```
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;
```

# Name – Pranay B Shah Leetcode SQL problems 183. Customers who never Order **Table: Customers** Table: Orders +----+ +----+ | Column Name | Type | | Column Name | Type | +----+ +----+ | int | | int | | id | id | customerId | int | | name | varchar | +----+ +----+ id is the primary key (column with unique values) for id is the primary key (column with unique values) for this table. this table. Each row of this table indicates the ID and name of a customerId is a foreign key (reference columns) of the ID from the Customers table. customer. 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 Table: Orders +----+

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: |4 |3 | Orders table: +----+ +----+ Output: | order\_number | customer\_number | +----+ +----+ | customer\_number | |1 | | 1 +----+ | 2 | 2 | 3 | | 3 | 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 |

## Example 2:

Employee table:

+----+

| id | salary |

+----+

- -- # Write your MySQL query statement below
- -- with nullcte as (
- -- select salary as
  SecondHighestSalary,
  row\_number() over(order by
  salary desc) as rownum

+---+

#### **Output:**

+----+

| SecondHighestSalary |

| 200 | +-----+

+----+

#### Input:

| 1 | 100 |

+----+

Output:

+----+

-- from employee

-- order by salary desc)

-- select

-- case when rownum is null then 'null'

| SecondHighestSalary |

+----+

| null |

+----+

-- when rownum = 2 thenSecondHighestSalary

-- else null

-- end as 'SecondHighestSalary'

-- from nullcte

-- limit 1 offset 1;

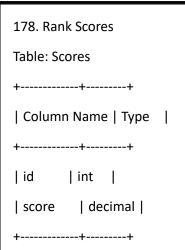
select

(select distinct Salary

from Employee order by salary desc

limit 1 offset 1)

as SecondHighestSalary;



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:

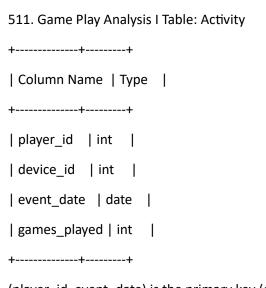
- The scores should be ranked from the highest to the lowest.
- If there is a tie between two scores, both should have the same ranking.
- After a tie, the next ranking number should be the next consecutive integer value. In other words, there should be no holes between ranks.

Return the result table ordered by score in descending order.

The result format is in the following example.

### Example 1:

Input:		Output:
Scores table:		++
++		score   rank
id   score		++
++		4.00   1
1   3.50		4.00   1
2   3.65		3.85   2
3   4.00		3.65   3
4   3.85		3.65   3
5   4.00		3.50   4
6   3.65		++
++		
Solution –		
	select score, dense_rank() over(orde	er by score desc) as 'rank
	from scores;	



(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:			++	
++			Output:	
player_id   device_id   event_date   games_played			++	
+	+	+	+	player_id   first_login
1	2	2016-03-01   5	1	++
1	2	2016-05-02   6	1	1   2016-03-01
2	3	2017-06-25   1	1	2   2017-06-25
3	1	2016-03-02   0	1	3   2016-03-02
3	4	2018-07-03   5	1	++

# 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		
Table: MyNumbers		
++		
Column Name   Type		
++		
num		
++		
This table may contain duplicates (In other words, there is no primary key for this table in SQL).	Find the largest <b>single number</b> . If there is no <b>single number</b> , report null.	
Each row of this table contains an integer.	The result format is in the following example.	
A <b>single number</b> is a number that appeared only once in the MyNumbers table.		
Example 1:		
Input:	4	
MyNumbers table:	5	
++	6	
num	++	
++	Output:	
8	++	
8	num	
3	++	
3	6	
1	++	
<b>Explanation:</b> The single numbers are 1, 4, 5, and 6.	3	
Since 6 is the largest single number, we return it.	3	
Example 2:	3	
Input:	++	
MyNumbers table:	Output:	
++	++	
num	num	
++	++	
8	null	
8	++	
7	<b>Explanation:</b> There are no single numbers in the input table so we return null.	
7		

```
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
```

Name – Pranay B Shah	Leetcode SQL problems
UNIO	N ALL
SELEC	Γ'High Salary',
cc	OUNT(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 v	with unique values) for this table.
'conditions' contains 0 or more code se	eparated by spaces.
This table contains information of the	patients in the hospital.
Write a solution to find the patient_id, Diabetes always starts with DIAB1 pref	patient_name, and conditions of the patients who have Type I Diabetes. Type I ix.
Return the result table in any order.	
The result format is in the following ex	ample.
	# 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 %';