【简单】175. 组合两个表

编写一个 SQL 查询，满足条件：无论 person 是否有地址信息，都需要基于上述两表提供 person 的以下信息：

```
FirstName, LastName, City, State
```

```
select FirstName,LastName,City,State
from Person  p
left join  Address a
on a.PersonId = p.PersonId
```

176. 第二高的薪水

```
SELECT
    IFNULL(
      (SELECT DISTINCT Salary
       FROM Employee
       ORDER BY Salary DESC
        LIMIT 1 OFFSET 1),

    NULL) AS SecondHighestSalary
```

177. 第N高的薪水

```
CREATE FUNCTION getNthHighestSalary(N INT) RETURNS INT
BEGIN
  RETURN (
      SELECT IFNULL(
        (select salary
        from(
        select salary,
        rank() over(order by salary desc) rk
        from Employee
        group by salary// 这句话我觉得可以不要
        )t1
        where rk=N),NULL) SecondHighestSalary
  );
END
```

【简单】178. 分数排名

重要提示：对于 MySQL 解决方案，如果要转义用作列名的保留字，可以在关键字之前和之后使用撇号。例如 `Rank`

```
select Score,
dense_rank() over(order by Score desc) `rank`
from Scores
```

180. 连续出现的数字

```
select distinct Num ConsecutiveNums
from
(
select
Num,
lead(Num,1,null) over(order by id) n2,
lead(Num,2,null) over(order by id) n3
from Logs
)t1
where Num = n2 and Num = n3
```

【简单】181. 超过经理收入的员工

```
select a.Name  Employee
from Employee a
join Employee b
on a.ManagerId = b.id
where a.Salary>b.Salary
```

【简单】182. 查找重复的电子邮箱

```
select Email
from Person
group by Email
having count(*)>1
```

【简单】183. 从不订购的客户

```
select  c.Name Customers
from Customers  c left join Orders  o
on c.id = o.CustomerId
where o.id is null
```

184. 部门工资最高的员工

```
select Department,Employee,Salary
from (
select d.Name  Department,e.Name Employee, e.Salary,
rank() over(partition by d.id order by Salary desc) rk // 这句话在这里就是筛选条件
from Employee e join Department d
on e.DepartmentId=d.id
)tmp
where rk = 1
```

185. 部门工资前三高的所有员工

```
select Department,Employee,Salary
from (
select d.Name  Department,e.Name Employee, e.Salary,
dense_rank() over(partition by d.id order by Salary desc) rk
from Employee e join Department d
on e.DepartmentId=d.id
)tmp
where rk <=3
```

196. 删除重复的电子邮箱

```
DELETE p1 FROM Person p1,
    Person p2
WHERE
    p1.Email = p2.Email AND p1.Id > p2.Id
```

197. 上升的温度

```
select
Id
from
(
select Id,RecordDate,Temperature,
lag(RecordDate,1,9999-99-99) over (order by RecordDate) yd,
lag(Temperature,1,999) over(order by RecordDate ) yt
from Weather
)tmp
where Temperature >yt
and datediff(RecordDate,yd)=1
```

【实践】262. 行程和用户

```
SELECT T.request_at AS `Day`,
    ROUND(
            SUM(
                IF(T.STATUS = 'completed',0,1) // 这里就很妙了！！！统计数量就可以用这
种技巧
            )
            /
            COUNT(T.STATUS),
            2
    ) AS `Cancellation Rate`
FROM trips AS T
WHERE
T.Client_Id NOT IN (
    SELECT users_id
    FROM users
    WHERE banned = 'Yes'
)
AND
T.Driver_Id NOT IN (
    SELECT users_id
    FROM users
    WHERE banned = 'Yes'
)
```

```
AND T.request_at BETWEEN '2013-10-01' AND '2013-10-03'
GROUP BY T.request_at
```

511. 游戏玩法分析 I

写一条 SQL 查询语句获取每位玩家 **第一次登陆平台的日期。**

```
select player_id ,event_date first_login
from (
select player_id ,event_date,
rank() over(partition by player_id order by event_date) rk
from Activity
) tmp
where rk = 1
```

2.最优 （选最小日期）

```
select player_id ,min(event_date) first_login
from Activity
group by player_id
```

512. 游戏玩法分析 II

请编写一个 SQL 查询，描述每一个玩家首次登陆的设备名称

```
select player_id ,device_id
from (
select player_id ,event_date,device_id,
rank() over(partition by player_id order by event_date) rk
from Activity
) tmp
where rk = 1
```

534. 游戏玩法分析 III

编写一个 SQL 查询，同时报告每组玩家和日期，以及玩家到目前为止玩了多少游戏。也就是说，在此日期之前玩家所玩的游戏总数。详细情况请查看示例。

```
select player_id,event_date ,
sum(games_played) over(partition by player_id order by  event_date
)games_played_so_far
from Activity
```

```
select
    a1.player_id,
    a1.event_date,
    sum(a2.games_played) games_played_so_far
from Activity a1,Activity a2
where a1.player_id=a2.player_id and  //这里就是笛卡尔积了
      a1.event_date>=a2.event_date  //看似只是一个简单的日期比较缺筛选出了1，2，3然后进行sum即可
group by 1,2; // 个人觉得group by 日期，但是其它的就取不出来了，所以1，2有啥用
```

550. 游戏玩法分析 IV

```
select round(avg(a.event_date is not null), 2) fraction
from
    (select player_id, min(event_date) as login
    from activity
    group by player_id) p
left join activity a
on p.player_id=a.player_id and datediff(a.event_date, p.login)=1
```

569. 员工薪水中位数

请编写SQL查询来查找每个公司的薪水中位数。

```
select Id,Company,Salary
from (
select Id,Company,Salary,
ROW_NUMBER() over(partition by Company order by Salary) rk,
count(*) over(partition by Company) cnt
from Employee
)t1
where rk IN (FLOOR((cnt + 1)/2), FLOOR((cnt + 2)/2))
```

```
//order by是由顺序要求的,自己的解法
select Id,Company,Salary
from(
select Id,Company,Salary,count(*) over(partition by company)
rk_count,row_number() over(partition by company order by salary) rk
from employee
)tmp1
where rk between ceil(rk_count/2) and floor(rk_count/2)+1
```

570. 至少有5名直接下属的经理

编写一个SQL查询来查找至少有5名直接下属的经理。

```
select Name
from Employee
where Id in (
select ManagerId
from Employee
group by ManagerId
having count(*)>=5
)
```

【难】571. 给定数字的频率查询中位数

请编写一个查询来查找所有数字的中位数并将结果命名为 `median` 。

```
select
    avg(cast(number as float)) median
from
    (
        select
```

```
            Number,
            Frequency,
            sum(Frequency) over(order by Number) - Frequency prev_sum,
            sum(Frequency) over(order by Number) curr_sum
        from Numbers
    ) t1,
    (
        select
            sum(Frequency) total_sum
        from Numbers
    ) t2
where
    t1.prev_sum <= (cast(t2.total_sum as float) / 2) and
    t1.curr_sum >= (cast(t2.total_sum as float) / 2)

//只要理解prev_sum和curr_sum是频数即可
```

```
SELECT
AVG(Number)median
FROM
(SELECT n1.Number FROM Numbers n1 JOIN Numbers n2 ON n1.Number>=n2.Number
 GROUP BY
 n1.Number
 HAVING
 SUM(n2.Frequency)>=(SELECT SUM(Frequency) FROM Numbers)/2
 AND
 SUM(n2.Frequency)-AVG(n1.Frequency)<=(SELECT SUM(Frequency) FROM Numbers)/2
)s
```

574. 当选者

```
select
    Name
from Candidate c
    left join Vote v
    on c.id = v.CandidateId
    group by Name
    order by count(*) desc
    limit 1
```

先过滤再 效率高很多

```
select Name
from Candidate
where id =
(
select CandidateId
    from
        (
        select CandidateId,
        count(*) over(partition by CandidateId ) cnt
        from Vote
        order by cnt desc
        limit 1
        )t1
```

```
)
```

577. 员工奖金

```
select name,bonus
from Employee e
left join Bonus b on  e.empId=b.empId
where bonus<1000 or bonus is null
```

578. 查询回答率最高的问题

```
select question_id  survey_log
from (
  select
      question_id,
      sum(if(action = 'answer', 1, 0)) as AnswerCnt,
      sum(if(action = 'show', 1, 0)) as ShowCnt
  from
      survey_log
  group by question_id
) as tbl
order by (AnswerCnt / ShowCnt) desc
limit 1
```

直接不嵌套

```
select question_id  survey_log
from survey_log
group by question_id
order by sum(if(action = 'answer', 1, 0)) / sum(if(action = 'show', 1, 0)) desc
limit 1
```

579. 查询员工的累计薪水

```
select Id,Month,
sum(Salary) over(partition by Id order by Month ROWS BETWEEN 2 PRECEDING AND
CURRENT ROW) Salary
from
(
    select Id,Month,Salary,
    lead(Month,1,0) over(partition by Id order by Month) lm //排除最近一个月
    from Employee
)t1
where lm != 0
order by  Id,Month desc
```

【简单】580. 统计各专业学生人数

```
select  dept_name ,count(student_id) student_number
from department d left join student s
on d.dept_id=s.dept_id
group by dept_name
order by student_number desc
```

【简单】584. 寻找用户推荐人

```
SELECT name FROM customer WHERE referee_id != 2 OR referee_id IS NULL;
```

【有意思】585. 2016年的投资

```
select sum(TIV_2016) TIV_2016
from (
    select PID,TIV_2016,cnt,
    count(*) over(partition by loc ) lcnt
    from (
        select PID,TIV_2016,
        count(TIV_2015) over(partition by TIV_2015 ) cnt,
        concat_ws(",",LAT,LON) loc
        from insurance
    )t1
)t2
where lcnt=1 and cnt!=1
```

优化 窗口

```
SELECT
    ROUND(SUM(TIV_2016), 2) as TIV_2016
FROM(
    SELECT
        *,
        count(*) over(partition by TIV_2015) as cnt_1,
        count(*) over(partition by LAT, LON) as cnt_2
    FROM
        insurance
) a
WHERE a.cnt_1 > 1 AND a.cnt_2 < 2
```

【简单】586. 订单最多的客户

```
select customer_number
from orders
group by customer_number
order by count(*)  desc
limit 1
```

【简单】596. 超过5名学生的课

```
select  class
from courses
group by class
having count(distinct student)>=5
```

597. 好友申请 I： 总体通过率

```
select
round(
    ifnull(
    (select count(*) from (select distinct requester_id, accepter_id from
request_accepted) as A)
    /
    (select count(*) from (select distinct sender_id, send_to_id from
friend_request) as B),
    0)
, 2) as accept_rate;
```

【难】601. 体育馆的人流量

```
select distinct t1.*
from stadium t1, stadium t2, stadium t3
where t1.people >= 100 and t2.people >= 100 and t3.people >= 100
and
(
        (t1.id - t2.id = 1 and t1.id - t3.id = 2 and t2.id - t3.id =1)  -- t1,
t2, t3
    or
    (t2.id - t1.id = 1 and t2.id - t3.id = 2 and t1.id - t3.id =1) -- t2, t1, t3
    or
    (t3.id - t2.id = 1 and t2.id - t1.id =1 and t3.id - t1.id = 2) -- t3, t2, t1
)
order by t1.id
```

窗口函数(272 ms)

```
select id,visit_date,people from
(
    select id
    ,lead(people,1) over(order by id) ld
    ,lead(people,2) over(order by id) ld2
    ,visit_date
    ,lag(people,1) over(order by id) lg
    ,lag(people,2) over(order by id) lg2
    ,people
    from stadium
    ) a
where (a.ld>=100 and a.lg>=100 and a.people>=100)
or (a.ld>=100 and a.ld2>=100 and a.people>=100)
or (a.lg>=100 and a.lg2>=100 and a.people>=100)
```

602. 好友申请 II： 谁有最多的好友

```sql
select rid as `id`,count(aid) as `num`
from
(
    select R1.requester_id as rid,R1.accepter_id as aid
    from request_accepted as R1
    UNION all
    select R2.accepter_id as rid,R2.requester_id as aid
    from request_accepted as R2
) as A
group by rid
order by num desc
limit 0,1
```

603. 连续空余座位

```sql
select seat_id
from (
select seat_id,
lag(seat_id,1,-99) over(order by seat_id) ls,
lead(seat_id,1,-99) over(order by seat_id) rs
from cinema
where free=1
)t1
where  seat_id-ls = 1 or rs-seat_id =1
```

607. 销售员

```sql
select name
from salesperson
where sales_id not in
(
    select sales_id
    from orders
    where com_id =
            (
            select com_id
            from company
            where name ='RED'
            )
)
```

608. 树节点

```sql
select id,
    (case when p_id is null then "Root"
    when id not in (select ifnull(p_id,0) from tree) then "Leaf"
    else "Inner" end)  Type
from tree
```

610. 判断三角形

```
select x,y,z,
if(x+y>z && x+z>y && y+z>x,'Yes','No') triangle
from triangle
```

612. 平面上的最近距离

```
SELECT
    ROUND(SQRT(MIN((POW(p1.x - p2.x, 2) + POW(p1.y - p2.y, 2)))), 2) AS shortest
FROM
    point_2d p1
        JOIN
    point_2d p2 ON p1.x != p2.x OR p1.y != p2.y
```

优化： 减少重复计算

```
SELECT
    ROUND(SQRT(MIN((POW(p1.x - p2.x, 2) + POW(p1.y - p2.y, 2)))),2) AS shortest
FROM
    point_2d p1
        JOIN
    point_2d p2 ON (p1.x <= p2.x AND p1.y < p2.y)
        OR (p1.x <= p2.x AND p1.y > p2.y)
        OR (p1.x < p2.x AND p1.y = p2.y)
```

613. 直线上的最近距离

开窗方法 178m

```
select min(l-x) shortest
from(
select x,lead(x,1,null) over(order by x) l
from point
)t1
```

join方法 268m

```
SELECT
    MIN(ABS(p1.x - p2.x)) AS shortest
FROM
    point p1
        JOIN
    point p2 ON p1.x != p2.x
;
```

614. 二级关注者

```
select followee follower,count(distinct follower) num
from follow
where followee  in (
    select follower
    from follow
    group by follower
)
group by followee
order by follower
```

【有意思】615. 平均工资：部门与公司比较

```
select
    pay_month,
    department_id,
     (case when avgs>ts then 'higher'
           when avgs<ts then 'lower'
           else 'same' end) as comparison
from
(
    select
        date_format(pay_date,'%Y-%m')pay_month,
        department_id,
        avg(amount) over(partition by date_format(pay_date,'%Y-%m') )ts,
        avg(amount) over(partition by date_format(pay_date,'%Y-
%m'),department_id) avgs
    from salary s
    left join employee e
    on s.employee_id = e.employee_id

)t1
group by pay_month, department_id
```

> 也可以用if

```
IF(avgs>ts,'higher',IF(avgs=ts,'same','lower')) AS comparison
```

【有问题】618. 学生地理信息报告

开窗

```
//不懂max子啊这里啥意思，获得分组最大值？
select
max(if(continent='America',name,null)) America,
max(if(continent='Asia',name,null)) Asia,
max(if(continent='Europe',name,null)) Europe
from
    (select *, row_number() over(partition by continent order by name) rk
    from student) t
group by rk

select
if(continent='America',name,null) America,
if(continent='Asia',name,null) Asia,
if(continent='Europe',name,null) Europe
```

```
from
student
```

变量

```
SELECT
    America, Asia, Europe
FROM
    (SELECT @as:=0, @am:=0, @eu:=0) t,
    (SELECT
        @as:=@as + 1 AS asid, name AS Asia
    FROM
        student
    WHERE
        continent = 'Asia'
    ORDER BY Asia) AS t1
        RIGHT JOIN
    (SELECT
        @am:=@am + 1 AS amid, name AS America
    FROM
        student
    WHERE
        continent = 'America'
    ORDER BY America) AS t2 ON asid = amid
        LEFT JOIN
    (SELECT
        @eu:=@eu + 1 AS euid, name AS Europe
    FROM
        student
    WHERE
        continent = 'Europe'
    ORDER BY Europe) AS t3 ON amid = euid
```

> 官方给出的。。同下方开窗

```
select America,Asia,Europe
from(
    select row_number() over(order by name) as rn,name as America from student
    where continent='America'
) a
left join(
    select row_number() over(order by name) as rn,name as Asia from student
    where continent='Asia'
) b on a.rn=b.rn
left join(
    select row_number() over(order by name) as rn,name as Europe from student
    where continent='Europe'
) c on a.rn=c.rn
```

619. 只出现一次的最大数字

```
SELECT
    MAX(num) AS num
FROM
    (SELECT
        num
    FROM
        my_numbers
    GROUP BY num
    HAVING COUNT(num) = 1) t1
```

620. 有趣的电影

```
select  id,movie,description,rating
from cinema
where id%2=1 and description !='boring'
order by rating desc,id,movie,description
```

【有意思】626. 换座位

```
select id,
(case when id%2=0 then f
      when id%2=1 && b is not null then b
      else student end) student
from(
    select id,student,
    lag(student,1,null) over(order by id) f,
    lead(student,1,null) over(order by id) b
    from seat
)t1
```

非嵌套

```
select
    if(id%2=0,
        id-1,
        if(id=(select count(distinct id) from seat),
            id,
            id+1))
    as id,student
from seat
order by id;
```

用异或

```
select b.id,a.student from
seat as a,seat as b,(select count(*) as cnt from seat) as c
where b.id=1^(a.id-1)+1
-- where a.id=1^(b.id-1)+1; 也可以这样写，更容易理解
 || (c.cnt%2 && b.id=c.cnt && a.id=c.cnt);
```

627. 交换工资

```
UPDATE salary
SET
    sex = CASE sex
        WHEN 'm' THEN 'f'
        ELSE 'm'
    END;
```

1045. 买下所有产品的客户

```
select customer_id
from Customer
group by customer_id
having count(distinct product_key)=(
select count(*) cnt
from Product)
```

1050. 合作过至少三次的演员和导演

```
select actor_id,director_id
from ActorDirector
group by actor_id,director_id
having count(*)>=3
```

1068. 产品销售分析 I

```
select product_name,year,price
from Sales s left join Product p
on s.product_id  = p.product_id
```

1069. 产品销售分析 II

编写一个 SQL 查询，按产品 id `product_id` 来统计每个产品的销售总量。

```
select s.product_id,sum(quantity) total_quantity
from Sales s left join Product p
on s.product_id  = p.product_id
group by s.product_id
```

1070. 产品销售分析 III

编写一个 SQL 查询，选出每个销售产品的 **第一年** 的 **产品 id**、**年份**、**数量** 和 **价格**。

```
select product_id,year first_year,quantity,price
from (
select s.product_id,year,quantity,price,
rank() over(partition by product_id order by year) rk
from Sales s left join Product p
on s.product_id  = p.product_id
)t1
where rk = 1
```

1075. 项目员工 I

请写一个 SQL 语句，查询每一个项目中员工的 **平均** 工作年限，**精确到小数点后两位。**

```
select project_id,round(avg(experience_years),2) average_years
from Project p join Employee e
on p.employee_id=e.employee_id
group by project_id
```

1076. 项目员工II

编写一个SQL查询，报告所有雇员最多的项目。

```
select project_id
from Project
group by project_id
having count(*) =
(select count(*) `num` from Project group by project_id order by count(*) desc
limit 1);
```

开窗

```
select project_id from
(select project_id,rank()over(order by count(employee_id) desc) ranking from
Project group by project_id) temp where ranking=1
```

1077. 项目员工 III

写 一个 SQL 查询语句，报告在每一个项目中经验最丰富的雇员是谁。如果出现经验年数相同的情况，请报告所有具有最大经验年数的员工。

```
select project_id ,employee_id
from(
select project_id,e.employee_id,
rank()over(partition by project_id order by experience_years desc) rk
from Project p join Employee e
on p.employee_id=e.employee_id
)t1
where rk=1
```

1082. 销售分析 I

编写一个 SQL 查询，查询总销售额最高的销售者，如果有并列的，就都展示出来。

```
select seller_id
from (
select seller_id ,sum(price) tp,rank() over(order by sum(price) desc) rk
from Sales
group by seller_id
)t1
where rk =1
```

1083. 销售分析 II

编写一个 SQL 查询，查询购买了 S8 手机却没有购买 iPhone 的买家。注意这里 S8 和 iPhone 是 Product 表中的产品。

```
select t.buyer_id from(
select s.buyer_id, p.product_name
from sales s
inner join
product p
on s.product_id=p.product_id and (p.product_name='S8' or
p.product_name='iPhone')
group by s.buyer_id
having count(distinct p.product_name) = 1
) t
where t.product_name='S8'
```

效率低

```
select s.buyer_id
from sales as s left join product as p
on s.product_id=p.product_id
group by buyer_id
having sum(p.product_name='S8')>0 and sum(p.product_name='iPhone')=0
```

1084. 销售分析III

编写一个SQL查询，报告2019年春季才售出的产品。即**仅**在**2019-01-01**至**2019-03-31**（含）之间出售的商品。

876ms

正常解法

```
select product_id, product_name
from product
where product_id not in (
select distinct product_id from sales
where sale_date > '2019-03-31' or sale_date < '2019-01-01')
```

867ms

sum = count

```
select product_id, product_name
from Sales join Product
using(product_id)
group by product_id
having sum(sale_date between "2019-01-01" and "2019-03-31") = count(sale_date)
```

987ms

sum 为0 类似于第一种解法，计算了sum效率就低了

```
select p.product_id, p.product_name
from sales s, product p
where s.product_id = p.product_id
group by s.product_id
having sum(s.sale_date> '2019-03-31')=0 and sum(s.sale_date < '2019-01-01') = 0
```

上一解法sum换成max

859ms

```
select p.product_id, p.product_name
from sales s, product p
where s.product_id = p.product_id
group by s.product_id
having min(s.sale_date)>='2019-01-01' and max(s.sale_date) <= '2019-03-31'
```

1097. 游戏玩法分析 V

编写一个 SQL 查询，报告每个安装日期、当天安装游戏的玩家数量和第一天的留存时间。

```
select install_dt,count(distinct player_id)installs,
       round(sum(if(datediff(event_date,install_dt)=1,1,0))/count(distinct
player_id),2) Day1_retention
from
(
    select *,min(event_date) over(partition by player_id) install_dt
    from Activity
)t1
group by install_dt
```

1098. 小众书籍

筛选出过去一年中订单总量 **少于10本** 的 **书籍** 。

```
SELECT a.book_id, a.name
FROM books a LEFT JOIN orders b ON a.book_id=b.book_id
AND dispatch_date BETWEEN DATE_ADD('2019-06-23', INTERVAL -1 YEAR) AND '2019-06-
23'
WHERE a.available_from <= DATE_ADD('2019-06-23',INTERVAL -1 MONTH)
GROUP BY a.book_id, a.name
HAVING SUM(IFNULL(b.quantity,0)) < 10
ORDER BY a.book_id;
```

1107. 每日新用户统计

查询从今天起最多 90 天内，每个日期该日期首次登录的用户数。假设今天是 **2019-06-30**.

```
select login_date,count(*) user_count
from(
    select user_id,min(activity_date) login_date
    from Traffic
    where activity = 'login'
    group by user_id
    having login_date>= DATE_ADD('2019-06-30',INTERVAL -90 DAY)
)t1
group by login_date
```

1112. 每位学生的最高成绩

```
select student_id,course_id ,grade
from (
select student_id,course_id ,grade,
rank()over(partition by student_id order by grade desc,course_id) rk
from Enrollments
)t1
where rk =1
```

1113. 报告的记录

查询每种 ***报告理由*** （report reason）在昨天的报告数量。假设今天是 **2019-07-05。**

```
SELECT
    extra AS report_reason,
    COUNT(distinct post_id) AS report_count
FROM
    Actions
WHERE
    `action` = 'report' AND action_date = date_add('2019-07-05',Interval -1 day)
GROUP BY
    extra;
```

1132. 报告的记录 II

在被报告为垃圾广告的帖子中，被移除的帖子的每日平均占比，**四舍五入到小数点后 2 位。**

```
SELECT ROUND(AVG(proportion) * 100, 2) AS average_daily_percent
FROM (
    SELECT actions.action_date, COUNT(DISTINCT removals.post_id)/COUNT(DISTINCT
actions.post_id) AS proportion
    FROM actions
    LEFT JOIN removals
    ON actions.post_id = removals.post_id
    WHERE extra = 'spam'
    GROUP BY actions.action_date
) a
```

1126. 查询活跃业务

```
select business_id
from (
    select business_id,occurences,
    avg(occurences) over(partition by event_type) avo
    from Events
)t1
where  occurences > avo
group by business_id
having count(*)>=2
```

1127. 用户购买平台

写一段 SQL 来查找每天 **仅** 使用手机端用户、**仅** 使用桌面端用户和 **同时** 使用桌面端和手机端的用户人数和总支出金额。

```
select
    spend_date, platform,
    ifnull(sum(total_am),0) total_amount,
    ifnull(sum(total_u),0) total_users
from
(
    select p.spend_date, p.platform, t.total_am, t.total_u
    from
    (
        select distinct spend_date, "desktop" platform from Spending
        union
        select distinct spend_date, "mobile" platform from Spending
        union
        select distinct spend_date, "both" platform from Spending
    ) p
    left join
    (
        select spend_date,
            if(count(distinct platform)=1, platform, 'both') plat,
            sum(amount) total_am,
            count(distinct user_id) total_u
        from Spending
        group by spend_date, user_id
    ) t
    on p.platform = t.plat and p.spend_date = t.spend_date
) temp
group by spend_date, platform
```

1141. 查询近30天活跃用户数

请写SQL查询出截至 **2019-07-27**（包含2019-07-27），**近** 30天的每日活跃用户数（当天只要有一条活动记录，即为活跃用户）。

```
select activity_date day,count(distinct user_id) active_users
from Activity
where activity_date > date_add('2019-07-27',INTERVAL -1 MONTH)
group by activity_date
```

1142. 过去30天的用户活动 II

```
SELECT IFNULL(ROUND(COUNT(DISTINCT session_id) / COUNT(DISTINCT user_id), 2), 0)
AS average_sessions_per_user
from Activity
where activity_date > date_add('2019-07-27',INTERVAL -1 MONTH)
```

1148. 文章浏览 I

请编写一条 SQL 查询以找出所有浏览过自己文章的作者，结果按照 id 升序排列。

```
select distinct author_id id
from Views
where author_id=  viewer_id
order by id
```

1149. 文章浏览 II

编写一条 SQL 查询来找出在同一天阅读至少两篇文章的人，结果按照 id 升序排序。

```
SELECT DISTINCT viewer_id AS id
FROM Views
GROUP BY view_date, viewer_id
HAVING COUNT(DISTINCT article_id) >= 2
ORDER BY viewer_id
```

1158. 市场分析 I

请写出一条SQL语句以查询每个用户的注册日期和在 **2019** 年作为买家的订单总数。

```
select user_id buyer_id,join_date, ifnull(cnt,0)orders_in_2019
from Users u left join
(select buyer_id,count(*) cnt
from Orders
where year(order_date) = 2019
group by  buyer_id
)t1
on u.user_id =t1.buyer_id
```

1159. 市场分析 II

写一个 SQL 查询确定每一个用户按日期顺序卖出的第二件商品的品牌是否是他们最喜爱的品牌。如果一个用户卖出少于两件商品，查询的结果是 `no` 。

```
select user_id seller_id, if(item_brand=favorite_brand,'yes','no')
2nd_item_fav_brand
from Users u
left join
(
    select i.item_id,seller_id,item_brand
    from Items i
    join
    (
```

```
        select seller_id,item_id,rank() over(partition by seller_id  order by
order_date ) rk
        from Orders
    )t1
    on i.item_id = t1.item_id
    where rk  =2
)t2
on u.user_id = t2.seller_id
```

1164. 指定日期的产品价格

写一段 SQL来查找在 **2019-08-16** 时全部产品的价格，假设所有产品在修改前的价格都是 **10。**

```
select distinct p.product_id,ifnull(t1.new_price,10) price
from Products p
left join
(
    select product_id,new_price
    from (
        select product_id,new_price,change_date,Max(change_date) over(partition
by product_id  ) md
        from  Products
        where change_date<='2019-08-16'
    )tmp
    where change_date = md
)t1
on p. product_id = t1.product_id
order by price desc
```

1173. 即时食物配送 I

写一条 SQL 查询语句获取即时订单所占的百分比， **保留两位小数。**

```
select round(sum(if(order_date=customer_pref_delivery_date,1,0))/count(*)*100,2)
immediate_percentage
from Delivery
```

1174. 即时食物配送 II

写一条 SQL 查询语句获取即时订单在所有用户的首次订单中的比例。**保留两位小数。**

开窗

```
select round(sum(if(order_date = fo && fo=d,1,0))/count(distinct
customer_id)*100,2) immediate_percentage
from
(
select customer_id,order_date,min(order_date)over ( partition by customer_id)
fo,
    if(order_date=customer_pref_delivery_date,order_date ,null) d
from Delivery
)t1
```

另一种思路

```sql
select round (
    sum(order_date = customer_pref_delivery_date) * 100 /
    count(*),
    2
) as immediate_percentage
from Delivery
where (customer_id, order_date) in (
    select customer_id, min(order_date)
    from delivery
    group by customer_id
)
```

1179. 重新格式化部门表

编写一个 SQL 查询来重新格式化表，使得新的表中有一个部门 id 列和一些对应 **每个月** 的收入（revenue）列。

查询结果格式如下面的示例所示：

```sql
SELECT id,
SUM(CASE `month` WHEN 'Jan' THEN revenue END) Jan_Revenue,
SUM(CASE `month` WHEN 'Feb' THEN revenue END) Feb_Revenue,
SUM(CASE `month` WHEN 'Mar' THEN revenue END) Mar_Revenue,
SUM(CASE `month` WHEN 'Apr' THEN revenue END) Apr_Revenue,
SUM(CASE `month` WHEN 'May' THEN revenue END) May_Revenue,
SUM(CASE `month` WHEN 'Jun' THEN revenue END) Jun_Revenue,
SUM(CASE `month` WHEN 'Jul' THEN revenue END) Jul_Revenue,
SUM(CASE `month` WHEN 'Aug' THEN revenue END) Aug_Revenue,
SUM(CASE `month` WHEN 'Sep' THEN revenue END) Sep_Revenue,
SUM(CASE `month` WHEN 'Oct' THEN revenue END) Oct_Revenue,
SUM(CASE `month` WHEN 'Nov' THEN revenue END) Nov_Revenue,
SUM(CASE `month` WHEN 'Dec' THEN revenue END) Dec_Revenue
FROM Department
GROUP BY id;
```

1193. 每月交易 I

编写一个 sql 查询来查找每个月和每个国家/地区的事务数及其总金额、已批准的事务数及其总金额。

```sql
SELECT DATE_FORMAT(trans_date, '%Y-%m') AS month,
    country,
    COUNT(*) AS trans_count,
    COUNT(IF(state = 'approved', 1, NULL)) AS approved_count,
    SUM(amount) AS trans_total_amount,
    SUM(IF(state = 'approved', amount, 0)) AS approved_total_amount
FROM Transactions
GROUP BY month, country
```

1205. 每月交易II

编写一个 SQL 查询，以查找每个月和每个国家/地区的已批准交易的数量及其总金额、退单的数量及其总金额。

```sql
select
    date_format(trans_date, '%Y-%m') month,
```

```
    country,
    sum(state = 'approved') approved_count,
    sum(if(state = 'approved', amount, 0)) approved_amount,
    sum(state = 'chargeback') chargeback_count,
    sum(if(state = 'chargeback', amount, 0)) chargeback_amount
from (
    select * from transactions
    union all
    select id, country, 'chargeback' state, amount, c.trans_date
    from chargebacks c left join transactions t
    on c.trans_id = t.id
) tmp
group by month, country
having approved_amount or chargeback_amount
```

1194. 锦标赛优胜者

编写一个 SQL 查询来查找每组中的获胜者。

union all

```
SELECT group_id, player_id
FROM (
    SELECT group_id, player_id, SUM(score) AS score
    FROM (
        -- 每个用户总的 first_score
        SELECT Players.group_id, Players.player_id, SUM(Matches.first_score) AS
score
        FROM Players JOIN Matches ON Players.player_id = Matches.first_player
        GROUP BY Players.player_id

        UNION ALL

        -- 每个用户总的 second_score
        SELECT Players.group_id, Players.player_id, SUM(Matches.second_score) AS
score
        FROM Players JOIN Matches ON Players.player_id = Matches.second_player
        GROUP BY Players.player_id
    ) s
    GROUP BY player_id
    ORDER BY score DESC, player_id
) result
GROUP BY group_id
```

```
select group_id, player_id
from (
    select players.*, sum(if(player_id = first_player, first_score,
second_score)) score
    from players join matches
    on player_id = first_player or player_id = second_player
    group by player_id
    order by score desc, player_id
) tmp
group by group_id
```

1204. 最后一个能进入电梯的人

写一条 SQL 查询语句查找最后一个能进入电梯且不超过重量限制的 `person_name` 。题目确保队列中第一位的人可以进入电梯 。

```
select person_name
from(
    select person_name  ,sum(weight) over(order by turn) t
    from Queue
)t1
where t<=1000
order by t desc
limit 1
```

1211. 查询结果的质量和占比

编写一组 SQL 来查找每次查询的`名称(query_name)`、`质量(quality)`和`劣质查询百分比(poor_query_percentage)`。

```
select query_name,
    round(avg(rating/position),2) quality,
    round(sum(if(rating<3,1,0))/count(*)*100,2) poor_query_percentage
from Queries
group by query_name
```

1212. 查询球队积分

写出一条SQL语句以查询每个队的 **team_id**，**team_name** 和 **num_points**。结果根据 num_points **降序排序**，如果有两队积分相同，那么这两队按 team_id **升序排序**。

```
select team_id , team_name ,sum(
if(team_id = host_team && host_goals>guest_goals ,3,0)+
if(team_id = host_team && host_goals=guest_goals,1,0)+
if(team_id = guest_team && host_goals=guest_goals,1,0)+
if(team_id = guest_team && host_goals<guest_goals ,3,0)
)num_points
from Teams t left join Matches m
on t.team_id =m.host_team  or t.team_id =m.guest_team
group by team_id ,team_name
order by num_points desc,team_id
```

1225. 报告系统状态的连续日期

编写一个 SQL 查询 **2019-01-01** 到 **2019-12-31** 期间任务连续同状态 `period_state` 的起止日期（`start_date` 和 `end_date`）。即如果任务失败了，就是失败状态的起止日期，如果任务成功了，就是成功状态的起止日期。

开窗函数

```
select type  period_state, min(date) start_date, max(date) as end_date
from
(
    select type, date, subdate(date,row_number()over(partition by type order by
date)) as diff
    from
```

```
    (
        select 'failed' as type, fail_date as date from Failed
        where fail_date between '2019-01-01' and '2019-12-31'
        union all
        select 'succeeded' as type, success_date as date from Succeeded
        where success_date between '2019-01-01' and '2019-12-31'
    ) a
)b
group by type,diff
order by start_date
```

1241. 每个帖子的评论数

后join

```
select s.sub_id post_id,ifnull(number_of_comments,0) number_of_comments
from Submissions s
left join (
  select parent_id ,count(distinct sub_id) number_of_comments
  from Submissions
  group by parent_id
) t1
on s.sub_id = t1.parent_id
where s.parent_id  is null
group by post_id,number_of_comments
order by sub_id
```

先join

```
SELECT post_id, COUNT(sub_id) AS number_of_comments
FROM (
    SELECT DISTINCT post.sub_id AS post_id, sub.sub_id AS sub_id
    FROM Submissions post
    LEFT JOIN Submissions sub
    ON post.sub_id = sub.parent_id
    WHERE post.parent_id is null
) T
GROUP BY post_id
ORDER BY post_id ASC
```

1251. 平均售价

```
SELECT
    product_id,
    Round(SUM(sales) / SUM(units), 2) AS average_price
FROM (
    SELECT
        Prices.product_id AS product_id,
        Prices.price * UnitsSold.units AS sales,
        UnitsSold.units AS units
    FROM Prices
    JOIN UnitsSold ON Prices.product_id = UnitsSold.product_id
    WHERE UnitsSold.purchase_date BETWEEN Prices.start_date AND Prices.end_date
) T
GROUP BY product_id
```

1264. 页面推荐

写一段 SQL 向`user_id` = 1 的用户，推荐其朋友们喜欢的页面。不要推荐该用户已经喜欢的页面。

```
select distinct page_id  recommended_page
from Likes
where user_id in(
select if(user1_id=1,user2_id,user1_id) user_id
from  Friendship
where  user1_id =1 or user2_id =1
)
and page_id not in (select page_id from Likes where user_id=1)
```

1270. 向公司CEO汇报工作的所有人

```
select employee_id
from
(
select a.employee_id
from Employees a
left join Employees b on a.manager_id = b.employee_id
left join Employees c on b.manager_id = c.employee_id
where a.manager_id=1 or b.manager_id=1  or c.manager_id=1
)t1
where employee_id!=1
```

1280. 学生们参加各科测试的次数

```
SELECT a.student_id, a.student_name, b.subject_name, COUNT(e.subject_name) AS
attended_exams
FROM Students a CROSS JOIN Subjects b
LEFT JOIN Examinations e ON a.student_id = e.student_id AND b.subject_name =
e.subject_name
GROUP BY a.student_id, b.subject_name
ORDER BY a.student_id, b.subject_name
```

1285. 找到连续区间的开始和结束数字

```
SELECT
    MIN(log_id) start_id,
    MAX(log_id) end_id
FROM
    (SELECT
        log_id,
        log_id - row_number() OVER(ORDER BY log_id) as diff
    FROM Logs) t
GROUP BY diff
```

1294. 不同国家的天气类型

写一段 SQL 来找到表中每个国家在 2019 年 11 月的天气类型。

```
select country_name,( case when avg(weather_state)<=15 then 'Cold'
                           when avg(weather_state)>=25 then 'Hot'
                           else 'Warm' end ) weather_type
from Countries c join Weather w on c.country_id = w.country_id
where date_format(day,"%Y-%m")='2019-11'
group by country_name
```

1303. 求团队人数

编写一个 SQL 查询，以求得每个员工所在团队的总人数。

```
select employee_id,count(*) over(partition by team_id) team_size
from Employee
```

1308. 不同性别每日分数总计

写一条SQL语句查询每种性别在每一天的总分，并按性别和日期对查询结果排序

```
select gender , day  ,sum( score_points) over(partition by gender order by day)
total
from Scores
```

1321. 餐馆营业额变化增长

写一条 SQL 查询计算以 7 天（某日期 + 该日期前的 6 天）为一个时间段的顾客消费平均值

```
select  visited_on,amount,round(amount/7,2) average_amount
from (
    select visited_on,ant,lag(visited_on,6,null) over(order by visited_on) lg,
            sum(ant) over(order by visited_on rows between 6 PRECEDING and
current row) amount
    from(
        select  visited_on  ,sum(amount) ant
        from Customer
        group by visited_on
    )t1
)t2
where lg is not null
```

1322. 广告效果

写一条SQL语句来查询每一条广告的 `ctr` ， `ctr` 要保留两位小数。结果需要按 `ctr` **降序**、按 `ad_id` **升序** 进行排序。

精简

```
SELECT ad_id,
    ROUND(IFNULL(SUM(action = 'Clicked') /
        (SUM(action = 'Clicked') + SUM(action = 'Viewed')) * 100, 0), 2) AS ctr
FROM Ads
GROUP BY ad_id
ORDER BY ctr DESC, ad_id ASC;
```

笨方法

```
select a.ad_id,ifnull(ctr,0) ctr
from Ads a
left join (
    select ad_id,round(sum(if(action='Clicked',1,0))/count(*)*100,2) ctr
    from Ads
    where action !='Ignored'
    group by ad_id
)t1
on a.ad_id= t1.ad_id
group by ad_id,ctr
order by ctr desc,ad_id
```

1327. 列出指定时间段内所有的下单产品

写一个 SQL 语句，要求获取在 2020 年 2 月份下单的数量不少于 100 的产品的名字和数目。

```
select product_name,sum(unit) unit
from Orders o left join Products p
on o.product_id=p.product_id
where date_format(order_date,'%Y-%m')='2020-02'
group by product_name
having unit>=100
```

1336. 每次访问的交易次数

写一条 SQL 查询多少客户访问了银行但没有进行任何交易，多少客户访问了银行进行了一次交易等等

```
SELECT *
FROM
(
    SELECT t5.rnb AS transactions_count, IFNULL(visits_count, 0) AS visits_count
    FROM
    (
        SELECT 0 AS rnb
        UNION
        SELECT ROW_NUMBER() OVER () AS rnb
        FROM Transactions
    ) t5
    LEFT JOIN
    (
        SELECT
            cnt AS transactions_count
            ,COUNT(user_id) AS visits_count
        FROM
        (
            SELECT t1.user_id, COUNT(t2.amount) AS cnt
```

```
            FROM Visits t1
            LEFT JOIN Transactions t2
            ON t1.user_id = t2.user_id AND t1.visit_date = t2.transaction_date
            GROUP BY user_id, visit_date
        ) t3
        GROUP BY cnt
    ) t4
    ON t5.rnb = t4.transactions_count
) t6
WHERE transactions_count <= (
    SELECT COUNT(t2.amount) AS cnt
    FROM Visits t1
    LEFT JOIN Transactions t2
    ON t1.user_id = t2.user_id AND t1.visit_date = t2.transaction_date
    GROUP BY t1.user_id, visit_date
    ORDER BY cnt DESC
    LIMIT 1)
```

> 难点 从0自增序列，2交易的人数为0

```
select pcnt transactions_count,count(*) visits_count
from (
select visit_date,
        sum(if(amount is  null,0,1)) over(partition by transaction_date ) pcnt,
        count(*) over(partition by  visit_date ) tcnt
from Visits v left join Transactions t
on v.user_id= t.user_id and v.visit_date=t.transaction_date
)t1
group by pcnt
```

1341. 电影评分

请你编写一组 SQL 查询：

- 查找评论电影数量最多的用户名。如果出现平局，返回字典序较小的用户名。
- 查找在 2020 年 2 月 平均评分最高的电影名称。如果出现平局，返回字典序较小的电影名称。

查询分两行返回，查询结果格式如下例所示：

```
select name results
from
(
    select m.user_id ,u.name
    from Movie_Rating m left join Users u
    on m.user_id = u.user_id
    group by user_id
    order by count(*) desc,name
    limit 1
)t1
union
(
select title results
from Movie_Rating r left join Movies m
on r.movie_id =m.movie_id
where date_format(created_at,'%Y-%m')='2020-02'
group by r.movie_id
order by avg(rating) desc,title
```

```
limit 1
)
```

1350. 院系无效的学生

写一条 SQL 语句以查询那些所在院系不存在的学生的 id 和姓名

```
select id,name
from Students
where department_id not in
(
    select id
    from Departments
)
```

1355. 活动参与者

写一条 SQL 查询那些既没有最多，也没有最少参与者的活动的名字

```
    select activity
    from (
        select activity,
        rank()over(order by cnt) rk1,
        rank()over(order by cnt desc) rk2
        from
        (
            select  activity  ,count(*) cnt
            from  Friends
            group by  activity
        )t1
    )t2
    where rk1 !=1 and rk2 != 1
```

1364. 顾客的可信联系人数量

```
select invoice_id ,customer_name,price,ifnull(cnt,0) contacts_cnt,ifnull(bc,0)
trusted_contacts_cnt
from Invoices i
left join (
select user_id ,count(*) cnt
from Contacts
group by user_id
) t1
on i.user_id=t1.user_id
left join (
select  user_id ,count(*) bc
from Contacts
    where contact_name in
    (
        select customer_name
        from Customers
    )
group by user_id
)t2
on i.user_id = t2.user_id
left join Customers c
```

```
on i.user_id= c.customer_id
order by invoice_id
```

1369. 获取最近第二次的活动

```
select username, activity ,startDate,endDate
from
(
select username, activity ,startDate,endDate ,
    rank()over(partition by username order by startDate desc) rk,
    lag( startDate ,1,null)over(partition by username order by startDate ) lg
from UserActivity
)t1
where rk=2 or  (rk = 1 &&  lg is null)
```

1378. 使用唯一标识码替换员工ID

```
select unique_id,e.name
from Employees  e left join EmployeeUNI u
on e.id = u.id
```

1384. 按年度列出销售总额

```
(
 select Sales.product_id, product_name, '2018' as 'report_year',
if(period_start<'2019-01-01', (datediff(if(period_end<'2019-01-01', period_end,
date('2018-12-31')), if(period_start>='2018-01-01', period_start, date('2018-01-
01')))+1)*average_daily_sales, 0) as total_amount
from Sales
join Product on Sales.product_id = Product.product_id
having  total_amount>0
)
union(
select Sales.product_id, product_name, '2019' as 'report_year', if(
period_start<'2020-01-01', (datediff(if(period_end<'2020-01-01', period_end,
date('2019-12-31')), if(period_start>='2019-01-01', period_start, date('2019-01-
01')))+1)*average_daily_sales , 0) as total_amount
from Sales
join Product on (Sales.product_id = Product.product_id )
having  total_amount>0
)
union(
select Sales.product_id, product_name, '2020' as 'report_year',
(datediff(if(period_end<'2021-01-01', period_end, date('2020-12-31')),
if(period_start>='2020-01-01', period_start, date('2020-01-
01')))+1)*average_daily_sales as total_amount
from Sales
join Product  on (Sales.product_id = Product.product_id)
having total_amount>0
)
order by product_id, report_year
```

1393. 股票的资本损益

```
select stock_name,sell-buy capital_gain_loss
from(
select stock_name ,
       sum(if(operation='Buy', price,0))over(partition by stock_name ) buy,
       sum(if(operation='Sell',price,0))over(partition by stock_name) sell
from Stocks s
)t1
group by stock_name,buy,sell
```

1398. 购买了产品A和产品B却没有购买产品C的顾客

```
select  o.customer_id, customer_name
from Orders o left join  Customers c
on o.customer_id=c.customer_id
group by customer_id
having sum(product_name ='A')>=1 and sum(product_name='B')>=1 and
sum(product_name='C')=0
```

1407. 排名靠前的旅行者

```
select name,sum(ifnull(distance,0)) travelled_distance
from Users u left join Rides r
on u.id = r.user_id
group by name
order by travelled_distance  desc, name
```

1412. 查找成绩处于中游的学生

```
select e.student_id,student_name
from Exam e left join Student s
on e.student_id=s.student_id
where e.student_id not in(
    select student_id
    from(
        select student_id,rank() over(partition by exam_id order by score desc)
rkmax, rank() over(partition by exam_id order by score ) rkmin
        from Exam
    )t1
    where rkmax = 1 or rkmin =1
)
group by e.student_id,student_name
order by e.student_id
```

1421. 净现值查询

```
select q.id,q.year,ifnull(npv,0) npv
from Queries q left join NPV n
on q.id = n.id and q.year = n.year
```

1435. 制作会话柱状图

Union

```
select '[0-5>' as bin, count(*) as total from Sessions where duration/60>=0 and
duration/60<5
union
select '[5-10>' as bin, count(*) as total from Sessions where duration/60>=5 and
duration/60<10
union
select '[10-15>' as bin, count(*) as total from Sessions where duration/60>=10
and duration/60<15
union
select '15 or more'as bin, count(*) as total from Sessions where duration/60>=15
```

还有很多其他解法

```
select a.bin, count(b.bin) as total
from
(
    select '[0-5>' as bin union select '[5-10>' as bin union select '[10-15>' as
bin union select '15 or more' as bin
)a
left join
(
    select case
        when duration < 300 then '[0-5>'
        when duration >= 300 and duration < 600 then '[5-10>'
        when duration >= 600 and duration < 900 then '[10-15>'
        else '15 or more'
        end bin
    from Sessions
)b
on a.bin = b.bin
group by a.bin
```

1440. 计算布尔表达式的值

```
select e.left_operand,e.operator,e.right_operand,
case e.operator
    when '>' then if(v1.value>v2.value,'true','false')
    when '<' then if(v1.value<v2.value,'true','false')
    else  if(v1.value=v2.value,'true','false')
end value
from Expressions e
left join Variables v1 on v1.name = e.left_operand
left join Variables v2 on v2.name = e.right_operand
```

1445. 苹果和桔子

写一个 SQL 查询, 报告每一天 **苹果** 和 **桔子** 销售的数目的差异.

```
select  sale_date,sold_num-ld diff
from
(
select sale_date,sold_num , fruit ,lead(sold_num ,1,null) over(partition by
sale_date ) ld
from Sales
)t1
where fruit='apples'
```

1454. 活跃用户

```
select t3.id,name
from
(
    select distinct id
    from
    (
        select id,login_date,lead(login_date,4,null) over(partition by id order
by login_date) ld
        from
        (
            select id,login_date
            from Logins
            group by id,login_date
        )t1
    )t2
    where datediff(ld,login_date)=4
)t3
left join Accounts a
on t3.id = a.id
```

1459. 矩形面积

```
select a.id P1,b.id P2,abs(a.x_value-b.x_value)*abs(a.y_value-b.y_value) as area
from Points a,Points b
where a.id<b.id and a.x_value != b.x_value and a.y_value != b.y_value
order by area desc,P1 ,P2
```

1468. 计算税后工资

```
select company_id,employee_id , employee_name,
round(case when maxsalary<1000 then salary
       when maxsalary<10000 then salary*(1-0.24)
       else salary*(1-0.49) end ,0)salary
from(
    select *,max(salary) over(partition by company_id ) maxsalary
    from Salaries
)t1
```

1479. 周内每天的销售情况

```
select item_category as category,
sum(case when num = 2 then quantity else 0 end) as Monday,
sum(case when num = 3 then quantity else 0 end) as Tuesday,
sum(case when num = 4 then quantity else 0 end) as Wednesday,
sum(case when num = 5 then quantity else 0 end) as Thursday,
sum(case when num = 6 then quantity else 0 end) as Friday,
sum(case when num = 7 then quantity else 0 end) as Saturday,
sum(case when num = 1 then quantity else 0 end) as Sunday
from
(select item_category, quantity,dayofweek(order_date) as num from
items i left join orders o
on i.item_id=o.item_id) t
group by item_category
order by item_category
```

1485. 按日期分组销售产品

```
select sell_date, count(distinct product) num_sold,
    group_concat(distinct product order by product) products
from Activities
group by sell_date
```

1495. 上月播放的儿童适宜电影

```
select distinct title
from TVProgram t left join Content c
on t.content_id  = c.content_id
where  Kids_content ='Y'
and date_format(program_date ,'%Y-%m')='2020-06'
and content_type='Movies'
```

1501. 可以放心投资的国家

笛卡尔积

```
select c2.name as country
from Calls c1,Person p,Country c2
where (p.id=c1.caller_id or p.id=c1.callee_id) and
c2.country_code=left(p.phone_number,3)
group by c2.name
having avg(duration)>(select avg(duration) from Calls)
```

思路更清晰

```
with people_country as
(
    select id, c.name country
    from Person p left join Country c
    on left(p.phone_number,3) = c.country_code
)

select country
```

```
from
(
    select country, avg(duration) avgtime
    from
    (
        select caller_id id, duration
        from Calls
        union all
        select callee_id, duration
        from Calls
    ) t left join people_country
    using(id)
    group by country
) temp
where avgtime >
    (
        select avg(duration) avgtime
        from
        (
            select caller_id, duration
            from Calls
            union all
            select callee_id, duration
            from Calls
        ) t
    )
```

1511. 消费者下单频率

写一个 SQL 语句, 报告消费者的 id 和名字, 其中消费者在 2020 年 6 月和 7 月, 每月至少花费了$100.

```
select customer_id,name
from Customers
where customer_id in
(select customer_id
    from
        (select customer_id, month(order_date) as month , sum(quantity*price) as
total
        from Orders o left join Product p on o.product_id = p.product_id
        where month(order_date) = 6 or month(order_date)=7
        group by customer_id,month(order_date)
        ) as t1
    where total >=100
    group by customer_id
    having count(*)>=2
)
```

1517. Find Users With Valid E-Mails

考察正则表达式的使用

```
SELECT *
FROM Users
WHERE mail REGEXP '^[a-zA-Z]+[\\w_\\.\\-]*@leetcode.com$'
ORDER BY user_id;
```

```
select * from Users
where mail regexp '^[a-zA-Z]+[a-zA-Z0-9_\\./\\-]{0,}@leetcode.com$'
order by user_id
```

1527. Patients With a Condition

```
select  patient_id , patient_name ,conditions
from Patients
where conditions like '%DIAB1%'
```

1532. The Most Recent Three Orders

```
select name customer_name ,customer_id,order_id,order_date
from (
select  name ,o.customer_id,order_id,order_date ,rank()over(partition by
o.customer_id order by order_date desc) rk
from Orders o left join Customers c
on o.customer_id=c.customer_id
)t1
where rk <=3
order by customer_name ,customer_id,order_date desc
```

1543. Fix Product Name Format

```
select trim(lower(product_name)) as product_name,
        date_format(sale_date,'%Y-%m') as sale_date,
        count(*) as total
from Sales
group by trim(lower(product_name)), date_format(sale_date,'%Y-%m')
order by product_name asc, sale_date asc
```

1549. The Most Recent Orders for Each Product

```
select product_name,product_id,order_id,order_date
from
(
select product_name ,o.product_id ,order_id,order_date ,
    rank() over(partition by o.product_id order by order_date desc) rk
from Orders o left join Products p
on o.product_id =p.product_id
)t1
where rk =1
order by product_name,product_id,order_id
```