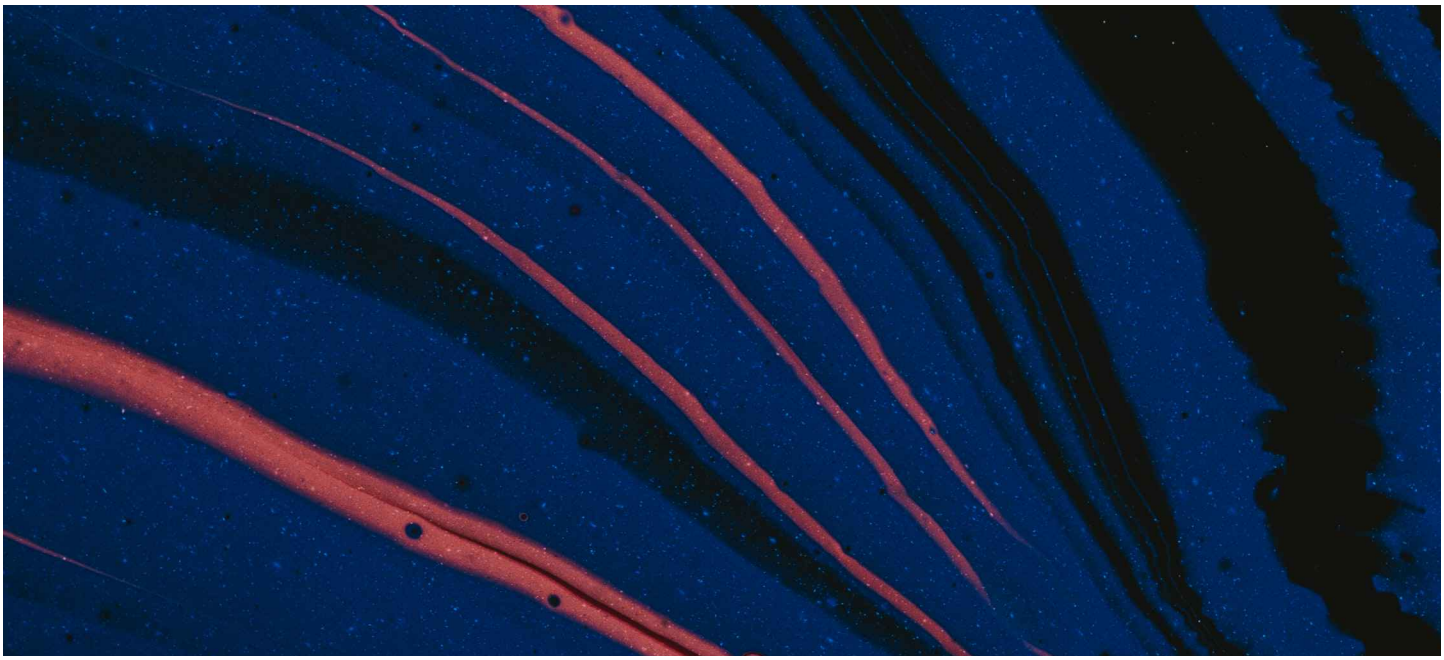


作业 锤子小行

DAY1（完整）

Question 1

1068. 产品销售分析 I

思路

通过product id连接两表

作答

```
SELECT Product.product_name, Sales.year, Sales.price
FROM Sales LEFT JOIN Product ON Sales.product_id = Product.product_id
```

用时

6min

反思

JOIN的写法不熟练

Question 2

1069. 产品销售分析 II

思路

用group by 合并id，sum一下加起来

作答

```
SELECT product_id, SUM(quantity) AS total_quantity
FROM Sales
GROUP BY product_id
```

用时

1min

Question 3

577. 员工奖金

思路

通过empId连接两表，null可能要处理一下

作答

```
SELECT Employee.name, Bonus.bonus
FROM Employee LEFT JOIN Bonus ON Employee.empId = Bonus.empId
WHERE Bonus.bonus<1000 OR Bonus.bonus IS NULL
```

用时

8min

反思

1. 题目没看清，要求是bouns<1000
2. 在处理join的时候犹豫了
3. 处理NULL不熟练，应该马上想到IS NULL

```
-- 复盘
SELECT t1.name, t2.bonus
FROM Employee t1 LEFT JOIN Bonus ON t1.empId = Bonus.empId
WHERE IFNULL(Bonus,0)<1000
```

Question 4

584. 寻找用户推荐人

思路

where符合条件就ok

作答

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

用时

2min

反思

1. 看清题目
2. 应该马上想到IS NULL

```sql
-- 复盘
SELECT name
FROM customer
WHERE IFNULL(referee_id,0) != 2
-- != or <>
```

Question 5

511. 游戏玩法分析 I

思路

用窗口函数，先对同一个player的event date进行排序，然后选出ranking最高的一个作为最早登录的日期

作答

```sql
SELECT player_id, event_date AS first_login
FROM(
    SELECT player_id, event_date,
        row_number() OVER (partition by player_id order by event_date ASC ) AS ran
    FROM Activity)a
WHERE ranking<=1
```

用时

18min

反思

1. 区分不熟

   ROW_NUMBER() (123456)

   RANK() (US NEWS 1114)

   DENSE_RANK() (1112)
2. order by不是group by，窗口函数不熟练
3. SELECT写窗口函数的那行之前一定记得加逗号啊
4. 好像这个方法比较麻烦，下面的方法更好

```sql
-- 复盘
SELECT player_id, min(event_date) AS first_login
FROM activity
GROUP BY player_id
```

Question 6

512. 游戏玩法分析 II

思路

用窗口函数和上题思路一样

作答

```sql
SELECT player_id, device_id
```

```
FROM(
    SELECT player_id, device_id,
    row_number() over(partition by player_id order by event_date) as ranking
    FROM Activity
)a
WHERE ranking<=1
```

用时

2min

反思

1. 掌握了上面的注意事项很快写出了思路
2. 还是要注意熟练代码完整性，争取一次编译通过

```
-- 复盘
-- 窗口函数
SELECT player_id, device_id
FROM(
    SELECT player_id, device_id,
    dense_rank() over(partition by player_id order by event_date ASC) as rnk
    FROM Activity
)a
WHERE rnk=1
-- 子查询

event_date = min(event_date)

SELECT player_id, device_id
FROM activity
WHERE (player_id,event_date) IN(
    SELECT player_id, min(event_date)
    FROM activity
    GROUP BY player_id
    )
```

错误示例

```sql
select
    player_id, device_id
from activity
where event_date in
(
    select min(event_date)
    from activity
    group by player_id
)
```

player_id和event_date是联合主键

```sql
select
    player_id,device_id
from
(select
    player_id, device_id, min(event_date)
from activity
group by player_id) t
```

并没有group by device_id，所以只会显示每个group的第一行的device_id

Question 7

534. 游戏玩法分析 III

思路

需要对游戏的数量进行一个累加

作答

```sql
SELECT player_id, event_date, games_played_so_far
FROM(
    SELECT player_id, event_date,
        SUM(games_played) OVER (partition by player_id order by event_date ASC ) a
    FROM Activity
    )a
```

用时

10min

反思

1. 其实一开始就写对了
2. 后面自己多想了好多步骤以为题目也问了，假如要再进一步，只保留累积最大值的时候该怎么办呢?

```sql
-- 复盘
-- 如果不让窗口函数，可以用 Cartesian product
SELECT player_id, event_date, sum(t2.games_played) as games_played_so_far
FROM Activity t1, Activity t2
WHERE t1.player_id = t2.player_id AND t1.event_date>=t2.event_date
GROUP BY t1.player_id, t1.event_date
```

```
["player_id", "event_date", "games_played", "player_id", "event_date", "games_played"]
[[1, "2016-03-01", 5, 1, "2016-03-01", 5],
[1, "2016-05-02", 6, 1, "2016-03-01", 5],
[1, "2016-05-02", 6, 1, "2016-05-02", 6],
[1, "2017-06-25", 1, 1, "2016-03-01", 5],
[1, "2017-06-25", 1, 1, "2016-05-02", 6],
[1, "2017-06-25", 1, 1, "2017-06-25", 1],
[3, "2016-03-02", 0, 3, "2016-03-02", 0],
[3, "2018-07-03", 5, 3, "2016-03-02", 0],
[3, "2018-07-03", 5, 3, "2018-07-03", 5]]
```

5
11
12
0
5

Question 8

550. 游戏玩法分析 IV

思路

用窗口函数，但是有个问题就是第二天再次登录如何表示

要求fraction的话就要求玩家数量和总数

作答

```sql
-- 没写出来
SELECT COUNT(player_id)/COUNT(DISTINCT player_id) AS fraction
FROM(
    SELECT DISTINCT event_date, player_id
    FROM(
        SELECT player_id,
            row_number() over(partition by player_id order by event_date) AS rank
        FROM Activity
        )a
    WHERE ranking<=2
)a
# WHERE DATEDIFF(event_date) = 1
```

用时

>10min

反思

1. 查询是否为空
2. DATE的处理方式及查询方式where in
3. ROUND的用法

```sql
-- 复盘
-- 子查询
SELECT
    IFNULL(
    ROUND(COUNT(distinct player_id)/ (SELECT COUNT(distinct player_id) FROM activ
FROM activity
WHERE (player_id, event_date) IN
(
    SELECT player_id, DATE(min(event_date) + 1) AS second_login
    FROM activity
    GROUP BY player_id
)a
```

Question 9

570. 至少有5名直接下属的经理

思路

连接

作答

```sql
SELECT A.name
FROM Employee AS A, Employee AS B
WHERE A.Id = B.managerId
GROUP BY B.managerId
HAVING COUNT(*) >= 5
```

```sql
SELECT A.name
FROM Employee A left join Employee B on A.Id = B.ManagerId
GROUP BY A.Id
HAVING count(*)>=5
```

用时

7min

反思

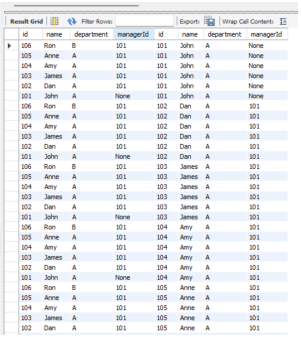
1. JOIN还是不太清晰，做了些test case

```
-- CASE 1
SELECT *
FROM  Employee AS A LEFT JOIN Employee AS B on A.Id = B.ManagerId
-- GROUP BY A.Id
-- HAVING count(*)>=5
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	name	department	managerId	id	name	department	managerId
101	John	A	None	106	Ron	B	101
101	John	A	None	105	Anne	A	101
101	John	A	None	104	Amy	A	101
101	John	A	None	103	James	A	101
101	John	A	None	102	Dan	A	101
102	Dan	A	101	NULL	NULL	NULL	NULL
103	James	A	101	NULL	NULL	NULL	NULL
104	Amy	A	101	NULL	NULL	NULL	NULL
105	Anne	A	101	NULL	NULL	NULL	NULL
106	Ron	B	101	NULL	NULL	NULL	NULL

```
-- CASE 2
SELECT *
FROM Employee AS A, Employee AS B
WHERE A.Id = B.managerI
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	name	department	managerId	id	name	department	managerId
101	John	A	None	102	Dan	A	101
101	John	A	None	103	James	A	101
101	John	A	None	104	Amy	A	101
101	John	A	None	105	Anne	A	101
101	John	A	None	106	Ron	B	101

问题：如果是这样子该怎么写？

```
FROM  employee.manager AS A LEFT JOIN employee.manager AS B on A.managerId = B.id
```

```
-- 复盘
-- 子查询匹配
SELECT name
```

```
FROM Employee
WHERE Id IN
(
    SELECT DISTINCT managerId
    FROM Employee
    GROUP BY managerId
    HAVING COUNT(mangaerId) >= 5
 )a
```

Question 10

569. 员工薪水中位数

思路

涉及group by 以及处理中位数，如何比较

作答

```
SELECT id, company, salary
FROM Employee
WHERE id in (
    SELECT e1.Id
    FROM Employee e1, Employee e2
    WHERE e1.Company = e2.Company
    GROUP BY e1.Id
    HAVING SUM(CASE WHEN e1.Salary >= e2.Salary THEN 1 ELSE 0 END) >= COUNT(*)/2
    AND SUM(CASE WHEN e1.Salary <= e2.Salary THEN 1 ELSE 0 END) >= COUNT(*)/2)
GROUP BY Company, Salary
ORDER BY Company
```

用时

》10min

反思

1. 运用CASE表达式，非等值自连接和HAVING子句来找中位数
2. 通过 `WHERE e1.Company = e2.Company` 进行分组
3. 最后通过GROUP BY 去重

```
--复盘
```

```
SELECT id, company, salary
FROM
(
    SELECT a.*,
           row_number() over (partition by Company order by Salary) as rnk,
           count(Salary) over (partition by Company) as cnt
    FROM Employee AS a
)a
WHERE rnk in (cnt/2, cnt/2+1,(cnt+1)/2)
```

Question 11

571. 给定数字的频率查询中位数

思路

作答

```
SELECT AVG(num) AS median
FROM(
    SELECT  num,
            SUM(frequency) OVER (order by num ASC) as asc_amount,
            SUM(frequency) OVER (order by num DESC) as desc_amount,
            SUM(frequency) OVER () as total_num
    FROM Numbers
    ) a
WHERE asc_amount >= total_num/2 and desc_amount >= total_num / 2
```

用时

>10min

反思

理解窗口函数+aggregate的作用

```
select
    round(avg(num),1) as median
from
(select
    a.*,
    sum(frequency) over(order by num) as rnk1,
    sum(frequency) over(order by num desc) as rnk2,
```

```
    sum(frequency) over() as s
from Numbers a) tmp
where rnk1>=s/2 and rnk2>=s/2
```

DAY2（完整）

Question 1

586. 订单最多的客户

思路

通过窗口函数，对order number 进行count，算哪个customer number最多

作答

```
SELECT customer_number
FROM(
    SELECT customer_number,
    dense_rank() over (order by count(order_number) DESC ) as rnk
    FROM Orders
    GROUP BY customer_number
)a
WHERE rnk = 1
```

用时

10min （审题2+调试8）

反思

搞清楚 count的是order_number, 且要DESC

```
-- 复盘
-- 方法 1， 有局限性
-- order by count
select customer_number
from Orders
group by customer_number
order by count(order_number) desc  -- 这里只适用于本题，不能出来多个第一
limit 1
```

```sql


-- 方法2， 窗口函数，构造新列，普适性
select customer_number from
(select
    customer_number,
    -- 组间排序，对于customer number中 1，2，3 分别的 order number的数量排序
    dense_rank() over(order by count(order_number) desc) as ranking
    -- 窗口函数适用范围广，在数据层面先构造再筛选
from orders
group by customer_number) t
where ranking = 1

-- 方法 3 子查询匹配
select customer_number
from Orders
group by customer_number
having count(*) >=
all(select count(*)
    from orders
    group by customer_number)
   -- all 满足所有的条件就成立 AND
   -- any 满足一个条件就成立 OR

   -- 这里主要如果orders是一个复杂的table，建议用cte封装一下
   -- ?
```

Question 2

1075. 项目员工 I

思路

用GROUP BY把每个项目合并，链接另一个表，求得员工平均工作年数

作答

```sql
SELECT A.project_id, ROUND(AVG(B.experience_years),2) AS average_years
FROM Project AS A LEFT JOIN Employee AS B ON A.employee_id = B.employee_id
GROUP BY A.project_id
```

用时

6min （审题2+调试4）

反思

看清是average_years不是average_year

```sql
-- 复盘
SELECT A.project_id, ROUND(AVG(B.experience_years),2) AS average_years
FROM Project JOIN Employee USING employee_id
GROUP BY project_id
```

Question 3

1076. 项目员工II

思路

和上上题一个意思

作答

```sql
SELECT project_id
FROM(
    SELECT project_id,
    dense_rank() over (order by count(employee_id) DESC ) as rnk
    FROM Project
    GROUP BY project_id
)a
WHERE rnk=1
```

用时

4min （审题2+调试2）

反思

构造新列做为外层的筛选，属于窗口函数 order by count题型

```sql
-- 方法1 - 子查询匹配
SELECT project_id
FROM project
GROUP BY project_id
HAVING count(employee_id ) >=
ALL(
    SELECT count(employee_id)
    FROM project GROUP BY project_id
   )
```

```sql


-- 方法2 - 窗口函数
SELECT project_id
FROM(
    SELECT project_id, dense_rank() over(order by count(employee_id) desc) as ran
        -- 窗口函数适用范围广，在数据层面先构造再筛选
    FROM project
    GROUP BY project_id
    ) t
WHERE ranking = 1
```

Question 4

1077. 项目员工 III

思路

两边left join，用rank窗口函数DESC求出最有经验的人

作答

```sql
SELECT project_id, employee_id
FROM(
    SELECT  P.project_id, P.employee_id,
    rank() over (partition by P.project_id order by E.experience_years DESC ) as r
    FROM Project AS P LEFT JOIN Employee AS E ON P.employee_id = E.employee_id
)a
WHERE rnk = 1
```

用时

5min

反思

还是要看好名字，减少编译次数

```sql
-- 方法1 - 链接2表
WITH t AS(
    SELECT a.project_id, a.employee_id, b.experience_years
FROM project a JOIN employee b
USING(employee_id)
)

```

```sql
-- 方法2 - 子查询匹配
SELECT project_id,employee_id
FROM t
WHERE (project_id, experience_years) IN(
    SELECT project_id, max(experience_years)
    FROM t
    GROUP BY 1)

-- 方法3 -  窗口函数
SELECT project_id,employee_id
FROM(
    SELECT project_id, employee_id,
    dense_rank() over(partition by project_id order by experience_years desc) as
    FROM t) tmp
WHERE rnk =1


-- 注意联合主键

-- 两种排序
-- 组内排序 - 窗口 partition by vs 组查询 where in
-- 组间排序 - Group by vs having >= all
```

Question 5

619. 只出现一次的最大数字

思路

首先考虑到是最大的且是单一是数字，第二考虑的是要报告null值

作答

```sql
SELECT IFNULL(
    (SELECT num
    FROM MyNumbers
    GROUP BY num
    HAVING count(num) = 1
    ORDER BY num DESC LIMIT 1), null) as num
```

用时

7min

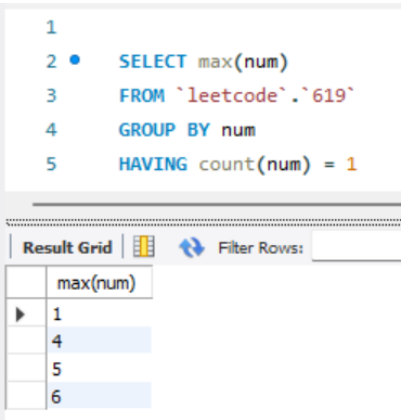
反思

LIMIT 1选定只返回一行

先找单一数字（group by having count），再找最大的

```sql
-- 复盘
SELECT max(num) AS num
FROM(
      SELECT num
      FROM MyNumbers
      GROUP BY num
      HAVING count(num) = 1
) t

-- 为什么不能在 t表内部直接写 max(num) 呢?
-- 因为此时选中的是每一个 group 的 max，结果为1 4 5 6，与 t 表一致
```



Question 6

1141. 查询近30天活跃用户数

思路

需要用datediff函数找出小于30的天数

作答

```sql
SELECT activity_date AS day, count(DISTINCT user_id) AS active_users
FROM activity
WHERE datediff('2019-07-27', activity_date) >= 0 AND
      datediff('2019-07-27', activity_date) < 30
GROUP BY activity_date
```

```
-- 会筛选出负数  所以要大于等于0
```

用时

>10min

反思

1. 需要注意distinct去重
2. 需要注意既要大于等于0又要小于30

Question 7

574. 当选者

思路

两个表合并，用order by count DESC 选出票最高的

作答

```
SELECT C.name
FROM Candidate AS C JOIN Vote AS V ON C.id = V.CandidateId
GROUP BY V.CandidateId
ORDER BY count(V.id) DESC
LIMIT 1
```

用时

>10min

反思

1. 对于此类表的合并比较迷。
2. 然后order by count没有理解

```
-- 方法1 窗口函数
SELECT name
FROM (
    SELECT name, dense_rank() over(order by count(b.id) desc) as rnk
    FROM Candidate a JOIN Vote b ON a.id = b.candidateid
    GROUP BY candidateid) t
WHERE rnk=1

```

```sql
-- 方法2 order by count
SELECT name
FROM Candidate a JOIN Vote b ON a.id = b.candidateid
GROUP BY candidateid
ORDER BY count(b.id) DESC LIMIT 1

-- 方法3 having count
SELECT name
FROM candidate as c right join vote as v on c.id=v.candidateid
GROUP BY v.candidateid
HAVING count(*)>=all(
    SELECT count(*)
    FROM vote
    GROUP BY candidateid)
```

Question 8

1107. 每日新用户统计

思路

查询activity是登录的用户，group by user id，算出最小的login date，然后看是否是在90天内的，最后count一下user id的个数

作答

```sql
SELECT login_date, COUNT(user_id) AS user_count
FROM(
    SELECT user_id, min(activity_date) AS login_date
    FROM Traffic
    WHERE activity = 'login'
    GROUP BY user_id
)a
WHERE datediff('2019-06-30', a.login_date) <= 90
GROUP BY a.login_date
```

用时

>10min

反思

注意思考的层次问题，要考虑清楚

```sql
 SELECT first_date as login_date, count(user_id) as user_count
```

```
 FROM(
     SELECT user_id, min(activity_date) as first_date
     FROM Traffic
     WHERE activity = 'login'
     GROUP BY user_id) tmp
WHERE datediff("2019-06-30", first_date) <= 90
GROUP BY 1
-- 为什么user_id不是主键不用 distinct?
-- 已经 Group by 过了
```

Question 9

578. 查询回答率最高的问题

思路

先group by question id，然后按照回答率顺序排序，DESC， 选第一行

作答

```
SELECT question_id AS survey_log
FROM Surveylog
GROUP BY question_id
ORDER BY sum(action = 'answer') / sum(action = 'show') DESC, question_id
LIMIT 1
```

用时

>10min

反思

读题的理解时间比较久，要理解回答率的含义

```
-- 方法1
SELECT question_id AS survey_log
FROM Surveylog
GROUP BY question_id
ORDER BY sum(action = 'answer') / sum(action = 'show') DESC, question_id
LIMIT 1
-- sum是求和 count是计数，求行

-- 方法2
SELECT question_id as survey_log
```

```
FROM(
    SELECT a.*,
    dense_rank() over (order by sum(action = 'answer')/sum(action = 'show') desc,
    FROM SurveyLog a
    GROUP BY question_id) t
WHERE rnk = 1
```

Question 10

579. 查询员工的累计薪水

思路

用窗口函数查询累计的salary，难点在于如何表示最近一个月之外的薪水。

作答

```
SELECT id,
        month,
        sum(salary) over(partition by id order by month range 2 preceding) as 'sa
FROM Employee
WHERE (id, month) NOT IN
(
    SELECT id, max(month)
    FROM Employee AS e
    GROUP BY id
    )
ORDER BY id, month DESC
```

用时

>10min

反思

1. 注意理解rows和range的区别

"rows 2 PRECEDING"计算的是3,4,7月份的工资

"range 2 PRECEDING"计算的是5,6,7月份的工资

range是逻辑窗口，是指定当前行对应值的范围取值，列数不固定，只要行值在范围内，对应列都包含在内

rows是物理窗口，即根据order by 子句排序后，取的前N行及后N行的数据计算（与当前行的值无关，只与排序后的行号相关）

```
SELECT
  a.Id AS id, a.Month AS month,SUM(b.Salary) AS Salary
FROM Employee a, Employee b
WHERE a.Id = b.Id AND a.Month >= b.Month AND a.Month < b.Month+3
AND (a.Id, a.Month) NOT IN (
    SELECT Id, MAX(Month)
    FROM Employee
    GROUP BY Id)
GROUP BY a.Id, a.Month
ORDER BY a.Id, a.Month DESC
```

DAY3（完整）

Question 1

603. 连续空余座位

思路

用自连接，思考如何表示0->1的转变

作答

```
SELECT DISTINCT A.seat_id
FROM Cinema AS A, Cinema AS B
WHERE abs(A.seat_id - B.seat_id) = 1 and A.free = 1 and B.free = 1
ORDER BY 1 ASC
```

用时

>10min

反思

1. 搞不清楚自连接，需要看表
2. 需要考虑到distinct

```sql
SELECT DISTINCT (A.seat_id)
FROM Cinema AS A, Cinema AS B
WHERE abs(A.seat_id - B.seat_id) = 1 and A.free = 1 and B.free = 1
ORDER BY a.seat_id
```

Question 2

610. 判断三角形

思路

两边之和大于第三边，用case when， if yes then yes, no otherwise

作答

```sql
SELECT a.*,
    CASE WHEN x+y>z and x+z>y and y+z>x THEN "Yes"
    ELSE "No"
    END AS triangle
FROM Triangle AS a
```

用时

2min

反思

主要考察CASE WHEN 函数的运用

Question 3

613. 直线上的最近距离

思路

用子连接，算最短距离

作答

```sql
SELECT min(abs(A.x-B.x)) as shortest
FROM point as A, point as B
WHERE A.x != B.x
```

用时

5min

反思

注意，自己和自己的距离就是0了不能算

Question 4

1050. 合作过至少三次的演员和导演

思路

Group by having count>=3

作答

```
SELECT actor_id, director_id
FROM ActorDirector
GROUP BY actor_id,director_id
HAVING COUNT(*)>=3
```

用时

2min

反思

注意，同时group by两个列的时候别加括号，Operand should contain 1 column(s)

```
sum(if(a='A',1,0))
=
sum(a='A')
=
count(if(a='A',1,null))
```

Question 5

1082. 销售分析 I

思路

用窗口函数，order by sum，然后group by 取rank

作答

```
SELECT seller_id
FROM(
    SELECT seller_id,
    DENSE_RANK() over (order by sum(price) DESC ) as rnk
    FROM Sales
    GROUP BY seller_id
)a
WHERE rnk = 1
```

用时

5min

反思

注意，partition by XXX order by， 是分成组间，然后组内自己排名

```
SELECT seller_id
FROM Sales
GROUP BY seller_id
HAVING sum(price)>= all(
    SELECT sum(price)
    FROM Sales
    GROUP BY seller_id
    )
```

Question 6

1083. 销售分析 II

思路

用子查询

作答

```
SELECT DISTINCT buyer_id
FROM Product AS A LEFT JOIN Sales AS B ON A.Product_id = B.Product_id
WHERE A.Product_name ='S8'
```

```
AND buyer_id NOT IN(
    SELECT buyer_id
    FROM Product AS A LEFT JOIN Sales AS B ON A.Product_id = B.Product_id
    WHERE A.Product_name = 'iPhone'
)
```

用时

>10min

反思

1. 反复调试主要要考虑的是子查询中要查的是什么, 是buyer_id，手机只是条件
2. 搞清楚要哪个手机不要哪个手机

```
-- 方法一 sum
select b.buyer_id
from Product a
join Sales b using(product_id)
group by buyer_id
having sum(product_name = 'S8')>0 and sum(product_name = 'iphone') = 0

-- 方法二 count
select s.buyer_id
from product p, sales s
where p.product_id = s.product_id
group by s.buyer_id
having count(if(p.product_name='S8',1,null)) > 0
and count(if(p.product_name='iPhone',1,null)) < 1
```

> sum if 和 count if 的区别和注意事项：
>
> - sum if 对出现次数求和（出现一次计1，加起来得到该条件共出现几次）
> - count if 对非null情况全部计数，所以在不满足条件时，一定要定义为null，才能避免count对其计数。

Question 7

1084. 销售分析III

思路

子查询

作答

```sql
SELECT product_id, product_name
FROM product
WHERE product_id IN
(
    select distinct product_id
    from sales
) AND product_id NOT IN
(
    SELECT product_id
    FROM sales
    WHERE datediff(sale_date, '2019-01-01') < 0 or datediff(sale_date, '2019-03-3
)
```

用时

5min

反思

理解datediff的含义，not in 相反区间

```sql
SELECT
    p.product_id,
    s.product_name
FROM sales s,product p
WHERE s.product_id=p.product_id
GROUP BY p.product_id
HAVING SUM(sale_date < '2019-01-01')=0
AND SUM(sale_date>'2019-03-31')=0;

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)
```

Question 8

612. 平面上的最近距离

思路

用自连接，找出距离并排序，用round保留2位小数

作答

```
SELECT ROUND(min(sqrt(pow(B.x-A.x,2)+pow(B.y-A.y,2))),2) AS shortest
FROM Point2D as A, Point2D as B
WHERE (A.x,A.y) != (B.x,B.y)
```

用时

>10min

反思

和上面的613类似，但是不知道为什么下面的写法不对

```
WHERE A.x!=B.x AND A.y!=B.y
```

Question 9

180. 连续出现的数字

思路

用聚合窗口函数

作答

```
select
    distinct num as ConsecutiveNums
from
(select
    num,
    row_number() over(order by id) -
    row_number() over(partition by num order by id) as rnk_diff
from Logs) t
group by num,rnk_diff
having count(*) >= 3
```

用时

>10min

反思

没有做出来,思路是通过rnk_diff, count>=3的就是连续的

Question 10

1285. 找到连续区间的开始和结束数字

思路

窗口函数

作答

```sql
SELECT min(log_id) AS start_id, max(log_id) AS end_id
FROM(
    SELECT log_id,
    rank() over(ORDER BY log_id asc) AS rnk
    FROM logs
    ) a
GROUP BY (log_id-rnk)
```

用时

5min

反思

运用log_id-rank的group形式可以区分是否连续

```sql
SELECT min(log_id) AS start_id, max(log_id) AS end_id
FROM(
    SELECT log_id,
    log_id - row_number() over(ORDER BY log_id asc) AS diff
    FROM logs
    ) a
GROUP BY diff
```

Question 11

580. 统计各专业学生人数

思路

ifnull和left join即可

作答

```
select d.dept_name dept_name, ifnull(count(student_name),0) student_number
from department d left join student s on d.dept_id = s.dept_id
group by d.dept_id
order by student_number desc, dept_name
```

用时

5min

反思

注意有的部门是没有学生的

```
select d.dept_name, ifnull(count(student_id),0) as student_number
from Department d
left join Student s
using(dept_id)
group by 1
order by 2 desc,1
```

Question 12

601. 体育馆的人流量

思路

没做出来

作答

```
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)
    or
    (t2.id - t1.id = 1 and t2.id - t3.id = 2 and t1.id - t3.id =1)
    or
    (t3.id - t2.id = 1 and t2.id - t1.id =1 and t3.id - t1.id = 2)
)
order by t1.id
```

用时

>10min

反思

```
with cte as(
    select a.*, id - row_number() over(order by id) as rnk_diff
from Stadium a
where people >= 100
)

select id, visit_date,people
from cte
where rnk_diff in(
    select rnk_diff
    from cte
    group by rnk_diff
    having count(id)>=3
)
order by visit_date
```

Question 13

1225. 报告系统状态的连续日期

思路

窗口函数排序，然后where日期的区间

作答

```
select state period_state, min(date) start_date, max(date) end_date
from (
    select *,
        row_number() over (partition by state order by date asc) rk1,
        row_number() over (order by date asc) rk2
```

```sql
    from (
        select fail_date 'date', 'failed' state from failed
        union all select success_date, 'succeeded' from succeeded
    ) t
) t2
where date between '2019-01-01' and '2019-12-31'
group by state, rk2-rk1
```

用时

>10min

反思

```sql
-- 打标签
with cte as
(select fail_date as date, 'failed' as period_state
from Failed
union
select success_date, 'succeeded'
from Succeeded
order by 1)

select period_state, min(date) as start_date, max(date) as end_date
from(
    select period_state, date,
    row_number() over(order by date) -
    row_number() over(partition by period_state order by date) as rnk_diff
    from cte
    where date between '2019-01-01' and '2019-12-31'
    ) t
group by period_state, rnk_diff -- 不同period_state时rnk_diff一样
order by 2
```