**( Here are the references of leet-code problems that I am pasting to understand the HOTS (high order thinking questions) and approaches towards the questions )**

**Leetcode**

**1.( Duplicate Emails )**

SELECT DISTINCT P1.Email FROM person P1,Person P2

WHERE P1.id <>P2.id AND P1.Email=P2.Email;

**2.(Customers who never order)**

# Write your MySQL query statement below

Select C.name AS Customers

from Customers C

LEFT JOIN Orders O ON C.id = O.customerId

where O.CustomerID is NULL;

**alternative solution**

SELECT name AS Customers FROM Customers WHERE Customers.id

NOT IN (SELECT CustomerId FROM Orders);

**3. Employees Earning More Than Their Managers** **( Round 2 done)**

SELECT EMP.NAME AS Employee FROM EMPLOYEE EMP, Employee MGR

WHERE EMP.managerId=MGR.id AND EMP.salary>MGR.salary;

**4.**

[**Rising Temperature**](https://leetcode.com/problems/rising-temperature/) **( Round 2 done)**

**5.(Rising Temperature)**

SELECT w1.id

FROM Weather w1, Weather w2

WHERE DATEDIFF(w1.recordDate, w2.recordDate) = 1 AND w1.temperature > w2.temperature;

**6.Delete Duplicate Emails (round 2)**

delete P1 from person P1, person  P2

where P1.email = P2.email  and P1.id > P2.id;

**7.(Game play Analysis ) (round 2)**

select player\_id,min(event\_date) as first\_login

from Activity

group by player\_id;

**8.(Employee Bonus) (round 2)**

# Write your MySQL query statement below

SELECT Employee.name,Bonus.bonus FROM Employee

LEFT JOIN Bonus ON Employee.empID = Bonus.empID

WHERE bonus < 1000 OR Bonus IS NULL ; providing this is necessary .

**9.Second Highest Salary (Medium)**

select max(salary) as SecondHighestSalary from Employee where salary <> (select max(salary) from  Employee);

**(Find customer Referee)**

# Write your MySQL query statement below

SELECT  name FROM Customer

WHERE referee\_id  <> 2 or referee\_id is null ;

**(Customers placing largest number of orders) (round 2)**

# Write your MySQL query statement below

SELECT customer\_number

FROM Orders

GROUP BY customer\_number

ORDER BY COUNT(\*) DESC LIMIT 1

(BIG Countries)

# Write your MySQL query statement below

select name, area, population

from world

where area >= 3000000 or population >= 25000000

**Classes more than 5 students**

# Write your MySQL query statement below

SELECT class FROM Courses

GROUP BY class

HAVING COUNT(student) >= 5;

**Sales Person**

Need to practice this again

# Write your MySQL query statement below

select s.name from salesperson as s

where s.sales\_id not in

(

    select sales\_id from orders as o left join company as c

    on o.com\_id = c.com\_id where c.name = 'RED'

)

**(ALL triangle solution )**

# Write your MySQL query statement below

select \*, if (x+y>z and y+z>x and z+x >y ,'Yes','No') as triangle from triangle

**(BIGGEST SINGLE NUMBER)**

# Write your MySQL query statement below

SELECT MAX(num) AS num

 FROM MyNumbers

 WHERE num IN (SELECT num FROM MyNumbers GROUP BY num HAVING COUNT(\*) = 1);

**(Not boring movies)**

# Write your MySQL query statement below

select \* from Cinema

where id % 2 != 0 and description != 'boring'

order by rating desc;

**(Swap salary ) (round 2)**

UPDATE salary

set sex = case

    when sex = 'm'then 'f'

    when sex = 'f'then 'm'

    else sex

end;

**(Actors and directors who have cooperated atleast three times ) (round 2)**

# Write your MySQL query statement below

select actor\_id, director\_id

from ActorDirector

group by actor\_id,director\_id

having count(timestamp) >= 3;

**(Product sales Analysis ) (round 2)**

# Write your MySQL query statement below

SELECT P.product\_name, S.year, S.price

FROM Sales AS S

LEFT JOIN Product AS P ON S.product\_id = P.product\_id;

**(Project Employees) (round 2)**

# Write your MySQL query statement below

SELECT Project.project\_id, ROUND(AVG(Employee.experience\_years), 2) AS average\_years

FROM Project

INNER JOIN Employee

ON Project.employee\_id = Employee.employee\_id

GROUP BY Project.project\_id;

**(Sales analysis ) (round 2)**

**Using the natural join**

select product\_id,product\_name

from product natural join sales

group by product\_id

having min(sale\_date)>='2019-01-01' and max(sale\_date)<='2019-03-31'

**(Article Views ) (round 2)**

# Write your MySQL query statement below

select distinct author\_id as id from Views

where author\_id = viewer\_id

order by id;

**Reformat department Table (round 2)**

# Write your MySQL query statement below

select id,

sum(case when month = "Jan" then revenue else NULL end) as Jan\_Revenue,

sum(case when month = "Feb" then revenue else NULL end) as Feb\_Revenue,

sum(case when month = "Mar" then revenue else NULL end) as Mar\_Revenue,

sum(case when month = "Apr" then revenue else NULL end) as Apr\_Revenue,

sum(case when month = "May" then revenue else NULL end) as May\_Revenue,

sum(case when month = "Jun" then revenue else NULL end) as Jun\_Revenue,

sum(case when month = "Jul" then revenue else NULL end) as Jul\_Revenue,

sum(case when month = "Aug" then revenue else NULL end) as Aug\_Revenue,

sum(case when month = "Sep" then revenue else NULL end) as Sep\_Revenue,

sum(case when month = "Oct" then revenue else NULL end) as Oct\_Revenue,

sum(case when month = "Nov" then revenue else NULL end) as Nov\_Revenue,

sum(case when month = "Dec" then revenue else NULL end) as Dec\_Revenue

from Department

group by id

order by id;

**(Query and quality percentage )**

# Write your MySQL query statement below

select

    query\_name,

    round(avg(rating/position), 2) as quality,

    round(sum(if(rating < 3,1,0)) \* 100 / count(\*), 2) as poor\_query\_percentage

from

    Queries

where

    query\_name is not null

group by

    query\_name;

**(Avg Selling price )**

# Write your MySQL query statement below

SELECT p.product\_id, IFNULL(ROUND(SUM(units\*price)/SUM(units),2),0) AS average\_price

FROM Prices p LEFT JOIN UnitsSold u

ON p.product\_id = u.product\_id AND

u.purchase\_date BETWEEN start\_date AND end\_date

group by product\_id;

**Students and Examinations**

# Write your MySQL query statement below

SELECT s.student\_id, s.student\_name, sub.subject\_name, COUNT(e.student\_id) AS attended\_exams

FROM Students s

CROSS JOIN Subjects sub

LEFT JOIN Examinations e ON s.student\_id = e.student\_id AND sub.subject\_name = e.subject\_name

GROUP BY s.student\_id, s.student\_name, sub.subject\_name

ORDER BY s.student\_id, sub.subject\_name;

**List the Products Ordered in a Period**

SELECT product\_name, SUM(unit) AS unit

FROM Orders

INNER JOIN Products USING(product\_id)

WHERE MONTH(order\_date) = '02' AND YEAR(order\_date) = '2020'

GROUP BY product\_id HAVING SUM(unit) >= 100;

**Replace Employee ID With The Unique Identifier**

# Write your MySQL query statement below

SELECT e.name, u.unique\_id

FROM Employees e LEFT JOIN EmployeeUNI u ON e.id = u.id;

**Top Travellers**

# Write your MySQL query statement below

SELECT a.name,CASE WHEN b.user\_id IS NULL THEN 0 ELSE SUM(distance) END as travelled\_distance FROM Users as a

LEFT JOIN Rides as b

ON a.id=b.user\_id

GROUP BY a.id

ORDER BY travelled\_distance DESC,name ASC;

**Group Sold Products By The Date**

# Write your MySQL query statement below

select sell\_date, count(distinct(product)) as num\_sold,

group\_concat(distinct(product) order by product) as products

from Activities group by sell\_date order by sell\_date;

**(Find Users With Valid E-Mails)**

select

    \*

from

    users

where

    mail REGEXP '^[a-zA-Z][a-zA-Z0-9\_.-]\*@leetcode[.]com$';

**Regex Pattern**: ^[A-Za-z][A-Za-z0-9\_\.\-]\*@leetcode(\?com)?\.com$

Let's break down the regex pattern step by step to clearly explain each part:

* **^**: Anchor the regex pattern to match from the start of the string.
* **[A-Za-z]**: Match any single uppercase or lowercase letter. The email prefix name must start with a letter.
* **[A-Za-z0-9\_.-]**\*: Match any number of characters following the first letter in the email prefix name. It includes letters (upper or lower case), digits, underscore '\_', period '.', and/or dash '-'.
* **@**: Match the literal '@' character, which separates the prefix name and the domain.
* **leetcode**: Match the literal 'leetcode', which is part of the email domain.
* **(?com)?**: Make the sequence ?com optional in the email domain. Allows the pattern to match both '@leetcode.com' and '@leetcode?com'.
* **.** : Match the literal '.' character, which separates the 'leetcode' part from the 'com' part of the domain.
* **com**: Match the literal 'com' at the end of the email domain.
* **$**: Anchor the regex pattern to match until the end of the string.

**Patients With a Condition**

# Write your MySQL query statement below

select patient\_id, patient\_name, conditions

from Patients

where conditions like 'DIAB1%' or conditions like '% DIAB1%'

**Customer Who Visited but Did Not Make Any Transactions**

# Write your MySQL query statement below

SELECT v.customer\_id, COUNT(v.visit\_id) AS count\_no\_trans

from Visits v

LEFT JOIN Transactions t

ON v.visit\_id = t.visit\_id

WHERE t.transaction\_id IS NULL

GROUP BY v.customer\_id;

**Bank Account Summary II**

# Write your MySQL query statement below

SELECT u.name, SUM(t.amount) AS balance

FROM Users u

LEFT JOIN Transactions t

ON u.account = t.account

GROUP BY u.account

HAVING balance >10000;

**Percentage of Users Attended a Contest**

# Write your MySQL query statement below

select

    contest\_id,

    round(100 \* count(distinct user\_id)/(select count(user\_id) from Users) ,2) as percentage

from

    Register

group by

    contest\_id

order by

    percentage desc,contest\_id

**Average Time of Process per Machine**

select a1.machine\_id, round(avg(a2.timestamp-a1.timestamp), 3) as processing\_time

from Activity a1

join Activity a2

on a1.machine\_id=a2.machine\_id and a1.process\_id=a2.process\_id

and a1.activity\_type='start' and a2.activity\_type='end'

group by a1.machine\_id;

**Fix Names in a Table**

1. **SELECT Clause**:
   * **user\_id**: This selects the **user\_id** column from the **Users** table.
   * **CONCAT(UPPER(LEFT(name, 1)), LOWER(SUBSTR(name, 2))) AS name**: This part is used to format the **name** column. Let's break it down further:
     + **UPPER(LEFT(name, 1))**: This takes the first character of the **name** column (**LEFT(name, 1)**) and converts it to uppercase (**UPPER**).
     + **LOWER(SUBSTR(name, 2))**: This takes the substring of **name** starting from the second character to the end (**SUBSTR(name, 2)**) and converts it to lowercase (**LOWER**).
     + **CONCAT(...)**: This concatenates the uppercase first letter and the rest of the name in lowercase.
   * **AS name**: This gives the resulting concatenated value an alias **name**.
2. # Write your MySQL query statement below
3. select
4. user\_id,
5. Concat(Upper(Left(name, 1)), Lower(Substr(name, 2))) AS name
6. from
7. users
8. order by
9. user\_id;

[**Invalid Tweets**](https://leetcode.com/problems/invalid-tweets/)

# Write your MySQL query statement below

select tweet\_id from Tweets

where char\_length(content) > 15;

[**Daily Leads and Partners**](https://leetcode.com/problems/daily-leads-and-partners/)

# Write your MySQL query statement below

SELECT

    date\_id,

    make\_name,

    COUNT(DISTINCT lead\_id) AS unique\_leads,

    COUNT(DISTINCT partner\_id) AS unique\_partners

FROM DailySales

GROUP BY 1, 2; group by 1,2 means column 1 and 2.

[**Find Followers Count**](https://leetcode.com/problems/find-followers-count/)

# Write your MySQL query statement below

select user\_id, count(follower\_id) as followers\_count

from Followers

group by user\_id

order by user\_id;

**The Number of Employees Which Report to Each Employee**

select mgr.employee\_id, mgr.name, COUNT(emp.employee\_id) as reports\_count, ROUND(AVG(emp.age)) as average\_age

from employees emp join employees mgr

on emp.reports\_to = mgr.employee\_id

group by employee\_id

order by employee\_id

**Find Total Time Spent by Each Employee**

SELECT

    event\_day AS day,

    emp\_id,

    SUM(out\_time - in\_time) AS total\_time

FROM Employees

GROUP BY 1, 2;

Recyclable and Low Fat Products

# Write your MySQL query statement below

select product\_id

from Products

where low\_fats = 'Y' AND recyclable = 'Y';

**Primary Department for Each Employee**

# Write your MySQL query statement below

SELECT employee\_id,department\_id from Employee

group by employee\_id

having count(employee\_id)=1

UNION

SELECT employee\_id,department\_id from Employee

where primary\_flag='Y';

**Calculate Special Bonus**

select employee\_id,

case when employee\_id%2<>0 and name not like "M%" then salary

else 0

end as bonus

from employees

order by employee\_id;

**The Latest Login in 2020**

# Write your MySQL query statement below

Select user\_id, max(time\_stamp) as last\_stamp

From Logins

Where year(time\_stamp) = 2020

Group By user\_id;

**Employees With Missing Information**

SELECT e.employee\_id

FROM employees AS e

LEFT JOIN salaries AS s

ON e.employee\_id = s.employee\_id

WHERE s.salary IS NULL

UNION

SELECT s.employee\_id

FROM salaries AS s

LEFT JOIN employees AS e

ON s.employee\_id = e.employee\_id

WHERE e.name IS NULL

ORDER BY employee\_id

**Employees Whose Manager Left the Company**

# Write your MySQL query statement below

select employee\_id from Employees

where salary  < 30000 and manager\_id not in (select employee\_id from Employees)

order by employee\_id;

**Number of Unique Subjects Taught by Each Teacher**

# Write your MySQL query statement below

SELECT teacher\_id,COUNT(DISTINCT subject\_id) as cnt

FROM Teacher

GROUP BY teacher\_id;

[**Managers with at Least 5 Direct Reports**](https://leetcode.com/problems/managers-with-at-least-5-direct-reports/)

# Write your MySQL query statement below

select m.name from Employee as E

inner join Employee as m

on e.managerId = m.id

group by e.managerId

having count(E.id)>=5;