Write a solution to display the records with three or more rows with consecutive id's, and the number of people is greater than or equal to 100 for each.

Return the result table ordered by visit_date in ascending order.

```
with cte as (

select

row_number() over (order by id) as rownum,

id, visit_date,people

from Stadium

where people>=100)
```

--

Write a solution to find the nth highest distinct salary from the Employee table. If there are less than n distinct salaries, return null.

```
CREATE FUNCTION getNthHighestSalary(N INT) RETURNS INT

BEGIN

  RETURN (

      # Write your MySQL query statement below.

        select distinct(salary)

        from (

            select salary,

            dense_rank() over(order by salary desc) as ranked

            from employee) as temp

            where ranked= N

        );

END
```

--

For this problem, we will consider a manager an employee who has at least 1 other employee reporting to them.

Write a solution to report the ids and the names of all managers, the number of employees who report directly to them, and the average age of the reports rounded to the nearest integer.

Return the result table ordered by employee_id.

```
select a.employee_id, a.name, count(b.reports_to) as reports_count, round(avg(b.age)) as average_age

from employees a

join employees b

on a.employee_id=b.reports_to

group by a.employee_id

order by a.employee_id
```

--

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.

```
select email

from person

group by email

having count(email)>1
```

--

Write a solution to find the employees who earn more than their managers.

Return the result table in any order.

```
select e1.name as employee

from employee e1

inner join employee e2

on e1.managerId= e2.id

where e1.salary>e2.salary
```

--

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

Compute the moving average of how much the customer paid in a seven days window (i.e., current day + 6 days before). average_amount should be rounded to two decimal places.

Return the result table ordered by visited_on in ascending order.

```
select distinct c.visited_on as visited_on,

Round(sum(c.amount) over (order by visited_on range between interval 6 day preceding and current row),2 ) as amount,

round((sum(c.amount) over (order by visited_on range between interval 6 day preceding and current row )/7),2) as average_amount

from customer c

order by c.visited_on

limit 10000000000000 offset 6
```

--

Write a solution to show the unique ID of each user, If a user does not have a unique ID replace just show null.

Return the result table in any order.

```
select u.unique_id, e.name from employeeuni u

right join employees e

on u.id=e.id
```

--

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

Return the result table in any order.

```
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
```

--

Write a solution to delete all duplicate emails, keeping only one unique email with the smallest id.

For SQL users, please note that you are supposed to write a DELETE statement and not a SELECT one.

For Pandas users, please note that you are supposed to modify Person in place.

After running your script, the answer shown is the Person table. The driver will first compile and run your piece of code and then show the Person table. The final order of the Person table does not matter.

```
delete a from person a
join person b
on a.email=b.email
where a.id>b.id
```

--

Write a solution to find the number of times each student attended each exam.

Return the result table ordered by student_id and subject_name.

```
select s.student_id,s.student_name,ss.subject_name,count(e.subject_name) as attended_exams
from students s
cross join subjects ss
left join examinations e
on s.student_id=e.student_id and ss.subject_name=e.subject_name
group by s.student_id,s.student_name,ss.subject_name
order by s.student_id,ss.subject_name
```

--

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.

Write a solution to report the type of each node in the tree. Return the result table in any order.

```
select id, case
when p_id is null then "Root"
when id in(
select distinct p_id
from tree
where p_id is not null) then "Inner"
else "Leaf"
end as type
from Tree;
```

Write a solution to 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 temporary tables.

Note that you must write a single update statement, do not write any select statement for this problem.

```
update salary

set sex= case

when sex='m' then 'f'

when sex='f' then 'm'

end
```

--

Write a solution to find the daily active user count for a period of 30 days ending 2019-07-27 inclusively. A user was active on someday if they made at least one activity on that day.

Return the result table in any order.

```
select activity_date as day, count(distinct user_id) as active_users

from activity

group by day

having activity_date between '2019-06-28' and '2019-07-27'
```

--

Write a solution to get the names of products that have at least 100 units ordered in February 2020 and their amount.

Return the result table in any order.

```
select p.product_name, sum(o.unit) as unit from products p

join orders o

on p.product_id= o.product_id

where o.order_date LIKE '%2020-02-%'

group by p.product_name

having sum(o.unit)>=100
```

--

Write a solution to find the ids of products that are both low fat and recyclable.

Return the result table in any order.

```
select product_id

from products

where low_fats='Y' and recyclable='Y'
```

--

A country is big if:

- it has an area of at least three million (i.e., 3000000 km^2), or
- it has a population of at least twenty-five million (i.e., 25000000).

Write a solution to find the name, population, and area of the big countries.

Return the result table in any order.

```
select name,population,area from world

where area>=3000000 or population >=25000000
```

--

There is a factory website that has several machines each running the same number of processes. Write a solution to find the average time each machine takes to complete a process.

The time to complete a process is the 'end' timestamp minus the 'start' timestamp. The average time is calculated by the total time to complete every process on the machine divided by the number of processes that were run.

The resulting table should have the machine_id along with the average time as processing_time, which should be rounded to 3 decimal places.

Return the result table in any order.

```
SELECT MACHINE_ID, ROUND(SUM(IF (ACTIVITY_TYPE="START",-
TIMESTAMP,TIMESTAMP))/COUNT(DISTINCT PROCESS_ID),3) AS PROCESSING_TIME

FROM ACTIVITY

GROUP BY MACHINE_ID
```

--

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

Return the result table in any order.

```
select c.name as Customers from customers c

where id not in(select o.customerID from orders o)
```

--

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.

```
select s.score, DENSE_RANK() over (order by s.score desc) as 'rank'

from scores s

order by s.score desc
```

Write a solution to report the first name, last name, city, and state of each person in the Person table. If the address of a personId is not present in the Address table, report null instead.

Return the result table in any order.

```sql
select p.firstname,p.lastname, a.city, a.state
from person p
left join address a
on p.personid=a.personid
```

--

Write a solution to report the sum of all total investment values in 2016 tiv_2016, for all policyholders who:

- have the same tiv_2015 value as one or more other policyholders, and
- are not located in the same city as any other policyholder (i.e., the (lat, lon) attribute pairs must be unique).

Round tiv_2016 to two decimal places.

```sql
select round(sum(tiv_2016),2) as tiv_2016
from (
select tiv_2016,
count(*) over (partition by tiv_2015 ) as cnt15,
count(*) over (partition by lat, lon ) as citycnt
from insurance) as sub
where cnt15>1 and citycnt=1
```

--

Write a solution to find the people who have the most friends and the most friends number.

The test cases are generated so that only one person has the most friends.

```sql
with a as(select requester_id as id from requestaccepted
union all
select accepter_id as id from requestaccepted)
select a.id, count( a.id ) as num
from a
group by a.id
order by num desc
```

--

Write a solution to:

- Find the name of the user who has rated the greatest number of movies. In case of a tie, return the lexicographically smaller user name.
- Find the movie name with the highest average rating in February 2020. In case of a tie, return the lexicographically smaller movie name.

```
(select name results

from users u

join movierating mr

on u.user_id=mr.user_id

group by 1

order by count(rating)  desc, 1 asc

limit 1)
```

--

Write a solution to swap the seat id of every two consecutive students. If the number of students is odd, the id of the last student is not swapped.

Return the result table ordered by id in ascending order.

```
with cte as (select id, student, coalesce(if(id%2<>0,id+1,id-1),student) as swap,

(select student from seat where swap=id) as neww

from seat)

select id, coalesce(neww,student) as student

from cte
```

--

Find the IDs of the employees whose salary is strictly less than $30000 and whose manager left the company. When a manager leaves the company, their information is deleted from the Employees table, but the reports still have their manager_id set to the manager that left.

Return the result table ordered by employee_id.

```
select employee_id

from employees

where salary<30000 and manager_id not in(select employee_id

from employees)

order by employee_id
```

--

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.

```
select "Low Salary" as category, sum(if(income<20000,1,0)) as accounts_count

from Accounts

union

select "Average Salary" as category, sum(if(income>=20000 and income<=50000,1,0)) as accounts_count

from Accounts

union

select "High Salary" as category, sum(if(income>50000,1,0)) as accounts_count

from Accounts

--
```

There is a queue of people waiting to board a bus. However, the bus has a weight limit of 1000 kilograms, so there may be some people who cannot board.

Write a solution to find the person_name of the last person that can fit on the bus without exceeding the weight limit. The test cases are generated such that the first person does not exceed the weight limit.

Note that *only one* person can board the bus at any given turn.

```
with cte as(select turn, person_id as id, person_name as name, weight,

sum(weight) over(order by turn asc) as total_weight


from queue

order by turn)

select name as person_name

from cte

where total_weight<=1000

order by total_weight desc

limit 1

--
```

Write a solution to find the prices of all products on the date 2019-08-16.

Return the result table in any order.

```
SELECT distinct a.product_id, coalesce(b.new_price,10) as price
from products a
left join (select product_id,
rank() over (partition by product_id order by change_date desc) as xrank,
new_price from products
where change_date<="2019-08-16" ) as b
on a.product_id=b.product_id and b.xrank=1
order by product_id
```

--

Find all numbers that appear at least three times consecutively.

Return the result table in any order.

```
WITH CTE AS(SELECT ID, NUM,
LAG (NUM) OVER (ORDER BY ID) AS PREV,
LEAD (NUM) OVER (ORDER BY ID) AS AFTER
FROM LOGS)
SELECT DISTINCT CTE.NUM AS ConsecutiveNums
FROM CTE
WHERE CTE.NUM=CTE.PREV AND CTE.NUM=CTE.AFTER
```

--

Report for every three line segments whether they can form a triangle.

Return the result table in any order.

```
SELECT X,Y,Z, IF(X+Y>Z AND Y+Z>X AND X+Z>Y, "Yes","No") AS TRIANGLE
FROM TRIANGLE
```

--

Write a solution to report the customer ids from the Customer table that bought all the products in the Product table.

Return the result table in any order.

```
SELECT CUSTOMER_ID
FROM CUSTOMER
GROUP BY CUSTOMER_ID
HAVING COUNT(DISTINCT PRODUCT_KEY)=(SELECT COUNT(PRODUCT_KEY)FROM PRODUCT)
```

--