<https://dsfaisal.com/articles/2020-11-06-leetcode-sql-problem-solving/#180-consecutive-numbers--medium--leetcode>

with agg as (

select dpid, name, salary, dense\_rank() over (partition by dpid order by salary desc) as rank from employee)

select a.salary as sl

from agg a

inner join dpt d on a.dpid = d.id

where a.rank <4

window function, dense\_rank vs rank

how they handle identical values,

rank will skip the next available ranking values

dense\_rank would still use the next chronological ranking value.

create function name (n int) returns int

begin

set n = n - 1;

return (select distinct salary from employee

order by salary desc

limit 1 offerset n

);

end

select email

from person

group by email

having count(\*) > 1

with cte as (

select email, row\_number() over (partition by email order by email)

as rn from person)

select email

from cte

where rn >1;

select name as customers

from customers

left join orders

on customer.id = orders.customerid

where orders.customerid is null;

select name as customers

from customres

where id not in (select customerid from orders);

seelect dp.name as dptname

, employe.name as empl

, salary

from employee em

inner join department as dp

on em.deptid = dp.id

where (dpid, salary) in (select departmentid, max(salary) as salary

from employee

group by dpartmentid

);

delete p2

from person p1

join person p2

on p1.email = p2.email

and p1.id < p2.id

select t.id

from weather t

join weather y

on datediff(t.data, y.date) = 1

and t.tmp > y.tmp

select t.id

from weather as t,

weather as y

where datediff(t., y.) = 1

and t.t > y.t;

SELECT Request\_at AS Day,

ROUND(SUM(IF(Status<>"completed", 1, 0))/COUNT(Status),2) AS "Cancellation Rate"

FROM Trips

WHERE Request\_at BETWEEN "2013-10-01" AND "2013-10-03"

AND Client\_Id NOT IN (SELECT Users\_Id FROM Users WHERE Banned = 'Yes')

AND Driver\_Id NOT IN (SELECT Users\_Id FROM Users WHERE Banned = 'Yes')

GROUP BY Request\_at;

SELECT

a.id,

a.month,

SUM(b.salary) Salary

FROM

Employee a JOIN Employee b ON

a.id = b.id AND

a.month - b.month >= 0 AND

a.month - b.month < 3

GROUP BY

a.id, a.month

HAVING

(a.id, a.month) NOT IN (SELECT id, MAX(month) FROM Employee GROUP BY id)

ORDER BY

a.id, a.month DESC

**Department highest 3 salaries**

select d.Name as Department, a. Name as Employee, a. Salary

from (

select e.\*, dense\_rank() over (partition by DepartmentId order by Salary desc) as DeptPayRank

from Employee e

) a

join Department d

on a. DepartmentId = d. Id

where DeptPayRank <=3;

**# Write your MySQL query statement below**

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

**Find consercutive numbers**

select distinct a.num as ConsecutiveNums

from logs as a

inner join logs as b

on a.id + 1 = b.id and a.num = b.num

inner join logs as c

on a.id + 2 = c.id and a.num = c.num

**Cancellation Rate**

SELECT t.request\_at AS Day,

round(avg(case when t.status = 'completed' then 0 else 1 end), 2) as "Cancellation Rate"

FROM Trips t

inner JOIN Users u1 ON t.client\_id = u1.users\_id and u1.banned = 'No'

inner JOIN Users u2 ON t.driver\_id = u2.users\_id and u2.banned = 'No'

WHERE t.request\_at BETWEEN "2013-10-01" AND "2013-10-03"

GROUP BY t.request\_at;

with CTE as (

select \* from Trips

where client\_id in (select users\_id from Users where role='client' and banned='No')

and driver\_id in (select users\_id from Users where role='driver' and banned='No')

and (Request\_at BETWEEN '2013-10-01' AND '2013-10-03')

)

select Request\_at Day,

cast(cast(sum(case when status!='completed' then 1 else 0 end) as decimal(10,2)) / count(\*) as decimal(10,2)) as "Cancellation Rate"

from CTE group by Request\_at

SELECT request\_at AS Day,

ROUND((SUM(CASE WHEN t.status LIKE "cancelled%" THEN 1 ELSE 0 END))/COUNT(t.status), 2) AS "Cancellation Rate"

FROM Trips t

JOIN Users u1 ON t.client\_id = u1.users\_id

JOIN Users u2 ON t.driver\_id = u2.users\_id

WHERE u1.banned = "No"

AND u2.banned = "No"

AND t.request\_at BETWEEN "2013-10-01" AND "2013-10-03"

GROUP BY request\_at;

**Customers who never ordered**

#Solution- 1:

SELECT Name AS Customers

FROM Customers

LEFT JOIN Orders

ON Customers.Id = Orders.CustomerId

WHERE CustomerId IS NULL;

#Solution- 2:

SELECT Name as Customers

FROM Customers

WHERE Id NOT IN(

SELECT CustomerId

FROM Orders

)

**Department’s highest salary**

SELECT Department.Name AS Department, Employee.Name AS Employee, Salary

FROM Employee

JOIN Department

ON Employee.DepartmentId = Department.Id

WHERE (DepartmentId, Salary) IN(

SELECT DepartmentId, MAX(Salary) AS Salary

FROM Employee

GROUP BY DepartmentId

);

**Delete duplicated records**

DELETE p2

FROM Person p1

JOIN Person p2

ON p1.Email = p2.Email

AND p1.id < p2.id

**Select days (IDs) with temperate higher than yesterday**

#Solution- 1:

SELECT t.Id

FROM Weather AS t, Weather AS y

WHERE DATEDIFF(t.RecordDate, y.RecordDate) = 1

AND t.Temperature > y.Temperature;

#Solution- 2:

SELECT t.Id

FROM Weather t

JOIN Weather y

ON DATEDIFF(t.recordDate, y.recordDate) = 1 AND

t.temperature > y.temperature;

**cancellation rates**

SELECT Request\_at AS Day,

ROUND(SUM(IF(Status<>"completed", 1, 0))/COUNT(Status),2) AS "Cancellation Rate"

FROM Trips

WHERE Request\_at BETWEEN "2013-10-01" AND "2013-10-03"

AND Client\_Id NOT IN (SELECT Users\_Id FROM Users WHERE Banned = 'Yes')

AND Driver\_Id NOT IN (SELECT Users\_Id FROM Users WHERE Banned = 'Yes')

GROUP BY Request\_at;

First loged-in device by player ID

#Solution- 1:

SELECT DISTINCT player\_id, device\_id

FROM Activity

WHERE (player\_id, event\_date) in (

SELECT player\_id, min(event\_date)

FROM Activity

GROUP BY player\_id)

#Solution- 2:

SELECT a.player\_id, b.device\_id

FROM

(SELECT player\_id, MIN(event\_date) AS event\_date FROM Activity

GROUP BY player\_id) a

JOIN Activity b

ON a.player\_id = b.player\_id AND a.event\_date = b.event\_date;

#Solution- 3:

SELECT player\_id, device\_id

FROM

(SELECT player\_id, device\_id, event\_date,

ROW\_NUMBER() OVER (PARTITION BY player\_id ORDER BY event\_date) AS r

FROM Activity) lookup

WHERE r = 1;

**Games played so far**

#Solution- 1:

SELECT t1.player\_id, t1.event\_date, SUM(t2.games\_played) as games\_played\_so\_far

FROM Activity t1

JOIN Activity t2

ON t1.player\_id = t2.player\_id

WHERE t1.event\_date >= t2.event\_date

GROUP BY t1.player\_id, t1.event\_date;

#Solution- 2:

SELECT player\_id, event\_date,

SUM(games\_played) OVER (PARTITION BY player\_id ORDER BY event\_date) AS games\_played\_so\_far

FROM Activity;

**% of users logged on in the 2nd day**

#Solution- 1:

SELECT ROUND(sum(CASE WHEN t1.event\_date = t2.first\_event+1 THEN 1 ELSE 0 END)/COUNT(DISTINCT t1.player\_id), 2) AS fraction

FROM Activity t1

JOIN

(SELECT player\_id, MIN(event\_date) AS first\_event

FROM Activity

GROUP BY player\_id) t2

ON t1.player\_id = t2.player\_id;

#Solution- 2:

SELECT ROUND(COUNT(DISTINCT b.player\_id)/COUNT(DISTINCT a.player\_id),2) AS fraction

FROM

(SELECT player\_id, MIN(event\_date) AS event\_date FROM Activity

GROUP BY player\_id) a

LEFT JOIN Activity b

ON a.player\_id = b.player\_id AND a.event\_date+1 = b.event\_date;