses (ids: 5).
On Saturday there are no items sold.
On Sunday (2020-06-14, 2020-06-21) were sold a total of 10 units (5 +5) in the category Phone (ids: 3, 4).
There are no sales of T-Shirt.
```

Difficulty:

Hard

```
with cte as(select o.order_id,o.item_id,datename(weekday,order_date) as day_name,i.item_category as category,quantity
from Orders_1479 o
right join Items_1479 i
on o.item_id=i.item_id),
cte1 as
(select distinct category,day_name,sum(quantity) over (partition by day_name,category) as total_quantity from cte),

cte2 as
(Select category,
Case when day_name= 'Monday' then total_quantity else 0 end as [Monday] ,
Case when day_name= 'Tuesday' then total_quantity else 0 end as [Tuesday] ,
Case when day_name= 'Wednesday' then total_quantity else 0 end as [Wednesday] ,
Case when day_name= 'Thursday' then total_quantity else 0 end as [Thursday] ,
Case when day_name= 'Friday' then total_quantity else 0 end as [Friday] ,
Case when day_name= 'Saturday' then total_quantity else 0 end as [Saturday] ,
Case when day_name= 'Sunday' then total_quantity else 0 end as [Sunday]
from cte1)

Select category,
sum(Monday) as 'Monday',
sum(Tuesday) as 'Tuesday',
sum(Wednesday) as 'Wednesday',
sum(Thursday) as 'Thursday',
sum(Friday) as 'Friday',
sum(Saturday) as 'Saturday',
sum(Sunday) as 'Sunday'
from cte2
group by category
```

100 %

Results | Messages

	category	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
1	Book	20	5	0	0	10	0	0
2	Glasses	0	0	0	0	5	0	0

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

1484. Group Sold Products By The Date

SQL Schema ›
Table `Activities` :

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| sell_date   | date    |
| product     | varchar |
+-------------+---------+
There is no primary key for this table, it may contains duplicates.
Each row of this table contains the product name and the date it was sold in a market.
```

Write an SQL query to find for each date, the number of distinct products sold and their names.

The sold-products names for each date should be sorted lexicographically.

Return the result table ordered by `sell_date` .

The query result format is in the following example.

```
Activities table:
+------------+-------------+
| sell_date  | product     |
+------------+-------------+
| 2020-05-30 | Headphone   |
| 2020-06-01 | Pencil      |
| 2020-06-02 | Mask        |
| 2020-05-30 | Basketball  |
| 2020-06-01 | Bible       |
| 2020-06-02 | Mask        |
| 2020-05-30 | T-Shirt     |
+------------+-------------+

Result table:
+------------+----------+------------------------------+
| sell_date  | num_sold | products                     |
+------------+----------+------------------------------+
| 2020-05-30 | 3        | Basketball,Headphone,T-shirt |
| 2020-06-01 | 2        | Bible,Pencil                 |
| 2020-06-02 | 1        | Mask                         |
+------------+----------+------------------------------+
For 2020-05-30, Sold items were (Headphone, Basketball, T-shirt), we sort them lexicographically and separate them by comma.
For 2020-06-01, Sold items were (Pencil, Bible), we sort them lexicographically and separate them by comma.
For 2020-06-02, Sold item is (Masks), we just return it.
```

Difficulty:

Easy

```sql
with cte as(
select sell_date,product from Activities_1484 group by sell_date,product)

select sell_date,count(product) as num_sold,string_agg(product,',') within group (order by product) as products
from cte
group by sell_date
order by sell_date;
```

100 %

Results | Messages

	sell_date	num_sold	products
1	2020-05-30	3	Basketball,Headphone,T-Shirt
2	2020-06-01	2	Bible,Pencil
3	2020-06-02	1	Mask

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

1495. Friendly Movies Streamed Last Month

SQL Schema ›
Table: `TVProgram`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| program_date  | date    |
| content_id    | int     |
| channel       | varchar |
+---------------+---------+
(program_date, content_id) is the primary key for this table.
This table contains information of the programs on the TV.
content_id is the id of the program in some channel on the TV.
```

Table: `Content`

```
+------------------+---------+
| Column Name      | Type    |
+------------------+---------+
| content_id       | varchar |
| title            | varchar |
| Kids_content     | enum    |
| content_type     | varchar |
+------------------+---------+
```

```
content_id is the primary key for this table.
Kids_content is an enum that takes one of the values ('Y', 'N') where:
'Y' means is content for kids otherwise 'N' is not content for kids.
content_type is the category of the content as movies, series, etc.
```

Write an SQL query to report the distinct titles of the kid-friendly movies streamed in June 2020.

Return the result table in any order.

The query result format is in the following example.

```
TVProgram table:
+--------------------+--------------+-------------+
| program_date       | content_id   | channel     |
+--------------------+--------------+-------------+
| 2020-06-10 08:00   | 1            | LC-Channel  |
| 2020-05-11 12:00   | 2            | LC-Channel  |
| 2020-05-12 12:00   | 3            | LC-Channel  |
| 2020-05-13 14:00   | 4            | Disney Ch   |
| 2020-06-18 14:00   | 4            | Disney Ch   |
| 2020-07-15 16:00   | 5            | Disney Ch   |
+--------------------+--------------+-------------+
```


LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```
Content table:
+------------+----------------+---------------+---------------+
| content_id | title          | Kids_content  | content_type  |
+------------+----------------+---------------+---------------+
| 1          | Leetcode Movie | N             | Movies        |
| 2          | Alg. for Kids  | Y             | Series        |
| 3          | Database Sols  | N             | Series        |
| 4          | Aladdin        | Y             | Movies        |
| 5          | Cinderella     | Y             | Movies        |
+------------+----------------+---------------+---------------+

Result table:
+--------------+
| title        |
+--------------+
| Aladdin      |
+--------------+
"Leetcode Movie" is not a content for kids.
"Alg. for Kids" is not a movie.
"Database Sols" is not a movie
"Alladin" is a movie, content for kids and was streamed in June 2020.
"Cinderella" was not streamed in June 2020.
```

Difficulty:

Easy

```
113 select distinct title
114 from TVProgram_1495 t
115 left join Content_1495 c
116 on t.content_id=c.content_id
117 where Kids_content='Y' and content_type='Movies' and month(program_date)=6 and year(program_date)=2020;
118
```

100 %

Results | Messages

	title
1	Aladdin

1501. Countries You Can Safely Invest In

SQL Schema

Table `Person`:

```
+----------------+---------+
| Column Name    | Type    |
+----------------+---------+
| id             | int     |
| name           | varchar |
| phone_number   | varchar |
+----------------+---------+
id is the primary key for this table.
Each row of this table contains the name of a person and their phone number.
Phone number will be in the form 'xxx-yyyyyyy' where xxx is the country code (3 characters) and yyyyyyy is the phone number (7 characters) w
```

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

Table `Country`:

```
+----------------+---------+
| Column Name    | Type    |
+----------------+---------+
| name           | varchar |
| country_code   | varchar |
+----------------+---------+
country_code is the primary key for this table.
Each row of this table contains the country name and its code. country_code will be in the form 'xxx' where x is digits.
```

Table `Calls`:

```
+-------------+------+
| Column Name | Type |
+-------------+------+
| caller_id   | int  |
| callee_id   | int  |
| duration    | int  |
+-------------+------+
There is no primary key for this table, it may contain duplicates.
Each row of this table contains the caller id, callee id and the duration of the call in minutes. caller_id != callee_id
```

A telecommunications company wants to invest in new countries. The country intends to invest in the countries where the average call duration of the calls in this country is strictly greater than the global average call duration.

Write an SQL query to find the countries where this company can invest.

Return the result table in any order.

The query result format is in the following example.

```
Person table:
+----+----------+--------------+
| id | name     | phone_number |
+----+----------+--------------+
| 3  | Jonathan | 051-1234567  |
| 12 | Elvis    | 051-7654321  |
| 1  | Moncef   | 212-1234567  |
| 2  | Maroua   | 212-6523651  |
| 7  | Meir     | 972-1234567  |
| 9  | Rachel   | 972-0011100  |
+----+----------+--------------+

Country table:
+----------+--------------+
| name     | country_code |
+----------+--------------+
| Peru     | 051          |
| Israel   | 972          |
| Morocco  | 212          |
| Germany  | 049          |
| Ethiopia | 251          |
+----------+--------------+
```

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```
Calls table:
+-----------+-----------+----------+
| caller_id | callee_id | duration |
+-----------+-----------+----------+
| 1         | 9         | 33       |
| 2         | 9         | 4        |
| 1         | 2         | 59       |
| 3         | 12        | 102      |
| 3         | 12        | 330      |
| 12        | 3         | 5        |
| 7         | 9         | 13       |
| 7         | 1         | 3        |
| 9         | 7         | 1        |
| 1         | 7         | 7        |
+-----------+-----------+----------+

Result table:
+----------+
| country  |
+----------+
| Peru     |
+----------+
The average call duration for Peru is (102 + 102 + 330 + 330 + 5 + 5) / 6 = 145.666667
The average call duration for Israel is (33 + 4 + 13 + 13 + 3 + 1 + 1 + 7) / 8 = 9.37500
The average call duration for Morocco is (33 + 4 + 59 + 59 + 3 + 7) / 6 = 27.5000
Global call duration average = (2 * (33 + 3 + 59 + 102 + 330 + 5 + 13 + 3 + 1 + 7)) / 20 = 55.70000
Since Peru is the only country where average call duration is greater than the global average, it's the only recommended country.
```

Difficulty:

Medium

```
147
148 with cte as(select substring(p.phone_number,1,3) as country_code,c.duration
149 from  Person_1501 p
150 left join Calls_1501 c
151 on c.caller_id=p.id
152
153 union all
154
155 select substring(p.phone_number,1,3) as country_code,c.duration
156 from  Person_1501 p
157 left join Calls_1501 c
158 on c.callee_id=p.id),
159
160 cte1 as(select country_code,
161       round(avg(cast(duration as float)) over (partition by country_code),2) as avg_duration,
162       round(avg(cast(duration as float)) over (),2) as global_avg_duration
163 from cte)
164
165 select distinct c.name as country
166 from cte1
```

100 %

Results | Messages

	country
1	Peru

Follow me: https://www.linkedin.com/in/ankesh-vaibhav/ **Like, Comment &Repost!**

LEETCODE SQL QUESTIONS ALONG WITH SOLUTIONS

```
147
148 with cte as(select substring(p.phone_number,1,3) as country_code,c.duration
149 from  Person_1501 p
150 left join Calls_1501 c
151 on c.caller_id=p.id or c.callee_id=p.id
152 ),
153
154 cte1 as(select country_code,
155       round(avg(cast(duration as float)) over (partition by country_code),2) as avg_duration,
156       round(avg(cast(duration as float)) over (),2) as global_avg_duration
157 from cte)
158
159 select distinct c.name as country
160 from cte1
161 left join Country_1501 c
162 on cte1.country_code=c.country_code
163 where avg_duration>global_avg_duration
164
165
```

100 %

Results | Messages

	country
1	Peru