SQL

Occupation

1.SET @r1=0, @r2=0

2.SELECT CASE…THEN…END FROM…

[3.@r1:=@r1+1](mailto:3.@r1:=@r1+1)

4.SELECT min(string1), min(string2),…

1.标号 2.分组并建成一个temp表 3. Using either max()/min() can get first non-null value after group by.

**Binary Trees**

语句优化 in,exist, not exist, left join..on

SELECT…,(CASE WHEN… THEN…

WHEN…THEN…)

SELECT T1.N,

CASE

WHEN T1.P IS NULL THEN 'Root'

WHEN EXISTS (SELECT \* FROM BST T2 WHERE T1.N = T2.P) THEN 'Inner'

ELSE 'Leaf'

END

FROM BST AS T1

ORDER BY T1.N

exists用法是把主查询中的字段传入到子查询中去。如果有符合的条件，会停止全表检索，返回TRUE。所以效率才要高于IN，IN是要进行完全表检索得到集合才会结束执行。而EXISTS遇到符合的 条件，就会停止执行子查询。

[EXISTS](https://www.baidu.com/s?wd=EXISTS&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4nHK9mH6knj6sPvc3ujn30ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWc3n1DYPH6L)、IN与[JOIN](https://www.baidu.com/s?wd=JOIN&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4nHK9mH6knj6sPvc3ujn30ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWc3n1DYPH6L" \t "_blank)，都可以用来实现形如“查询A表中在（或不在）B表中的记录”的查询逻辑。  
  
在查询的两个表大小相当的情况下，3种查询方式的执行时间通常是：  
[EXISTS](https://www.baidu.com/s?wd=EXISTS&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4nHK9mH6knj6sPvc3ujn30ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWc3n1DYPH6L) <= IN <= [JOIN](https://www.baidu.com/s?wd=JOIN&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4nHK9mH6knj6sPvc3ujn30ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWc3n1DYPH6L)  
NOT [EXISTS](https://www.baidu.com/s?wd=EXISTS&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4nHK9mH6knj6sPvc3ujn30ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWc3n1DYPH6L) <= NOT IN <= LEFT [JOIN](https://www.baidu.com/s?wd=JOIN&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4nHK9mH6knj6sPvc3ujn30ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWc3n1DYPH6L)  
只有当表中字段允许[NULL](https://www.baidu.com/s?wd=NULL&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4nHK9mH6knj6sPvc3ujn30ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWc3n1DYPH6L" \t "_blank)时，NOT IN的方式最慢：  
NOT EXISTS <= LEFT JOIN <= NOT IN  
  
但是如果两个表中一个较小，一个较大，则[子查询](https://www.baidu.com/s?wd=%E5%AD%90%E6%9F%A5%E8%AF%A2&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4nHK9mH6knj6sPvc3ujn30ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWc3n1DYPH6L" \t "_blank)表大的用exists，[子查询](https://www.baidu.com/s?wd=%E5%AD%90%E6%9F%A5%E8%AF%A2&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4nHK9mH6knj6sPvc3ujn30ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWc3n1DYPH6L" \t "_blank)表小的用in，因为in 是把外表和内表作hash 连接，而exists是对外表作loop循环，每次loop循环再对内表进行查询。而无论那个表大，用not exists都比not in要快。这是因为如果查询语句使用了not in 那么内外表都进行[全表扫描](https://www.baidu.com/s?wd=%E5%85%A8%E8%A1%A8%E6%89%AB%E6%8F%8F&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4nHK9mH6knj6sPvc3ujn30ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWc3n1DYPH6L" \t "_blank)，没有用到索引；而not extsts 的[子查询](https://www.baidu.com/s?wd=%E5%AD%90%E6%9F%A5%E8%AF%A2&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4nHK9mH6knj6sPvc3ujn30ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWc3n1DYPH6L" \t "_blank)依然能用到表上的索引。  
  
IN的好处是逻辑直观简单（通常是独立子查询）；缺点是只能判断单字段，并且当NOT IN时效率较低，而且[NULL](https://www.baidu.com/s?wd=NULL&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4nHK9mH6knj6sPvc3ujn30ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWc3n1DYPH6L" \t "_blank)会导致不想要的结果。  
EXISTS的好处是效率高，可以判断单字段和组合字段，并不受[NULL](https://www.baidu.com/s?wd=NULL&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4nHK9mH6knj6sPvc3ujn30ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWc3n1DYPH6L" \t "_blank)的影响；缺点是逻辑稍微复杂（通常是相关子查询）。  
JOIN用在这种场合，往往是吃力不讨好。JOIN的用途是联接两个表，而不是判断一个表的记录是否在另一个表。

运算： SUM, AVG, COUNT 语句后通常会接 GROUP BY

##NEW Companies

INNER JOIN:

SELECT… FROM… INNER JOIN… ON …

返回所有ON条件的行

INNER JOIN Orders

ON Persons.Id\_P=Orders.Id\_P

连接两表，返回所有满足Persons.Id\_P =Orders.Id\_P的SELECT元素

2. key points for this question: inner join; group by company\_code and founder. Think about why founder should be grouped. It is because this column has no algorithm, SUM AVG COUNT…, so it should be grouped otherwise it cannot be printed.

**The Blunder:** 1.CEIL(234.0) = 234, CEIL(234.1)=235. Ceil: round up to next integer

Replace(salary,’0’,’’)

**Top Earners:** MAX…GROUP BY…. 返回的值是每组中最大的值，而不是全部数据的最大值

COUNT… GROUP BY… 每组的数量

**Weather Observation Station13**: The BETWEEN operator is inclusive: begin and end values are included.

HAVING 前提是有GROUP BY, 可以多个条件，where只能一个条件。。

**Weather Observation Station 18:** ABS, SQRT,POWER(…,2)

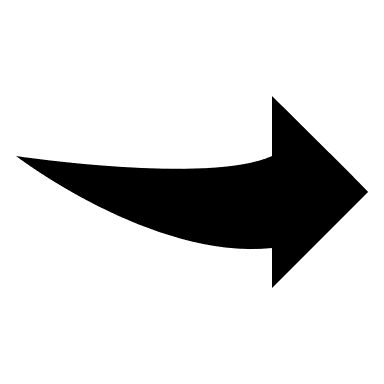
## Ollivander's Inventory： Understand the difference between group by and where clause

## Group by will return all rows , Where clause will return one row.

## In coins\_needed = …, we need only one row, so that’s why we use where clause.

And how about min(w1.coins\_needed)… WHERE w.age=w1.age and wp.power=wp1.power? In the where condition, the data may have three or four or even more rows, min selects the min among these rows. Similar with the question Binary Trees.

Sometimes WHERE clause and JOIN can be exchanged. But we prefer to use JOIN clause(ANSI).



/\* \*\*\*\*Median\*\*\*\* **Weather Observation Station 20**\*/

/\*1. where can use the elements from subquery directly even these elements

are not appeared in select

2. WHERE A IN (VAR1, VAR2, VAR3...)

WHERE A IN (SELECT... FROM ...)\*/

**The Report**

/\* T1

JOIN T2

ON T1. ... BETWEEN T2.... AND T2.... \*/ /\*ON 可以是个范围\*/

**Draw The Triangle1**

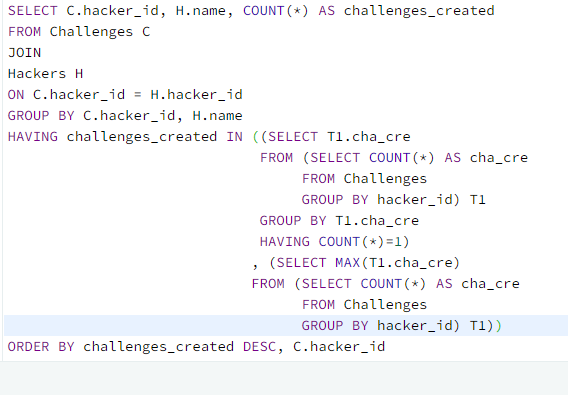
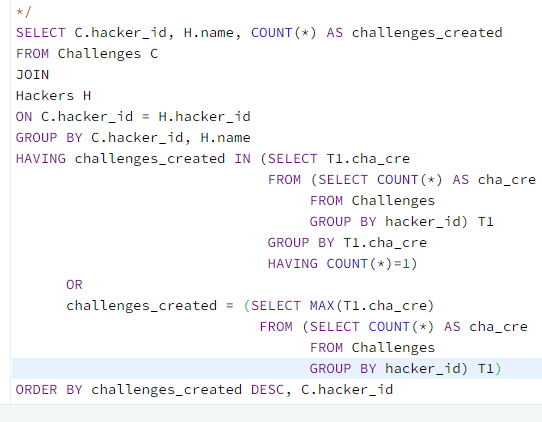
SET @row:=0;

SELECT Repeat(‘\* ’, @row:=@row+1) FROM…

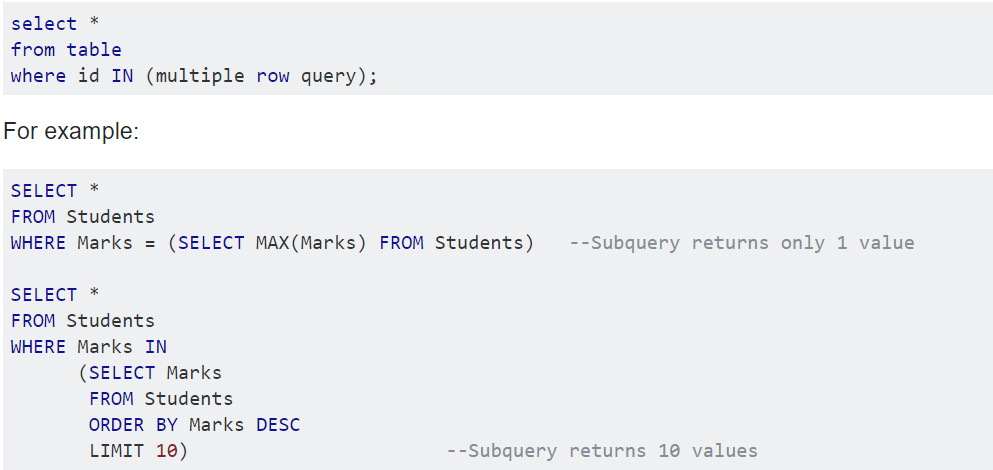
WHERE @row<20;

**Challenges:** 1. where a IN(Val1, Val2, Val3) 等价于 a=Val1 OR a=Val2 OR a=Val3，

黄记号只能是一个数值val,不能是（select…）的range,黄几号直接是range也不行

2. difference between IN and = ,(=后 只能跟1value)



**Projects**

SELECT T1.id,T2.id

FROM T1, T2;

This will give the output: T1.id\_1 ---- T2.id\_1

T1.id\_2 ----T2.id\_1

T1.id\_3 ----T2.id\_1

T1.id\_1 ----T2.id\_2

T1.id\_2 ---- T2.id\_2

T1.id\_3 ---- T2.id\_2

Interviews: Left join

|  |  |  |  |
| --- | --- | --- | --- |
| T1.ID | T1.col2 | T2.ID | T2.col2 |
| 1 | 33 | 1 | Aa |
| 1 | 33 | 1 | Df |
| 1 | 33 | 1 | hyh |
| 2 | 56 | 2 | dfg |
| 3 | 20 | 3 | dfk |
| 3 | 20 | 3 | lj |
| 4 | 78 | Null | null |
| 5 | 23 | 5 | gh |
| 5 | 23 | 5 | jk |
| 5 | 66 | 5 | gh |
| 5 | 66 | 5 | jk |

T1 T2 T1 Left Join T2

No this line if use INNER JOIN

|  |  |
| --- | --- |
| T1.ID | T1.col2 |
| 1 | 33 |
| 2 | 56 |
| 3 | 20 |
| 4 | 78 |
| 5 | 23 |
| 5 | 66 |

|  |  |  |  |
| --- | --- | --- | --- |
| T2.ID | | T2.col2 | |
| 1 | | Aa | |
| 1 | | Df | |
| 1 | | hyh | |
| 2 | | dfg | |
| 3 | | dfk | |
| 3 | | li | |
| 5 | | gh | |
| 5 | jk | |

Using GROUP BY challenge\_id in View\_Stats Table

And GROUP BY challenge\_id in Submission\_Stats Table first can avoid the above situation that many data repetition so that it will affect summarization if we use LEFT JOIN AND GROUP BY.

GROUP BY challenge\_id in View\_Stats Table

And GROUP BY challenge\_id in Submission\_Stats Table first can give us 1 to 1 match when we use LEFT JOIN, so we get only one row for eachchallenge\_id, then GROUP BY college\_id to get the final answer.

Leetcode

627. Swap Salary

Update… SET …= (CASE ….)

176. Second Highest Salary

Delete FROM … WHERE…

177.Nth Highest Salary

CREATE FUNCTION function\_name(input\_1 int(2), input\_2 int(2)) returns int(11)

BEGIN

RETURN input\_1+input\_2 ;

END;

Leetcode 586. Customer Placing the Largest Number of Orders

Max, pick up the largest number,(WHEN more than one customer have the largest number)

……..GROUP BY… HAVING…. = (SELECT MAX(T1.COUNT) FROM(SELECT COUNT()…FROM…)T1)

WHEN only one customer having the largest number

1. GROUP BY, ORDER BY可以跟聚合函数（aggregate function）;
2. 用….GROUP BY… **ORDER BY COUNT(**customer\_number), LIMIT 1； 可选出最大，但前提是这个最大只有一个值

See similar ones: 619. Biggest Single Number

|  |  |
| --- | --- |
| T1 | T2 |
| 1 | 3 |
|  | 4 |
| 3 | 1 |
|  | 4 |
| 4 | 1 |
|  | 3 |

613.Shortest Distance in a Line

mins between TWO tables

SELECT T1.x-T2.x FROM T1, T2 (WHERE T1.x<>T2.x)

610. Triangle Judgement

SELECT…,(CASE WHEN… THEN... ELSE… END) AS…

FROM…;

607. Sales Person

SELECT name FROM salesperson

WHERE sales\_id NOT IN (

SELECT sales\_id

FROM orders o JOIN company c

ON o.com\_id = c.com\_id

WHERE c.name = 'RED')

SELECT s.name

FROM orders o JOIN company c

ON o.com\_id = c.com\_id AND c.name = 'RED'

RIGHT JOIN salesperson s

ON s.sales\_id=o.sales\_id

WHERE o.sales\_id IS NULL

597. Friend Request I: Overall Acceptance Rate

ONLY when both of requester\_id and accepter\_id are identical at the same time, then using distinct will select one of the identical row.

|  |  |
| --- | --- |
| 2 | 3 |
| 2 | 5 |
|  |  |

|  |  |
| --- | --- |
| 2 | 3 |
| 2 | 3 |
| 2 | 5 |

Method1:

SELECT IFNULL(

(SELECT ROUND (COUNT(DISTINCT requester\_id, accepter\_id) /

COUNT(DISTINCT sender\_id, send\_to\_id), 2)

),0.00)

AS accept\_rate

FROM request\_accepted, friend\_request

Method2:

SELECT IFNULL(ROUND(acc/req,2),0.00) accept\_rate

FROM (SELECT COUNT(DISTINCT requester\_id, accepter\_id) AS acc FROM request\_accepted) AS T1,

(SELECT COUNT(DISTINCT sender\_id, send\_to\_id) AS req FROM friend\_request) AS T2

**Follow-up:**

 Can you write a query to return the accept rate but for every month?

WHERE YEAR(request\_date)=2016 AND YEAR(accept\_date) = 2016

GROUP BY MONTH(request\_date), MONTH(accept\_date)

OR DATEDD(MONTH,(DATEDIFF(MONTH,0,...),0))

<https://www.glassdoor.com/Interview/Write-a-sql-query-to-find-out-the-overall-friend-acceptance-rate-for-a-given-date-Table-User-id-who-sent-User-id-to-wh-QTN_2241356.htm> FACEBOOK interview

602. Friend Request2: Who has the most friends

The UNION operator selects only distinct values by default. To allow duplicate values, use UNION ALL

SELECT id1 AS id, COUNT(id1) AS num FROM

(

select requester\_id as id1, accepter\_id as id2

from request\_accepted

union

select accepter\_id as id1, requester\_id as id2

from request\_accepted) T1

GROUP BY id1

ORDER BY num DESC

LIMIT 1;

如果出现以下情况，用select requester\_id as id1, accepter\_id as id2

from request\_accepted

union

select accepter\_id as id1, requester\_id as id2

from request\_accepted

|  |  |
| --- | --- |
| Requester\_id | Accepter\_id |
| 2 | 3 |
| 3 | 2 |

|  |  |
| --- | --- |
| Id1 | Id2 |
| 2 | 3 |
| 3 | 2 |

|  |  |
| --- | --- |
| Id1 | num |
| 2 | 1 |
| 3 | 1 |

|  |  |
| --- | --- |
| Accepter\_id | Requester\_id |
| 3 | 2 |
| 2 | 3 |

|  |
| --- |
| id |
| 2 |
| 3 |
| 3 |
| 2 |

题目给的solution:

|  |  |
| --- | --- |
| Requester\_id | Accepter\_id |
| 2 | 3 |
| 3 | 2 |

|  |  |
| --- | --- |
| Id | Num |
| 2 | 2 |
| 3 | 2 |

603. Consecutive Available Seats

SELECT DISTINCT T1.seat\_id

FROM cinema T1

JOIN cinema T2

ON ABS(T1.seat\_id-T2.seat\_id)=1

WHERE T1.free=1 AND T2.free=1

ORDER BY T1.seat\_id

SELECT seat\_id

FROM cinema

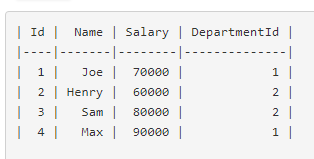
WHERE free=1 AND

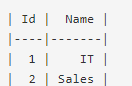
(seat\_id+1 IN (SELECT seat\_id FROM cinema WHERE free=1)

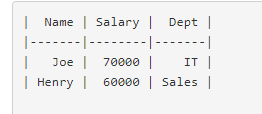
OR seat\_id-1 IN (SELECT seat\_id FROM cinema WHERE free =1))

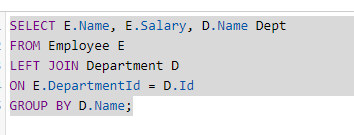
180. Consecutive Numbers

184. Department Highest Salary

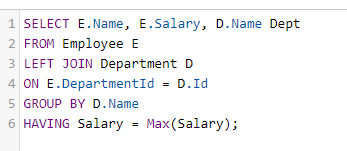


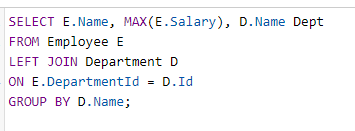


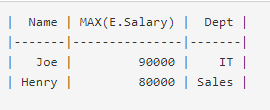




**→→**



**→→→ No records.** 因为7000 不是 IT Salary最大的， 6000 也不是Sales里最大的



**→→→**

根据代码顺序，先找MAX, 然后附上Name of first row of dept. 所以Salary和Name并不匹配

稍稍改下dataset Employee中 inverse Sam and henry

