# SQL Practice

1. Select 是程序运行的最后一步，select后面的程序是赋值的过程。
2. AND 和(，)不同， AND只能用于条件语句 eg：where， 用select时需用 (，)。
3. X % y : 表示x除y余数。
4. Distinct：表示选取不重复的元素，exclude duplicates from your answer。
5. Length()：表示字符串的长度
6. Like: string like “\*%”，字符串像\*开头一样。
7. Regexp: string like “^[\*\*\*].\*”，字符串像regular expression 一样。
   1. [^\*\*\*]：表示不以\*\*\*开头。
8. Right(name, x): 在使用 order by的时候可以使用字符串的指定位置（从右数x个）来排序。
   1. substring(name,len(name)-2, 3): 用同样功用，意思是：选取name，从name的倒数第三个（len(name)-2）开始，往右数3个。
9. 判断三角形：
   1. 使用 Then “\*\*\*” 来代替 print “\*\*\*”
   2. 使用when …Then 来表达 if else
   3. 在else…end 搭配使用
   4. Select case… from xxx: 在…输入case的内容

SELECT CASE

WHEN A + B > C THEN CASE WHEN A = B AND B = C THEN 'Equilateral'

WHEN A = B OR B = C OR A = C THEN 'Isosceles'

WHEN A != B OR B != C OR A != C THEN 'Scalene' END

ELSE 'Not A Triangle'

END

FROM TRIANGLES;

1. 判断三角形2：
   1. 这里使用了**嵌套，每个case对应一个else**

SELECT CASE

WHEN A + B > C AND A + C > B AND B + C > A THEN

CASE

WHEN A = B AND B = C THEN 'Equilateral'

WHEN A = B OR B = C OR A = C THEN 'Isosceles'

ELSE 'Scalene'

END

ELSE 'Not A Triangle'

END

FROM TRIANGLES;

1. 判断三角形3 ：
   1. 运用了if， 逻辑和case..end 一样： if1(CASE 1, if2(case1, if3(case2, case 3)), CASE 2 ):
      1. 解释：当CASE 1 发生，进入if2, 如果case1不发生，发生if3, 如果case2不发生，发生case3 (else)；如果CASE 1 都不发生，直接CASE 2。
      2. If(条件，条件发生的结论，else)

SELECT IF((A+ B) > C and (A + C)> B and (B + C) > A ,

IF(A = B and A = C and C=B,'Equilateral',

IF(A=B OR B = C OR A = C,'Isosceles','Scalene'))

,'Not A Triangle') TRIANGLES

FROM TRIANGLES

1. Concat (sth, sth)：将字符串整合在一起 (“sth”+”sth”).
   1. Sth可以是另一个concat()。
   2. Concat (sth ,sth ,sth ,...)
   3. **用于去空格的整合**
2. Left(sth,1):输出sth的第一个字符 == substring (sth,1,1)
3. Lower(), upper() :大小写
4. Round(): 四舍五入
5. Replace(name,’x’,’y’): 把name里的字符x替换成Y
   1. Name是数字型也没关系，avg()也可以直接在replace后使用
6. (salary\*months) as earning: 计算结果可以用as赋值，并在下面的where引用
7. Group by 1 : 意思是选择select里的第一个元素（earning）作为分割的标准

select (salary\*months) as earning, count(\*) from employee

group by 1

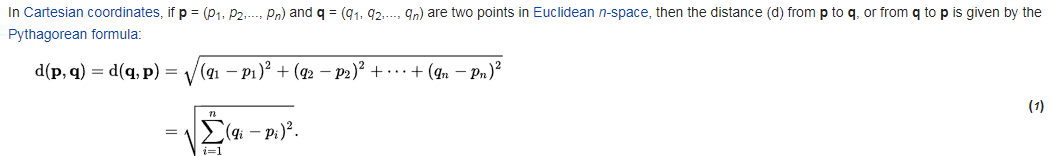
order by earning desc

limit 1

1. ><： 同时满足大于x和小于y需要分开写，不能 y>n >x。
2. Where 内可以嵌套 select，注意括号的使用

where lat\_n = (select max(lat\_n) from station

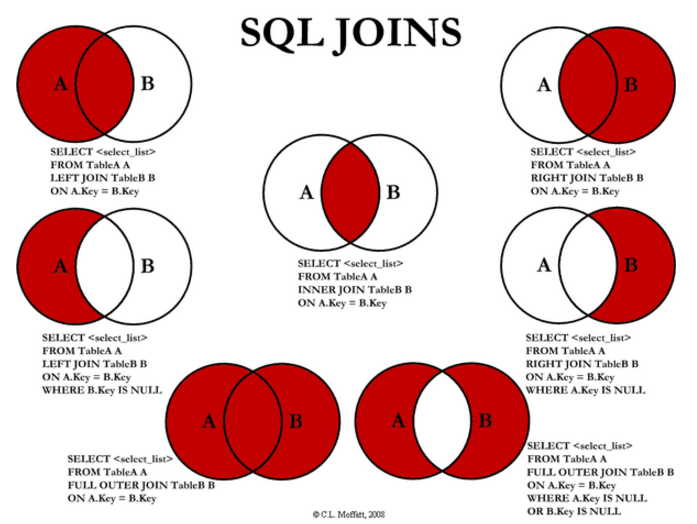
where lat\_n < 137.2345)

1. Manhattan distance:  The distance between two points measured along axes at right angles. In a plane with p1 at (x1, y1) and p2 at (x2, y2), it is |x1 - x2| + |y1 - y2|.
2. The **Euclidean distance** : between points **p** and **q** is the length of the [line segment](https://en.wikipedia.org/wiki/Line_segment) connecting them ({\displaystyle {\overline {\mathbf {p} \mathbf {q} }}}pq). 
3. As 可以省略 station as s 🡺 station s
4. 调用自身用来作比较并记数。
   1. 注意这里有外层的station as S, 内层的是station, 引用的时候用lat\_n 和 s.lat\_n 分开两层。

Select round(S.LAT\_N,4) as median from station as S

(select count(Lat\_N) from station where Lat\_N < S.LAT\_N ) = (select count(Lat\_N) from station where Lat\_N > S.LAT\_N)

1. Left join, right join, inner join, outer join:



1. X Between (y1) and (y2) ：可以用来代替：(X >= y1) and (X <= y2)

on (students.marks >= grades.min\_mark) and (students.marks <= grades.max\_mark)

🡺 on s.marks BETWEEN min\_mark AND max\_mark

1. Select (case ),….: 可以用于逻辑判断。Eg

select (CASE grades.grade>=8 WHEN TRUE THEN students.name ELSE null END)

1. Oder by x,y, z: 先按x排序，若x一样，则按y排序，若y一样，则按z排序。
2. Group by x,y: GROUP BY X, Y意思是将所有具有相同X字段值和Y字段值的记录放到一个分组里。 如果是aggregate function，就不用group by 两列。

select h.hacker\_id, h.name

FROM submissions s

JOIN challenges c

ON s.challenge\_id = c.challenge\_id

JOIN difficulty d

ON c.difficulty\_level = d.difficulty\_level

JOIN hackers h

ON s.hacker\_id = h.hacker\_id

where s.score = d.score

and c.difficulty\_level = d.difficulty\_level

group by h.hacker\_id, h.name

having count(s.hacker\_id) > 1

order by count(s.hacker\_id) desc, s.hacker\_id asc

* 1. when you do group by you need to pull all the things you select
  2. Because you cannot select hacker\_id and Hak.name at once if group by is on a single column.We can select only functions and column in a Query if group by is used on a single column.
  3. Yeah, I had the same question. I think I understand now. We thought it would be OK to group by id only because we KNEW that id and name are uniquely paired to each other. However sql does not know that and cannot assume that. If we group by id only, then from sql's point of view, it may well be possible that multiple different names corresponding to the same id are counted together. Therefore sql cannot print a separate column for name.
  4. GROUP BY without aggregate function

<https://stackoverflow.com/questions/20074562/group-by-without-aggregate-function>

1. Group by having: having 指分组时满足的条件
2. Join..on..join..on…：可以不断地加入table
3. 在两个table中选取最小值: There is a sub-query to make sure that if the wand age and wand power matches with another, to return the cheapest (lowest coins\_needed) wand with.

select w.id, wp.age, w.coins\_needed, w.power

from wands w

inner join wands\_property wp

on w.code = wp.code

where wp.is\_evil = 0 and w.coins\_needed =(

select min(coins\_needed)

from wands w1

join wands\_property wp1

on w1.code = wp1.code

where w1.power = w.power and wp1.age = wp.age)

group by w.id, wp.age, w.coins\_needed, w.power

order by w.power desc,wp.age desc

1. Group by… having…or…：可以增加多个条件进行分组 （筛选输出与不输出）
2. Select…from (select)：from的table也可以通过select自己创建，里面的元素列可以作为select的值输出或作为聚组函数输出。
3. Group by… having .. in sth：满足在 sth里的元素才能被输出，sth 是table，可以用 select 构建。

select h.hacker\_id, h.name ,count(c.hacker\_id) as count1

from Hackers as h

join Challenges as c

on c.hacker\_id = h.hacker\_id

group by h.hacker\_id, h.name

having count1 =

(select max(temp1.cnt)

from (select count(\*) as cnt

from challenges

group by hacker\_id

order by hacker\_id) temp1)

or count1 in

(select t.cnt

from (select count(\*) as cnt

from challenges

group by hacker\_id) t

group by t.cnt

having count(t.cnt) = 1)

order by count1 desc, c.hacker\_id

1. Where 和 having 的区别：

　　WHERE语句在GROUP BY语句之前；SQL会在分组之前计算WHERE语句（不包含聚组函数）。     
  
　　HAVING语句在GROUP BY语句之后；SQL会在分组之后计算HAVING语句（经常包含聚组函数）。

<http://keep.iteye.com/blog/240665>

<http://www.cnblogs.com/s021368/articles/1288720.html>

1. Select..from x,y,z : from可以多个table。
2. Datediff (end\_date, start\_date) : 用来做日期的差值。
3. 每一个创建的table都需要自己的名称：Every derived table must have its own alias。
4. Where X in (not in) Some\_table : 作用类似 having sth in…，但 这是在分组之前完成的。

select start\_date, min(end\_date)

from

(select start\_date from projects where start\_date not in (select end\_date from projects)) a,

(select end\_date from projects where end\_date not in (select start\_date from projects)) b

where start\_date < end\_date

group by start\_date

order by datediff(min(end\_date),start\_date) asc, start\_date

1. Subquery returns more than 1 row ： 子查询返回值不唯一，意思是必须唯一才正确。
2. Using ():可以在join.. on x.id = y.id 时代替on.. 变成 join…using(id)

select s.name

from students s

join friends f using(id)

join packages p1 using(id)

join packages p2 on p2.id = f.friend\_id

where p2.salary > p1.salary

order by p2.salary