

-SQL211

获取当前薪水第二多的员工的emp_no以及其对应的薪水salary

SQL211 获取当前薪水第二多的员工的emp_no以及其对应的薪水salary

编辑 收藏 报错 新建

题目

题解(167)

讨论(727)

排行

面经

new

简单 通过率: 41.26% 时间限制: 1秒 空间限制: 32M

描述

有一个薪水表salaries简况如下:

| emp_no | salary | from_date | to_date |
|--------|--------|------------|------------|
| 10001 | 88958 | 2002-06-22 | 9999-01-01 |
| 10002 | 72527 | 2001-08-02 | 9999-01-01 |
| 10003 | 43311 | 2001-12-01 | 9999-01-01 |

请你获取薪水第二多的员工的emp_no以及其对应的薪水salary,

若有多个员工的薪水为第二多的薪水，则将对应的员工的emp_no和salary全部输出，并按emp_no升序排序。

| emp_no | salary |
|--------|--------|
| 10002 | 72527 |

SQL211 获取当前薪水第二多的员工的emp_no以及其对应的薪水salary

题目 题解(167) 讨论(727) 排行 面经 new

发表于 2020-08-10 13:15

你夕

```
1 select c.emp_no, c.salary from
2 (select emp_no, salary,
3 dense_rank() over(order by salary desc) as rk from salaries) as c
4 where c.rk=2 #限定条件 第二名
5 limit 1 #如果有多个同值（多个第二名），只取第一个值
```

你们不要打啦，你们不要打啦，我看窗口函数就挺好的

▲ 30 ▼ 16 举报

MiracleXY

```
1 select emp_no,salary
2 from salaries
3 where to_date = '9999-01-01',
4 order by salary desc limit 1,1 //salary降序排列，从1+1的位置取一个记录
```

TOP

发表于 2017-07-10 04:35

shiaogo

```
13 ▲ 7 ▼ 通过全部用例 运行时间 41ms 占用内存 6532KB
```

TOP

注: 如果不限定当前条件

练习模式 提示未通过的测试用例，便于代码调试
您的代码已保存

执行结果 自测输入 提交记录 自测运行 保存并提交

解法一

```
#通过子查询查出薪水第二多的数值salary  
#然后让salaries的记录中员工的salary=第二多薪水；  
#按emp_no升序排序  
select emp_no,salary  
from salaries  
where salary=  
(  
    select distinct salary  
    from salaries  
    order by salary desc  
    limit 1,1  
)  
order by emp_no asc
```

解法二

```
#第二种解法使用窗口函数，在缓存给salary进行降序排序并给上序列号标签  
#给这个窗口表取别名c  
#where限定条件对这个窗口表筛选从而查询到符合题目条件的记录  
select c.emp_no, c.salary from  
(select emp_no, salary,  
dense_rank() over(order by salary desc) as rk from salaries) as c  
where c.rk=2 #限定条件 第二名  
limit 1 #如果有多个同值（多个第二名），只取第一个值
```

-SQL212

获取当前薪水第二多的员工的emp_no以及其对应的薪水salary

题目

题解(338)

讨论(897)

排行

面经

new

描述

有一个员工表employees简况如下:

| emp_no | birth_date | first_name | last_name | gender | hire_date |
|--------|------------|------------|-----------|--------|------------|
| 10001 | 1953-09-02 | Georgi | Facello | M | 1986-06-26 |
| 10002 | 1964-06-02 | Bezalel | Simmel | F | 1985-11-21 |
| 10003 | 1959-12-03 | Parto | Bamford | M | 1986-08-28 |
| 10004 | 1954-05-01 | Chirstian | Koblick | M | 1986-12-01 |

有一个薪水表salaries简况如下:

| emp_no | salary | from_date | to_date |
|--------|--------|------------|------------|
| 10001 | 88958 | 2002-06-26 | 9999-01-01 |
| 10002 | 72527 | 2001-08-02 | 9999-01-01 |
| 10003 | 43311 | 2001-12-01 | 9999-01-01 |
| 10004 | 74057 | 2001-11-27 | 9999-01-01 |

请你查找薪水排名第二多的员工编号emp_no、薪水salary、last_name以及first_name, **不能使用order by完成**, 以上例子输出为:

(温馨提示:sqlite通过的代码不一定能通过mysql, 因为SQL语法规规定, 使用聚合函数时, select子句中一般只能存在以下三种元素: 常数、聚合函数, group by 指定的列名。如果使用非group by的列名, sqlite的结果和mysql可能不一样)

思考思路(不能使用order by)

目标字段-select: emp_no,salary,last_name,first_name

库表来源-from: employees&salaries

连接关系-join on: employees join salaries on employees.emp_no=salaries.emp.no

筛选条件 WHERE/having(先列出再决定): salary排名第二多

聚合依据-group by: 好像不用聚合

梳理思路:

1、先使用Max函数查询出薪水最高的员工的薪水

2、然后以这个最高薪水为where的筛选条件, s.salary !=

3、查询出在剔除最高薪水的数据记录下剩下的员工中薪水最高的即为薪水第二多

4、然后以第二多的薪水的值为where的筛选条件s.salary=第二多的薪水;连接两张表格查询出题目要求的字段

分段编辑:

```
1、select max(salary) from salaries where to_date='9999-01-01'
```

```
2、where salary not in (select max(salary) from salaries where to_date='9999-01-01')
```

```
3、
```

```
select max(salary)
from salaries
where salary !=(
    select max(salary) from salaries where to_date='9999-01-01'
)
```

```
#组合代码
```

```
4、
```

```
select em.emp_no,sa.salary,em.last_name,em.first_name
from employees em join salaries sa
on em.emp_no=sa.emp_no
where sa.salary=(
    select max(salary)
    from salaries
    where salary !=
    (
        select max(salary)
        from salaries
        where to_date='9999-01-01'
    )
)
and to_date='9999-01-01'
```

-SQL213

查找所有员工的last_name和first_name以及对应的dept_name

SQL213 查找所有员工的last_name和first_name以及对应的dept_name

收藏 分享 新

题目

题解(157)

讨论(548)

排行

面经

通过

有一个员工表employees简况如下:

| emp_no | birth_date | first_name | last_name | gender | hire_date |
|--------|------------|------------|-----------|--------|------------|
| 10001 | 1953-09-02 | Georgi | Facello | M | 1986-06-26 |
| 10002 | 1964-06-02 | Bezalel | Simmel | F | 1985-11-21 |
| 10003 | 1959-12-03 | Parto | Bamford | M | 1986-08-28 |
| 10004 | 1954-05-01 | Chirstian | Koblick | M | 1986-12-01 |

有一个部门表departments表简况如下:

| dept_no | dept_name |
|---------|-----------------|
| d001 | Marketing |
| d002 | Finance |
| d003 | Human Resources |

有一个，部门员工关系表dept_emp简况如下:

| emp_no | dept_no | from_date | to_date |
|--------|---------|------------|------------|
| 10001 | d001 | 1986-06-26 | 9999-01-01 |
| 10002 | d001 | 1996-08-03 | 9999-01-01 |

请你查找所有员工的last_name和first_name以及对应的dept_name，也包括暂时没有分配部门的员工，以上例子输出如下:

| last_name | first_name | dept_name |
|-----------|------------|-----------|
| Facello | Georgi | Marketing |
| Simmel | Bezalel | Marketing |
| Bamford | Parto | Finance |
| Koblick | Chirstian | NULL |

思考思路

目标字段:employees.last_name;employees.first_name;departments.dept_name

库表来源: employees&departments&dept_emp

连接关系: employees.emp_no=dept_emp.emp_no;department.dept_no=dept_emp.dept_no

筛选条件: 无

聚合依据：无

梳理思路：1、员工表employees先和dept_emp左连接;2、然后新连接好的组合表再和departments表左连接在一起

分段编辑：

#1&2

```
select employees.last_name,employees.first_name,departments.dept_name
from employees left join dept_emp
on employees.emp_no=dept_emp.emp_no
left join departments
on dept_emp.dept_no=departments.dept_no
```

组合代码

```
select employees.last_name,employees.first_name,departments.dept_name
from employees left join dept_emp
on employees.emp_no=dept_emp.emp_no
left join departments
on dept_emp.dept_no=departments.dept_no
```

SQL213 查找所有员工的last_name和first_name以及对应的dept_name

题目 题解(157) 讨论(548) 排行 面经 new

中等 通过率: 36.63% 时间限制: 1秒 空间限制: 32M

描述

有一个员工表employees简况如下:

| emp_no | birth_date | first_name | last_name | gender | hire_date |
|--------|------------|------------|-----------|--------|------------|
| 10001 | 1953-09-02 | Georgi | Facello | M | 1986-06-26 |
| 10002 | 1964-06-02 | Bezalel | Simmel | F | 1985-11-21 |
| 10003 | 1959-12-03 | Porto | Bamford | M | 1986-08-28 |
| 10004 | 1954-05-01 | Christian | Koblick | M | 1986-12-01 |

有一个部门表departments表简况如下:

| dept_no | dept_name |
|---------|-----------------|
| d001 | Marketing |
| d002 | Finance |
| d003 | Human Resources |

有一个，部门员工关系表dept_emp简况如下:

| emp_no | dept_no | from_date | to_date |
|--------|---------|-----------|---------|
|--------|---------|-----------|---------|

-SQL215(困难)

查找在职工自入职以来的薪水涨幅情况

SQL215 查找在职员工自入职以来的薪水涨幅情况

困难 通过率: 20.95% 时间限制: 1秒 空间限制: 32M

题目

题解(433)

讨论(997)

排行

面经

new

描述

有一个员工表employees简况如下:

| emp_no | birth_date | first_name | last_name | gender | hire_date |
|--------|------------|------------|-----------|--------|------------|
| 10001 | 1953-09-02 | Georgi | Facello | M | 2001-06-22 |
| 10002 | 1964-06-02 | Bezalel | Simmel | F | 1999-08-03 |

有一个薪水表salaries简况如下:

| emp_no | salary | from_date | to_date |
|--------|--------|------------|------------|
| 10001 | 85097 | 2001-06-22 | 2002-06-22 |
| 10001 | 88958 | 2002-06-22 | 9999-01-01 |
| 10002 | 72527 | 1999-08-03 | 2000-08-02 |
| 10002 | 72527 | 2000-08-02 | 2001-08-02 |

请你查找在职员工自入职以来的薪水涨幅情况，给出在职员工编号emp_no以及其对应的薪水涨幅growth，并按照growth进行升序，以上例子输出为

(注: to_date为薪资调整某个结束日期，或者为离职日期，to_date='9999-01-01'时，表示依然在职，无后续调整记录)

| emp_no | growth |
|--------|--------|
| 10001 | 3861 |

思考思路

目标字段: emp_no&growth--(salary的差值)

库表来源: employees&salaries

连接关系: inner join on employees.emp_no=salaries.emp_no

筛选条件: 无

聚合依据: 无

梳理思路:

1、先连接两张表格

2、开两个滑动窗口函数max&min(salary)over(partition by emp_no order by to_date rows between unbounded preceding and unbounded following)

3、将上面操作当作一个子查询放在from后的表格中

分段编辑:

1、

```

select emp_no,salary
from employees join salaries on employees.emp_no=salaries.emp_no

2、

select emp_no,max(salary)over(partition by emp_no order by to_date rows between unbounded preceding and unbounded following) 最高工资, min(salary)over(partition by emp_no order by to_date rows between unbounded preceding and unbounded following) 最低工资

from employees join salaries on employees.emp_no=salaries.emp_no

3、

select emp_no,(h_l.最高工资-h_l.最低工资) growth
from
(子查询2) h_l
where (h_l.最高工资-h_l.最低工资) !=0
order by growth asc

```

组合代码1

```

select emp_no,(h_l.最高工资-h_l.最低工资) growth
from
(
    select emp_no,max(salary)over(partition by emp_no order by to_date rows
between unbounded preceding and unbounded following) 最高工资,
min(salary)over(partition by emp_no order by to_date rows between unbounded
preceding and unbounded following) 最低工资
        from employees join salaries on employees.emp_no=salaries.emp_no
) h_l
where (h_l.最高工资-h_l.最低工资) !=0
order by growth asc

```

其他解法

目标字段：emp_no&growth--(salary的差值)

库表来源：employees&salaries

连接关系：inner join on employees.emp_no=salaries.emp_no

筛选条件：无

聚合依据：emp_no;根据员工聚合：求出员工工资的最小值

梳理思路：

- 1、直接用第二张表salaries；首先计算出员工刚入职的初始工资
- 2、然后再计算出员工最后的薪资调整的结束日期时候的工资即为当前工资最高点
- 3、将两个工资值相减即可得到growth

分段编辑：

1、

```
(  
select emp_no,min(salary) 初始工资  
from salaries  
group by 1  
) a
```

2、

```
(  
select emp_no,to_date,salary 调整后工资  
from salaries  
group by 1,2,3  
having to_date='9999-01-01'  
) b
```

3、

```
select emp_no,(b.调整后工资-a.初始工资) growth  
from  
(  
子查询1  
) ,  
(  
子查询2  
)  
where a.emp_no=b.emp_no  
order by growth
```

组合代码2

```
#写法2  
select a.emp_no,(b.调整后工资 - a.初始工资) growth  
from  
(  
select emp_no,min(salary) 初始工资  
from salaries  
group by 1  
) a,  
(  
select emp_no,to_date,salary 调整后工资  
from salaries  
group by 1,2,3  
having to_date = "9999-01-01"  
) b  
where a.emp_no = b.emp_no  
order by growth
```

#写法3 通过employees和salaries表连接起来，以连接键hire_date和from_date相连再查出salary即是入职时初始工资

```
select e.emp_no, j.salary - s.salary as growth  
from employees e
```

```
join salaries s on e.emp_no = s.emp_no and e.hire_date = s.from_date
join salaries j on e.emp_no = j.emp_no and j.to_date = '9999-01-01'
order by growth;
```

-SQL216(中等)

统计各个部门的工资记录数

思考思路

目标字段：dept_no、dept_name、count(工资记录)

库表来源：部门表departments、部门员工关系表dept_emp、薪水表salaries

连接关系：inner join 需要三表连接

salaries.emp_no=dept_emp.emp_no;departments.dept_no=dept_emp.dept_no

筛选条件：where/having 暂无

聚合依据：dept_no&dept_name:按照部门号进行group by的聚合依据；

梳理思路：连接dept_emp、salaries和departments这三张表格，根据dept_no和dept_name为聚合的依据，count(salary)聚合函数计算工资记录字段sum，由此可以找出每个部门发工资数的记录和具体的部门号以及部门名称

分段编辑：无

组合代码

```
select dep.dept_no
,dep.dept_name
,count(sa.salary) sum
from salaries sa join dept_emp de
on sa.emp_no=de.emp_no join
departments dep on
de.dept_no=dep.dept_no
group by dep.dept_no,dep.dept_name
order by dep.dept_no
```

-SQL217(较难)

对所有员工的薪水按照salary降序进行1-N的排名

题目

题解(174)

讨论(534)

排行

面经

new

较难 通过率: 31.17% 时间限制: 1秒 空间限制: 32M

描述

有一个薪水表salaries简况如下:

| emp_no | salary | from_date | to_date |
|--------|--------|------------|------------|
| 10001 | 88958 | 2002-06-22 | 9999-01-01 |
| 10002 | 72527 | 2001-08-02 | 9999-01-01 |
| 10003 | 43311 | 2001-12-01 | 9999-01-01 |
| 10004 | 72527 | 2001-12-01 | 9999-01-01 |

对所有员工的薪水按照salary降序先进行1-N的排名, 如果salary相同, 再按照emp_no升序排列:

| emp_no | salary | t_rank |
|--------|--------|--------|
| 10001 | 88958 | 1 |
| 10002 | 72527 | 2 |
| 10004 | 72527 | 2 |
| 10003 | 43311 | 3 |

思考思路1-窗口函数

目标字段: emp_no、salary、工资排名t_rank

库表来源: salaries

连接关系: 无

筛选条件: 无

聚合依据: 无

梳理思路:

1、用dense_rank()over(order by salary desc) t_rank; 注解:over里面不用partition by emp_no,是因为如果这样的话就是按照各个员工的工资的记录为对象一个一个区域去分配排名序号, 这样整张表所有的员工的工资排名都是1了。

2、先按照salary 降序排序;然后按照emp_no升序排序;

分段编辑:无

组合代码

```
select emp_no,salary,dense_rank()over(order by salary desc) t_rank  
from salaries  
order by salary desc,emp_no asc
```

思考思路2-复用salaries表

梳理思路：

- 1、从两张相同的表salaries(代码中简写为a,b)进行对比分析，先加一个表格的限定条件：where a.to_date = '9999-01-01' and b.to_date='9999-01-01'
- 2、核心步骤：在where筛选时添加一个条件：a.salary<=b.salary；意思是通过比较两张表的salary的值，找出b表中有多少个b.salary大于等于a.salary，然后通过合计函数count()计算大于等于a.salary的数量作为工资的排名，因为有多少个大于等于一个值排名就是count(distinct b.salary)，用distinct是因为可能会出现多个相同的salary值，那么按照题目意思排名可以不唯一，相同值的排名需相同；比如：当a.salary=94409，有三个b.salary (94692,94409,94409) 大于等于a.salary，但是由于94409是重复的，所以利用count(distinct b.salary)去重可得工资为94409的rank等于2.其余的排名同理可得。
- 3、对a.emp_no进行聚合依据group by

a.emp_no,否则输出的记录只有一条，因为使用了合计函数count()

-SQL218(较难)

获取所有非manager员工当前的薪水情况

SQL218 获取所有非manager员工当前的薪水情况

收藏 分享 新闻

题目

题解(253)

讨论(698)

排行

面经

new

较难

通过率: 31.00%

时间限制: 1秒

空间限制: 32M

描述

有一个员工表employees简况如下:

| emp_no | birth_date | first_name | last_name | gender |
|--------|------------|------------|-----------|--------|
| 10001 | 1953-09-02 | Georgi | Facello | M |
| 10002 | 1964-06-02 | Bezalel | Simmel | F |

有一个，部门员工关系表dept_emp简况如下:

| emp_no | dept_no | from_date | to_date |
|--------|---------|------------|------------|
| 10001 | d001 | 1986-06-26 | 9999-01-01 |
| 10002 | d001 | 1996-08-03 | 9999-01-01 |

有一个部门经理表dept_manager简况如下:

| dept_no | emp_no | from_date | to_date |
|---------|--------|------------|------------|
| d001 | 10002 | 1996-08-03 | 9999-01-01 |

有一个薪水表salaries简况如下:

| emp_no | salary | from_date | to_date |
|--------|--------|------------|------------|
| 10001 | 88958 | 1986-06-26 | 9999-01-01 |

思考思路

目标字段：非manager员工薪水的情况;dept_no、emp_no以及salary

库表来源：员工表employees(员工的基本信息，员工姓名、性别&员工号和出生日期)、部门员工关系表dept_emp(员工号、部门号和入职离职日期)、部门经理表dept_manager(经理部门号、经理的员工号、入职离职日期)、薪水表salaries(员工号、工资、入职离职日期)

连接关系 无

筛选条件 无

聚合依据 无

梳理思路：1、先从部门经理表中查出所有的经理的emp_no；2、再连接部门员工关系表dept_emp和薪水表Salaries并添加筛选条件where emp_no not in (子查询1)

分段编辑:

```
#1、先从部门经理表中查出所有的经理的emp_no  
select emp_no  
from dept_manager
```

```
#2、再连接部门员工关系表dept_emp和薪水表salaries并添加筛选条件where emp_no not in (子查询1)  
select demp.dept_no  
,demp.emp_no  
,sa.salary  
from dept_emp demp join salaries sa on  
demp.emp_no=sa.emp_no  
where demp.emp_no not in  
(  
    #子查询1  
)
```

组合代码

```
select demp.dept_no  
,demp.emp_no  
,sa.salary  
from dept_emp demp join salaries sa on  
demp.emp_no=sa.emp_no  
where demp.emp_no not in  
(  
    select emp_no  
    from dept_manager  
)
```

-SQL219 (困难)

获取员工其当前的薪水比其manager当前薪水还高的相关信息

SQL219 获取员工其当前的薪水比其manager当前薪水还高...

new

题目

题解(384)

讨论(964)

排行

面经

困难 通过率: 29.58% 时间限制: 1秒 空间限制: 32M

描述

有一个，部门关系表dept_emp简况如下:

| emp_no | dept_no | from_date | to_date |
|--------|---------|------------|------------|
| 10001 | d001 | 1986-06-26 | 9999-01-01 |
| 10002 | d001 | 1996-08-03 | 9999-01-01 |

有一个部门经理表dept_manager简况如下:

| dept_no | emp_no | from_date | to_date |
|---------|--------|------------|------------|
| d001 | 10002 | 1996-08-03 | 9999-01-01 |

有一个薪水表salaries简况如下:

| emp_no | salary | from_date | to_date |
|--------|--------|------------|------------|
| 10001 | 88958 | 2002-06-22 | 9999-01-01 |
| 10002 | 72527 | 1996-08-03 | 9999-01-01 |

获取员工其当前的薪水比其manager当前薪水还高的相关信息,

第一列给出员工的emp_no,

第二列给出其manager的manager_no,

第三列给出该员工当前的薪水emp_salary,

第四列给该员工对应的manager当前的薪水manager_salary

以上例子输出如下:

思考思路

目标字段：员工号:emp_no;经理号:manager_no;员工当前的薪水:emp_salary;员工对应的经理manager当前的薪水manager_salary;

库表来源:dept_emp、 dept_manager、 salaries三表

连接关系 inner join ,连接字段on emp_salary.dept_no=manager_salary.dept_no

筛选条件 where emp_salary.salary > manager_salary.salary

聚合依据 无

梳理思路:1、先通过部门关系表dept_emp和salaries连接起来，查出emp_no还有对应的salary;2、将部门经理表和薪水表salaries连接起来，查出emp_no和对应的salary; 3、将两表以同一部门号dept_no连接on emp_salary.dept_no=manager_salary.dept_no作为子查询作为下一步的库表来源；4、将题目要求：员工其当前的薪水比其同部门的manager当前的薪水还高作为外部查询的筛选条件：where

```
emp_salary.salary > manager_salary.salary
```

分段编辑:

#1、先通过部门关系表dept_emp和salaries连接起来，查出emp_no还有对应的salary

```
select demp.emp_no  
,sa.salary  
,demp.dept_no  
from dept_emp demp join salaries sa on  
demp.emp_no=sa.emp_no
```

#2、将部门经理表和薪水表salaries连接起来，查出emp_no和对应的salary

```
select dman.emp_no  
,sa.salary  
,dman.dept_no  
from dept_manager dman join salaries sa on  
dman.emp_no=sa.emp_no
```

#3、将两表以同一部门号dept_no连接on emp_salary.dept_no=manager_salary.dept_no作为子查询
作为下一步的库表来源

```
from  
(  
    子查询1  
) emp_salary  
join  
(  
    子查询2  
) manager_salary  
on emp_salary.dept_no=manager_salary.dept_no
```

#4、将题目要求：员工其当前的薪水比其同部门的manager当前的薪水还高作为外部查询的筛选条件：where
emp_salary.salary > manager_salary.salary

```
where emp_salary.salary > manager_salary.salary
```

组合代码

#组合以上分段编辑的代码

```
select emp_salary.emp_no  
,manager_salary.emp_no  
,emp_salary.salary  
,manager_salary.salary  
from  
(  
    select demp.emp_no  
,sa.salary  
,demp.dept_no  
    from dept_emp demp join salaries sa on  
    demp.emp_no=sa.emp_no  
) emp_salary  
join  
(  
    select dman.emp_no  
,sa.salary
```

```

,dman.dept_no
from dept_manager dman join salaries sa on
dman.emp_no=sa.emp_no
) manager_salary
on emp_salary.dept_no=manager_salary.dept_no
where emp_salary.salary > manager_salary.salary

```

SQL219 获取员工其当前的薪水比其manager当前薪水还高...  Mysql SQL 编程

困难 通过率: 29.58% 时间限制: 1秒 空间限制: 32M

描述

有一个 部门关系表dept_emp简况如下:

| emp_no | dept_no | from_date | to_date |
|--------|---------|------------|------------|
| 10001 | d001 | 1986-06-26 | 9999-01-01 |
| 10002 | d001 | 1996-08-03 | 9999-01-01 |

有一个部门经理表dept_manager简况如下:

| dept_no | emp_no | from_date | to_date |
|---------|--------|------------|------------|
| d001 | 10002 | 1996-08-03 | 9999-01-01 |

有一个薪水表salaries简况如下:

| emp_no | salary | from_date | to_date |
|--------|--------|------------|------------|
| 10001 | 88958 | 2002-06-22 | 9999-01-01 |
| 10002 | 72527 | 1996-08-03 | 9999-01-01 |

获取员工其当前的薪水比其manager当前薪水还高的相关信息,
第一列给出员工的emp_no,
第二列给出其manager的manager_no,
第三列给出该员工当前的薪水emp_salary,
第四列给出该员工对应的manager当前的薪水manager_salary
以上例子输出如下:



恭喜你通过本题

邀请朋友来挑战吧:

分享解题思路

练习模式 提示未通过的测试用例, 便于代码调试
您的代码已保存

通过全部用例 运行时间 44ms 占用内存 6488KB

自测运行 保存并提交

-SQL220 (困难)

汇总各个部门当前员工的title类型的分配数目

题目

题解(260)

讨论(607)

排行

困难 通过率: 26.65% 时间限制: 1秒 空间限制: 32M

描述

有一个部门表departments简况如下:

| dept_no | dept_name |
|---------|-----------|
| d001 | Marketing |
| d002 | Finance |

有一个，部门员工关系表dept_emp简况如下:

| emp_no | dept_no | from_date | to_date |
|--------|---------|------------|------------|
| 10001 | d001 | 1986-06-26 | 9999-01-01 |
| 10002 | d001 | 1996-08-03 | 9999-01-01 |
| 10003 | d002 | 1995-12-03 | 9999-01-01 |

有一个职称表titles简况如下:

| emp_no | title | form_date | to_date |
|--------|-----------------|------------|------------|
| 10001 | Senior Engineer | 1986-06-26 | 9999-01-01 |
| 10002 | Staff | 1996-08-03 | 9999-01-01 |
| 10003 | Senior Engineer | 1995-12-03 | 9999-01-01 |

思考思路

目标字段:部门编号dept_no、dept_name部门名称、对应的部门下所有的员工title以及该类型title对应的数目count; 注意: 最终的结果按照dept_no升序排序, dept_no一样的再按照title升序排序;

库表来源: 部门表departments(dept_no部门编号、dept_name部门名称)、部门员工关系表dept_emp(emp_no员工编号、dept_no部门编号、入职离职日期)、职称表titles(emp_no、title职称类型、入职离职日期)

连接关系:join

筛选条件 无

聚合依据: group by deps.dept_no,deps.dept_name,tis.title

梳理思路:1、首先连接部门表departments、部门员工关系表dept_emp、职称表titles三张表格；2、步骤1作为步骤2的子查询,以部门编号和部门名称为聚合依据, count(title)合计函数计算出每一种职称类型的人数;3、使用order by,结果按照dept_no升序排序, dept_no一样的再按title升序排序:order by dept_no asc,title asc

分段编辑:

#1、首先连接部门表departments、部门员工关系表dept_emp、职称表titles三张表格

```
select deps.dept_no
,deps.dept_name
,tis.title
from dept_emp demp join departments deps on
demp.dept_no=deps.dept_no
join titles tis on
demp.emp_no=tis.emp_no
```

#2、步骤1作为步骤2的子查询,以部门编号和部门名称为聚合依据, count(title)合计函数计算出每一种职称类型的人数

```
select demp_ts.dept_no
,demp_ts.dept_name
,demp_ts.title
,count(demp_ts.title)
from
(
    子查询1
) demp_ts
group by 1,2,3
```

#3、使用order by,结果按照dept_no升序排序, dept_no一样的再按title升序排序:order by dept_no asc,title asc

组合代码

```
select demp_ts.dept_no
,demp_ts.dept_name
,demp_ts.title
,count(demp_ts.title)
from
(
    select deps.dept_no
,deps.dept_name
,tis.title
from dept_emp demp join departments deps on
demp.dept_no=deps.dept_no
join titles tis on
demp.emp_no=tis.emp_no
) demp_ts
group by 1,2,3
order by dept_no asc,title asc
```

SQL220 汇总各个部门当前员工的title类型的分配数目

题目 题解(260) 讨论(608) 排行

困难 通过率: 26.65% 时间限制: 1秒 空间限制: 32M

描述

有一个部门表departments简况如下:

| dept_no | dept_name |
|---------|-----------|
| d001 | Marketing |
| d002 | Finance |

有一个, 部门员工关系表dept_emp简况如下:

| emp_no | dept_no | from_date | to_date |
|--------|---------|------------|------------|
| 10001 | d001 | 1986-06-26 | 9999-01-01 |
| 10002 | d001 | 1996-08-03 | 9999-01-01 |
| 10003 | d002 | 1995-12-03 | 9999-01-01 |

有一个职称表titles简况如下:

| emp_no | title | from_date | to_date |
|--------|-----------------|------------|------------|
| 10001 | Senior Engineer | 1986-06-26 | 9999-01-01 |
| 10002 | Staff | 1996-08-03 | 9999-01-01 |
| 10003 | Senior Engineer | 1995-12-03 | 9999-01-01 |

SQL语句:

```
1 select d.emp_no
2 ,d.dept_name
3 ,d.title
4 ,count(d.title)
5 from
6 select deps.dept_no
7 ,deps.dept_name
8 ,tis.title
9 dept_emp demp join departments deps on
10 demp.dept_no = deps.dept_no
11 group by d.emp_no, d.dept_name, d.title
```

恭喜你通过本题

邀请朋友来挑战吧:

分享解题思路

练习模式 提示未通过的测试用例, 便于代码调试
您的代码已保存

通过全部用例 运行时间 52ms 占用内存 6396KB

自测运行 保存并提交

-SQL223(中等)

使用join查询方式找出没有分类的电影id以及名称

SQL223 使用join查询方式找出没有分类的电影id以及名称

收藏 分享 新闻

题目

题解(115)

讨论(315)

排行

面经

new

中等

通过率: 44.46%

时间限制: 1秒

空间限制: 32M

描述

现有电影信息表film，包含以下字段：

| 字段 | 说明 |
|-------------|--------|
| film_id | 电影id |
| title | 电影名称 |
| description | 电影描述信息 |

有类别表category，包含以下字段：

| 字段 | 说明 |
|-------------|------------|
| category_id | 电影分类id |
| name | 电影分类名称 |
| last_update | 电影分类最后更新时间 |

电影分类表film_category，包含以下字段：

| 字段 | 说明 |
|-------------|----------------------|
| film_id | 电影id |
| category_id | 电影分类id |
| last_update | 电影id和分类id对应关系的最后更新时间 |

思考思路

目标字段：没有分类的电影id以及对应的电影名称

库表来源：电影信息表film(字段、说明)、类别表category(字段、字段的对应含义说明)、电影分类表film_category(电影相关字段、说明)

连接关系

筛选条件

聚合依据

梳理思路：1、将电影信息表film和电影分类表film_category进行left join左连接；2、根据题目条件，需要查找出没有分类的电影id和其对应的电影名称添加筛选条件:where category_id is null

分段编辑：

#1、将电影信息表film和电影分类表film_category进行left join左连接

```
select fi.film_id  
,fi.title  
from film fi left join film_category fc on  
fi.film_id=fc.film_id
```

#2、根据题目条件，需要查找出没有分类的电影id和其对应的电影名称添加筛选条件:where category_id is null

```
where fc.category_id is null
```

组合代码

```
#组合以上步骤的代码  
select fi.film_id  
,fi.title  
from film fi left join film_category fc on  
fi.film_id=fc.film_id  
where fc.category_id is null
```

The screenshot shows a SQL competition interface for 'SQL223'. The user has successfully solved the question, which involved finding movies without categories. The solved code is:

```
1 select fi.film_id  
2 ,fi.title  
3 from film fi left join film_category fc on  
4 fi.film_id=fc.film_id  
5 where fc.category_id is null
```

A yellow trophy icon with a checkmark is displayed, indicating success. The message '恭喜你通过本题' (Congratulations, you have passed this question) is shown. Below the trophy, there are sharing options for WeChat, QQ, and other platforms, along with buttons for 'Share Solution思路' (Share Solution) and 'Enter Next Question'.

-SQL224(中等)

使用子查询的方式找出属于Action分类的所有电影对应的title,description

SQL224 使用子查询的方式找出属于Action分类的所有电影...

new

题目

题解(140)

讨论(372)

排行

面经

中等 通过率: 51.89% 时间限制: 1秒 空间限制: 256M

描述

film表

| 字段 | 说明 |
|-------------|--------|
| film_id | 电影id |
| title | 电影名称 |
| description | 电影描述信息 |

```
CREATE TABLE IF NOT EXISTS film (
    film_id smallint(5) NOT NULL DEFAULT '0',
    title varchar(255) NOT NULL,
    description text,
    PRIMARY KEY (film_id);
```

category表

| 字段 | 说明 |
|-------------|------------|
| category_id | 电影分类id |
| name | 电影分类名称 |
| last_update | 电影分类最后更新时间 |

思考思路

目标字段：属于Action分类的所有电影对应的title, description

库表来源：电影film表(film_id电影id、title电影名称、description电影描述信息)、category表(category_id 电影分类id、name电影分类名称、last_update电影分类最后更新时间)、film_category表(film_id电影id、category_id电影分类id、last_update电影id和分类id对应关系的最后更新时间)

连接关系

筛选条件

聚合依据

梳理思路:1、将category和film_category通过连接键category_id连接两张表格;2、通过添加筛选条件where name='Action'先在category表中查出该分类下的电影id;3、通过子查询，根据查出的电影id从film表中查出对应电影的名称和电影描述信息

分段编辑:

```
#1、将category和film_category通过连接键category_id连接两张表格;  
from category ca join film_category fica  
on ca.category_id=fica.category_id
```

```
#2、通过添加筛选条件where name='Action'先在category表中查出该分类下的电影id;  
select fica.film_id  
from category ca join film_category fica  
on ca.category_id=fica.category_id  
where ca.name='Action'
```

```
#3、通过子查询，根据查出的电影id从film表中查出对应电影的名称和电影描述信息
```

```
select film.title  
,film.description  
from film  
where film.film_id in  
(  
    select fica.film_id  
    from category ca join film_category fica  
    on ca.category_id=fica.category_id  
    where ca.name='Action'  
)
```

组合代码

```
select film.title  
,film.description  
from film  
where film.film_id in  
(  
    select fica.film_id  
    from category ca join film_category fica  
    on ca.category_id=fica.category_id  
    where ca.name='Action'  
)
```

SQL224 使用子查询的方式找出属于Action分类的电影

通过率: 51.69% 时间限制: 1秒 空间限制: 256M

描述

film表

| 字段 | 说明 |
|-------------|--------|
| film_id | 电影id |
| title | 电影名称 |
| description | 电影描述信息 |

category表

| 字段 | 说明 |
|-------------|------------|
| category_id | 电影分类id |
| name | 电影分类名称 |
| last_update | 电影分类最后更新时间 |

```
1 select film.title
2 ,film.description
3 from film
4 where film.film_id in
5 (
6 select fca.film_id
7 from film_category fca join film_category fica
8 on fca.category_id=fica.category_id
9 where fica.name='Action'
10 )
```

恭喜你通过本题

邀请朋友来挑战吧:

分享解题思路 进入下一题 ctrl +

练习模式 提示未通过的测试用例，便于代码调试
您的代码已保存

通过全部用例 (运行时间 45ms) 占用内存 6456KB

自测运行 保存并提交

-SQL226(简单)

将employees表的所有员工的last_name和first_name拼接起来作为Name

思路及代码

```
select concat(em.last_name, ' ', em.first_name) name
#concat函数可以用于字符串拼接，根据题目要求first_name和last_name之间需要加上空格 ' '
from employees em
```

-SQL227(中等)

创建一个actor表，包含如下列信息

SQL227 创建一个actor表，包含如下列信息

题目 题解(69) 讨论(161) 排行

中等 通过率: 32.44% 时间限制: 1秒 空间限制: 32M

描述

创建一个actor表，包含如下列信息

| 列表 | 类型 | 是否为NULL | 含义 |
|-------------|-------------|----------|--------------------|
| actor_id | smallint(5) | not null | 主键id |
| first_name | varchar(45) | not null | 名字 |
| last_name | varchar(45) | not null | 姓氏 |
| last_update | timestamp | not null | 最后更新时间， 默认是系统的当前时间 |

思路及代码

```
create table if not exists actor(
    actor_id smallint(5) not null primary key comment'主键id'
    ,first_name varchar(45) not null comment'名字'
    ,last_name varchar(45) not null comment'姓氏'
    ,last_update timestamp not null default (datetime('now','localtime'))
    comment'日期'
)
##在创建基本表的sql语句时注意
#设置主键primary key
#设置默认时间default (datetime('now','localtime'))
```

-SQL228(简单)

批量插入数据

SQL228 批量插入数据

收藏 分享 新建

new

题目

题解(69)

讨论(133)

排行

面经

简单 通过率: 47.30% 时间限制: 1秒 空间限制: 32M

描述

对于表actor批量插入如下数据

```
CREATE TABLE IF NOT EXISTS actor (
    actor_id smallint(5) NOT NULL PRIMARY KEY,
    first_name varchar(45) NOT NULL,
    last_name varchar(45) NOT NULL,
    last_update timestamp NOT NULL DEFAULT (datetime('now','localtime'))
```

| actor_id | first_name | last_name | last_update |
|----------|------------|-----------|---------------------|
| 1 | PENELOPE | GUINNESS | 2006-02-15 12:34:33 |
| 2 | NICK | WAHLBERG | 2006-02-15 12:34:33 |

示例1

```
输入: drop table if exists actor;
      CREATE TABLE actor (
          actor_id  smallint(5)  NOT NULL PRIMARY KEY,
          first_name  varchar(45)  NOT NULL,
          last_name  varchar(45)  NOT NULL,
          last_update  DATETIME NOT NULL)
```

复制

```
输出: 1|PENELOPE|GUINNESS|2006-02-15 12:34:33
      2|NICK|WAHLBERG|2006-02-15 12:34:33
```

复制

思路及代码

```
insert into actor(actor_id,first_name,last_name,last_update)
values(1,'PENELOPE','GUINNESS','2006-02-15 12:34:33'),(2,'NICK','WAHLBERG','2006-02-15 12:34:33')
```

-SQL229(中等)

批量插入数据，不使用replace操作

题目

题解(71)

讨论(141)

排行

中等

通过率: 45.04%

时间限制: 1秒

空间限制: 32M

描述

题目已经先执行了如下语句:

```
drop table if exists actor;
CREATE TABLE actor (
    actor_id smallint(5) NOT NULL PRIMARY KEY,
    first_name varchar(45) NOT NULL,
    last_name varchar(45) NOT NULL,
    last_update DATETIME NOT NULL);
insert into actor values ('3', 'WD', 'GUINNESS', '2006-02-15 12:34:33');
```

对于表actor插入如下数据,如果数据已经存在,请忽略(不支持使用replace操作)

| actor_id | first_name | last_name | last_update |
|----------|------------|-----------|-----------------------|
| '3' | 'ED' | 'CHASE' | '2006-02-15 12:34:33' |

示例1

输入: drop table if exists actor;

```
CREATE TABLE actor (
    actor_id smallint(5) NOT NULL PRIMARY KEY,
    first_name varchar(45) NOT NULL,
    last_name varchar(45) NOT NULL,
    last_update DATETIME NOT NULL);
insert into actor values ('3', 'WD', 'GUINNESS',
    '2006-02-15 12:34:33');
```

复制

输出: 3|WD|GUINNESS|2006-02-15 12:34:33

复制

对于表actor插入如下数据,如果数据已经存在,请忽略(不支持使用replace操作)

```
#忽略
insert ignore into actor
values(3, 'ED', 'CHASE', '2006-02-15 12:34:33')
```

-SQL230(中等)

创建一个actor_name表

SQL230 创建一个actor_name表



题目

题解(85)

讨论(164)

排行

中等

通过率: 47.16%

时间限制: 1秒

空间限制: 32M

描述

对于如下表actor，其对应的数据为：

| actor_id | first_name | last_name | last_update |
|----------|------------|-----------|---------------------|
| 1 | PENELOPE | GUINNESS | 2006-02-15 12:34:33 |
| 2 | NICK | WAHLBERG | 2006-02-15 12:34:33 |

请你创建一个actor_name表，并且将actor表中的所有first_name以及last_name导入该表。

actor_name表结构如下，题目最后会查询actor_name表里面的数据来对比结果输出：

| 列表 | 类型 | 是否为NULL | 含义 |
|------------|-------------|----------|----|
| first_name | varchar(45) | not null | 名字 |
| last_name | varchar(45) | not null | 姓氏 |

思路及代码

```
#创建一个actor_name表，并将actor表中所有first_name以及last_name导入actor_name表中
#创建一个actor_name表
drop table if exists actor_name;
create table actor_name(
    first_name varchar(45) not null comment'名字',
    last_name  varchar(45) not null comment'姓氏'
);
#插入/导入到新创建的actor_name表格
insert into
    actor_name
#从actor表中选择字段first_name和last_name导入到actor_name
select
```

```
first_name,  
last_name  
from  
actor;
```

-SQL231(中等)

对first_name创建唯一索引uniq_idx_firstname

思路及代码

```
#1. create (unique) index 索引名 on 表名(列名)  
create unique index uniq_idx_firstname on actor(first_name);  
create index idx_lastname on actor(last_name);  
  
#2. alter table 表名 add (unique) index 索引名(列名)  
alter table actor add unique index uniq_idx_firstname(first_name);  
alter table actor add index idx_lastname(last_name);
```

-SQL236(简单)

删除emp_no重复的记录，只保留最小的id对应的记录。

SQL236 删除emp_no重复的记录，只保留最小的id... ⚡ ☆ ↗ ⓘ

题目

题解(93)

讨论(287)

排行

简单 通过率: 36.99% 时间限制: 1秒 空间限制: 32M

描述

删除emp_no重复的记录，只保留最小的id对应的记录。

```
CREATE TABLE IF NOT EXISTS titles_test (
```

```
    id int(11) not null primary key,  
    emp_no int(11) NOT NULL,  
    title varchar(50) NOT NULL,  
    from_date date NOT NULL,  
    to_date date DEFAULT NULL);
```

```
insert into titles_test values ('1', '10001', 'Senior Engineer', '1986-06-26', '9999-01-01'),  
('2', '10002', 'Staff', '1996-08-03', '9999-01-01'),  
('3', '10003', 'Senior Engineer', '1995-12-03', '9999-01-01'),  
('4', '10004', 'Senior Engineer', '1995-12-03', '9999-01-01'),  
('5', '10001', 'Senior Engineer', '1986-06-26', '9999-01-01'),  
('6', '10002', 'Staff', '1996-08-03', '9999-01-01'),  
('7', '10003', 'Senior Engineer', '1995-12-03', '9999-01-01');
```

删除后titles_test表为(注: 最后会select * from titles_test表来对比结果)

| id | emp_no | title | from_date | to_ |
|----|--------|-----------------|------------|------|
| 1 | 10001 | Senior Engineer | 1986-06-26 | 9999 |
| 2 | 10002 | Staff | 1996-08-03 | 9999 |
| 3 | 10003 | Senior Engineer | 1995-12-03 | 9999 |

思路及代码

思路:

- 先查询出每组的最小值，但并不能直接在not in中直接使用，所有将其构建成一个临时表
- 根据临时表中的数据删除不需要的数据

```
#代码
#1、
with t as (
    select min(id) min_id
    from titles_test
    group by emp_no
)
#2、
delete from titles_test
where id not in (
    select min_id
    from t
)
```

-SQL237(简单)

将所有to_date为9999-01-01的全部更新为NULL

SQL237 将所有to_date为9999-01-01的全部更新为... ⚡ ☆ ↗ ⓘ

new

题目

题解(41)

讨论(127)

排行

面经

简单 通过率: 53.69% 时间限制: 1秒 空间限制: 32M

描述

将所有to_date为9999-01-01的全部更新为NULL,且 from_date更新为2001-01-01。

```
CREATE TABLE IF NOT EXISTS titles_test (
    id int(11) not null primary key,
    emp_no int(11) NOT NULL,
    title varchar(50) NOT NULL,
    from_date date NOT NULL,
    to_date date DEFAULT NULL
);

insert into titles_test values ('1', '10001', 'Senior Engineer', '1986-06-26', '9999-01-01'),
('2', '10002', 'Staff', '1996-08-03', '9999-01-01'),
('3', '10003', 'Senior Engineer', '1995-12-03', '9999-01-01'),
('4', '10004', 'Senior Engineer', '1995-12-03', '9999-01-01'),
('5', '10001', 'Senior Engineer', '1986-06-26', '9999-01-01'),
('6', '10002', 'Staff', '1996-08-03', '9999-01-01'),
('7', '10003', 'Senior Engineer', '1995-12-03', '9999-01-01');
```

更新后的值:

titles_test 表的值:

| id | emp_no | title | from_date | to_date |
|----|--------|-----------------|------------|---------|
| 1 | 10001 | Senior Engineer | 2001-01-01 | NULL |
| 2 | 10002 | Staff | 2001-01-01 | NULL |

```
update titles_test set to_date=null,from_date='2001-01-01'
where to_date='9999-01-01'
```

-SQL238(简单)

将id=5以及emp_no=10001的行数据替换成id=5以及emp_no=10005

```
REPLACE INTO titles_test VALUES(5,10005,...);
```

#另一种写法

```
#UPDATE titles_test SET emp_no = REPLACE(emp_no,10001,10005) WHERE id = 5;
```

#牛客漏洞：直接使用update也可以ac

```
UPDATE titles_test SET emp_no = 10005 WHERE id=5;
```

代码

```
#解法1
replace into titles_test
select 5,10005,title,from_date,to_date
from titles_test
where id=5;
#解法2
replace into titles_test values(5,1005,'Senior Engineer','1986-06-26','9999-01-01')
```

-SQL239(简单)

将titles_test表名修改为titles_2017

SQL239 将titles_test表名修改为titles_2017



题目

题解(31)

讨论(106)

排行

简单

通过率: 64.58%

时间限制: 1秒

空间限制: 32M

描述

将titles_test表名修改为titles_2017。

```
CREATE TABLE IF NOT EXISTS titles_test (
    id int(11) not null primary key,
    emp_no int(11) NOT NULL,
    title varchar(50) NOT NULL,
    from_date date NOT NULL,
    to_date date DEFAULT NULL);
```

```
insert into titles_test values ('1', '10001', 'Senior Engineer', '1986-06-26', '9999-01-01'),
('2', '10002', 'Staff', '1996-08-03', '9999-01-01'),
('3', '10003', 'Senior Engineer', '1995-12-03', '9999-01-01'),
('4', '10004', 'Senior Engineer', '1995-12-03', '9999-01-01'),
('5', '10001', 'Senior Engineer', '1986-06-26', '9999-01-01'),
('6', '10002', 'Staff', '1996-08-03', '9999-01-01'),
('7', '10003', 'Senior Engineer', '1995-12-03', '9999-01-01');
```

示例1

```
输入: drop table if exists titles_test;
      drop table if exists titles_2017;
      CREATE TABLE titles_test (
          id int(11) not null primary key,
          emp_no  int(11) NOT NULL,
          title   varchar(50) NOT NULL..
```

复制

题目是sqlite3，必须要加to

```
1 | alter table titles_test rename to titles_2017
```

如果是mysql，不用加to

```
1 | alter table titles_test rename titles_2017
```

代码

```
alter table titles_test rename titles_2017
```

-SQL240(中等)

在audit表上创建外键约束，其emp_no对应employees_test表的主键id

SQL240 在audit表上创建外键约束，其emp_no对...

new

题目

题解(45)

讨论(93)

排行

面经

中等

通过率: 38.82%

时间限制: 1秒

空间限制: 32M

描述

在audit表上创建外键约束，其emp_no对应employees_test表的主键id。
(以下2个表已经创建了)

```
CREATE TABLE employees_test(
    ID INT PRIMARY KEY NOT NULL,
    NAME TEXT NOT NULL,
    AGE INT NOT NULL,
    ADDRESS CHAR(50),
    SALARY REAL
);

CREATE TABLE audit(
    EMP_no INT NOT NULL,
    create_date datetime NOT NULL
);
```

后台会判断是否创建外键约束，创建输出1，没创建输出0

创建外键语句结构：

ALTER TABLE <表名>

ADD CONSTRAINT FOREIGN KEY (<列名>)

REFERENCES <关联表> (<关联列>)

代码

```
ALTER TABLE audit
ADD CONSTRAINT FOREIGN KEY (emp_no)
REFERENCES employees_test(id);
```

-SQL242(中等)

将所有获取奖金的员工当前的薪水增加10%

SQL242 将所有获取奖金的员工当前的薪水增加10% new

[题目](#)[题解\(60\)](#)[讨论\(177\)](#)[排行](#)[面经](#)

中等

通过率: 40.65%

时间限制: 1秒

空间限制: 32M

描述

现有员工获取到的奖金简表emp_bonus如下:

emp_no指获取到奖金的员工编号;

btype指获取到的奖金类型。

| emp_no | btype |
|--------|-------|
| 10001 | 1 |

有员工薪资简表salaries如下:

emp_no指员工编号;

salary指薪资;

from_date指该薪资的开始日期;

to_date指该薪资的结束日期。

| emp_no | salary | from_date | to_date |
|--------|---------|------------|------------|
| 10001 | 85097.0 | 2001-06-22 | 2002-06-22 |
| 10001 | 88958.0 | 2002-06-22 | 9999-01-01 |

请你写出更新语句，将所有获取奖金的员工当前的(salaries.to_date='9999-01-01')薪水增加10%。 (emp_bonus里面的emp_no都是当前获奖的所有员工，不考虑获取的奖金的类型)。

以上示例更新后的结果salaries为:

| emp_no | salary | from_date | to_date |
|--------|---------|------------|------------|
| 10001 | 85097.0 | 2001-06-22 | 2002-06-22 |
| 10001 | 97853.8 | 2002-06-22 | 9999-01-01 |

梳理思路

1、先从获取到奖金的员工表中查出emp_no

2、更新表格中的salary字段:

```
update salaries
```

```
set salary = salary*1.1
```

代码

```
#1、先从获取到奖金的员工表中查出emp_no
select emp_no
from emp_bonus

#2、更新表格中的salary字段
update salaries
set salary = salary*1.1
where salaries.emp_no in
(
    select emp_no
    from emp_bonus
) and to_date='9999-01-01'
```

-SQL244(中等)

将employees表中的所有员工的last_name和first_name通过引号连接起来。

SQL244 将employees表中的所有员工的last_name和first_name通过(')连接起来。

new

题目

题解(31)

讨论(142)

排行

面经

中等

通过率: 50.60%

时间限制: 1秒

空间限制: 32M

描述

将employees表中的所有员工的last_name和first_name通过(')连接起来。

```
CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,
`hire_date` date NOT NULL,
PRIMARY KEY (`emp_no`);
```

输出格式:

| name |
|-------------------|
| Facello'Georgi |
| Simmel'Bezalel |
| Bamford'Parto |
| Koblick'Chirstian |
| Maliniak'Kyoichi |
| Preusig'Anneke |
| Zielinski'Tzvetan |
| Kalloufi'Saniya |

代码

#concat函数,字符串拼接函数

```
select concat(em.last_name,' ',em.first_name)
from employees em
```

-SQL247(中等)

按照dept_no进行汇总

SQL247 按照dept_no进行汇总

☒ ☆ ☒ ⓘ

题目

题解(52)

讨论(128)

排行

中等 通过率: 49.90% 时间限制: 1秒 空间限制: 32M

描述

按照dept_no进行汇总，属于同一个部门的emp_no按照逗号进行连接，结果给出dept_no以及连接出的结果employees

```
CREATE TABLE `dept_emp` (
`emp_no` int(11) NOT NULL,
`dept_no` char(4) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`dept_no`);
```

输出格式:

| dept_no | employees |
|---------|-------------------|
| d001 | 10001,10002 |
| d002 | 10006 |
| d003 | 10005 |
| d004 | 10003,10004 |
| d005 | 10007,10008,10010 |
| d006 | 10009,10010 |

思路梳理

目标字段: dept_no;employees

库表来源: employees

连接关系: 无

筛选条件: 无

聚合依据: dept_no;

梳理思路:1、使用group_concat(emp_no) employees, group_concat的作用主要是将group by产生的同一个分组下的值都连接起来，返回一个字符串结果。2、对dept_no进行group by分组聚合，让group_concat发挥作用。

知识点：group_concat()函数;语法：group_concat([distinct] 要连接的字段[order by 排序字段 asc/desc] [separator '分隔符']); 通过使用distinct可以排除重复值；对结果中的值进行排序：使用order by子句；separator是一个字符串值，缺省为一个逗号。

分段编辑

组合代码

```
select dept_no,group_concat(emp_no) employees
from dept_emp
group by 1
```

如果只想展示一个或者两个角色，可以使用SUBSTRING_INDEX函数，SUBSTRING_INDEX(a,b,c)，a参数代表要截取的字符串，b参数代表分隔符，c参数代表截取几个字符

sql(截取两个):

```
SELECT
    user_no,
    SUBSTRING_INDEX(GROUP_CONCAT(DISTINCT role_name ORDER BY role_id desc ),',',2) AS role_name
FROM
    report_user_role_info
GROUP BY
    user_no;
```

-SQL248

查找排除在职(to_date = '9999-01-01')员工的最大、最小salary之后，其他的在职员工的平均工资
avg_salary

[题目](#)[题解\(125\)](#)[讨论\(348\)](#)[排行](#)[面经](#)

中等

通过率: 29.11%

时间限制: 1秒

空间限制: 32M

描述

查找排除在职(`to_date = '9999-01-01'`)员工的最大、最小`salary`之后，其他的在职员的平均工资`avg_salary`。

```
CREATE TABLE `salaries` ( `emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`from_date`));
```

如：

```
INSERT INTO salaries VALUES(10001,85097,'2001-06-22','2002-06-22');
INSERT INTO salaries VALUES(10001,88958,'2002-06-22','9999-01-01');
INSERT INTO salaries VALUES(10002,72527,'2001-08-02','9999-01-01');
INSERT INTO salaries VALUES(10003,43699,'2000-12-01','2001-12-01');
INSERT INTO salaries VALUES(10003,43311,'2001-12-01','9999-01-01');
INSERT INTO salaries VALUES(10004,70698,'2000-11-27','2001-11-27');
INSERT INTO salaries VALUES(10004,74057,'2001-11-27','9999-01-01');
```

输出格式：

| |
|------------|
| avg_salary |
| 73292 |

思路

目标字段：`avg_salary`:平均工资

库表来源: `salaries`

连接关系: 无

筛选条件 无

聚合依据 无

梳理思路：1、先分别查出表中目前在职工中的`max(salary)`最大、`min(salary)`最小`salary`； 2、步骤1作为步骤2的子查询作为筛选条件，`where salary not in(max(salary)) and salary not in (min(salary))`；
筛选条件2：`and to_date='9999-01-01'`

分段编辑：

```
#工资最大值
select max(salary)
from salaries
where to_date='9999-01-01'
#工资最小值
select min(salary)
from salaries
where to_date='9999-01-01'
```

组合代码

```
select avg(salary) avg_salary
from salaries
where salary not in
(
    select max(salary)
    from salaries
    where to_date='9999-01-01'
) and salary not in
(
    select min(salary)
    from salaries
    where to_date='9999-01-01'
)
and to_date='9999-01-01'
```

-SQL249

分页查询employees表，每5行一页，返回第2页的数据

SQL249 分页查询employees表，每5行一页，返回... ⌂ ☆ ↗ ⓘ

new

题目

题解(40)

讨论(134)

排行

面经

中等

通过率: 69.71%

时间限制: 1秒

空间限制: 32M

描述

分页查询employees表，每5行一页，返回第2页的数据

```
CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,
`hire_date` date NOT NULL,
PRIMARY KEY (`emp_no`));
```

示例1

思路

题目要求：返回第二页的数据，且每5行一页数据，题目翻译过来就是返回6-10共5行数据

1、通过limit 5, 5可以达到题目要求；从第6行开始输出一共输出5条数据即为第二页的数据

参考代码

```
select *
from employees
limit 5,5
```

-SQL251

使用含有关键字exists查找未分配具体部门的员工的所有信息。

题目

题解(49)

讨论(145)

排行

中等

通过率: 40.17%

时间限制: 1秒

空间限制: 32M

描述

使用含有关键字exists查找未分配具体部门的员工的所有信息。

```
CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,
`hire_date` date NOT NULL,
PRIMARY KEY (`emp_no`));
CREATE TABLE `dept_emp` (
`emp_no` int(11) NOT NULL,
`dept_no` char(4) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`, `dept_no`);
```

输出格式:

| emp_no | birth_date | first_name | last_name | ge |
|--------|------------|------------|-----------|----|
| 10011 | 1953-11-07 | Mary | Sluis | |

思路及代码

```
#思路1、使用not exists
select *
from employees em
where not exists
(
    select de.emp_no
    from dept_emp de
    where em.emp_no=de.emp_no
)
```

```
#思路2、使用not in
select *
from employees em
where em.emp_no not in
(
    select de.emp_no
    from dept_emp de
```

-SQL253

获取有奖金的员工相关信息。

SQL253 获取有奖金的员工相关信息。



new

题目

题解(140)

讨论(342)

排行

面经

较难

通过率: 30.30%

时间限制: 1秒

空间限制: 32M

描述

现有员工表employees如下:

| emp_no | birth_date | first_name | last_name | ge |
|--------|------------|------------|-----------|----|
| 10001 | 1953-09-02 | Georgi | Facello | |
| 10002 | 1964-06-02 | Bezalel | Simmel | |

有员工奖金表emp_bonus:

| emp_no | recevied | btype |
|--------|------------|-------|
| 10001 | 2010-01-01 | 1 |
| 10002 | 2010-10-01 | 2 |

有薪水表salaries:

| emp_no | salary | from_date | to_date |
|--------|--------|------------|------------|
| 10001 | 60117 | 1986-06-26 | 1987-06-26 |
| 10001 | 62102 | 1987-06-26 | 1988-06-25 |
| 10001 | 66074 | 1988-06-25 | 1989-06-25 |
| 10001 | 66596 | 1989-06-25 | 1990-06-25 |
| 10001 | 66961 | 1990-06-25 | 1991-06-25 |

思路

1、首先连接employees、emp_bonus、salaries三表；通过共同的连接键：

```
from employees ems join emp_bonus emb on  
ems.emp_no=emb.emp_no join salaries sa on  
emb.emp_no=sa.emp_no
```

2、根据题目要求：其中bonus类型btype为1其奖金为薪水salary的10%，btype为2其奖金为薪水的20%，其他类型均为薪水的30%。to_date='9999-01-01'表示当前薪水。本题核心逻辑代码用：

```
case...when...then...函数：,case when emb.btype=1 then salary*0.1 when emb.btype=2 then salary*0.2  
else salary*0.3 end
```

3、筛选条件为：where sa.to_date='9999-01-01'

组合代码

```
select ems.emp_no  
,ems.first_name  
,ems.last_name  
,emb.btype  
,sa.salary  
,case when emb.btype=1 then salary*0.1 when emb.btype=2 then salary*0.2 else  
salary*0.3  
end  
from employees ems join emp_bonus emb on  
ems.emp_no=emb.emp_no join salaries sa on  
emb.emp_no=sa.emp_no  
where sa.to_date='9999-01-01'
```

-SQL254

统计salary的累计和running_total

SQL254 统计salary的累计和running_total

收藏 分享 新建

题目

题解(94)

讨论(264)

排行

面经

new

较难 通过率: 41.01% 时间限制: 1秒 空间限制: 32M

描述

按照salary的累计和running_total, 其中running_total为前两个员工的salary累计和, 其他以此类推。具体结果如下Demo展示。。

```
CREATE TABLE `salaries` ( `emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`from_date`));
```

输出格式:

| emp_no | salary | running_total |
|--------|--------|---------------|
| 10001 | 88958 | 88958 |
| 10002 | 72527 | 161485 |
| 10003 | 43311 | 204796 |
| 10004 | 74057 | 278853 |
| 10005 | 94692 | 373545 |
| | | |

思路1

- 1、使用窗口函数, sum(sa.salary)over(order by emp_no asc rows between 2 preceding and current row) running_total; 计算当前员工和前两个员工的salary累计和
- 2、筛选条件: where sa.to_date='9999-01-01'

思路2

- 1、复用salary表, 用一个子查询代替running_total这个字段
- 2、子查询里筛选条件为where sin.emp_no <= sout.emp_no and sin.to_date='9999-01-01' and sout.to_date='9999-01-01'
(其中sin和sout为salary的两个表别名, 通过比较两张表的emp_no的大小来对salary求和, 并添加筛选条件to_date='9999-01-01')

组合代码

```
#思路1
select sa.emp_no
,sa.salary
,sum(sa.salary)over(order by emp_no asc rows between 2 preceding and current row)
running_total
from salaries sa
where sa.to_date='9999-01-01'

#思路2
select sout.emp_no ,sout.salary ,(select sum(sin.salary)  from salaries as sin
where sin.emp_no <= sout.emp_no and sin.to_date='9999-01-01' and
sout.to_date='9999-01-01') as running_total
from salaries as sout
where sout.to_date='9999-01-01'
```

-SQL255

给出employees表中排名为奇数行的first_name

SQL255 给出employees表中排名为奇数行的first_name

new

题目

题解(144)

讨论(463)

排行

面经

first_name

```
CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,
`hire_date` date NOT NULL,
PRIMARY KEY (`emp_no`);
```

如，输入为：

```
INSERT INTO employees VALUES(10001,'1953-09-02','Georgi','Facello','M','1986-06-26');
INSERT INTO employees VALUES(10002,'1964-06-02','Bezalel','Simmel','F','1985-11-21');
INSERT INTO employees VALUES(10005,'1955-01-21','Kyoichi','Maliniak','M','1989-09-12');
INSERT INTO employees VALUES(10006,'1953-04-20','Anneke','Preusig','F','1989-06-02');
```

输出格式：

| |
|--------|
| first |
| Georgi |
| Anneke |

请你在不打乱原序列顺序的情况下，输出：按first_name排序后，取奇数行的first_name。

如对以上示例数据的first_name排序后的序列为：Anneke、Bezalel、Georgi、Kyoichi。

则原序列中的Georgi排名为3，Anneke排名为1，所以按原序列顺序输出Georgi、Anneke。

思路

- 首先用排序窗口分析函数row_number()over(order by first_name) rk，给first_name排序
- 然后把步骤1当成一个子查询，给步骤1的表格一个别名，查出符合条件(排名为奇数的first_name)的全部first_name
- 根据题目要求，输出的first_name应该在不打乱原序列顺序的情况下，用where in (步骤2)，查出在原有的first_name顺序下的符合排名为奇数的first_name

4、如何判断first_name的排名是否为奇数；用where frk.rk % 2 !=0;其中rk为排名，frk为用了窗口分析函数的查询表格别名

组合代码

```
select first_name
from employees
where first_name
in
(
    select frk.first_name
    from
    (
        select *
        ,row_number()over(order by first_name) rk
        from employees
    ) frk
    where frk.rk % 2 !=0
)
```

-SQL256

出现三次以上相同积分的情况

题目

题解(56)

讨论(162)

排行

面经

new

简单 通过率: 53.00% 时间限制: 1秒 空间限制: 32M

描述

在牛客刷题的小伙伴们都有着牛客积分，积分(grade)表简化可以如下：

| id | number |
|----|--------|
| 1 | 111 |
| 2 | 333 |
| 3 | 111 |
| 4 | 111 |
| 5 | 333 |

id为用户主键id，number代表积分情况，让你写一个sql查询，积分表里面出现三次以及三次以上的积分，查询结果如下：

111

注意：若有多个符合条件的number，则按number升序排序输出。

梳理思路

本题需要有子查询

- 先对积分表grade的number进行计数count(number) cnt,以number为聚合依据进行分组聚合；
- 再对步骤1的查询进行嵌套，加上筛选条件where cnt>=3
- 根据题目意思，如果有多个符合条件的number，则按number升序排序输出;order by number asc

组合代码

```
select g_count.number
from
(
    select number, count(number) cnt
    from grade
    group by 1
) g_count
where g_count.cnt >= 3
order by number asc
```

-SQL257

刷题通过的题目排名

SQL257 刷题通过的题目排名

题目 题解(69) 讨论(166) 排行 面经 new

中等 通过率: 38.16% 时间限制: 1秒 空间限制: 32M

描述

在牛客刷题有一个通过题目个数的(passing_number)表，id是主键，简化如下：

| id | number |
|----|--------|
| 1 | 4 |
| 2 | 3 |
| 3 | 3 |
| 4 | 2 |
| 5 | 5 |
| 6 | 4 |

第1行表示id为1的用户通过了4个题目；
.....
第6行表示id为6的用户通过了4个题目；
请你根据上表，输出通过的题目的排名，通过题目个数相同的，排名相同，此时按照id升序排列，数据如下：

| id | number | t_rank |
|----|--------|--------|
| 5 | 5 | 1 |
| 1 | 4 | 2 |
| 6 | 4 | 2 |
| 2 | 3 | 3 |
| 3 | 3 | 3 |
| 4 | 2 | 4 |

id为5的用户通过了5个排名第1，
id为1和id为6的都通过了4个，并列第2

梳理思路

该题用到子查询和窗口函数

- 首先先使用窗口函数dense_rank()over()因为这个根据题目要求通过题目个数相同的，排名也相同；对牛客网刷题用户的通过题目个数number进行降序排名；
- 排序规则设置--通过题目个数相同的，排名相同，此时按照id升序排列：order by psr.number desc,id asc

组合代码

```
select psr.id
,psr.number
,psr.t_rank
from
(
    select id
    ,passing_number.number
    ,dense_rank()over(order by number desc) t_rank
    from passing_number
    group by 1,2
) psr
order by psr.number desc,id asc
```

-SQL258

找到每个人的任务

SQL258 找到每个人的任务

收藏 新

new

题目

题解(42)

讨论(137)

排行

面经

简单 通过率: 41.59% 时间限制: 1秒 空间限制: 32M

描述

有一个person表，主键是id，如下：

| id | name |
|----|------|
| 1 | fh |
| 2 | tm |

有一个任务(task)表如下，主键也是id，如下：

| id | person_id | content |
|----|-----------|----------------|
| 1 | 2 | tm1 works well |
| 2 | 2 | tm2 works well |

请你找到每个人的任务情况，并且输出出来，没有任务的也要输出，而且输出结果按照person的id升序排序，输出情况如下：

| id | name | content |
|----|------|----------------|
| 1 | fh | NULL |
| 2 | tm | tm1 works well |
| 2 | tm | tm2 works well |

梳理思路及代码

```
#使用左连接，因为根据题目要求，即使是为分配任务的人也要输出，且输出结果按照person的id升序排序
select p.id
,p.name
,t.content
from person p left join task t on
p.id=t.person_id
order by p.id asc
```

-SQL259

异常的邮件概率

写一个sql查询，每一个日期里，正常用户发送给正常用户邮件失败的概率是多少，结果保留到小数点后面3位(3位之后的四舍五入)，并且按照日期升序排序；

现在让你写一个sql查询，每一个日期里面，正常用户发送给正常用户邮件失败的概率是多少，结果保留到小数点后面3位(3位之后的四舍五入)，并且按照日期升序排序，上面例子查询结果如下：

| date | p |
|------------|-------|
| 2020-01-11 | 0.500 |
| 2020-01-12 | 0.000 |

结果表示：

2020-01-11失败的概率为0.500，因为email的第1条数据，发送的用户id为2是黑名单用户，所以不计入统计，正常用户发正常用户总共2次，但是失败了1次，所以概率是0.500；

2020-01-12没有失败的情况，所以概率为0.000。

(注意：sqlite 1/2得到的不是0.5，得到的是0，只有1*1.0/2才会得到0.5，sqlite四舍五入的函数为round)

思考思路

目标字段：date--邮件发送时间；p--正常用户之间邮件发送失败的概率

库表来源：Email表；User表

连接关系：无

筛选条件：where send_id not in (代表黑名单的用户) and receive_id not in (代表黑名单的用户)

聚合依据：date

梳理思路：1、首先先从user用户表中查出所有黑名单的用户id；2、然后步骤1作为步骤2的子查询，用not in对Email表格进行筛选，保证数据都是符合题目要求的：统计非黑名单(正常用户)发给正常用户邮件失败的概率是多少；3、在符合条件的数据下再以日期作为分组聚合依据，分别计算出每天都有多少个邮件发送记录的数量和发送的type为not_completed的邮件记录数量；4、最后，把步骤3作为步骤4的子查询，按照题目要求，查出需用于计算的字段，用round()函数计算异常邮件的概率；

分段编辑：

#1、首先先从user用户表中查出所有黑名单的用户id；

```
select user.id  
from user  
where is_blacklist=1
```

#2、然后步骤1作为步骤2的子查询，用not in对Email表格进行筛选，保证数据都是符合题目要求的：统计非黑名单(正常用户)发给正常用户邮件失败的概率是多少；3、在符合条件的数据下再以日期作为分组聚合依据，分别计算出每天都有多少个邮件发送记录的数量total_count和发送的type为not_completed的邮件记录数量fail_count；

```
select date  
,sum(case when type = 'no_completed' then 1 else 0 end) fail_count  
,count(date) total_count
```

```
from email
where email.send_id not in
(子查询1) and email.receive_id not in (子查询1) and type='no_completed'
group by 1
```

#4、最后，把步骤2&3作为步骤4的子查询，按照题目要求，查出需用于计算的字段，用round()函数计算异常邮件的概率；

```
select date
,round(fail_Count/total_count,3) p
from (子查询2&3) e_detail
```

#组合代码

```
select date
,round(fail_count/total_count,3) p
from
(
    select date
    ,sum(case when type = 'no_completed' then 1 else 0 end) fail_count
    ,count(date) total_count
    from email
    where email.send_id not in
    (
        select user.id
        from user
        where is_blacklist=1
    ) and email.receive_id not in
    (
        select user.id
        from user
        where is_blacklist=1
    )
    group by 1
) e_detail
```

组合代码

```
select date
,round(fail_count/total_count,3) p
from
(
    select date
    ,sum(case when type = 'no_completed' then 1 else 0 end) fail_count
    ,count(date) total_count
    from email
    where email.send_id not in
    (
        select user.id
        from user
        where is_blacklist=1
    ) and email.receive_id not in
    (
        select user.id
        from user
        where is_blacklist=1
    )
)
```

```
)  
group by 1  
) e_detail
```

-SQL260(简单)

牛客每个人最近的登录日期(一)

SQL260 牛客每个人最近的登录日期(一)

编辑 新星 热门 详细

new

题目

题解(44)

讨论(176)

排行

面经

简单 通过率: 44.50% 时间限制: 1秒 空间限制: 256M

描述

牛客每天有很多人登录，请你统计一下牛客每个用户最近登录是哪一天。

有一个登录(login)记录表，简况如下：

| id | user_id | client_id | date |
|----|---------|-----------|------------|
| 1 | 2 | 1 | 2020-10-12 |
| 2 | 3 | 2 | 2020-10-12 |
| 3 | 2 | 2 | 2020-10-13 |
| 4 | 3 | 2 | 2020-10-13 |

第1行表示user_id为2的用户在2020-10-12使用了客户端id为1的设备登录了牛客网

。。。

第4行表示user_id为3的用户在2020-10-13使用了客户端id为2的设备登录了牛客网

请你写出一个sql语句查询每个用户最近一天登录的日子，并且按照user_id升序排序，上面的例子查询结果如下：

| user_id | id |
|---------|------------|
| 2 | 2020-10-13 |
| 3 | 2020-10-13 |

查询结果表明：

user_id为2的最近的登录日期在2020-10-13

user_id为3的最近的登录日期也是2020-10-13

思考思路及代码

思路1：以user_id为聚合依据，并且按照user_id升序排序，根据题目要求和查询结果，使用max(date)计算最大日期；

思路2：使用窗口函数FIRST_VALUE(date)over(partition by user_id order by date desc)；以user_id为partition by以date降序为排序顺序；

```

#思路1
select user_id
,max(date) d
from login
group by user_id
order by user_id

#思路2
#用窗口函数FIRST_VALUE
select distinct user_id
,FIRST_VALUE(date) over(partition by user_id order by date desc) d
from login

```

-SQL261(较难)

牛客每个人最近的登录日期(二)

请你写出一个sql语句查询每个用户最近一天登录的日子，用户名，以及用户用的设备的名字，并且查询结果按照user的名字升序排序，上面的例子查询结果如下：

请你写出一个sql语句查询每个用户最近一天登录的日子，用户名，以及用户用的设备的名字，并且查询结果按照user的名字升序排序，上面的例子查询结果如下：

| u_n | c_n | date |
|----------|-----|------------|
| fh | ios | 2020-10-13 |
| wangchao | ios | 2020-10-13 |

查询结果表明：

fh最近的登录日期在2020-10-13，而且是使用ios登录的

wangchao最近的登录日期也是2020-10-13，而且是使用ios登录的

思考思路及代码

先根据sql260的思路，把sql260作为子查询和login、user、client三张表连接起来，先查出包含有用户登录最新的日期的登录记录表，然后根据日期去查询查找user name和client name

```

#1、先查出牛客用户最近的登录日期和对应的user_id
select user_id
,max(date) date
from login
group by 1

#2、步骤1的别名是result，分别以各自相同的连接键连接四张表；其中login和result通过date和user_id
两个连接键来连接；client和login则以client_id连接；user和login则以user_id连接
select u.name u_n,c.name c_n,l.date
from login l
join
(

```

```

select user_id
, max(date) date
from login
group by 1
) result
on (l.date=result.date and l.user_id=result.user_id)
join client c on l.client_id=c.id
join user u on l.user_id=u.id
order by u.name asc

```

-SQL262(较难)

牛客每个人最近的登录日期(三)

请你写出一个sql语句查询新登录用户次日成功的留存率，即第1天登陆之后，第2天再次登陆的概率,保存小数点后面3位(3位之后的四舍五入)，上面的例子查询结果如下：

SQL262 牛客每个人最近的登录日期(三)

new

题目

题解(185)

讨论(422)

排行

面经

取口11丁衣示user_id/11的用户在2020-10-14使用了客户端11/12的仅需求11牛客网

请你写出一个sql语句查询新登录用户次日成功的留存率，即第1天登陆之后，第2天再次登陆的概率,保存小数点后面3位(3位之后的四舍五入)，上面的例子查询结果如下：

| |
|-------|
| p |
| 0.500 |

查询结果表明：

user_id为1的用户在2020-10-12第一次新登录了，在2020-10-13又登录了，算是成功的留存
 user_id为2的用户在2020-10-12第一次新登录了，在2020-10-13又登录了，算是成功的留存
 user_id为3的用户在2020-10-12第一次新登录了，在2020-10-13没登录了，算是失败的留存
 user_id为4的用户在2020-10-13第一次新登录了，在2020-10-14没登录了，算是失败的留存
 故次日成功的留存率为 2/4=0.5

(sqlite里查找某一天的后一天的用法是:date('yyyy-mm-dd', '+1 day')，四舍五入的函数为round，sqlite 1/2得到的不是0.5，得到的是0，只有1*1.0/2才会得到0.5

mysql里查找某一天的后一天的用法是:DATE_ADD('yyyy-mm-dd', INTERVAL 1 DAY)，四舍五入的函数为round)

题目解释

查询出新登录用户（在当前日期之前没有过登录的记录即为新登录的用户）第二天成功登录的留存率
 （第一天登录之后，第二天再次登录的概率，保存小数点后面3位(3位之后的四舍五入)）；留存率=成功留存(新登录用户次日成功登录)的用户数量/新登录用户的总数量

思考思路及代码

- 1、分别计算出新登录用户的总数量和成功留存的用户的数量，最后再按照公式计算即可
- 2、首先计算新登录用户的总数量；思路：可以用去重后的登录用户计数count即可;count(distinct l1.user_id);
- 3、然后计算成功留存（新登录用户次日成功登录）的用户的数量；思路：新登录的用户次日还登录了，翻译过来-->即表示 (user_id,current_date+1) 还在login表格中，（其中user_id表示新登录用户id，current_date表示登录时间）
- 4、通过

```
count(distinct(l2.user_id))

from login l2
where (l2.user_id,l2.date) in
(
    select l3.user_id,date_add(min(l3.date),interval 1 day)
    from login l3
    group by 1
)
计算出新登录用户次日还登录的数量
```

```
#2、首先计算新登录用户的总数量；思路：可以用去重后的登录用户计数count即可
select count(distinct user_id)
from login

#3、然后计算成功留存（新登录用户次日成功登录）的用户的数量；思路：新登录的用户次日还登录了，翻译
过来-->即表示 (user_id,current_date+1) 还在login表格中，（其中user_id表示新登录用户id，
current_date表示登录时间）
from login l2
where (l2.user_id,l2.date) in
(
    select l3.user_id,date_add(min(l3.date),interval 1 day)
    from login l3
    group by 1
)

#根据留存率的公式计算
select round(count(distinct l2.user_id)*1.0/(select count(distinct l1.user_id)
from login l1),3)

#组合代码
select round(count(distinct l2.user_id)*1.0/(select count(distinct l1.user_id)
from login l1),3)
from login l2
where (l2.user_id,l2.date) in
(
    select l3.user_id,date_add(min(l3.date),interval 1 day)
    from login l3
    group by 1
)
```

-SQL263(较难)

牛客每个人最近的登录日期(四)

题目解释

写出一个sql语句查询每个日期登录的新用户个数，并且查询结果按照日期升序排序，例子的查询结果如下：

请你写出一个sql语句查询每个日期登录新用户个数，并且查询结果按照日期升序排序，上面的例子查询结果如下：

| date | new |
|------------|-----|
| 2020-10-12 | 3 |
| 2020-10-13 | 0 |
| 2020-10-14 | 1 |
| 2020-10-15 | 0 |

查询结果表明：

2020-10-12，有3个新用户(user_id为2, 3, 1)登录

2020-10-13，没有新用户登录

2020-10-14，有1个新用户(user_id为4)登录

2020-10-15，没有新用户登录

根据示例结果来看，需要以date为分组聚合的依据对新登录用户进行计数

思考思路及代码

需要计算出每个日期新登录用户的个数count

- 1、首先以user_id为聚合依据，min(date)函数计算出用户登录的最早日期(第一次登录的日期)；
- 2、然后从原表login中再查出在这个最早日期登录的user_id和在这个日期登录的用户的数量
count(user_id)
- 3、把原表login和步骤二得到的表用左连接根据date连接
- 4、在select字段中用ifnull(new,0)函数将字段值为null的字段转为0,如果不为null则查询输出new字段的内容

```
select l2.date
,ifnull(new,0)
from login l2 left join
(
    select l1.date
    ,count(user_id) new
```

```

from login l1
where (user_id,l1.date) in
(
    select user_id
    ,min(date)
    from login
    group by 1
    order by 1
)
group by 1
order by 1
) r_t on l2.date=r_t.date
group by 1
order by 1

```

-SQL264(困难)

牛客每个人最近的登录日期

请你写出一个sql语句查询每个日期新用户的次日留存率，结果保留小数点后面3位数(3位之后的四舍五入)，并且查询结果按照日期升序排序，上面的例子查询结果如下：

| date | p |
|------------|-------|
| 2020-10-12 | 0.667 |
| 2020-10-13 | 0.000 |
| 2020-10-14 | 1.000 |
| 2020-10-15 | 0.000 |

查询结果表明：

2020-10-12登录了3个(user_id为2, 3, 1)新用户，2020-10-13，只有2个(id为2,1)登录，故2020-10-12新用户次日留存率为2/3=0.667；

2020-10-13没有新用户登录，输出0.000；

2020-10-14登录了1个(user_id为4)新用户，2020-10-15，user_id为4的用户登录，故2020-10-14新用户次日留存率为1/1=1.000；

2020-10-15没有新用户登录，输出0.000；

(注意：sqlite里查找某一天的后一天的用法是:date('yyyy-mm-dd', '+1 day')，sqlite里1/2得到不是0.5，得到的是0，只有1*1.0/2才会得到0.5)



梳理思路

目标字段：用户登录日期date，新用户的次日留存率；

库表来源：login登录记录表；

连接关系

筛选条件

聚合依据

梳理思路:1、先理清留存率的计算公式：新用户次日登录的用户数量/当日新登录用户的数量，保留小数点后3位小数；2、分别思考如何计算分子和分母；3、先计算分母(当日新登录用户的数量)；4、再计算分子(新用户次日登录的用户数量)。

细分思路：

1、计算分母：①先以user_id为聚合依据筛选出在当前用户id的分组下，第一次登录的用户id和登录日期date；②where子查询筛选原表login中符合第一次登录(新用户)的date, count(user_id)，以date(登录日期为聚合依据)；③但是以上俩步只能筛选出有新用户登录的日期以及数量的记录而忽略了那些一天都无新用户登录的日期和数量，所以以原表login为主表和步骤②进行左连接left join操作，用ifnull(new_user,0)函数；

2、计算分子：①新登录用户次日还登录的用户的含义就是：**当日新登录用户的user_id和date**，user_id在login表中date+1:date_add(min(l3.date),interval 1 day)的位置还出现了**where(user_id,date) in.....;**②步骤①筛选出了新用户次日登录的用户user_id以及对应的次日date,**步骤②则从步骤①中计算出count(user_id)即为分子对应的数据**;③同理，此时也只筛选出了存在新用户第二天仍登录的数量忽略了0；

分段编辑：

```
#1、先计算分母(当日新登录用户的数量)
#①先以user_id为聚合依据筛选出在当前用户id的分组下，第一次登录的用户id和登录日期date:
select l1.user_id,min(l1.date)
from login l1
group by 1
order by 1
;
#②where子查询筛选原表login中符合第一次登录(新用户)的date, count(user_id)，以date(登录日期为聚合依据)
select l2.date,count(l2.user_id)
from login l2
where (l2.user_id,l2.date) in
(
    select l1.user_id,min(l1.date)
    from login l1
    group by 1
    order by 1
)
group by 1
order by 1
;
#③和原表左连接用ifnull把有新用户登录和无新用户登录的日期下登录的用户数量都查询出来
select l3.date,ifnull(rt.new_user,0) user_new
from login l3 left join
(
    select l2.date,count(l2.user_id) new_user
    from login l2
    where (l2.user_id,l2.date) in
    (
        select l1.user_id,min(l1.date)
```

```

        from login l1
        group by 1
        order by 1
    )
    group by 1
    order by 1
) rt on l3.date=rt.date
group by 1
order by 1
;

#2、再计算分子：新用户次日登录的用户数量
#@新登录用户次日还登录的用户的含义就是：当日新登录用户的user_id和date, user_id在login表中
date+1:date_add(min(l3.date),interval 1 day)的位置还出现了where(user_id,date)
in.....;
#@直接计算出新用户次日登录的数量count(user_id)
select l4.user_id,count(l4.user_id) sl_user
from login l4
where l4.user_id,l4.date in
(
    select l5.user_id,date_add(min(l5.date),interval 1 day)
    from login l5
    group by 1
    order by 1
)
group by 1
group by 1
;
select l4.date,count(l4.user_id) sl_user
from login l4
where l4.user_id,l4.date in
(
    select l5.user_id,date_add(min(l5.date),interval 1 day)
    from login l5
    group by 1
    order by 1
)
group by 1
order by 1
;
#同理，此时也只筛选出了存在新用户第二天仍登录的数量忽略了0，我们和原表进行左连接筛选出为null的
记录用ifnull函数查询出0或具体的sl_user数量;
select l6.date,ifnull(rt2.sl_user,0) user_sl
from login l6 left join
(
    select l4.date,count(l4.user_id) sl_user
    from login l4
    where (l4.user_id,l4.date) in
    (
        select l5.user_id,date_add(min(l5.date),interval 1 day)
        from login l5
        group by 1
        order by 1
    )
    group by 1
    order by 1
) rt2 on l6.date=rt2.date

```

```

group by 1
order by 1

#④因为以上步骤③查出的日期date是新用户第二天仍然登录的日期，所以应该减去一天的日期才是当日的新用户
在第二天还登录的数量
select date_sub(rt3.date,interval 1 day), rt3.user_s1
from
(
    select l6.date,ifnull(rt2.s1_user,0) user_s1
    from
    (
        select l4.date,count(l4.user_id) s1_user
        from login l4
        where (l4.user_id,l4.date) in
        (
            select l5.user_id,date_add(min(l5.date),interval 1 day)
            from login l5
            group by 1
            order by 1
        )
        group by 1
        order by 1
    ) rt2 right join
    login l6 on l6.date=rt2.date
    group by 1
    order by 1
) rt3

```

组合代码

```

select rt5.date2,round(rt4.s1_u/case when rt5.user_new=0 then 1 else rt5.user_new
end,3) p
from
(
    select date_sub(rt3.date,interval 1 day) date1, rt3.user_s1 s1_u
    from
    (
        select l6.date,ifnull(rt2.s1_user,0) user_s1
        from
        (
            select l4.date,count(l4.user_id) s1_user
            from login l4
            where (l4.user_id,l4.date) in
            (
                select l5.user_id,date_add(min(l5.date),interval 1 day)
                from login l5
                group by 1
                order by 1
            )
            group by 1
            order by 1
        ) rt2 right join
        login l6 on l6.date=rt2.date
        group by 1
        order by 1
    ) rt3
    group by 1
    order by 1
) rt4

```

```

        group by 1
        order by 1
    ) rt3
) rt4 join
(
    select l3.date date2,ifnull(rt.new_user,0) user_new
    from login l3 left join
    (
        select l2.date,count(l2.user_id) new_user
        from login l2
        where (l2.user_id,l2.date) in
        (
            select l1.user_id,min(l1.date)
            from login l1
            group by 1
            order by 1
        )
        group by 1
        order by 1
    ) rt on l3.date=rt.date
    group by 1
    order by 1
) rt5 on rt4.date1=rt5.date2

#####以上代码日期前后相差一日，故不符合题目要求#####
#####以下代码会从牛客每个人最近的登录日期(三)和牛客每个人最近的登录日期(四)俩题结合起来解决，这
三题是环环相扣、由内到外、知1推3的存在；
#牛客每个人最近的登录日期(三)--每个日期新用户次日还登录的人的个数
select login.date,ifnull(n1.new_num,0) as second_login_num
from login
left join
(
    select l1.date
    ,count(distinct l1.user_id) as new_num
    from login l1
    join login l2 on l1.user_id=l2.user_id and l2.date=date(l1.date,'+1 day')
    where l1.date =
    (
        select min(date) from login where user_id=l1.user_id
    )
    group by l1.date
) n1
on login.date = n1.date
group by login.date

#牛客每个人最近的登录日期(四)--每个日期的新登录用户总数
select login.date,ifnull(n1.new_num,0)
from login
left join
(
    select l1.date
    ,count(distinct l1.user_id) as new_num
    from login l1
    where l1.date =
    (
        select min(date)
    )
    group by l1.date
) n1
on login.date = n1.date
group by login.date

```

```

        from login
        where user_id=l1.user_id
    )
    group by l1.date
) n1
on login.date = n1.date
group by login.date

#牛客每个人最近的登录日期(五)--求每个日期新用户的留存率
select second_login.date, round(ifnull(second_login.second_login_num *1.0/
first_login.first_num,0),3)
from
(
    select login.date,ifnull(n1.new_num,0) as second_login_num
    from login
    left join
    (
        select l1.date,count(distinct l1.user_id) as new_num
        from login l1
        join login l2 on l1.user_id=l2.user_id and
l2.date=date_add((l1.date),interval 1 day)
        where l1.date =
        (
            select min(date)
            from login
            where user_id=l1.user_id
        )
    )#这一步where子查询是在连接完俩张表格之后进行筛选的,让表中l1.date都转换成首次登录的日期;
        #这样日期后面跟着的次日新用户登录数量就不是次日而是当日
        group by l1.date
) n1
on login.date = n1.date
group by login.date
) second_login

join
(
    select login.date,ifnull(n1.new_num,0) as first_num
    from login
    left join
    (
        select l1.date,count(distinct l1.user_id) as new_num
        from login l1
        where l1.date =
        (
            select min(date)
            from login
            where user_id=l1.user_id
        )
        group by l1.date
    ) n1
    on login.date = n1.date
    group by login.date
) first_login

```

```
on second_login.date=first_login.date
```

-SQL265(较难)

SQL265 牛客每个人最近的登录日期(六)

编辑 收藏 分享 报错

new

题目

题解(103)

讨论(229)

排行

面经

较难

通过率: 30.65%

时间限制: 1秒

空间限制: 256M

描述

牛客每天有很多人登录，请你统计一下牛客每个用户刷题情况，包括: 用户的名字，以及截止到某天，累计总共通过了多少题。不存在没有登录却刷题的情况，但存在登录了没刷题的情况，不会存在刷题表里面，会存在提交代码没有通过的情况并记录在刷题表里，通过数目是0。

有登录(login)记录表，简况如下:

| id | user_id | client_id | date |
|----|---------|-----------|------------|
| 1 | 2 | 1 | 2020-10-12 |
| 2 | 3 | 2 | 2020-10-12 |
| 3 | 1 | 2 | 2020-10-12 |
| 4 | 1 | 3 | 2020-10-13 |
| 5 | 3 | 2 | 2020-10-13 |

第1行表示user_id为2的用户在2020-10-12使用了客户端id为1的设备登录了牛客网

.....

第5行表示user_id为3的用户在2020-10-13使用了客户端id为2的设备登录了牛客网

请你写出一个sql语句查询刷题信息，包括：用户名，以及截止到某天，累计总共通过了多少题，并且查询结果先按照日期升序排序，再按照姓名升序排序，有登录却没有刷题的哪一天的数据不需要输出，上面的例子查询结果如下：

| u_n | date | ps_num |
|----------|------------|--------|
| fh | 2020-10-12 | 4 |
| wangchao | 2020-10-12 | 1 |
| tm | 2020-10-13 | 0 |
| wangchao | 2020-10-13 | 3 |

查询结果表明：

fh在2020-10-12为止，总共通过了4道题，输出为4

wangchao在2020-10-12为止，总共通过了1道题，总计为1

tm在2020-10-12为止只登陆了没有刷题，故没有显示出来

tm在2020-10-13为止刷了题，但是却没有通过任何题目，总计为0

wangchao在2020-10-13通过2道，但是加上前面2020-10-12通过1道，故在2020-10-13为止总共通过了3道题，总计为3



梳理思路

- 1、先连接login登录记录表和passing_number刷题表；连接键为user_id和date，筛选掉在login表中但是不在passing_number刷题表中的数据；
- 2、然后将user表和步骤1的连接后的表login_passing(表别名)连接起来，以user_id为连接键：on user.id=login_passing.user_id；
- 3、这样可以得到示例中的表数据，但是却不是该用户截至到当日所通过的题目的总计，所以使用我们的窗口函数：sum(login_passing.number)over(partition by user.name rows between unbounded preceding and current row)
以user.name为partition by的依据然后计算截至到当前行的刷题数量的总计；
- 4、

组合代码

```
select user.name
,login_passing.date
,sum(login_passing.number)over(partition by user.name rows between unbounded
preceding and current row)
from user join
(
    select pn.user_id
    ,pn.date
    ,pn.number
    from passing_number pn join login l1 on
    l1.date=pn.date and l1.user_id=pn.user_id
) login_passing on user.id=login_passing.user_id
order by 2,1
```

-SQL266(简单)

考试分数(一)

| | | |
|---|------|-------|
| 4 | Java | 12000 |
| 5 | Java | 13000 |
| 6 | JS | 12000 |
| 7 | JS | 11000 |
| 8 | JS | 9999 |

第1行表示用户id为1的用户选择了C++岗位并且考了11001分

....

第8行表示用户id为8的用户选择了JS岗位并且考了9999分

请你写一个sql语句查询各个岗位分数的平均数，并且按照分数降序排序，结果保留小数点后面3位(3位之后四舍五入):

| job | avg |
|------|-----------|
| Java | 12500.000 |
| JS | 10999.667 |
| C++ | 10000.333 |

(注意: sqlite 1/2得到的不是0.5, 得到的是0, 只有1*1.0/2才会得到0.5, sqlite四舍五入的函数为round)

写一个sql语句查询各个岗位分数的平均数，并且按照分数降序排序，结果保留小数点后面3位(3位之后四舍五入)

梳理思路及代码

```
select job
,round(avg(score),3)
from grade
group by 1#以job:岗位位聚合依据
order by 2 desc#以分数为排序依据, 按照分数降序排序
```

-SQL267(中等)

考试分数(二)

题目

题解(95)

讨论(252)

排行

面经

| | | |
|---|------|-------|
| 2 | C++ | 10000 |
| 3 | C++ | 9000 |
| 4 | Java | 12000 |
| 5 | Java | 13000 |
| 6 | JS | 12000 |
| 7 | JS | 11000 |
| 8 | JS | 9999 |
| 9 | Java | 12500 |

第1行表示用户id为1的用户选择了C++岗位并且考了11001分

。。。

第8行表示用户id为8的用户选择了前端岗位并且考了9999分

请你写一个sql语句查询用户分数大于其所在工作(job)分数的平均分的所有grade的属性，并且以id的升序排序，如下：

| id | job | score |
|----|------|-------|
| 1 | C++ | 11001 |
| 5 | Java | 13000 |
| 6 | JS | 12000 |
| 7 | JS | 11000 |

(注意: sqlite 1/2得到的不是0.5, 得到的是0, 只有 $1*1.0/2$ 才会得到0.5, sqlite四舍五入的函数为round)

写一个sql语句查询用户分数大于其所在工作(job)分数的平均分的所有grade的属性，并且以id的升序排序

梳理思路及代码

- 1、先查找计算出的avg(score) group by job每个岗位下的平均分数；
- 2、再把这个平均分数表和原表连接，连接键为job岗位，这样每个相同岗位的列都有一个avg平均分可以有助于下一步where筛选；
- 3、where筛选，根据题目要求，用户分数需要大于其所在工作(job)分数的平均分的所有grade的属性

4、以id的升序排序；

```
select g2.id
,g2.job
,g2.score
from grade g2
join
(
    select g1.job,avg(g1.score) avg_score
    from grade g1
    group by g1.job
) avg on g2.job=avg.job
where g2.score>avg.avg_score
order by g2.id
```

-SQL268(较难)

考试分数(三)

SQL268 考试分数(三)

困难 星级 分享 报错

new

题目

题解(118)

讨论(310)

排行

面经

较难

通过率: 23.90% 时间限制: 1秒 空间限制: 256M

描述

牛客每次举办企业笔试的时候，企业一般都会有不同的语言岗位，比如C++工程师，JAVA工程师，Python工程师，每个用户笔试完有不同的分数，现在有一个分数(grade)表简化如下：

| id | language_id | score |
|----|-------------|-------|
| 1 | 1 | 12000 |
| 2 | 1 | 13000 |
| 3 | 2 | 11000 |
| 4 | 2 | 10000 |
| 5 | 3 | 11000 |
| 6 | 1 | 11000 |
| 7 | 2 | 11000 |

第1行表示用户id为1的选择了language_id为1岗位的最后考试完的分数为12000，

....

第7行表示用户id为7的选择了language_id为2岗位的最后考试完的分数为11000，

不同的语言岗位(language)表简化如下：

| id | name |
|----|------|
| 1 | C++ |

SQL268 考试分数(三)

刷新 ⭐ ⌂ ⓘ

new

题目

题解(118)

讨论(310)

排行

面经

| id | name |
|----|--------|
| 1 | C++ |
| 2 | JAVA |
| 3 | Python |

请你找出每个岗位分数排名前2名的用户，得到的结果先按照language的name升序排序，再按照积分降序排序，最后按照grade的id升序排序，得到结果如下：

| id | name | score |
|----|--------|-------|
| 2 | C++ | 13000 |
| 1 | C++ | 12000 |
| 3 | JAVA | 11000 |
| 7 | JAVA | 11000 |
| 4 | JAVA | 10000 |
| 5 | Python | 11000 |

梳理思路及代码

- 1、先将grade表和language不同的语言岗位表以连接键id、language_id连接俩表；
- 2、并且给新连接的表开窗口分析函数dense_rank()over();
- 3、以步骤2为子查询，where为筛选条件，筛选出每个岗位分数排名前2名的用户；
- 4、得到的结果先按照language的name升序排序，再按照积分降序排序，最后按照grade的id升序排序；

组合代码

```
select rt.id  
,rt.name  
,rt.score  
from  
(  
    select g.id id
```

```
,l.name name
,g.score score
,dense_rank()over(partition by l.name order by score desc) rk
from grade g join language l on
g.language_id=l.id
) rt
where rt.rk<=2
order by rt.name,rt.score desc,rt.id
```

-SQL269(较难)

考试分数(四)

题目

题解(107)

讨论(246)

排行

面经

7

B

11000

8

B

9999

第1行表示用户id为1的用户选择了C++岗位并且考了11001分

....

第8行表示用户id为8的用户选择了B语言岗位并且考了9999分

请你写一个sql语句查询各个岗位分数升序排列之后的中位数位置的范围，并且按job升序排序，结果如下：

| job | start | end |
|------|-------|-----|
| B | 2 | 2 |
| C++ | 2 | 2 |
| Java | 1 | 2 |

解释：

第1行表示C++岗位的中位数位置范围为[2,2]，也就是2。因为C++岗位总共3个人，是奇数，所以中位数位置为2是正确的(即位置为2的10000是中位数)

第2行表示Java岗位的中位数位置范围为[1,2]。因为Java岗位总共2个人，是偶数，所以要知道中位数，需要知道2个位置的数字，而因为只有2个人，所以中位数位置为[1,2]是正确的(即需要知道位置为1的12000与位置为2的13000才能计算出中位数为12500)

第3行表示前端岗位的中位数位置范围为[2,2]，也就是2。因为B语言岗位总共3个人，是奇数，所以中位数位置为2是正确的(即位置为2的11000是中位数)

(注意：sqlite 1/2得到的不是0.5，得到的是0，只有 $1*1.0/2$ 才会得到0.5，sqlite四舍五入的函数为round，sqlite不支持floor函数，支持cast(x as integer) 函数，不支持ceil函数，支持case when ...then ...else ..end函数)



梳理思路及代码

- 1、使用向上ceil(),向下floor()取整函数;
- 2、以岗位job为group by的聚合依据;

```
select job
,floor((count(*)+1)/2) 'start'
,ceil((count(*)+1)/2) 'end'
from grade
group by job
order by job
```

-SQL270(困难)

考试分数(五)

SQL270 考试分数(五)



new

题目

题解(149)

讨论(395)

排行

面经

7

B

11000

8

B

9999

第1行表示用户id为1的用户选择了C++岗位并且考了11001分

...

第8行表示用户id为8的用户选择了B语言岗位并且考了9999分

请你写一个sql语句查询各个岗位分数的中位数位置上的所有grade信息，并且按id升序排序，结果如下：

| id | job | score | t_rank |
|----|------|-------|--------|
| 2 | C++ | 10000 | 2 |
| 4 | Java | 12000 | 2 |
| 5 | Java | 13000 | 1 |
| 7 | B | 11000 | 2 |

解释：

第1行表示C++岗位的中位数位置上的为用户id为2，分数为10000，在C++岗位里面排名是第2

第2、3行表示Java岗位的中位数位置上的为用户id为4,5，分数为12000,13000，在Java岗位里面排名是第2,1

第4行表示B语言岗位的中位数位置上的为用户id为7，分数为11000，在前端岗位里面排名是第2

(注意：sqlite 1/2得到的不是0.5，得到的是0，只有 $1*1.0/2$ 才会得到0.5，sqlite四舍五入的函数为round，sqlite不支持floor函数，支持cast(x as integer) 函数，不支持if函数，支持case when ...then ...else ..end函数，sqlite不支持自定义变量)

写一个sql语句查询各个岗位分数的中位数位置上的所有grade信息，并且按照id升序排序



梳理思路及代码

- 1、首先，按照题目例题截图结果可知，需要使用窗口函数对job岗位进行分组聚合然后内部根据score得分降序排序；
- 2、然后，把上一题SQL269(较难)这一题当作一个新表，别名(rt1)，跟步骤1的表进行内连接，连接键为on rt.job=rt1.job；
- 3、添加筛选条件where，让rk(窗口函数的排名)=start or rk(窗口函数的排名)=end；
- 4、order by rt.id,根据id升序排序；

组合代码

```
select rt.id
,rt.job
,rt.score
,rt.rk
from
(
    select id
    ,job
    ,score
    ,dense_rank()over(partition by job order by score desc) rk
    from grade
    order by id
) rt join
(
    select job
    ,floor((count(*)+1)/2) 'start'
    ,ceil((count(*)+1)/2) 'end'
    from grade
    group by job
    order by job
) rt1 on rt.job=rt1.job
where rt.rk=rt1.start or rt.rk=rt1.end
order by rt.id
```

-SQL271(简单)

牛客的课程订单分析(一)

| 题目 | 题解(46) | 讨论(107) | 排行 | 面经 | new |
|----|--------|---------|--------------|----|------------|
| 4 | 57 | C++ | completed | 3 | 2025-10-23 |
| 5 | 557336 | Java | completed | 1 | 2025-10-23 |
| 6 | 557336 | Python | no_completed | 1 | 2025-10-24 |

第1行表示user_id为557336的用户在2025-10-10的时候使用了client_id为1的客户端下了C++课程的订单，但是状态为没有购买成功。

第2行表示user_id为230173543的用户在2025-10-12的时候使用了client_id为2的客户端下了Python课程的订单，状态为购买成功。

...

最后1行表示user_id为557336的用户在2025-10-24的时候使用了client_id为1的客户端下了Python课程的订单，状态为没有购买成功。

请你写出一个sql语句查询在2025-10-15以后状态为购买成功的C++课程或者Java课程或者Python的订单，并且按照order_info的id升序排序，以上例子查询结果如下：

| id | user_id | product_name | status | client_id | date |
|----|---------|--------------|-----------|-----------|------------|
| 4 | 57 | C++ | completed | 3 | 2025-10-23 |
| 5 | 557336 | Java | completed | 1 | 2025-10-23 |

梳理思路及代码

核心逻辑：在2025-10-15以后状态为购买成功的C++课程或者Java课程或者Python的订单，并且按照order_info的id升序排序；对应的sql操作就是添加三个where的筛选条件和一个排序条件order by；

组合代码

```
select id
,user_id
,product_name
,status
,client_id
,order_info.date
from order_info
where (order_info.date>'2025-10-15') and (product_name='C++' or
product_name='Java' or product_name='Python') and status='completed'
order by 1
```

-SQL272(中等)

牛客的课程订单分析

SQL272 牛客的课程订单分析(二)

分享 收藏 新建

题目

题解(71)

讨论(155)

排行

面经

new

| | | | | |
|---|--------|------|-----------|---|
| 6 | 57 | Java | completed | 1 |
| 7 | 557336 | C++ | completed | 1 |

第1行表示user_id为557336的用户在2025-10-10的时候使用了client_id为1的客户端下了C++课程的订单，但是状态为没有购买成功。

第2行表示user_id为230173543的用户在2025-10-12的时候使用了client_id为2的客户端下了Python课程的订单，状态为购买成功。

...

最后1行表示user_id为557336的用户在2025-10-25的时候使用了client_id为1的客户端下了C++课程的订单，状态为购买成功。

请你写出一个sql语句查询在2025-10-15以后，同一个用户下单2个以及2个以上状态为购买成功的C++课程或Java课程或Python课程的user_id，并且按照user_id升序排序，以上例子查询结果如下：

| user_id |
|---------|
| 57 |
| 557336 |

解析：

id为4, 6的订单满足以上条件，输出对应的user_id为57；

id为5, 7的订单满足以上条件，输出对应的user_id为557336；

按照user_id升序排序。

写一个sql语句查询在2025-10-15之后，同一个用户下单2个以及2个以上状态为购买成功的C++课程或java课程或python课程的user_id,并且按照user_id升序排序；

思考思路

目标字段：满足题目要求的user_id,

库表来源：订单信息表(order_info)

连接关系: 无

筛选条件：date>'2025-10-15';同一用户下单2个以及2个以上；状态status为completed购买成功；课程product_name为C++课程/Java课程/python课程；

聚合依据：user_id;

梳理思路：1、首先，以user_id为partition by聚合依据开窗聚合，count(*)计算每个用户的分组下记录的数量是否 ≥ 2 (外查询的筛选条件)，内查询把筛选条件date&status&product_name三个题目要求的条件添加到where后然后根据user_id、product_name两个字段进行group by；2、将查询出用户下单量count(**)的查询作为内查询，别名为count_sale，添加上筛选条件where ct ≥ 2 ，并以user_id为聚合依据将内查询的user_id记录进行去重；

分段编辑：

#1、首先，以user_id为partition by聚合依据开窗聚合，count(*)计算每个用户的分组下记录的数量是否>=2(外查询的筛选条件)，内查询把筛选条件date&status&product_name三个题目要求的条件添加到where后然后根据user_id、product_name俩个字段进行group by;

```
select user_id
,product_name
, count(*) over(partition by user_id) ct
from order_info
where date > '2025-10-15' and status = 'completed'
and (product_name = 'C++' or product_name = 'Java' or product_name = 'Python')
group by 1,2
```

#2、将查询出用户下单量count(**)的查询作为内查询，别名为count_sale，添加上筛选条件where ct>=2；

```
select count_sale.user_id
from
(
    select user_id
    ,product_name
    , count(*) over(partition by user_id) ct
    from order_info
    where date > '2025-10-15' and status = 'completed'
    and (product_name = 'C++' or product_name = 'Java' or product_name = 'Python')
    group by 1,2
) count_sale
where count_sale.ct >= 2
group by 1
order by 1
```

组合代码：

```
select count_sale.user_id
from
(
    select user_id
    ,product_name
    , count(*) over(partition by user_id) ct
    from order_info
    where date > '2025-10-15' and status = 'completed'
    and (product_name = 'C++' or product_name = 'Java' or product_name = 'Python')
    group by 1,2
) count_sale
where count_sale.ct >= 2
group by 1
order by 1
```

-SQL273(中等)

牛客的课程订单分析(三)

SQL273 牛客的课程订单分析(三)

分享 星级 收藏 新闻

| 题目 | 题解(98) | 讨论(185) | 排行 | 面经 | new |
|----|--------|---------|-----------|----|-----|
| 6 | 57 | Java | completed | 1 | |
| 7 | 557336 | C++ | completed | 1 | |

第1行表示user_id为557336的用户在2025-10-10的时候使用了client_id为1的客户端下了C++课程的订单，但是状态为没有购买成功。

第2行表示user_id为230173543的用户在2025-10-12的时候使用了client_id为2的客户端下了Python课程的订单，状态为购买成功。

...
最后1行表示user_id为557336的用户在2025-10-25的时候使用了client_id为1的客户端下了C++课程的订单，状态为购买成功。

请你写出一个sql语句查询在2025-10-15以后，同一个用户下单2个以及2个以上状态为购买成功的C++课程或Java课程或Python课程的订单信息，并且按照order_info的id升序排序，以上例子查询结果如下：

| id | user_id | product_name | status | client_id |
|----|---------|--------------|-----------|-----------|
| 4 | 57 | C++ | completed | 3 |
| 5 | 557336 | Java | completed | 1 |
| 6 | 57 | Java | completed | 1 |
| 7 | 557336 | C++ | completed | 1 |

解析：

id为4, 6的订单满足以上条件，输出它们的对应的信息；

id为5, 7的订单满足以上条件，输出它们的对应的信息；

按照id升序排序



写一个sql语句查询在2025-10-15之后，同一个用户下单2个以及2个以上状态为购买成功的C++课程或Java课程或python课程的订单信息，并按照order_info的id升序排序；

思考思路

跟sql272相比，sql273也就是比sql272多了id、status、client_id和date这些在原表中存在的字段；初步思路是将sql272的内查询select查询语句作为一张新表和原表进行连接，最后再将用户的所有这些信息从连接好的表格中查询出来；但是需要添加筛选条件：where count_sale.ct>=2 and status='completed'；

组合代码

```
select order_info.id  
,order_info.user_id
```

```

,order_info.product_name
,order_info.status
,order_info.client_id
,order_info.date
from
(
    select user_id
    ,product_name
    ,count(*)over(partition by user_id) ct
    from order_info
    where date>'2025-10-15' and status='completed'
    and (product_name='C++' or product_name='Java' or product_name='Python')
    group by 1,2
) count_sale join order_info on count_sale.user_id=order_info.user_id and
count_sale.product_name=order_info.product_name
where count_sale.ct>=2 and status='completed'
order by order_info.id

```

-SQL274(较难)

SQL274 牛客的课程订单分析(四)



new

题目

题解(89)

讨论(251)

排行

面经

最后1行表示user_id为557336的用户在2025-10-25的时候使用了client_id为1的客户端下了Python课程的订单，状态为购买成功。

请你写出一个sql语句查询在2025-10-15以后，如果有一个用户下单2个以及2个以上状态为购买成功的C++课程或Java课程或Python课程，那么输出这个用户的user_id，以及满足前面条件的第一次购买成功的C++课程或Java课程或Python课程的日期first_buy_date，以及所有日期里购买成功的C++课程或Java课程或Python课程的次数cnt，并且输出结果按照user_id升序排序，以上例子查询结果如下：

| user_id | first_buy_date | cnt |
|---------|----------------|-----|
| 57 | 2025-10-23 | 2 |
| 557336 | 2025-10-23 | 3 |

解析：

id为4, 6的订单满足以上条件，输出57，id为4的订单为第一次购买成功，输出first_buy_date为2025-10-23，总共成功购买了2次；
 id为5, 7, 8的订单满足以上条件，输出557336，id为5的订单为第一次购买成功，输出first_buy_date为2025-10-23，总共成功购买了3次；

请你写出一个sql语句查询在2025-10-15以后，如果有一个用户下单2个以及2个以上状态为购买成功的C++课程或Java课程或Python课程，那么输出这个用户的user_id，以及满足前面条件的第一次购买成功的C++课程或Java课程或Python课程的日期first_buy_date，以及所有日期里购买成功的C++课程或Java课程或Python课程的次数cnt，并且输出结果按照user_id升序排序

梳理思路

- 1、跟上题的思路类似，将sql272的内查询select查询语句group by聚合依据增加一个日期date字段，即可查出牛客用户在下定符合题目条件的订单的具体日期；
- 2、然后在select位置加上窗口函数row_number()over(partition by user_id order by) rk，对用户下订单的日期date进行排序,为下一步提供筛选条件:排序=1的，符合题目要求，被筛选出来；
- 3、以步骤1&2结合的代码为子查询，外查询将符合题目要求的字段筛选出来，并添加筛选条件：wheresale1.rk=1 and sale1.ct>=2即可；
- 4、最后输出结果按照user_id升序排序；

组合代码

```
select sale1.user_id
,sale1.date first_buy_date
,sale1.ct
from
(
    select user_id
    ,product_name
    ,date
    ,count(*)over(partition by user_id) ct
    ,row_number()over(partition by user_id order by date asc) rk
    from order_info
    where date>'2025-10-15' and status='completed'
    and (product_name='C++' or product_name='Java' or product_name='Python')
    group by 1,2,3
) sale1
where sale1.rk=1 and sale1.ct>=2
order by user_id
```

SQL275(困难)

SQL275 牛客的课程订单分析(五)

困难

new

题目

题解(158)

讨论(361)

排行

面经

最后1行表示user_id为557336的用户在2025-10-26的时候使用了client_id为1的客户端下了Python课程的订单，状态为购买成功。

请你写出一个sql语句查询在2025-10-15以后，如果有一个用户下单2个以及2个以上状态为购买成功的C++课程或Java课程或Python课程，那么输出这个用户的user_id，以及满足前面条件的第一次购买成功的C++课程或Java课程或Python课程的日期first_buy_date，以及满足前面条件的第二次购买成功的C++课程或Java课程或Python课程的日期second_buy_date，以及购买成功的C++课程或Java课程或Python课程的次数cnt，并且输出结果按照user_id升序排序，以上例子查询结果如下：

| user_id | first_buy_date | second_buy_date | cnt |
|---------|----------------|-----------------|-----|
| 57好 | 2025-10-23 | 2025-10-24 | 2 |
| 557336 | 2025-10-23 | 2025-10-25 | 3 |

解析：

id为4, 6的订单满足以上条件，输出57，id为4的订单为第一次购买成功，输出first_buy_date为2025-10-23，id为6的订单为第二次购买，输出second_buy_date为2025-10-24，总共成功购买了2次；

id为5, 7, 8的订单满足以上条件，输出557336，id为5的订单为第一次购买成功，输出first_buy_date为2025-10-23，id为7的订单为第二次购买，输出second_buy_date为2025-10-25，总共成功购买了3次；

请你写出一个sql语句查询在2025-10-15以后，如果有一个用户下单2个以及2个以上状态为购买成功的C++课程或Java课程或Python课程，那么输出这个用户的user_id，以及满足前面条件的第一次购买成功的C++课程或Java课程或Python课程的日期first_buy_date，以及满足前面条件的第二次购买成功的C++课程或Java课程或Python课程的日期second_buy_date，以及购买成功的C++课程或Java课程或Python课程的次数cnt，并且输出结果按照user_id升序排序。

梳理思路

- 1、把筛选日期排名rk=1的查询语句和日期排名rk=2的查询语句分开编辑；
- 2、然后将两个表以连接键sales1.id=sales2.id and sales1.cnt=sales2.cnt来连接；
- 3、from步骤1&2连接后的表格，查询出题目截图所示的所有字段；

代码组合

```
select sales1.id
,sales1.first_buy_date
,sales2.second_buy_date
,sales1.cnt
from
(
    select sale1.user_id id
    ,sale1.date first_buy_date
    ,sale1.ct cnt
    from
    (
        select user_id
        ,product_name
        ,date
        ,count(*)over(partition by user_id) ct
        ,row_number()over(partition by user_id order by date asc) rk
        from order_info
        where date>'2025-10-15' and status='completed'
        and (product_name='C++' or product_name='Java' or product_name='Python')
        group by 1,2,3
    ) sale1
    where sale1.rk=1 and sale1.ct>=2
    order by id
) sales1

join

(
    select sale2.user_id id
    ,sale2.date second_buy_date
    ,sale2.ct cnt
    from
    (
        select user_id
        ,product_name
        ,date
        ,count(*)over(partition by user_id) ct
        ,row_number()over(partition by user_id order by date asc) rk
        from order_info
        where date>'2025-10-15' and status='completed'
        and (product_name='C++' or product_name='Java' or product_name='Python')
        group by 1,2,3
    ) sale2
    where sale2.rk =2 and sale2.ct>=2
    order by id
) sales2 on sales1.id=sales2.id and sales1.cnt=sales2.cnt
```

-SQL276(中等)

SQL276 牛客的课程订单分析(六)

分享 收藏 报错

new

题目

题解(113)

讨论(252)

排行

面经

购买成功。

有一个客户端表(client)，简况如下：

| id | name |
|----|---------|
| 1 | PC |
| 2 | Android |
| 3 | IOS |
| 4 | H5 |

请你写出一个sql语句查询在2025-10-15以后，同一个用户下单2个以及2个以上状态为购买成功的C++课程或Java课程或Python课程的订单id，是否拼团以及客户端名字信息，最后一列如果是非拼团订单，则显示对应客户端名字，如果是拼团订单，则显示NULL，并且按照order_info的id升序排序，以上例子查询结果如下：

| id | is_group_buy | client_name |
|----|--------------|-------------|
| 4 | No | IOS |
| 5 | Yes | NULL |
| 6 | No | PC |
| 7 | Yes | NULL |

解析：

id为4, 6的订单满足以上条件，且因为4是通过IOS下单的非拼团订单，输出对应信息，6是通过PC下单的非拼团订单，输出对应的信息以及客户端名字；

id为5, 7的订单满足以上条件，且因为5与7都是拼团订单，输出对应的信息以及NULL；

按照id升序排序



写一个sql语句查询在2025-10-15以后，同一个用户下单2个及2个以上状态为购买成功的C++课程或Java课程或Python课程的订单id，是否拼团以及客户端名字信息，最后一列如果是非拼团订单，则显示对应客户端名字，如果是拼团订单，则显示null，并且按照order_info的id升序排序

梳理思路

- 1、复用sql272中使用窗口函数对user_id进行分组对用户下单数进行计数的代码；将查询字段改成并添加，字段一共有：user_id、client_id、id、is_group_buy、count(*)over(partition by user_id) ct；
- 2、将步骤1作为子查询，新建一个外查询，选择字段：id、is_group_buy、client_id查询，其中client_id为和client表格的连接键；添加筛选条件：where rt1.ct>=2；添加排序条件：order by id；
- 3、将步骤2作为新表和client表格连接，连接键为client_id；但是题目要求，拼团成功不计客户端，拼团不成功则计算客户端，所以
思路是：以步骤2的新表作为主表左连接left join client；
- 4、给外查询添加一个升序排序条件：order by id；

组合代码

```
select rt2.id
,rt2.is_group_buy
,c.name
from
(
    select id
    ,is_group_buy
    ,client_id
    from
    (
        select user_id
        ,client_id
        ,id
        ,is_group_buy
        ,count(*)over(partition by user_id) ct
        from order_info
        where date>'2025-10-15' and status='completed'
        and (product_name='C++' or product_name='Java' or product_name='Python')
    ) rt1
    where rt1.ct>=2
    order by id
) rt2 left join client c on rt2.client_id=c.id
order by id
```

-SQL277(较难)

请你写出一个sql语句查询在2025-10-15以后，同一个用户下单2个以及2个以上状态为购买成功的C++课程或Java课程或Python课程的来源信息，第一列是显示的是客户端名字，如果是拼团订单则显示GroupBuy，第二列显示这个客户端(或者是拼团订单)有多少订单

请你写出一个sql语句查询在2025-10-15以后，同一个用户下单2个以及2个以上状态为购买成功的C++课程或Java课程或Python课程的来源信息，第一列是显示的是客户端名字，如果是拼团订单则显示GroupBuy，第二列显示这个客户端(或者是拼团订单)有多少订单，最后结果按照第一列(source)升序排序，以上例子查询结果如下：

| source | cnt |
|----------|-----|
| GroupBuy | 2 |
| IOS | 1 |
| PC | 1 |



解析:

id为4, 6的订单满足以上条件, 且因为4是通过IOS下单的非拼团订单, 则记: IC
, 6是通过PC下单的非拼团订单, 则记: PC 1;
id为5, 7的订单满足以上条件, 且因为5与7都是拼团订单, 则记: GroupBuy 2;
最后按照source升序排序。

梳理思路

首先，先梳理一下该题跟前几个同类型的sql题的区别：题目要求，拼团订单和客户端单独下单的订单数量需要分开计算，但是题目的要求“同一个用户下单2个及2个以上状态为购买成功的C++.....”没有改变；

- 1、复用sql276的代码部分，先筛选出同一个用户下单2个及2个以上状态为购买成功的C++课程或Java课程或python课程的来源信息等...并且以连接键client_id和client表左连接；
 - 2、以source(client.name)为group by聚合依据，cout(*) cnt对分组的数据进行计数，从而符合题目要求；
 - 3、对于client.name为null的值用`ifnull(c.name,'GroupBuy')`函数来替换None值为'GroupBuy'；
 - 4、添加升序排序条件`order by source`；

组合代码

```
select ifnull(c.name,'GroupBuy') source
, count(*) cnt
from
(
    select id
    , is_group_buy
    , client_id
    from
    (
        select user_id
        , client_id
        , id
        , is_group_buy
```

```

, count(*) over(partition by user_id) ct
from order_info
where date>'2025-10-15' and status='completed'
and (product_name='C++' or product_name='Java' or product_name='Python')
) rt1
where rt1.ct>=2
order by id
) rt2 left join client c on rt2.client_id=c.id
group by 1
order by source

```

-SQL278(简单)

请你写出SQL语句查询在2025年内投递简历的岗位和数量，并且按数量降序排序

SQL278 实习广场投递简历分析(一)



new

题目

题解(54)

讨论(128)

排行

面经

第1行表示，在2025年1月2号，C++岗位收到了53封简历

...

最后1行表示，在2026年1月4号，Java岗位收到了230封简历

请你写出SQL语句查询在2025年内投递简历的岗位和数量，并且按数量降序排序，
以上例子查询结果如下：

| job | cnt |
|--------|-----|
| C++ | 141 |
| Java | 128 |
| Python | 111 |

梳理思路

根据题目要求，查询2025年内--“where筛选条件1”、查询出对应年份下的岗位和数量--“group by job,sum(...)"、并且按照数量降序排序"order by sum(...) desc";

- 1、首先对年份的筛选条件进行思考--“where year(date)=2025”;
- 2、sum(num) cnt group by job对job岗位进行分组去重统计sum计算总的投递数量;
- 3、按照总投递数量降序排序order by cnt desc;

组合代码

```
select job
,sum(num) cnt
from resume_info
where year(date)=2025
group by 1
order by cnt desc
```

-SQL279(中等)

请你写出SQL语句查询在2025年内投递简历的每个岗位，每一个月内收到简历的数量，并且按先按月份降序排序，再按简历数目降序排序

SQL279 实习广场投递简历分析(二)



new

题目

题解(79)

讨论(151)

排行

面经

请你写出SQL语句查询在2025年内投递简历的每个岗位，每一个月内收到简历的数量，并且按先按月份降序排序，再按简历数目降序排序，以上例子查询结果如下：

| job | mon | cnt |
|--------|---------|-----|
| Java | 2025-03 | 126 |
| C++ | 2025-03 | 99 |
| Python | 2025-03 | 88 |
| Python | 2025-02 | 93 |
| C++ | 2025-02 | 68 |
| Java | 2025-02 | 66 |
| C++ | 2025-01 | 107 |
| Python | 2025-01 | 66 |
| Java | 2025-01 | 53 |

梳理思路

先解读一下题目的意思： "按照job岗位和月份mon进行分组聚合，从而对每个岗位每个月份所投递的简历数量进行sum求和，并先按照月份降序排序然后再按照简历数目降序排序"；

1、本题的核心就是要设法将date日期中的月份给筛出来，并对其进行分组聚合，使用日期格式化函数 date_format(date,format)--将日期格式化成"year-month"；

2、然后按照Job岗位和月份month进行group by分组聚合，然后再select使用聚合函数sum(num) cnt对投递数量进行求和；

组合代码

```
select job
,date_format(date, '%Y-%m') mon
,sum(num) cnt
from resume_info
where year(date)=2025
group by 1,2
order by mon desc,cnt desc
```

-SQL280(困难)

请你写出SQL语句查询在2025年投递简历的每个岗位，每一个月内收到简历的数目，和对应的2026年的同一个月同岗位，收到简历的数目，最后的结果先按first_year_mon月份降序，再按job降序排序显示

SQL280 实习广场投递简历分析(三)

收藏 分享 新

题目

题解(170)

讨论(337)

排行

面经

最后1行表示，在2027年2月6号，C++岗位收到了231封简历

请你写出SQL语句查询在2025年投递简历的每个岗位，每一个月内收到简历的数目，和对应的2026年的同一个月同岗位，收到简历的数目，最后的结果先按first_year_mon月份降序，再按job降序排序显示，以上例子查询结果如下：

| job | first_year_mon | first_year_cnt | second_year_mon | second_year_cnt |
|--------|----------------|----------------|-----------------|-----------------|
| Python | 2025-02 | 93 | 2026-02 | 846 |
| Java | 2025-02 | 66 | 2026-02 | 1649 |
| C++ | 2025-02 | 68 | 2026-02 | 394 |
| Python | 2025-01 | 66 | 2026-01 | 1268 |
| Java | 2025-01 | 53 | 2026-01 | 1478 |
| C++ | 2025-01 | 107 | 2026-01 | 470 |

解析：

第1行表示Python岗位在2025年2月收到了93份简历，在对应的2026年2月收到了846份简历

.....

最后1行表示C++岗位在2025年1月收到了107份简历，在对应的2026年1月收到了470份简历

梳理思路

- 1、先进行题目意思解读，“将两个不同年份的同一月份的同一岗位的投递简历数量总和放在同一行中查询出来，并分别对当年的该月份进行计数”；
- 2、主要的思路还是跟SQL279一样，对job和对应年份的month进行group by分组聚合，并使用聚合函数sum对对应分组下的num进行sum求和；如何求第二年的对应岗位和月份的简历投递数量呢？
- 3、将SQL279的解题套路再实施一遍，修改where的年份筛选条件，筛选出2026年的简历投递记录，求和计数sum跟上题SQL279的思路一样，但需要添加一个月份字段month1&2，因为题目要求将两个不同年份的同一个月份的同一个岗位的投递简历的数量总和放在同一行中查询出来，为了后续的表连接；然后将此表与SQL279的表格进行连接，连接键为job和month，最后根据题目要求把需要的字段查询出结果；
- 4、修改排序规则：最后的结果先按first_year_mon月份降序，再按job降序排序；

组合代码

```
select rt1.job  
,first_year_mon
```

```
,first_year_cnt
,second_year_mon
,second_year_cnt
from
(
    select job
    ,date_format(date, '%Y-%m') first_year_mon
    ,month(date) month1
    ,sum(num) first_year_cnt
    from resume_info
    where year(date)=2025
    group by 1,2,3
    order by first_year_mon desc,job desc
) rt1

join

(
    select job
    ,date_format(date, '%Y-%m') second_year_mon
    ,month(date) month2
    ,sum(num) second_year_cnt
    from resume_info
    where year(date)=2026
    group by 1,2,3
    order by second_year_mon desc,job desc
) rt2 on rt1.job=rt2.job and rt1.month1=rt2.month2
```

-SQL281(中等)

最差是第几名

题目

题解(50)

讨论(128)

排行

面经

中等 通过率: 61.01% 时间限制: 1秒 空间限制: 256M

描述

TM小哥和FH小妹在牛客大学若干年后成立了牛客SQL班，班的每个人的综合成绩用A,B,C,D,E表示，90分以上都是A，80~90分都是B，70~80分为C，60~70为D，E为60分以下

假设每个名次最多1个人，比如有2个A，那么必定有1个A是第1名，有1个A是第2名(综合成绩同分也会按照某一门的成绩分先后)。

每次SQL考试完之后，老师会将班级成绩表展示给同学看。

现在有班级成绩表(class_grade)如下：

| grade | number |
|-------|--------|
| A | 2 |
| D | 1 |
| C | 2 |
| B | 2 |

第1行表示成绩为A的学生有2个

.....

最后1行表示成绩为B的学生有2个

请你写出一个SQL查询，如果一个学生知道了自己综合成绩以后，最差是排第几名？结果按照grade升序排序，以上例子查询如下：

| grade | t_rank |
|-------|--------|
| A | 2 |
| B | 4 |

梳理思路

- 1、题目解读：“写一个SQL查询，如果一个学生知道了自己综合成绩grade之后，还能推测出自己最差是排第几名，结果按照**grade升序排序**”；
- 2、直接使用窗口函数的rows指定顺序和范围运算，`sum(number)over(order by grade rows between unbounded preceding and current row) t_rank;`
- 3、本质上题目的意思还是根据grade排序，然后根据每个grade有多少number数量进行从上到下的数量累加；

组合代码

```
select grade
,sum(class_grade.number)over(order by grade rows between unbounded preceding and
current row) t_rank
from class_grade
```

SQL282(较难)

SQL282 最差是第几名(二)



题目

题解(98)

讨论(172)

排行

面经

new

现在有班级成绩表(class_grade)如下：

| grade | number |
|-------|--------|
| A | 2 |
| C | 4 |
| B | 4 |
| D | 2 |

第1行表示成绩为A的学生有2个

.....

最后1行表示成绩为D的学生有2个

老师想知道学生们综合成绩的中位数是什么档位，请你写SQL帮忙查询一下，如果只有1个中位数，输出1个，如果有2个中位数，按grade升序输出，以上例子查询结果如下：

| grade |
|-------|
| B |
| C |

解析：

总体学生成绩排序如下：A, A, B, B, B, B, C, C, C, C, D, D，总共12个数，取中间的2个，取6, 7为：B, C

老师想知道学生们的综合成绩的中位数是什么档位的，如果只有一个中位数，那么输出1个，如果有2个中位数，按grade升序输出；

梳理思路

- 1、题目解读：“总的来说，题目的意思就是根据班级成员数量number/成绩表中的number是偶数还是单数来匹配回原表中去，筛选出这个综合成绩的中位数会落在哪个档位上：例如：A,A,B,B,B,C,C,C,C,D,D，共12个数，取中间的2个，取6, 7为B, C”；

- 2、首先，先对班级成绩表class_grade中number进行sum(number)求和，以及round(sum(number)/2,0) , round((sum(number)+1)/2,0)对，对sum(number)进行四舍五入，结果如果有小数则向上取整、若是整数则直接得出结果，保留小数点后0位；
- 3、其次，对步骤2的查询作为一个子查询，新建一个外查询(在SQL281的题的背景下)，计算每个grade的最差排名，用lag(number1,1,0)查询当前行的上一行的值(最差排名)；
- 4、添加一个where的筛选条件：where (number2 < s1 and number1 >= s1) or (number2 < s2 and number1 >= s2)，这一句代码的目的是：判断中位数会落在哪个等级上，当sum(number)为偶数时，那么有俩中位数，那么就得限制这俩数(排名)落在哪个排名的范围中，其中较小的那个数和以grade分组的number1作比较：number1 >= s1，但是也得和number1前面那个等级的排名number2对比：number2 < s1；其中较大的那个数也同理，如果s1=s2的话那么sum(number)为奇数，按照题目意思：奇数只有一位中位数，此时or两边的条件取值范围是一致的，结果只会落到一个成绩等级上；

组合代码

```
select grade
from
(
    select *
    ,lag(number1,1,0) over() as number2
    from
    (
        select *, sum(number) over(order by grade) number1
        ,(select round(sum(number)/2,0) from class_grade) s1
        ,(select round((sum(number)+1)/2,0) from class_grade) s2
        from class_grade
    ) s
) s3
where (number2 < s1 and number1 >= s1) or (number2 < s2 and number1 >= s2)
```

-SQL283(中等)

获得积分最多的人(一)

题目

题解(105)

讨论(233)

排行

面经

还有一个积分表(grade_info), 简况如下:

| user_id | grade_num | type |
|---------|-----------|------|
| 1 | 3 | add |
| 2 | 3 | add |
| 1 | 1 | add |
| 3 | 3 | add |
| 4 | 3 | add |
| 5 | 3 | add |

第1行表示, user_id为1的用户积分增加了3分。

第2行表示, user_id为2的用户积分增加了3分。

第3行表示, user_id为1的用户积分又增加了1分。

.....

最后1行表示, user_id为5的用户积分增加了3分。

请你写一个SQL查找积分增加最高的用户名, 以及他的总积分是多少(此题数据保证积分最高的用户有且只有1个), 以上例子查询结果如下:

| name | grade_num |
|------|-----------|
| tm | 4 |

解释:

user_id为1的总计加了4分, 其他的都是3分, user_id为1的name为tm

输出tm|4



梳理思路

- 1、按题目意思, 核心思路: 先以user_id为聚合依据, 对grade_num进行求和并根据grade_num进行降序排序sum(grade_num)over(partition by user_id order by grade_num);
- 2、然后再以步骤1为子查询, 新建一个外查询, 并且新建窗口函数row_number()over(partition by user_id order by rt1.sum_grade(对积分的求和));
- 3、再新建一层外查询, 以步骤2为子查询, 筛选出积分的和sum最高的user_id和sum值(唯一值);
- 4、与用户表user进行连接, 连接键就为user.id, 最终查询出题目要求的字段;

组合代码

```
select user.name
```

```
,rt3.sum_grade
from
(
    select rt2.user_id user_id
    ,rt2.sum_grade sum_grade
    from
    (
        select rt1.user_id user_id
        ,rt1.sum_grade sum_grade
        ,row_number()over(order by sum_grade desc) rk
        from
        (
            select user_id
            ,sum(grade_num)over(partition by user_id order by grade_num desc)
            sum_grade
            from grade_info
        ) rt1
    ) rt2
    where rt2.rk=1
) rt3 join user on rt3.user_id=user.id
```

-SQL284(较难)

请你写一个SQL查找积分增加最高的用户的id(可能有多个), 名字, 以及他的总积分是多少, 查询结果按照id升序排序

SQL284 获得积分最多的人(二)

收藏

new

题目

题解(141)

讨论(256)

排行

面经

还有一个积分表(grade_info), 简况如下:

| user_id | grade_num | type |
|---------|-----------|------|
| 1 | 3 | add |
| 2 | 3 | add |
| 1 | 1 | add |
| 3 | 3 | add |
| 4 | 3 | add |
| 5 | 3 | add |
| 3 | 1 | add |

第1行表示, user_id为1的用户积分增加了3分。

第2行表示, user_id为2的用户积分增加了3分。

第3行表示, user_id为1的用户积分又增加了1分。

.....

最后1行表示, user_id为3的用户积分增加了1分。

请你写一个SQL查找积分增加最高的用户的id(可能有多个), 名字, 以及他的总积分是多少, 查询结果按照id升序排序, 以上例子查询结果如下:

| id | name | grade_num |
|----|------|-----------|
| 1 | tm | 4 |
| 3 | zk | 4 |



解释:

user_id为1和3的2个人, 和公都为4, 都被输出

梳理思路

- 乍一看, 跟上一题sql283在字段查询上就增加一个用户id字段以及积分增加最高的用户不止一个了(可能有多个);
- 核心思路: 修改排序窗口函数为dense_rank()over(), 这个排序序号不唯一的排序函数;
- 复用SQL283的代码;

组合代码

```
select user.id  
, user.name
```

```
,rt3.sum_grade
from
(
    select rt2.user_id user_id
    ,rt2.sum_grade sum_grade
    from
    (
        select rt1.user_id user_id
        ,rt1.sum_grade sum_grade
        ,dense_rank()over(order by sum_grade desc) rk
        from
        (
            select user_id
            ,sum(grade_num)over(partition by user_id order by grade_num desc)
            sum_grade
            from grade_info
        ) rt1
    ) rt2
    where rt2.rk=1
) rt3 join user on rt3.user_id=user.id
```

-SQL285(困难)

请你写一个SQL查找积分最高的用户的id，名字，以及他的总积分是多少(可能有多个)，查询结果按照id升序排序

SQL285 获得积分最多的人(三)

置顶 收藏 分享 复制

new

题目

题解(169)

讨论(330)

排行

面经

| | | |
|---|---|--------|
| 2 | 3 | add |
| 1 | 1 | reduce |
| 3 | 3 | add |
| 4 | 3 | add |
| 5 | 3 | add |
| 3 | 1 | reduce |

第1行表示， user_id为1的用户积分增加了3分。

第2行表示， user_id为2的用户积分增加了3分。

第3行表示， user_id为1的用户积分减少了1分。

.....

最后1行表示， user_id为3的用户积分减少了1分。

请你写一个SQL查找积分最高的用户的id，名字，以及他的总积分是多少(可能有多个)，查询结果按照id升序排序，以上例子查询结果如下：

| id | name | grade_num |
|----|------|-----------|
| 2 | wwy | 3 |
| 4 | qq | 3 |
| 5 | lm | 3 |

解释：

user_id为1和3的先加了3分，但是后面又减了1分，他们2个是2分，其他3个都是3分，所以输出其他三个的数据。



梳理思路

- 1、按照题目意思，总思路跟SQL284的核心思路差不多，但是就是得区分type用户反馈类型是add还是reduce，如果是add则sum求和如果是reduce则减去对应数字；
- 2、核心思路：对add和reduce两个类型的记录分别求和，然后根据user_id连接起来，连接关系为左连接，因为如果为内连接的话，当type为reduce的积分为0的话就匹配不上对应的user_id，会被剔除掉记录，取求和的字段，用add类型的求和值减去reduce类型的求和值即为最终的积分增长的数值，然后对这个数值再进行排序取数值最高(排名第一的)；
- 3、给type为reduce的sum值加上ifnull函数，当reduce_grade为null时，赋值0；具体写法：
 $(g1.add_grade - ifnull(g2.reduce_grade,0)) last_grade$
- 4、再和用户user表连接，连接键为user_id，查询出题目要求的所有字段；
- 5、最后添加user.id,user.name,rt2.last_grade为group by的聚合依据：group by 1,2,3;

组合代码

```
select user.id
, user.name
, rt2.last_grade
from
(
    select rt1.user_id
    ,rt1.last_grade last_grade
    ,dense_rank()over(order by last_grade desc) rk
    from
    (
        select g1.user_id user_id
        ,(g1.add_grade - ifnull(g2.reduce_grade,0)) last_grade
        from
        (
            select user_id
            ,sum(grade_num)over(partition by user_id order by grade_num desc)
add_grade
            from grade_info
            where grade_info.type='add'
        ) g1 left join
        (
            select user_id
            ,sum(grade_num)over(partition by user_id order by grade_num desc)
reduce_grade
            from grade_info
            where grade_info.type='reduce'
        ) g2 on g1.user_id=g2.user_id
    ) rt1
) rt2 join user on user.id=rt2.user_id
where rt2.rk=1
group by 1,2,3
```

-SQL286(中等，网易校招笔试题)

SQL286 商品交易(网易校招笔试真题)

收藏 new

题目

题解(97)

讨论(147)

排行

面经

描述

如下有一张商品表 (goods) , 字段依次为: 商品id、商品名、商品质量

| id | name | weight |
|----|------|--------|
| 1 | A1 | 100 |
| 2 | A2 | 20 |
| 3 | B3 | 29 |
| 4 | T1 | 60 |
| 5 | G2 | 33 |
| 6 | C0 | 55 |

还有一张交易表 (trans) , 字段依次为: 交易id、商品id、这个商品购买个数

| id | goods_id | count |
|----|----------|-------|
| 1 | 3 | 10 |
| 2 | 1 | 44 |
| 3 | 6 | 9 |
| 4 | 1 | 2 |
| 5 | 2 | 65 |
| 6 | 5 | 23 |
| 7 | 3 | 20 |

查找购买个数超过20,质量小于50的商品, 按照商品id升序排序,如:

| id | name | weight | total |
|----|------|--------|-------|
| 2 | A2 | 20 | 81 |
| 3 | B3 | 29 | 30 |
| 5 | G2 | 33 | 23 |

思考思路

目标字段：商品id;商品名称name;商品质量weight;商品购买个数total

库表来源:商品表goods (商品id、商品名、商品质量) 、交易表trans (交易id、商品id、这个商品购买个数)

连接关系:内连接

筛选条件: having 购买个数>20;weight<50

聚合依据：商品id:根据商品id计算sum(count)每一件商品的购买数量total

梳理思路：1、将goods和trans两表以商品id为连接键连接；2、以goods_id为聚合依据group by;3、having筛选；4、order by商品id asc

分段编辑：

组合代码

```
select g.id
,g.name
,g.weight
,sum(count) total
from goods g join trans t on
g.id=t.goods_id
group by 1
having total>20 and weight<50
order by g.id asc
```

-SQL287(网易云音乐推荐(网易校招笔试真题))

SQL287 网易云音乐推荐(网易校招笔试真题)

分享 收藏 新闻

题目

题解(175)

讨论(322)

排行

面经

new

这张表的第一行代表着用户id为1的喜欢music_id为17的音乐

....

这张表的第五行代表着用户id为4的喜欢music_id为17的音乐

音乐music表，第一列是音乐id，第二列是音乐name,id是主键

| id | music_name |
|----|------------|
| 17 | yueyawang |
| 18 | kong |
| 19 | MOM |
| 20 | Sold Out |

请你编写一个SQL，查询向user_id = 1 的用户，推荐其关注的人喜欢的音乐。

不要推荐该用户已经喜欢的音乐，并且按music的id升序排列。你返回的结果中不应当包含重复项

上面的查询结果如下：

| music_name |
|------------|
| kong |
| MOM |

请你编写一个SQL，查询向user_id=1的用户，推荐其关注的人喜欢的音乐。不要推荐该用户已经喜欢的音乐，并且按照music的id升序排列。返回的结果中不应当包含重复项；

梳理思路

1、根据题目要求，先查询出user_id=1的用户的关注者的follow_id；然后根据这个id和音乐music_likes表连接，查询出music_id；然后根据这个id和音乐music表连接，查询出music_name:最后符合题目要求查询的字段；-----修改为where in的思路，具体看组合代码；

2、但是题目要求不能推荐用户已经喜欢的音乐，所以得在music_likes表中筛选出user_id=1的用户喜欢的音乐id,然后在下一步的查询中使用where not in(...)即可筛选掉用户自己已经喜欢的音乐；

组合代码

```
select music_name  
from music
```

```
where id in
(
    select music_id
    from music_likes
    where user_id in
    (
        select follower_id
        from follow
        where user_id=1
    )
)
and id not in
(
    select music_id
    from music_likes
    where user_id=1
)
```

-sql288(中等)

SQL288 今天的刷题量(一)

□ ☆ ☰ ⓘ

new

题目

题解(44)

讨论(104)

排行

面经

3 huaweijsjhu

4 top101

第一行表示:题单id为1的是剑指offer

...

最后一行表示:题单id为4的是面试笔刷TOP101

请你写出一个SQL，查找出当天(对，就是你现在写代码的这一天，实现原理就是后台有特殊程序会将'2999-02-22'这个东西变为今天的日期，并且将'2999-02-21'变为昨天的日期)的每个题单的刷题量，先按提交数量降序排序，如果提交数量一样的话，再按subject_id升序排序，以上例子查询如下：

| name | cnt |
|-------------|-----|
| tiba | 2 |
| jzoffer | 1 |
| huaweijsjhu | 1 |

解释：

第一行表示题霸这个专题一天的提交量为2，排名最靠前

第二行，第三行表示剑指offer，华为机试这2个专题一天的提交量都为1，但是剑指offer的subject_id比较小，排在前面

注：由于后台有程序会将'2999-02-22'这个东西变为今天的日期，并且将'2999-02-21'变为昨天的日期，请写出通用的代码，不然可能你的代码只有今天可以通过哟~



请你写出一个SQL，查找出当天(对，就是你现在写代码的这一天，实现原理就是后台有特殊程序会将'2999-02-22'这个东西变为今天的日期，并且将'2999-02-21'变为昨天的日期)的每个题单的刷题量，先按提交数量降序排序，如果提交数量一样的话，再按subject_id升序排序

梳理思路

- 1、题目一直有在强调，查找出当天的每个题单的刷题量，每天在这题就用'2999-02-22'来表示，依据这个提示可知应该添加一个where的筛选条件“where create_time='2999-02-22'”；
- 2、根据题目要求，首先在submission(代码提交表)中查询，以subject_id为聚合依据，并在select添加聚合函数count(*) cnt；
- 3、步骤2查询出结果之后和题单(subject)连接，连接键为id，即为上一步的subject_id；查找出题单名称name以及cnt数量；

组合代码

```
#程序出现bug，代码无法提交通过
select subject.name name
,rt1.cnt cnt
from
(
    select subject_id
    ,count(*)over(partition by subject_id) cnt
    from submission
    where create_time='2999-02-22'
) rt1 join subject on rt1.subject_id=subject.id
order by cnt desc,subject.id
```