SQL50-----还是得都看一下对应的python怎么写

按题型和知识点进行总结和练习

[**584. Find Customer Referee**](https://leetcode.com/problems/find-customer-referee/)

这里的问题在于 null值怎么select ---用coalesce()获取该列的所有null值并赋值

coalesce(referee,0)**如果 referee\_id 是 NULL**，COALESCE(referee\_id, 0) 返回 0

[**1148. Article Views I**](https://leetcode.com/problems/article-views-i/)

**Select Distinct**  a, b, c from

Distinct: remove duplicated rows in a col.

[**1683. Invalid Tweets**](https://leetcode.com/problems/invalid-tweets/)

Length(): to get the length of char

图示

AI 生成的内容可能不正确。

[**1378. Replace Employee ID With The Unique Identifier**](https://leetcode.com/problems/replace-employee-id-with-the-unique-identifier/)

简单理解：

Left join B

on B.id=A.id

就是将B中的元素加入到A中，A列不做任何变动。如果A列中有些没有对应B元素，则该处为null.

在select from 的时候，即使表格进行了left join，要挑选出的col还是带原表头 eg：u.id, s.unique\_id

给一个例子在这：

select p.product\_name, s.year, s.price

from Sales s

join Product p

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

[**1581. Customer Who Visited but Did Not Make Any Transactions**](https://leetcode.com/problems/customer-who-visited-but-did-not-make-any-transactions/)

这题的问题是一开始没搞懂题目表格中的cols所代表的意思。这是有关顾客交易记录的统计表格，每一次visit都有一个不同的id记录（unique values），transaction也是每次记录一个不同的id。

所以对所选的每位顾客的visit次数进行统计，需要先group by每位顾客的id，然后count visit这一列一共有多少行，出现了多少不同的id就是访问了多少次。

查找null值 用 where x is null

按知识点练习

**self-join方法- from w1，w2**, 根据查找目的注意join condition

**197**:

select w1.id

from weather w1, weather w2 #self join 直接‘copy’成两份

where datediff(w1.recordDate, w2.recordDate) = 1

and w1.temperature > w2.temperature;

**181**: (self-join方法) ---from 表1，表2; where A=B and…

**182**．法一：Self-join/inner join+ group by 确保最后的查询结果只输出一次

法二：group by + having count

**1661**: self-join + group by , round(avg(),3) 注意join conditions

上述使用self-join方法的题目也可以用inner join

Self- join： from 表1， 表2

**Inner join**： from表1 inner join表2 on 列关系

**补充各种join使用场景：**

* **INNER JOIN**：当你只关心两个表之间匹配的数据时使用。
* **LEFT JOIN**：当你需要保留左表的所有记录，并查看右表是否有匹配时使用。
* **RIGHT JOIN**：当你需要保留右表的所有记录，并查看左表是否有匹配时使用。
* **FULL JOIN**：当你需要保留两个表的所有记录，匹配的行才显示在结果中，没匹配的部分用 NULL 填充时使用。
* **CROSS JOIN**：当你需要生成所有可能的行组合时使用。

**577.**left join + need to query NULL value for the result

**1280**. cross join, left join, group by, order by, count

思路：统计每门科目的次数，首先进行cross join全部组合排序（因为即使没有考过试的科目也要统计为null），然后连接考试信息，按id和科目排序后count。（仅按照id进行count**不会混淆不同课程的记录**，因为 GROUP BY 已按 student\_id 和 subject\_name 进行分组，count会统计每个id每门科目的行数）

**Group by** 之后用**Having**筛选 （必须与group by一起使用）

语法：

SELECT column1, column2, ..., AGGREGATE\_FUNCTION(column)

FROM table

WHEREcondition -- (可选，过滤原始数据)

GROUP BY column1, column2, ...

**HAVING** condition -- 过滤分组后的结果 可与聚合函数配合使用COUNT(), SUM(), AVG(), MAX(), MIN()

**182**: 法二：group by + having count

196: use delete from: 仅保留min即delete所有其他行

法一：self-join（选取min即删除所有>的）

法二：group by 将min创建为一个新的列表，not in 则delete

197: