模式：mysql

# 176. 第二高的薪水

<https://leetcode-cn.com/problems/second-highest-salary/>

## limit，IFNULL

答案：

select max(Salary) SecondHighestSalary

from Employee

where Salary < (select max(Salary) from Employee)

SELECT

(SELECT DISTINCT

Salary

FROM

Employee

ORDER BY Salary DESC

LIMIT 1 OFFSET 1) AS SecondHighestSalary

select IFNULL((SELECT distinct Salary from Employee order by Salary desc limit 1 offset 1),NULL)

# 177. 第N高的薪水

答案;

CREATE FUNCTION getNthHighestSalary(N INT) RETURNS INT

BEGIN

SET N := N-1;

RETURN (

# Write your MySQL query statement below.

SELECT

salary

FROM

employee

ORDER BY

salary DESC

LIMIT N, 1

);

END

这种解法形式最为简洁直观，但仅适用于查询全局排名问题，如果要求各分组的每个第N名，则该方法不适用；而且也不能处理存在重复值的情况。

## SET N := N-1;

CREATE FUNCTION getNthHighestSalary(N INT) RETURNS INT

BEGIN

RETURN (

SELECT IF(count<N,NULL,min)

FROM

(SELECT MIN(Salary) AS min, COUNT(1) AS count

FROM

(SELECT DISTINCT Salary

FROM Employee ORDER BY Salary DESC LIMIT N) AS a

) as b

);

END

<https://leetcode-cn.com/problems/nth-highest-salary/solution/liang-ge-hao-li-jie-de-fang-an-by-postor/>

# 178. 分数排名



**使用库函数：**

Mysql 8.0 四大排名函数

## ROW\_NUMBER、RANK、DENSE\_RANK、NTILE简介

ROW\_NUMBER()函数作用就是将select查询到的数据进行排序，每一条数据加一个序号。

RANK()函数，顾名思义排名函数，可以对某一个字段进行排名。ROW\_NUMBER()是排序，当存在相同成绩的学生时，ROW\_NUMBER()会依次进行排序，他们序号不相同，而Rank()则出现相同的，他们的排名是一样的。

DENSE\_RANK()密集的排名他和RANK()区别在于，排名的连续性，DENSE\_RANK()排名是连续的，RANK()是跳跃的排名，所以一般情况下用的排名函数就是RANK()。





**答案：**

select Score, dense\_rank() over (order by Score desc) as `rank` from Scores;

## 敏感字别名 `rank`

复杂度较高，但不用库函数的写法。

select a.Score as Score,

(select count(distinct b.Score) from Scores b where b.Score >= a.Score) as `Rank`

from Scores a

order by a.Score DESC

相当于两层循环。可以化为join 内连接（未看）

# 180. 连续出现的数字

答案：

select distinct(l1.num) as ConsecutiveNums from

logs l1,logs l2,logs l3

where l1.id+1=l2.id

and l2.id+1=l3.id

and l1.num=l2.num

and l2.num=l3.num

其实是三表连接。

## If函数

其他答案：

思路解析：

#①首先遍历一遍整张表，找出每个数字的连续重复次数

#具体方法为：

#初始化两个变量，一个为pre，记录上一个数字；一个为count，记录上一个数字已经连续出现的次数。

#然后调用if()函数，如果pre和当前行数字相同，count加1极为连续出现的次数；如果不同，意味着重新开始一个数字，count重新从1开始。

#最后，将当前的Num数字赋值给pre，开始下一行扫描。

select

Num, #当前的Num 数字

if(@pre=Num,@count := @count+1,@count := 1) as nums, #判断 和 计数

@pre:=Num #将当前Num赋值给pre

from Logs as l ,

(select @pre:= null,@count:=1) as pc #这里需要别名

#上面这段代码执行结果就是一张三列为Num,count as nums,pre的表。

#②将上面表的结果中，重复次数大于等于3的数字选出，再去重即为连续至少出现三次的数字。

select

distinct Num as ConsecutiveNums

from

(select Num,

if(@pre=Num,@count := @count+1,@count := 1) as nums,

@pre:=Num

from Logs as l ,

(select @pre:= null,@count:=1) as pc

) as n

where nums >=3;

每个派生表必须有自己的别名。

## 定义变量：

<https://www.cnblogs.com/caicaizi/p/9803013.html>

1、在MySQL中实现ROW\_NUMBER普通排名函数

要在mysql中声明一个变量，你必须在变量名之前使用@符号。FROM子句中的(@curRank := 0)部分允许我们进行变量初始化，而不需要单独的SET命令。当然，也可以使用SET，但它会处理两个查询：

（1）

SELECT pid, name, age, @curRank := @curRank + 1 AS rank

FROM players p, (

SELECT @curRank := 0

) q

ORDER BY age

（2）

SET @curRank := 0;

SELECT pid, name, age, @curRank := @curRank + 1 AS rank

FROM players

ORDER BY age

3、在MySQL中实现DENSE\_RANK()排名函数

现在，如果我们希望为并列数据的行赋予相同的排名，则意味着那些在排名比较列中具有相同值的行应在MySQL中计算排名时保持相同的排名(例如在我们的例子中的age)。为此，我们使用了一个额外的变量。

## When函数

SELECT pid, name, age,

CASE

WHEN @prevRank = age THEN @curRank

WHEN @prevRank := age THEN @curRank := @curRank + 1

END

AS rank

FROM players p,

(SELECT @curRank :=0, @prevRank := NULL) r

ORDER BY age

=和：=的区别

WHEN @prevRank = age THEN @curRank：此时=是判断的作用，@num不等于@num+1，所以始终返回0，如果改为@num=@num,始终返回1了。mysql数据库中，用1表示真，0表示假。

4、在MySQL中实现Rank高级并列排名函数

当使用RANK()函数时，如果两个或以上的行排名并列，则相同的行都会有相同的排名，但是实际排名中存在有关系的差距。

SELECT pid, name, age, rank FROM --这个select是为了剔除中间变量

(SELECT pid, name, age,

@curRank := IF(@prevRank = age, @curRank, @incRank) AS rank,

@incRank := @incRank + 1,

@prevRank := age --运用select实现循环

FROM players p, (

SELECT @curRank :=0, @prevRank := NULL, @incRank := 1 --select实现变量初始化

) r

ORDER BY age) s

# 185. 部门工资前三高的所有员工

我做出了，答案如下：

select department.name as department,a.name as employee, salary

from employee a join department on a.departmentid=department.id

where 3>=(select count(distinct b.salary) from employee b where b.departmentid=a.departmentid and b.salary>=a.salary )

# where 3>(select count(distinct b.salary) from employee b where b.departmentid=a.departmentid and a.id!=b.id and b.salary>a.salary ) 两种where写法都可以

order by department.id asc, salary desc

别人的解法：

/\*\*

解题思路:先对Employee表进行部门分组工资排名，再关联Department表查询部门名称，再使用WHERE筛选出排名小于等于3的数据（也就是每个部门排名前3的工资）。

\*\*/

SELECT

B.Name AS Department,

A.Name AS Employee,

A.Salary

FROM (SELECT DENSE\_RANK() OVER (partition by DepartmentId order by Salary desc) AS ranking,DepartmentId,Name,Salary

FROM Employee) AS A

JOIN Department AS B ON A.DepartmentId=B.id

WHERE A.ranking<=3

解工资在前百分之三的员工

SELECT top 3 percent

B.Name AS Department,

A.Name AS Employee,

A.Salary

FROM (SELECT DENSE\_RANK() OVER (partition by DepartmentId order by Salary desc) AS ranking,DepartmentId,Name,Salary

FROM Employee order by ranking) AS A

JOIN Department AS B ON A.DepartmentId=B.id

# 196. 删除重复的电子邮箱

官方题解：自连接

DELETE p1 FROM Person p1,

Person p2

WHERE

p1.Email = p2.Email AND p1.Id > p2.Id

DELETE FROM Person

WHERE Id NOT IN ( -- 删除不在查询结果中的值

SELECT id FROM

(

SELECT MIN(Id) AS Id -- 排除Email相同时中Id较大的行

FROM Person

GROUP BY Email

) AS temp -- 此处需使用临时表，否则会发生报错

)

我的解法：

正确：

delete from person

where exists (select \* from (select a.id as id from person a,person b where a.id>b.id and a.email=b.email ) as temp where temp.id=person.id)

错误：

delete from person

where id in (select a.id from person a,person b where a.id>b.id and a.email=b.email )

必须用临时表来存数据，因为mysql不允许同时对一个表delete和select联合操作

# 197. 上升的温度

最佳写法：

select a.Id from Weather a join Weather b on a.Temperature > b.Temperature and dateDiff(a.RecordDate,b.RecordDate) = 1

用and连接比where要快

时间比较函数：

Date1 = Date\_add(date2, interval 1 day)

Datediff(date1, date2) = 1

# 262. 行程和用户

我的解法：971ms

select request\_at as day, round(count(status!="completed" or null)/count(\*),2) as "Cancellation Rate" from trips

where not exists (select \* from (select users\_id bid from users where Banned="Yes") t where driver\_id=bid or client\_id=bid)

group by request\_at #临时表可以去掉

官方解答：295ms

SELECT

request\_at as 'Day', round(avg(Status!='completed'), 2) as 'Cancellation Rate'

FROM

trips t JOIN users u1 ON (t.client\_id = u1.users\_id AND u1.banned = 'No')

JOIN users u2 ON (t.driver\_id = u2.users\_id AND u2.banned = 'No')

WHERE

request\_at BETWEEN '2013-10-01' AND '2013-10-03'

GROUP BY

request\_at

1. 巧用avg 计算 count(status!="completed" or null)/count(\*)
2. 三表连接 是一个不那么好悟到的三表连接，不可以改写成两表
3. 时间范围限定函数