Leelet code exercise

UPDATE AD CASE USING SAME TIME

update TD\_XXX  
set  
djyzmdm=null,  
djyzmsj=null,  
DLCS= case when DLCS is null then 1 else DLCS+1 end   
where djsj='xxx'

2、  
update `TD\_XXX`  
set  
`datetime` = 1434567890,  
`status` = case `id` when 12 Then 1 when 13 Then 2 when 14 Then 1 end   
where `id` in (12,13,14)

Given a table salary, such as the one below, that has m=male and f=female values. 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 temp table.

update salary

set

sex = case sex

when sex = 'm' then 'f'

else 'm'

end

UPDATE salary  
SET sex = IF(sex='f','m','f');

UPDATE salary

SET sex =

CASE sex

when 'f' then 'm'

when 'm' then 'f'

else sex -- just in case it contains a new value, we keep it

END

;

Write a SQL query for a report that provides the pairs (actor\_id, director\_id) where the actor have cooperated with the director at least 3 times.

select ACTOR\_ID , DIRECTOR\_ID from ActorDirector

group by actor\_id, director\_id

having count(actor\_id) >= 3 and count(director\_id) >=3

**SELECT** project\_id **FROM** (**SELECT** project\_id, DENSE\_RANK () OVER (**ORDER** **BY** COUNT(employee\_id) **DESC**) r **FROM** project **GROUP** **BY** project\_id) t **WHERE** r = 1;

Write an SQL query that reports all the ****projects**** that have the most employees.

The query result format is in the following example:

select project\_id from project group by project\_id having

count(employee\_id)=

(select count(distinct(employee\_id)) as cnt from project group by project\_id order by

count(distinct(employee\_id)) desc limit 1);

Write an SQL query that reports the ****buyers**** who have bought S8 but not iPhone. Note that S8 and iPhone are products present in the Product table.

1. 在发现题目问法为"是……而不是…… "时，一般情况下考虑使用NOT IN语句，即利用子查询找出需要排除的选项，然后通过NOT IN语句将主查询中不符合题意的部分排除出去。

select distinct s.buyer\_id from Product p

join Sales s

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

where p.product\_name = 'S8'

and s.buyer\_id not in

(select distinct s.buyer\_id from Product p

join Sales s

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

where p.product\_name = 'Iphone')

Write an SQL query that reports the ****products**** that were ****only**** sold in spring 2019. That is, between ****2019-01-01**** and ****2019-03-31**** inclusive.

1. 题意中最重要的一点就是“只在2019年1月1日-2019年3月31日被售出”。也就是，该产品所有的销售记录都在此时间段，此外的时间段无销售记录。
2. 要实现这个目的，对product\_id进行GROUP BY，只要要求每个分组中最大的销售时间小于等于"2019-03-31"，最小的销售时间大于等于"2019-01-01"即可。
3. 为了在最后显示出产品的名称，还需要对两表进行内连接JOIN。
4. 为防止出现重复记录，使用DISTINCT去重。

select distinct s.product\_id, p.product\_name from Product p

join Sales s

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

group by s.product\_id

#having '2019-01-01' <= s.sale\_date

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

Write an SQL query that reports the number of posts reported yesterday for each report reason. Assume today is ****2019-07-05****.

DATEDIFF ( datepart , startdate , enddate )

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 extra

order by extra

'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  
order by activity\_date'''

SELECT activity\_date AS day, COUNT(DISTINCT user\_id) AS active\_usersFROM activityWHERE activity\_date > DATE\_SUB('2019/07/27', INTERVAL 30 DAY)GROUP BY activity\_date

select cast(cast(count(distinct(session\_id)) as decimal(2))/cast(count(distinct(user\_id)) as decimal(2)) as decimal(8,2)) as average\_sessions\_per\_user  
from Activity  
where activity\_date between '2019-06-27' and '2019-07-27'

CAST() function converts a value (of any type) into a specified datatype.

SELECT CAST(25.65 AS varchar);

SELECT Round(Coalesce(Count(DISTINCT( session\_id )) / Count(DISTINCT user\_id),0) , 2)

average\_sessions\_per\_user

FROM activity

WHERE activity\_date > date\_sub( '2019-07-27', interval 30 day)

# Write your MySQL query statement below

select ifnull(round((tab.c / count(d.delivery\_id))\*100,2),0) as immediate\_percentage from delivery d ,

(select count(order\_date) as c from delivery

where order\_date = customer\_pref\_delivery\_date) as tab

If the preferred delivery date of the customer is the same as the order date then the order is called immediate otherwise it's called scheduled.

Write an SQL query to find the percentage of immediate orders in the table, ****rounded to 2 decimal places****.

We define query quality as:

The average of the ratio between query rating and its position.

We also define poor query percentage as:

The percentage of all queries with rating less than 3.

Write an SQL query to find each query\_name, the quality and poor\_query\_percentage.

Both quality and poor\_query\_percentage should be ****rounded to 2 decimal places****.

select query\_name, round(avg(rating/position),2) as quality,round(((sum(case when rating < 3 then 1 else 0 end))/count(1))\*100,2) as poor\_query\_percentagefrom queriesgroup by query\_name

select query\_name, round(avg(rating/position), 2) quality , round(avg(rating<3)\*100,2) as poor\_query\_percentage

from Queries

group by query\_name

Write an SQL query to find the type of weather in each country for November 2019.

The type of weather is ****Cold**** if the average weather\_state is less than or equal 15, ****Hot**** if the average weather\_state is greater than or equal 25 and ****Warm**** otherwise.

select t.country\_name,

(CASE

when t.avg\_weather <= 15 then 'Cold'

when t.avg\_weather >= 25 then 'Hot'

else 'Warm'

end) as weather\_type

from

(select c.country\_name, avg(w.weather\_state) as avg\_weather from Countries c

left join Weather w

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

where w.day between '2019-11-01' and '2019-11-30'

group by country\_name

) as t

Write an SQL query to find the team size of each of the employees.

select employee\_id, t.team\_size from Employee e

left join

(SELECT team\_id , count(employee\_id) as team\_size from Employee

group by team\_id) as t

on e.team\_id = t.team\_id

select e1.employee\_id, count(e2.team\_id) as team\_size from Employee e1

inner join

Employee e2

on e1.team\_id = e2.team\_id

group by e1.employee\_id

select ad\_id,

round(coalesce(

sum(case when action = 'Clicked' then 1 else 0 end)/

sum(case when action in( 'Viewed', 'Clicked') then 1 else 0 end),0)\*100, 2)

as ctr

from Ads

group by ad\_id

order by ctr desc, ad\_id asc

Write an SQL query to find the ctr of each Ad.

****Round**** ctr to 2 decimal points. ****Order**** the result table by ctr in descending order and by ad\_id in ascending order in case of a tie.

Write an SQL query to get the names of products with greater than or equal to 100 units ordered in February 2020 and their amount.

Return result table in any order.

The query result format is in the following example:

select p.product\_name, sum(o.unit) as unit from Products p

left join

Orders o

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

where order\_date between '2020-02-01' and '2020-02-29'

group by p.product\_name

having sum(o.unit) >= 100

Write an SQL query to find the id and the name of all students who are enrolled in departments that no longer exists.

select s.id, s.name from Students s

left join Departments d

on s.department\_id = d.id

where s.department\_id not in (select id from Departments)

select id, name from Students where department\_id not in(select id from Departments);

SELECT S.id, S.name

FROM Students S

LEFT JOIN Departments D

ON S.department\_id=D.id

WHERE D.id is null

Write an SQL query to show the ****unique ID****of each user, If a user doesn't have a unique ID replace just show null.

select u.unique\_id, e.name from Employees e

left join EmployeeUNI u

on u.id = e.id

Write an SQL query to find for each date, the number of distinct products sold and their names.

The sold-products names for each date should be sorted lexicographically.

Return the result table ordered by sell\_date.

select s.sell\_date, count(distinct s.product) as num\_sold, group\_concat(distinct s.product) as products from Activities s

group by s.sell\_date

**Group\_concat**

Write an SQL query to report the distinct titles of the kid-friendly movies streamed in June 2020.

Date could be expressed by two ways:

**month(t.program\_date) = 06**

**and (month(program\_date), year(program\_date)) = (6, 2020)**

Return the result table in any order.

The query result format is in the following example.

select distinct c.title as title from

Content c

inner join

TVProgram t

on t.content\_id = c.content\_id

where month(t.program\_date) = 06

and c.Kids\_content = 'Y' and c.Content\_type = 'Movies'

Write an SQL query to report the customer\_id and customer\_name of customers who have spent at least $100 in each month of June and July 2020.

# Write your MySQL query statement below

select c.customer\_id, c.name from

Customers c

join Orders o

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

join Product p

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

group by c.customer\_id

having sum(

case

when month(order\_date) = 06 then o.quantity\*p.price else 0 end) >= 100

and sum(

case

when month(order\_date) = 07 then o.quantity\*p.price else 0 end) >= 100

SELECT

o.customer\_id,

c.nameFROM Orders o JOIN Customers c ON o.customer\_id = c.customer\_id

JOIN Product p ON o.product\_id = P.product\_idGROUP BY o.customer\_idHAVING SUM(CASE WHEN DATE\_FORMAT(o.order\_date, '%Y-%m')='2020-06' THEN o.quantity\*p.price ELSE 0 END) >= 100

AND SUM(CASE WHEN DATE\_FORMAT(o.order\_date, '%Y-%m')='2020-07' THEN o.quantity\*p.price ELSE 0 END) >= 100;

Write an SQL query to find the users who have ****valid emails****.

A valid e-mail has a prefix name and a domain where:

* ****The prefix name**** is a string that may contain letters (upper or lower case), digits, underscore '\_', period '.' and/or dash '-'. The prefix name ****must**** start with a letter.
* ****The domain**** is '@leetcode.com'.

select \*

from users

where mail REGEXP '^[A-Za-z][A-Za-z0-9\.\\_\-]\*@leetcode.com$'

select \* from Users

where mail like '[a-zA-Z]%@leetcode.com' and left(mail, len(mail) - 13) not like '%[^0-9a-zA-Z\_.-]%'

**Date y 2 number of the year, Y all the whole year. M jan, m means 02-03**

**Trim remove the space**

**SELECT TRIM('#! ' FROM ' #SQL Tutorial! ') AS TrimmedString; remove the string from ..**

select lower(trim(product\_name)) as product\_name, DATE\_FORMAT(SALE\_DATE, '%Y-%m') AS sale\_date, count(sale\_id) as total from Sales

group by 1,2

order by 1, 2

SELECT LOWER(TRIM(product\_name)) product\_name, DATE\_FORMAT(sale\_date, "%y-%m") sale\_date, count(sale\_id) total

FROM sales

GROUP BY 1, 2

ORDER BY 1, 2

select trim(lower(product\_name)) as product\_name,  
left(sale\_date,7) as sale\_date,  
count(\*) as total  
from Sales  
group by trim(lower(product\_name)),left(sale\_date,7)  
order by product\_name,sale\_date

select trim(lower(product\_name)) as product\_name,

left(sale\_date,7) as sale\_date,

count(\*) as total

from Sales

group by trim(lower(product\_name)),left(sale\_date,7)

order by product\_name,sale\_date

Write an SQL query to find the number of ****unique orders**** and the number of ****unique customers**** with invoices ****> $20**** for each ****different month****.select

date\_format(order\_date,'%Y-%m') as month,

count(distinct order\_id) as order\_count,

count(distinct customer\_id) as customer\_count

from Orders

where invoice > 20

group by month

select left(order\_date, 7) month, count(distinct order\_id) order\_count, count(distinct customer\_id) customer\_count

from orders

where invoice > 20

group by 1;

Write an SQL query to report, How much cubic feet of ****volume****does the inventory occupy in each warehouse.

* warehouse\_name
* volume

SELECT name warehouse\_name,

SUM(units \* Width \* Length \* Height) volume

FROM Warehouse W

LEFT JOIN Products P

ON W.product\_id = P.product\_id

GROUP BY name

-- Solution 2

SELECT name warehouse\_name, SUM(units \* size) volume

FROM Warehouse W

LEFT JOIN

(

SELECT product\_id, Width \* Length \* Height size

FROM Products

) ps

ON W.product\_id = ps.product\_id

GROUP BY name

Write an SQL query to find the IDs of the users who visited without making any transactions and the number of times they made these types of visits.

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

select c.customer\_id, count(distinct c.visit\_id) as count\_no\_trans from Visits c

left join Transactions t

on c.visit\_id = t.visit\_id

where transaction\_id is null

group by c.customer\_id

SELECT customer\_id, COUNT(\*) as count\_no\_trans

FROM Visits

WHERE visit\_id NOT IN (SELECT DISTINCT visit\_id FROM Transactions)

GROUP BY customer\_id;

Write an SQL query to report the name and balance of users with a balance higher than 10000. The balance of an account is equal to the sum of the amounts of all transactions involving that account.

select u.name, sum(t.amount) as balance from Users u

left join Transactions t

on u.account = t.account

group by t.account

having balance >10000

Write an SQL query to report the names of all sellers who did not make any sales in 2020.

Return the result table ordered by seller\_name in ****ascending order****.

The query result format is in the following example.

select seller\_name from Seller

where seller\_id not in

(select s.seller\_id from Orders o

join Seller s

on s.seller\_id = o.seller\_id

where year(sale\_date) = 2020

group by s.seller\_name)

order by seller\_name asc

SELECT seller\_nameFROM Seller WHERE seller\_id NOT IN (

SELECT DISTINCT seller\_id

FROM Orders

WHERE LEFT(sale\_date, 4) = '2020'

)ORDER BY 1

Cross join same results from a,b,c or cross join b cross join c

select a.student\_name as member\_A, b.student\_name as member\_B, c.student\_name as member\_C from SchoolA a, SchoolB b, SchoolC c

where (a.student\_name != b.student\_name and a.student\_name != c.student\_name and b.student\_name!=c.student\_name)

and (a.student\_id!=b.student\_id and a.student\_id != c.student\_id and

b.student\_id != c.student\_id)

select contest\_id, round((

count(distinct user\_id)

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

group by contest\_id

order by percentage desc, contest\_id asc

Write an SQL query 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****.

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

The time to complete a process is the 'end' timestampminus 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****.

select machine\_id,

round((sum(case when activity\_type = 'end' then timestamp end) - sum(case when activity\_type = 'start' then timestamp end)) / count(distinct process\_id), 3) as processing\_time from Activity

group by machine\_id

SELECT machine\_id, ROUND((SUM(CASE WHEN activity\_type = 'end' THEN timestamp END)-SUM(CASE WHEN activity\_type = 'start' THEN timestamp END))/COUNT(DISTINCT process\_id), 3) processing\_time

FROM Activity

GROUP BY 1

select machine\_id,

round(avg(case when activity\_type = 'start' then -timestamp else timestamp end)\*2, 3) as processing\_time

from Activity

group by machine\_id

**Capitalize the first letger of name**

Write an SQL query to fix the names so that only the first character is uppercase and the rest are lowercase.

Return the result table ordered by user\_id.

SELECT user\_id, concat(upper(substring(name, 1,1)), lower(substring(name,2))) as name

from

Users

order by user\_id

select \* from Users

where name.formate('A-')

LECT user\_id, concat(upper(substring(name,1,1)),lower(substring(name,2,length(name)-1))) as name FROM Users

order by user\_id

select user\_id

, concat(upper(substring(name, 1,1)), lower(substring(name,2))) as name

from users

order by 1

Select sum groupby notice: join, left join

rite an SQL query that will, for all products, return each product name with total amount due, paid, canceled, and refunded across all invoices.

select p.name as name, sum(s.rest) as rest, sum(s.paid)as paid, sum(s.canceled) as canceled, sum(s.refunded) as refunded from Product p

join Invoice s

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

group by p.product\_id

order by p.name

Write an SQL query to find the IDs of the invalid tweets. The tweet is invalid if the number of characters used in the content of the tweet is ****strictly greater**** than 15.

Using LENGTH() is incorrect. The question is asking for the number of characters used in the content. LENGTH() returns the length of the string measured in bytes. CHAR\_LENGTH() returns the length of the string measured in characters.

select tweet\_id

from Tweets

where char\_length(content) > 15

Like and regexp

Write an SQL query that will, for each date\_id and make\_name, return the number of ****distinct**** lead\_id's and ****distinct**** partner\_id's.

select date\_id, make\_name, count(distinct lead\_id) as unique\_leads, count(distinct partner\_id) as unique\_partners from DailySales

group by date\_id, make\_name

SELECT \* FROM PatientsWHERE conditions REGEXP '^DIAB1| DIAB1';

Write an SQL query to find all active businesses.

An active business is a business that has more than one event type with occurences greater than the average occurences of that event type among all businesses.

ds

select e.business\_id from Events e

join

(select \*, avg(occurences) as avge from Events

group by event\_type) as t

on e.event\_type = t.event\_type

where e.occurences > t.avge

group by e.business\_id

having count(distinct t.event\_type) >1

Write an SQL query to find the id and the name of active users.

Active users are those who logged in to their accounts for 5 or more consecutive days.

Return the result table ****ordered**** by the id.

select distinct a.id, (select name from Accounts where id = a.id) as name

from Logins a

join Logins l

on a.id = l.id and datediff(a.login\_date, l.login\_date) between 1 and 4

group by a.id, a.login\_date

having count(distinct l.login\_date)=4

SELECT DISTINCT a.id

, (SELECT name FROM accounts WHERE id=a.id) AS name

FROM logins a, logins b

WHERE a.id = b.id AND DATEDIFF(a.login\_date, b.login\_date) BETWEEN 1 AND 4

GROUP BY a.id, a.login\_date

HAVING COUNT(DISTINCT b.login\_date) = 4

Write an SQL query to find the names of all the activities with neither maximum, nor minimum number of participants.

Return the result table in any order. Each activity in table Activities is performed by any person in the table Friends.

SELECT activity from Friends

group by activity

having count(\*) > (select count(\*) as num from Friends group by activity

order by num limit 1)

and count(\*) < (select count(\*) as num from Friends group by activity order by num desc limit 1)

Write an SQL query to find employee\_id of all employees that directly or indirectly report their work to the head of the company.

SELECT e1.employee\_id

FROM Employees e1

JOIN Employees e2

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

JOIN Employees e3

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

WHERE e3.manager\_id = 1 AND e1.employee\_id != 1

Write an SQL query to report the difference between number of ****apples**** and ****oranges**** sold each day.

Return the result table ****ordered**** by sale\_date in format ('YYYY-MM-DD').

select sale\_date, (

sum(case when fruit = 'apples' then sold\_num end) - sum(case when fruit = 'oranges' then sold\_num end)) as diff

from Sales

group by sale\_date

or select sale\_date,

(sum(case when fruit = 'aaple' then sold\_num else -sold\_num)) as diff

from Sales

group by

sale\_date