SQL Practice Questions Leetcode

1. Write a solution to report all the duplicate emails. Note that it's guaranteed that the email field is not NULL.

select Email

from person

group by email

having Count(\*)>1;

using joins

select distinct p1.Email

from person p1

join person p2

on p1.email = p2.email

and p1.id<>p2.id;

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

Return the result table in **any order**.

select  name as Customers

from Customers

left join Orders

on Customers.id = Orders.customerId

where Orders.customerId is NULL  ;

Without Join

select  name as Customers

from Customers

where Customers.id

NOT IN (Select CustomerId from Orders)

3. 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

delete p1

from person p1

join person p2

on p1.email = p2.email

where p1.id> p2.id;

Using Cartesian

delete p1

from person p1,person p2

where p1.email = p2.email

and p1.id>p2.id;

4. Write a solution to find the **first login date** for each player.

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

from Activity

group by player\_id

5. Write a solution to find all the classes that have **at least five students**.

Return the result table in **any order**.

select class

from Courses

group by class

having count(\*)>=5;

6. Write a solution to find the names of all the salespersons who did not have any orders related to the company with the name **"RED"**.

SELECT name from salesperson

where sales\_id not in

(

    select sales\_id from orders where com\_id in

    (select com\_id from company where name='RED')

)

7. A **single number** is a number that appeared only once in the MyNumbers table.

Find the largest **single number**. If there is no **single number**, report null.

SELECT MAX(num) AS num

FROM (

    SELECT num

    FROM MyNumbers

    GROUP BY num

    HAVING COUNT(\*) = 1

) AS single\_numbers;

8. Write a solution to find the average selling price for each product. average\_price should be **rounded to 2 decimal places**.

Return the result table in **any order**.

SELECT p.product\_id, IFNULL(round(SUM(p.price\*u.units)/sum(u.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 p.Start\_date and p.end\_date

GROUP BY p.product\_id

9. Write a solution to find the percentage of the users registered in each contest rounded to **two decimals**.

Return the result table ordered by percentage in **descending order**. In case of a tie, order it by contest\_id in **ascending order**.

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

from register

group by contest\_id

order by percentage desc,contest\_id

10. 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

WHERE

    activity\_date BETWEEN DATE\_SUB('2019-07-27', INTERVAL 29 DAY) AND '2019-07-27'

group by activity\_date

**Advace Query Window function and partition By clause**

Database Example :

| region | year | sales\_amount

| North | 2021 | 1000.00 |

| North | 2021 | 1500.00 |

| North | 2022 | 2000.00 |

| South | 2021 | 1200.00 |

| South | 2021 | 1800.00 |

| South | 2022 | 2500.00 |

**SUM()**

select

region ,

year ,

sales\_amount ,

sum(sales\_amount) over (partition by region) as sales\_per\_region

from sales;

**ROW\_NUMBER()**

select

region,

year,

sales\_amount,

row\_number() over (partition by region order by year) as sales\_row\_number

from sales;

**AVG()**

select

region,

year,

sales\_amount,

avg(sales\_amount) over (partition by region) as average\_sales

from sales;

**MIN()**

select

region,

year,

sales\_amount,

min(sales\_amount) over (partition by region) as mini\_sales

from sales;

**MAX()**

select

region,

year,

sales\_amount,

max(sales\_amount) over (partition by region) as mini\_sales

from sales;

**COUNT()**

select

region,

year,

sales\_amount,

count(\*) over (partition by region) as mini\_sales

from sales;

**Row \_NUM() , LAG () ,LEAD() , RANK() , DENSE\_RANK()**

**RANK()**

SELECT

employee\_id,

name,

department,

salary,

RANK() OVER (PARTITION BY department ORDER BY salary DESC) AS department\_rank

FROM

employees;

**DENSE\_RANK()**

SELECT

employee\_id,

name,

department,

salary,

DENSE\_RANK() OVER (PARTITION BY department ORDER BY salary DESC) AS dense\_department\_rank

FROM

employees;

**LEAD()**

SELECT

employee\_id,

name,

department,

salary,

LEAD(salary) OVER (ORDER BY salary DESC) AS next\_highest\_salary

FROM

employees;

**LAG()**

SELECT

employee\_id,

name,

department,

salary,

LAG(salary) OVER (ORDER BY salary DESC) AS previous\_highest\_salary

FROM

employees;