182. Duplicate Emails

+-------------+---------+

| Column Name | Type |

+-------------+---------+

| id | int |

| email | varchar |

+-------------+---------+

id is the primary key (column with unique values) for this table.

Each row of this table contains an email. The emails will not contain uppercase letters.

Write a solution to report all the duplicate emails. Note that it's guaranteed that the email field is not NULL.

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

**Input:**

Person table:

+----+---------+

| id | email |

+----+---------+

| 1 | a@b.com |

| 2 | c@d.com |

| 3 | a@b.com |

+----+---------+

**Output:**

+---------+

| Email |

+---------+

| a@b.com |

+---------+

**Explanation:** a@b.com is repeated two times.

Solution –

# Write your MySQL query statement below

**with duplicatecte as (**

**select id, email, row\_number() over(partition by email) as ranking**

**from person**

**)**

**select distinct email from duplicatecte where ranking >1;**

183 . Customers who never Order

Table: Customers

+-------------+---------+

| Column Name | Type |

+-------------+---------+

| id | int |

| name | varchar |

+-------------+---------+

id is the primary key (column with unique values) for this table.

Each row of this table indicates the ID and name of a customer.

Table: Orders

+-------------+------+

| Column Name | Type |

+-------------+------+

| id | int |

| customerId | int |

+-------------+------+

id is the primary key (column with unique values) for this table.

customerId is a foreign key (reference columns) of the ID from the Customers table.

Each row of this table indicates the ID of an order and the ID of the customer who ordered it.

Write a solution to find all customers who never order anything.

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

The result format is in the following example.

**# Write your MySQL query statement below**

**with neverordercte as (**

**select c.id as id, c.name as Customers, o.id as orderid**

**from customers c left join orders o on c.id = o.customerid**

**)**

**select Customers from neverordercte where orderid is NULL;**

[586. Customer placing largest number of orders](https://leetcode.com/problems/customer-placing-the-largest-number-of-orders/description/)

Table: Orders

+-----------------+----------+

| Column Name | Type |

+-----------------+----------+

| order\_number | int |

| customer\_number | int |

+-----------------+----------+

order\_number is the primary key (column with unique values) for this table.

This table contains information about the order ID and the customer ID.

Write a solution to find the customer\_number for the customer who has placed **the largest number of orders**.

The test cases are generated so that **exactly one customer** will have placed more orders than any other customer.

The result format is in the following example.

**Example 1:**

**Input:**

Orders table:

+--------------+-----------------+

| order\_number | customer\_number |

+--------------+-----------------+

| 1 | 1 |

| 2 | 2 |

| 3 | 3 |

| 4 | 3 |

+--------------+-----------------+

**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.

Solution –

**# Write your MySQL query statement below**

**with countcte as (**

**select customer\_number as cust, count( customer\_number) as customer\_number**

**from orders**

**group by customer\_number**

**order by customer\_number desc)**

**select cust as customer\_number from countcte limit 1 ;**

176. Second Highest Salary

Table: Employee

+-------------+------+

| Column Name | Type |

+-------------+------+

| id | int |

| salary | int |

+-------------+------+

id is the primary key (column with unique values) for this table.

Each row of this table contains information about the salary of an employee.

Write a solution to find the second highest **distinct** salary from the Employee table. If there is no second highest salary, return null (return None in Pandas).

The result format is in the following example.

**Example 1:**

**Input:**

Employee table:

+----+--------+

| id | salary |

+----+--------+

| 1 | 100 |

| 2 | 200 |

| 3 | 300 |

+----+--------+

**Output:**

+---------------------+

| SecondHighestSalary |

+---------------------+

| 200 |

+---------------------+

**Example 2:**

**Input:**

Employee table:

+----+--------+

| id | salary |

+----+--------+

| 1 | 100 |

+----+--------+

**Output:**

+---------------------+

| SecondHighestSalary |

+---------------------+

| null |

+---------------------+

-- # Write your MySQL query statement below

-- with nullcte as (

-- select salary as SecondHighestSalary, row\_number() over(order by salary desc) as rownum

-- from employee

-- order by salary desc)

-- select

--     case when rownum is null then 'null'

--      when rownum = 2 then SecondHighestSalary

--     else null

--     end as 'SecondHighestSalary'

-- from  nullcte

-- limit 1 offset 1;

**select**

**(select distinct Salary**

**from Employee order by salary desc**

**limit 1 offset 1)**

**as SecondHighestSalary;**

178. Rank Scores

Table: Scores

+-------------+---------+

| Column Name | Type |

+-------------+---------+

| id | int |

| score | decimal |

+-------------+---------+

id is the primary key (column with unique values) for this table.

Each row of this table contains the score of a game. Score is a floating point value with two decimal places.

Write a solution to find the rank of the scores. The ranking should be calculated according to the following rules:

* The scores should be ranked from the highest to the lowest.
* If there is a tie between two scores, both should have the same ranking.
* After a tie, the next ranking number should be the next consecutive integer value. In other words, there should be no holes between ranks.

Return the result table ordered by score in descending order.

The result format is in the following example.

**Example 1:**

**Input:**

Scores table:

+----+-------+

| id | score |

+----+-------+

| 1 | 3.50 |

| 2 | 3.65 |

| 3 | 4.00 |

| 4 | 3.85 |

| 5 | 4.00 |

| 6 | 3.65 |

+----+-------+

**Output:**

+-------+------+

| score | rank |

+-------+------+

| 4.00 | 1 |

| 4.00 | 1 |

| 3.85 | 2 |

| 3.65 | 3 |

| 3.65 | 3 |

| 3.50 | 4 |

+-------+------+

Solution –

**select score, dense\_rank() over(order by score desc) as 'rank'**

**from scores;**

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 (combination of columns with unique values) of this table.

This table shows the activity of players of some games.

Each row is a record of a player who logged in and played a number of games (possibly 0) before logging out on someday using some device.

Write a solution to find the **first login date** for each player.

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

The result format is in the following example.

**Example 1:**

**Input:**

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 |

+-----------+-----------+------------+--------------+

**Output:**

+-----------+-------------+

| player\_id | first\_login |

+-----------+-------------+

| 1 | 2016-03-01 |

| 2 | 2017-06-25 |

| 3 | 2016-03-02 |

+-----------+-------------+

**# Write your MySQL query statement below**

**with logincte as (**

**select \*, row\_number() over(partition by player\_id order by event\_date asc) as rownum**

**from activity)**

**select player\_id, event\_date as first\_login**

**from logincte**

**where rownum = 1;**

[**619. Biggest Single Number**](https://leetcode.com/problems/biggest-single-number/)

Table: MyNumbers

+-------------+------+

| Column Name | Type |

+-------------+------+

| num | int |

+-------------+------+

This table may contain duplicates (In other words, there is no primary key for this table in SQL).

Each row of this table contains an integer.

 A **single number** is a number that appeared only once in the MyNumbers table.

Find the largest **single number**. If there is no **single number**, report null.

The result format is in the following example.

**Example 1:**

**Input:**

MyNumbers table:

+-----+

| num |

+-----+

| 8 |

| 8 |

| 3 |

| 3 |

| 1 |

| 4 |

| 5 |

| 6 |

+-----+

**Output:**

+-----+

| num |

+-----+

| 6 |

+-----+

**Explanation:** The single numbers are 1, 4, 5, and 6.

Since 6 is the largest single number, we return it.

**Example 2:**

**Input:**

MyNumbers table:

+-----+

| num |

+-----+

| 8 |

| 8 |

| 7 |

| 7 |

| 3 |

| 3 |

| 3 |

+-----+

**Output:**

+------+

| num |

+------+

| null |

+------+

**Explanation:** There are no single numbers in the input table so we return null.

**with repeatcte AS (**

**SELECT num, COUNT(\*) AS count**

**FROM MyNumbers**

**group by num**

**)**

**SELECT max(num) AS num**

**FROM repeatcte**

**where count = 1;**

1907 : -

Table: Accounts

+-------------+------+

| Column Name | Type |

+-------------+------+

| account\_id | int |

| income | int |

+-------------+------+

account\_id is the primary key (column with unique values) for this table.

Each row contains information about the monthly income for one bank account.

Write a solution to calculate the number of bank accounts for each salary category. The salary categories are:

* "Low Salary": All the salaries **strictly less** than $20000.
* "Average Salary": All the salaries in the **inclusive** range [$20000, $50000].
* "High Salary": All the salaries **strictly greater** than $50000.

The result table **must** contain all three categories. If there are no accounts in a category, return 0.

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

The result format is in the following example.

**SELECT 'Low Salary' AS category,**

**COUNT(if(income<20000,1,null)) AS accounts\_count**

**FROM Accounts**

**UNION ALL**

**SELECT 'Average Salary',**

**COUNT(if(income>=20000 and income<=50000,1,null))**

**FROM Accounts**

**UNION ALL**

**SELECT 'High Salary',**

**COUNT(if(income>50000,1,null))**

**FROM Accounts;**

1527 :

Table: Patients

+--------------+---------+

| Column Name | Type |

+--------------+---------+

| patient\_id | int |

| patient\_name | varchar |

| conditions | varchar |

+--------------+---------+

patient\_id is the primary key (column with unique values) for this table.

'conditions' contains 0 or more code separated by spaces.

This table contains information of the patients in the hospital.

Write a solution to find the patient\_id, patient\_name, and conditions of the patients who have Type I Diabetes. Type I Diabetes always starts with DIAB1 prefix.

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

The result format is in the following example.

**# Write your MySQL query statement below**

**select patient\_id, patient\_name, conditions**

**from patients**

**where  conditions LIKE '% DIAB1%' OR conditions like 'DIAB1%'**

**OR conditions LIKE '% DIAB1 %' ;**