**597. Friend Requests I: Overall Acceptance Rate**

select ifnull(round((count(distinct requester\_id,accepter\_id)/count(distinct sender\_id,send\_to\_id)),2),0.00) as accept\_rate

from friend\_request, request\_accepted

The code is not working

**1083. Sales Analysis II**

SELECT b.buyer\_id

FROM Product AS a

JOIN Sales AS b

ON a.product\_id = b.product\_id

GROUP BY b.buyer\_id

HAVING SUM(a.product\_name = 'S8') > 0 and SUM(a.product\_name = 'iPhone') = 0;

**1322. Ads Performance**

SELECT ad\_id, IFNULL(ROUND(AVG(CASE WHEN action = 'Clicked' THEN 1

WHEN action = 'Viewed' THEN 0

ELSE NULL END)\*100,2),0) AS ctr

FROM Ads

GROUP BY ad\_id

ORDER BY ctr DESC, ad\_id

607. **Sales Person**

select name from salesperson where sales\_id not in

(select o.sales\_id from orders o

join company c

on c.com\_id = o.com\_id

where c.name = 'RED')

**1667. Fix Names in a Table**

select user\_id, concat(upper(substr(name,1,1)),lower(substr(name,2))) as name

from Users

order by 1

select user\_id, concat(upper(left(name,1)),lower(right(name,length(name)-1))) as name

from Users

order by 1

**1113. Reported Posts**

select extra as report\_reason, count(distinct post\_id) as report\_count

from Actions

where action\_date= '2019-07-04' and action = 'report'

group by 1

order by 1

**603. Consecutive Available Seats**

select distinct a.seat\_id from cinema a, cinema b

where abs(a.seat\_id - b.seat\_id) = 1 and

a.free = true and b.free = true

order by 1

**1294. Weather Type in Each Country**

select c.country\_name,

(case when avg(weather\_state) <=15 then 'Cold'

when avg(weather\_state) >=25 then 'Hot'

else 'Warm' end) as weather\_type

from Weather w

left join countries c

on c.country\_id = w.country\_id

where date\_format(day, '%Y-%m') = '2019-11'

group by 1

**580. Count Student Number in Departments**

**578. Get Highest Answer Rate Question**

select a.question\_id as survey\_log from

(select question\_id,answer\_id,q\_num,rank() over (order by count(answer\_id)/count(q\_num) desc) as 'rank'

from survey\_log

group by question\_id) a

where a.rank = 1

**574. Winning Candidate**

select Name from Candidate

where

id = (

select CandidateId from Vote

group by CandidateId

order by count (\*) desc

limit 1)

**570. Managers with at Least 5 Direct Reports**

select e2.Name

from employee e1

join employee e2

on e1.ManagerId=e2.Id

group by e2.name

having count(\*)>=5

**180. Consecutive Numbers**

select distinct a.Num as ConsecutiveNums

from Logs a,

Logs b,

Logs c

where a.Id - b.Id = 1 and b.Id - c.Id = 1

and a.Num = b.Num

and b.num = c.Num

**184. Department Highest Salary**

select Department, Employee, Salary from

(

select d.Name as Department, e.Name as Employee , e.Salary, rank() over(partition by d.Name order by e.Salary desc) as a

from department d

join Employee e

on e.DepartmentId = d.Id

) b

where b.a = 1

**1112.**

select student\_id, course\_id, grade from

(

select student\_id, course\_id, grade, rank() over(partition by student\_id order by grade desc, course\_id asc) as a

from Enrollments) e

where e.a=1

order by student\_id

**1107. New Users Daily Count**

select activity\_date as login\_date, count(distinct user\_id) as user\_count from

(

select user\_id, activity\_date, rank() over(partition by user\_id order by activity\_date) as a

from traffic

where activity = 'login'

) e

where e.a=1 and datediff('2019-06-30',e.activity\_date) <= 90

group by 1

order by login\_date

[1077. Project Employees III](https://leetcode.com/problems/project-employees-iii)

select b.project\_id, b.employee\_id from

(

select p.project\_id, p.employee\_id,e.experience\_years,rank() over(partition by p.project\_id order by experience\_years desc)as a

from Project p

join Employee e

on e.employee\_id = p.employee\_id

) b

where b.a = 1

**1070. Product Sales Analysis III**

SELECT product\_id, year AS first\_year, quantity, price

FROM Sales

WHERE (product\_id, year) IN

(

SELECT product\_id, MIN(year) as year

FROM Sales

GROUP BY product\_id

) ;

**1045. Customers Who Bought All Products**

select customer\_id from Customer

group by customer\_id

having count(distinct product\_key) = (select distinct count(product\_key)from Product )

**626. Exchange Seats**

select (case when counts != id and mod(id,2) !=0 then id+1

when counts = id and mod(id,2) !=0 then id

else id-1 end) as id, student

from

(seat, (select count(\*) as counts from seat)seat\_counts)

order by id asc

**614. Second Degree Follower**

select a.follower, count(distinct b.follower) as num from

follow a,

follow b

where a.follower = b.followee

group by a.follower

order by a.follower

**612. Shortest Distance in a Plane**

select round(min(sqrt(pow(b.x-a.x,2)+pow(b.y-a.y,2))),2) as 'shortest'

from point\_2d a, point\_2d b

where (a.x, a.y) < (b.x, b.y)

**608. Tree Node**

select a.id, (case when a.p\_id is null then 'Root'

when a.id in (select p\_id from tree) then 'Inner'

else 'Leaf' end) as Type

from tree a, tree b

group by id

order by id

**602. Friend Requests II: Who Has the Most Friends**

select ids as id, cnt as num

from

( select ids, count(ids) as cnt

from

(

(select requester\_id as ids from request\_accepted)

union all

(select accepter\_id from request\_accepted)

) as tbl1

group by ids

) as tbl2

order by cnt desc

Limit 1

[550. Game Play Analysis IV](https://leetcode.com/problems/game-play-analysis-iv)

select round((sum(if(datediff(event\_date,first\_login)=1,1,0))/count(distinct player\_id)),2) fraction

from (select \*, min(event\_date) over(partition by player\_id) first\_login

from activity) t

**178. Rank Scores**

select Score as score, dense\_rank() over(order by score desc) as 'Rank'

from Scores

order by score desc

**177. Nth Highest Salary**

CREATE FUNCTION getNthHighestSalary(N INT) RETURNS INT

BEGIN

DECLARE M INT;

set M = N-1;

RETURN (

select distinct Salary from Employee

group by id

order by Salary desc

LIMIT 1 OFFSET M

);

END

**586. Customer Placing the Largest Number of Orders**

select customer\_number from orders

group by customer\_number

order by count(order\_number) desc

limit 1

offset 0

**610. Triangle Judgement**

select x,y,z,

(case when x+y>z and y+z>x and x+z>y

and x-y<z and y-z<x and x-z<y

then 'Yes'

else 'No'

End) as triangle

from triangle

**613. Shortest Distance in a Line**

select min(abs(a.x-b.x) ) as shortest

from point a, point b

where a.x <> b.x

**620. Not Boring Movies**

select id, movie, description, rating

from cinema

where description <> 'boring'

and mod(id,2)=1

order by rating desc

**627. Swap Salary**

UPDATE salary

SET sex =

Case

When sex = 'm' Then 'f'

When sex = 'f' Then 'm'

End

WHERE sex ='f' or sex ='m'

**1050. Actors and Directors Who Cooperated At Least Three Times**

SELECT actor\_id, director\_id

FROM ActorDirector

GROUP BY actor\_id, director\_id

HAVING COUNT(1) >= 3

**1068. Product Sales Analysis I**

select p.product\_name, s.year, s.price

from Sales s

join Product p

on s.product\_id = p.product\_id;

**1069. Product Sales Analysis II**

select p.product\_id, sum(quantity) as total\_quantity

from Sales s

join Product p

on s.product\_id = p.product\_id

group by product\_id

order by product\_id;

**1075. Project Employees I**

select p.project\_id, round(avg(e.experience\_years),2) as average\_years

from Project p

join Employee e

on p.employee\_id= e.employee\_id

group by p.project\_id

order by p.project\_id

**1076. Project Employees II**

select project\_id

from Project

group by project\_id

having (count(project\_id) =

(select count(project\_id) as cnt

from Project

group by project\_id

order by count(project\_id) desc

LIMIT 1)

**1082. Sales Analysis I**

select seller\_id

from Sales

group by seller\_id

having sum(price) =

(select sum(price) from Sales

group by seller\_id

order by sum(price) desc

LIMIT 1);

**1148. Article Views I**

select distinct author\_id as id from Views

where author\_id = viewer\_id

order by author\_id

**1341. Movie Rating**

(select u.name as results

from Users u

join Movie\_Rating m

on u.user\_id = m.user\_id

group by u.user\_id

order by count(m.movie\_id) desc, u.name

LIMIT 1

)

union

(

select mv.title from

movies mv

join Movie\_Rating m

on mv.movie\_id = m.movie\_id

where date\_format(created\_at, '%Y-%m') = '2020-02'

group by m.movie\_id

order by avg(rating) desc, mv.title

limit 1

)

**1613. Find the Missing IDs**

WITH RECURSIVE cte AS

(SELECT 1 AS VALUE UNION ALL SELECT VALUE +1 FROM cte WHERE VALUE<(SELECT MAX(customer\_id) FROM Customers))

SELECT VALUE AS ids FROM cte WHERE

VALUE NOT IN (SELECT CUSTOMER\_ID FROM Customers)

**1596. The Most Frequently Ordered Products for Each Customer**

select customer\_id, product\_id, product\_name from

(

select o.customer\_id, p.product\_id, p.product\_name, rank() over(partition by o.customer\_id order by count(o.product\_id) desc) as a

from Orders o

left join Products p

on p.product\_id = o.product\_id

GROUP BY o.customer\_id, o.product\_id

) b

where b.a = 1

**1555. Bank Account Summary**

select u.user\_id, u.user\_name,

ifnull(sum(case when u.user\_id = t.paid\_by then - amount else amount end),0)+ u.credit as credit,

if ((ifnull(sum(case when u.user\_id = t.paid\_by then - amount else amount end),0)+ u.credit) < 0, 'Yes', 'No') as credit\_limit\_breached

from Users u

left join Transactions t

on u.user\_id = t.paid\_by

or u.user\_id = t.paid\_to

group by u.user\_id, u.user\_name

**1549. The Most Recent Orders for Each Product**

select product\_name, product\_id, order\_id, order\_date from

(

select p.product\_name, p.product\_id, o.order\_id, o.order\_date, rank() over(partition by p.product\_id order by o.order\_date desc) as a

from Orders o

left join Products p

on p.product\_id = o.product\_id

) b

where b.a = 1

order by product\_name, product\_id, order\_id

**1532. The Most Recent Three Orders**

select customer\_name, customer\_id, order\_id, order\_date from

(select c.name as customer\_name, c.customer\_id, o.order\_id, o.order\_date, rank() over(partition by customer\_id order by o.order\_date desc) as a

from Orders o

left join Customers c

on c.customer\_id = o.customer\_id

) b

where b.a <=3

order by customer\_name,customer\_id, order\_date desc

**1501. Countries You Can Safely Invest In**

SELECT Country.name AS country

FROM Person JOIN Calls ON Calls.caller\_id = Person.id OR Calls.callee\_id = Person.id

JOIN Country ON Country.country\_code = LEFT(Person.phone\_number, 3)

GROUP BY Country.name

HAVING AVG(duration) > (SELECT AVG(duration) FROM Calls)

**1468. Calculate Salaries**

with cte as

(

select company\_id,

(

case when max(salary)>10000 then .51

when max(salary) >=1000 and salary <=10000 then .76

else 1

end

)

as tax

from Salaries

group by company\_id

)

select s.company\_id, s.employee\_id, s.employee\_name,round((s.salary\*c.tax),0) as salary

from salaries s

join cte c

on c.company\_id = s.company\_id

**1459. Rectangles Area**

select distinct a.id as p1, b.id as p2,

abs((a.x\_value-b.x\_value) \* (a.y\_value-b.y\_value)) as area

from Points a, Points b

where a.x\_value != b.x\_value

and a.y\_value != b.y\_value

and a.id< b.id

order by area desc, p1, p2

**1445. Apples & Oranges**

select sale\_date, sum(case when fruit ='apples' then sold\_num else -sold\_num end) as diff from Sales

group by sale\_date

order by sale\_date

**1440. Evaluate Boolean Expression**

select e.left\_operand, e.operator, e.right\_operand,

(case when v1.value< v2.value and e.operator = '<' then 'true'

when v1.value> v2.value and e.operator = '>' then 'true'

when v1.value= v2.value and e.operator = '=' then 'true'

else 'false' end ) as value

from Expressions e

join Variables v1

on e.left\_operand = v1.name

join Variables v2

on e.right\_operand = v2.name

**1421. NPV Queries**

select q.id, q.year, ifnull(npv,0) as npv

from Queries q

left join NPV n

on n.id = q.id and n.year = q.year

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

select c.customer\_id, c.customer\_name

from Customers c

where customer\_id in (

select customer\_id from Orders where product\_name = ('A')

)

and customer\_id in(

select customer\_id from Orders where product\_name = ('B')

)

and customer\_id not in(

select customer\_id from Orders where product\_name = ('C')

)

**1393. Capital Gain/Loss**

select stock\_name, sum(case when operation = 'Buy' then -price else price end) as capital\_gain\_loss

from Stocks

group by stock\_name

**1364. Number of Trusted Contacts of a Customer**

select

i.invoice\_id,

c.customer\_name,

i.price,

count(con.user\_id) as contacts\_cnt,

count(c2.email) as trusted\_contacts\_cnt

from invoices i

join customers c on c.customer\_id = i.user\_id

left join contacts con on con.user\_id = c.customer\_id

left join customers c2 on c2.email = con.contact\_email

group by i.invoice\_id, c.customer\_name, i.price

order by i.invoice\_id

**1355. Activity Participants**

select activity from

(

select activity, rank() over(order by count(id) asc) as a,

rank() over(order by count(id) desc) as b

from Friends

group by activity)z

where z.a != 1 and z.b!= 1

**1321. Restaurant Growth**

select distinct visited\_on, amount, round(amount/7, 2) as average\_amount

from (select visited\_on,

sum(amount) over (order by visited\_on range between interval 6 day preceding and current row) as amount,

dense\_rank() over (order by visited\_on) as rk

from Customer) as t

where rk >= 7

**1454. Active Users**

select distinct id, name  
from  
(select id,login\_date,  
lead(login\_date, 4) over (partition by id order by login\_date) lag\_date  
from (select distinct id, login\_date from logins) t1) t2  
join accounts  
using (id)  
where datediff(lag\_date, login\_date) = 4  
order by id

**1285. Find the Start and End Number of Continuous Ranges**

SELECT min(log\_id) as start\_id,

max(log\_id) as end\_id

FROM (SELECT log\_id,

RANK() OVER(ORDER BY log\_id) as num

FROM Logs) a

GROUP BY log\_id – num

**1270. All People Report to the Given Manager**

SELECT e1.employee\_id

FROM Employees e1,

Employees e2,

Employees e3

WHERE e1.manager\_id = e2.employee\_id

AND e2.manager\_id = e3.employee\_id

AND e3.manager\_id = 1

AND e1.employee\_id != 1

**1264. Page Recommendations**

select DISTINCT page\_id as recommended\_page

from Likes

where user\_id in

(select user2\_id as id from Friendship where user1\_id = 1

union

select user1\_id as id from Friendship where user2\_id = 1) and Page\_id not in

(SELECT page\_id FROM Likes WHERE user\_id = 1)

**1205. Monthly Transactions II**

SELECT DATE\_FORMAT(trans\_date, '%Y-%m') as month,

country,

SUM(CASE WHEN state = 'approved' THEN 1 ELSE 0 END) AS approved\_count,

SUM(CASE WHEN state = 'approved' THEN amount ELSE 0 END)AS approved\_amount,

SUM(CASE WHEN state = 'chargeback' THEN 1 ELSE 0 END ) AS chargeback\_count,

SUM(CASE WHEN state = 'chargeback' THEN amount ELSE 0 END) AS chargeback\_amount

FROM(

SELECT c.trans\_id, t.country, 'chargeback' as state, t.amount, c.trans\_date

from Chargebacks as c JOIN Transactions t ON c.trans\_id = t.id

UNION ALL

SELECT \*

FROM Transactions) as t1

group by country, month

HAVING approved\_amount>0 OR chargeback\_amount>0

**1174. Immediate Food Delivery II**

SELECT

ROUND(100\*SUM(CASE WHEN order\_date = customer\_pref\_delivery\_date THEN 1

ELSE 0 END)/ COUNT(distinct customer\_id) ,2) AS immediate\_percentage

FROM

Delivery

WHERE

(customer\_id, order\_date)

IN

(SELECT

customer\_id, min(order\_date) as min\_date

FROM

Delivery

GROUP BY

customer\_id

);

**1164. Product Price at a Given Date**

SELECT product\_id, new\_price AS price FROM Products WHERE (product\_id, change\_date) IN

(SELECT product\_id, MAX(change\_date) FROM Products WHERE change\_date <= '2019-08-16' GROUP BY product\_id)

UNION

SELECT DISTINCT product\_id, 10 FROM Products GROUP BY Product\_id HAVING MIN(change\_date) > '2019-08-16'

**1158. Market Analysis I**

SELECT u.user\_id AS buyer\_id, join\_date,

IFNULL(COUNT(order\_date), 0) AS orders\_in\_2019

FROM Users as u

LEFT JOIN

Orders as o

ON u.user\_id = o.buyer\_id

and YEAR(order\_date) = '2019'

GROUP BY u.user\_id

[1132. Reported Posts II](https://leetcode.com/problems/reported-posts-ii)

SELECT ROUND(AVG(cnt), 2) AS average\_daily\_percent FROM

(SELECT (COUNT(DISTINCT r.post\_id)/ COUNT(DISTINCT a.post\_id))\*100 AS cnt

FROM Actions a

LEFT JOIN Removals r

ON a.post\_id = r.post\_id

WHERE extra='spam' and action = 'report'

GROUP BY action\_date)tmp