DAY 2

题目总结：

586、1077、574、578的题目方法都是一样的，都有三种方法。将586做透，另三道题就可以当练习了。

586. 订单最多的客户

表: `Orders`

Column Name	Type
order_number	int
customer_number	int

order_number是该表的主键。
此表包含关于订单ID和客户ID的信息。

编写一个SQL查询，为下了 最多订单 的客户查找 customer_number 。

测试用例生成后， 恰好有一个客户 比任何其他客户下了更多的订单。

查询结果格式如下所示。

示例 1:

输入:
`Orders` 表:

order_number	customer_number
1	1
2	2
3	3
4	3

输出:

customer_number
3

解释:
customer_number 为 '3' 的顾客有两个订单，比顾客 '1' 或者 '2' 都要多，因为他们只有一个订单。
所以结果是该顾客的 customer_number ，也就是 3 。

进阶： 如果有多位顾客订单数并列最多，你能找到他们所有的 customer_number 吗?

思路

1. 从数据入手，排序rank，**这里一定要注意区分group by 和partition by**，本题是group by
2. 从条件入手，**注意order by不一定是某个字段，也可以是聚合函数的计算结果。**

   group by customer_number　　order by count(order_number) desc　　limit 1
3. 从条件入手，group by ... having count(order_number) >=all(order_number)

算法

1. dense_rank() over(order by count(order_number) desc) as rnk

   ...

   where rnk = 1
2. group by customer_number
   order by count(order_number) desc
   limit 1
3. having count(order_number) >=all(...)

代码

```
# 方法1：注意partitioin by和group by的区别，注意倒序desc
# 正确写法:
select
    customer_number
from
(select
    customer_number
```

```sql
    , dense_rank() over(order by count(order_number) desc) as rnk
from Orders
group by customer_number) t
where rnk = 1
# 方法1的错误写法:
# select
#     customer_number
# from
# (select
#     customer_number
#     , dense_rank() over(partition by customer_number order by count(order_number)
desc) as rnk
# from Orders) t
# where rnk = 1

# 方法2，只能选出来最大的，并列情况无法判断
select
    customer_number
from Orders
group by customer_number
order by count(order_number) desc
limit 1

# 方法3，这种不好的地方：Orders可能很复杂，此时就cte吧
select
    customer_number
from Orders
group by customer_number
having count(order_number) >=
all(
    select count(order_number)
    from Orders
    group by customer_number)
# 方法3的错误写法，需要注意where 和having
# select
#     customer_number
# from Orders
# where count(order_number)>=
# all(
#     select
#         count(order_number)
#     from Orders
#     group by customer_number
# )
# group by customer_number
```

619. 只出现一次的最大数字

`MyNumbers` 表：

Column Name	Type
num	int

这张表没有主键。可能包含重复数字。
这张表的每一行都含有一个整数。
单一数字 是在 MyNumbers 表中只出现一次的数字。
请你编写一个 SQL 查询来报告最大的 单一数字 。如果不存在 单一数字 ， 查询需报告 null 。
查询结果如下例所示。
示例 1：
输入：
`MyNumbers` 表：

num
8
8
3
3
1
4
5
6

输出：

num
6

解释：单一数字有 1、4、5 和 6 。
6 是最大的单一数字，返回 6 。
示例 2：
输入：
`MyNumbers` table:

num
8
8
7
7
3
3
3

输出：

num
null
解释：输入的表中不存在单一数字，所以返回 null 。

思路：先找单一的，再找最大的，即使只有1列也可以group by，千万别忘别名

代码：

```
# 对的
select
    max(num) as num # 这里千万别忘别名
from
(   select
        num
    from MyNumbers
    group by num
    having count(num) = 1) tmp

# 错的，这样是组内最大值了
# select
#     max(num) as num

# from MyNumbers
# group by num
# having count(num) = 1
```

1075. 项目员工 I

项目表 `Project`：

Column Name	Type
project_id	int
employee_id	int

主键为 (project_id, employee_id)。
employee_id 是员工表 Employee 表的外键。
员工表 `Employee`：

Column Name	Type
employee_id	int
name	varchar
experience_years	int

主键是 employee_id。
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。
查询结果的格式如下：
`Project` 表：

project_id	employee_id
1	1
1	2
1	3
2	1
2	4

`Employee` 表：

employee_id	name	experience_years
1	Khaled	3
2	Ali	2
3	John	1
4	Doe	2

`Result` 表：

project_id	average_years
1	2.00
2	2.50

第一个项目中，员工的平均工作年限是 (3 + 2 + 1) / 3 = 2.00；第二个项目中，员工的平均工作年限是 (3 + 2) / 2 = 2.50

思路

1. 先看主键，注意 `Project` 表中的主键是联合主键 (project_id, employee_id)，考虑如何连接，会不会出现null问题；
2. **注意精确到小数点后两位。**

代码

```
select
    a.project_id
    , round(avg(b.experience_years),2) as average_years
from Project a
left join Employee b
on a.employee_id = b.employee_id
group by a.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。
查询结果格式如下所示：
`Project` table:

project_id	employee_id
1	1
1	2
1	3
2	1
2	4

`Employee` table:

employee_id	name	experience_years
1	Khaled	3
2	Ali	2
3	John	1
4	Doe	2

`Result` table:

project_id
1

第一个项目有3名员工，第二个项目有2名员工。

思路

同586题

代码

```
# 方法1:
select
    project_id
from Project
group by project_id
having count(employee_id)>=
all(
    select
        count(employee_id)
    from Project
    group by project_id
)

# 方法2:
select
    project_id
from
(
    select
      project_id
      , dense_rank() over(order by count(employee_id) desc) as rnk
    from Project
    group by project_id
) as tmp
where rnk = 1
```

```
# 方法3:(此题不行，因为会有并列)
# select
#     project_id
# from Project
# group by project_id
# order by count(employee_id) desc
# limit 1
```

1077. 项目员工 III【中等】

难度：中等

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。

查询结果格式在以下示例中：

`Project` 表：

project_id	employee_id
1	1
1	2
1	3
2	1
2	4

`Employee` 表：

employee_id	name	experience_years
1	Khaled	3
2	Ali	2
3	John	3
4	Doe	2

`Result` 表：

project_id	employee_id
1	1
1	3
2	1

employee_id 为 1 和 3 的员工在 project_id 为 1 的项目中拥有最丰富的经验。在 project_id 为 2 的项目中，employee_id 为 1 的员工拥有最丰富的经验。

```
# 思路1 ，条件入手，先连接，然后找到每个project_id中最大的experience_years对应的employee_id
# 1. join
# 2. max ... group by...
# 3. where in
# 正确写法:
with cte as(
    select
        a.project_id
        , a.employee_id
        , b.experience_years
    from Project as a
    left join Employee as b
    on a.employee_id = b.employee_id
)

select
    project_id
    , employee_id
from cte
where (project_id, experience_years) in
(
    select
    project_id
    , max(experience_years)
    from cte
    group by project_id
)

# 错误写法：(cte部分一致)
# select
#     project_id
#     , employee_id
# from Project
# where (project_id, employee_id) in
# (
#     select
#     project_id
#     , employee_id
#     from cte
#     group by project_id
#     having max(experience_years)
# )


# 思路2，利用窗口函数构造新列，组间排序
select
```

```
    project_id
    , employee_id
from
(select
    a.project_id
    , a.employee_id
    , dense_rank() over(partition by a.project_id order by b.experience_years desc) as
rnk
from Project a
left join Employee b
on a.employee_id = b.employee_id) as tmp
where rnk = 1
```

1141. 查询近30天活跃用户数

难度：简单

活动记录表：`Activity`

Column Name	Type
user_id	int
session_id	int
activity_date	date
activity_type	enum

该表是用户在社交网站的活动记录。
该表没有主键，可能包含重复数据。
activity_type 字段为以下四种值 ('open_session', 'end_session', 'scroll_down', 'send_message')。
每个 session_id 只属于一个用户。

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）。以 任意顺序 返回结果表。

查询结果示例如下

示例 1:

输入：
`Activity` table:

user_id	session_id	activity_date	activity_type
1	1	2019-07-20	open_session
1	1	2019-07-20	scroll_down
1	1	2019-07-20	end_session
2	4	2019-07-20	open_session
2	4	2019-07-21	send_message
2	4	2019-07-21	end_session
3	2	2019-07-21	open_session
3	2	2019-07-21	send_message
3	2	2019-07-21	end_session
4	3	2019-06-25	open_session
4	3	2019-06-25	end_session

输出：

day	active_users
2019-07-20	2
2019-07-21	2

解释：注意非活跃用户的记录不需要展示。

思路

先用datediff筛选合适的日期，然后再count一下**distinct** user_id。

代码

```
# 官方题库应该更新了，之前不需要加 and datediff('2019-07-27', activity_date) >= 0，现在需要加上。
# 因为select datediff('2019-07-27', '2019-07-28')的返回结果为-1
select
    activity_date day
    , count(distinct user_id) active_users
from activity
where datediff('2019-07-27', activity_date) < 30
and datediff('2019-07-27', activity_date) >= 0
group by activity_date
```

1107. 每日新用户统计【中等】

难度：中等

`Traffic` 表：

Column Name	Type
user_id	int
activity	enum
activity_date	date

该表没有主键，它可能有重复的行。
activity 列是 ENUM 类型，可能取 ('login', 'logout', 'jobs', 'groups', 'homepage') 几个值之一。

编写一个 SQL 查询，以查询从今天起最多 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.

查询结果格式如下例所示：

`Traffic` 表：

user_id	activity	activity_date
1	login	2019-05-01
1	homepage	2019-05-01
1	logout	2019-05-01
2	login	2019-06-21
2	logout	2019-06-21
3	login	2019-01-01
3	jobs	2019-01-01
3	logout	2019-01-01
4	login	2019-06-21
4	groups	2019-06-21
4	logout	2019-06-21
5	login	2019-03-01
5	logout	2019-03-01
5	login	2019-06-21
5	logout	2019-06-21

`Result` 表：

login_date	user_count
2019-05-01	1
2019-06-21	2

请注意，我们只关心用户数非零的日期.
ID 为 5 的用户第一次登陆于 2019-03-01，因此他不算在 2019-06-21 的的统计内。

思路

先找到每个用户的首次登陆日期，然后用datediff筛选合适用户，注意这道题是>=90不是>90

代码

```
select
    first_date as login_date
    , count(distinct user_id) as user_count
from
(
    select
        user_id
        , min(activity_date) as first_date
    from Traffic
    where activity = 'login'
    group by user_id
) as tmp
where datediff('2019-06-30', first_date)<=90
  and datediff('2019-06-30', first_date)>=0
group by first_date

# 这样写应该也可以，不过我感觉还是上面更清晰
select
    first_date as login_date
    , count(distinct user_id) as user_count
from
(
    select
        user_id
        , min(activity_date) as first_date
    from Traffic
    where activity = 'login'
    group by user_id
    having datediff('2019-06-30', min(activity_date))<=90
) as tmp
group by first_date
```

574. 当选者【中等】

难度：中等

表: `Candidate`

Column Name	Type
id	int
name	varchar

Id是该表的主键列。
该表的每一行都包含关于候选对象的id和名称的信息。

表: `Vote`

Column Name	Type
id	int
candidateId	int

Id是自动递增的主键。
candidateId是id来自Candidate表的外键。
该表的每一行决定了在选举中获得第i张选票的候选人。
编写一个SQL查询来报告获胜候选人的名字(即获得最多选票的候选人)。
生成测试用例以确保 只有一个候选人赢得选举。
查询结果格式如下所示。
示例 1:
输入:
`Candidate` table:

id	name
1	A
2	B
3	C
4	D
5	E

`Vote` table:

id	candidateId
1	2
2	4
3	3
4	2
5	5

输出:

name
B

解释:
候选人B有2票。候选人C、D、E各有1票。
获胜者是候选人B。

思路

同586题

代码

```
# 方法1:
select
    name
from
(
    select
        a.name
        , dense_rank() over(order by count(b.id) desc) as rnk
    from Candidate as a
    right join Vote as b
    on a.id = b.candidateId
    group by b.candidateId
) as tmp
where rnk = 1

# 方法2:
select
    a.name
from Candidate as a
right join Vote as b
on a.id = b.candidateId
group by b.candidateId
```

```
order by count(b.id) desc
limit 1

# 方法3:
with cte as(
    select
       a.name
       , b.id
       , b.candidateId
    from Candidate as a
    right join Vote as b
    on a.id = b.candidateId
)
select
    name
from cte
group by candidateId
having count(id)>=
all(
    select
        count(id)
    from cte
    group by candidateId
)

# 方法4
# 错误写法，好像是因为内层不能limit
# This version of MySQL doesn't yet support 'LIMIT & IN/ALL/ANY/SOME subquery'
# select
#     name
# from Candidate
# where id in(
#     select
#         candidateId
#     from Vote
#     group by candidateId
#     order by count(id)
#     limit 1
# )

# 正确写法
select
    name
from Candidate
where id in(
    select
        candidateId
    from Vote
    group by candidateId
```

```
    having count(id)>=all(
        select
            count(id)
        from Vote
        group by candidateId
    )
)
```

[578. 查询回答率最高的问题](#)【中等】

难度：中等

`SurveyLog` 表：

Column Name	Type
id	int
action	ENUM
question_id	int
answer_id	int
q_num	int
timestamp	int

这张表没有主键，其中可能包含重复项。
action 是一个 ENUM 数据，可以是 "show"、"answer" 或者 "skip" 。
这张表的每一行表示：ID = id 的用户对 question_id 的问题在 timestamp 时间进行了 action 操作。
如果用户对应的操作是 "answer" ，answer_id 将会是对应答案的 id ，否则，值为 null 。
q_num 是该问题在当前会话中的数字顺序。
回答率 是指：同一问题编号中回答次数占显示次数的比率。
编写一个 SQL 查询以报告 回答率 最高的问题。如果有多个问题具有相同的最大 回答率 ，返回 question_id 最小的那个。
查询结果如下例所示。
示例：
输入：
`SurveyLog` table:

id	action	question_id	answer_id	q_num	timestamp
5	show	285	null	1	123
5	answer	285	124124	1	124
5	show	369	null	2	125
5	skip	369	null	2	126

输出：

survey_log
285

解释：
问题 285 显示 1 次、回答 1 次。回答率为 1.0 。
问题 369 显示 1 次、回答 0 次。回答率为 0.0 。
问题 285 回答率最高。

思路

同586，一点小小的区别就是比较的地方计算变复杂了

代码

```
# 方法1：
select
    question_id as survey_log
from(
    select
        question_id
        , row_number() over (order by count(answer_id)/sum(if(action = 'show',1,0))
desc, question_id asc) as rnk
    from SurveyLog
    group by question_id
) as tmp
where rnk = 1

# 方法2：
select
    question_id as survey_log
from SurveyLog
group by question_id
# order by count(answer_id)/count(timestamp) desc, question_id
order by sum(action = 'answer')/sum(action = 'show') desc,question_id
limit 1
# order 不一定是某个字段，而是字段的计算

# 方法3：
select
    question_id as survey_log
from SurveyLog
group by question_id
having count(answer_id)/sum(if(action = 'show',1,0)) >=
all(
    select
        count(answer_id)/sum(if(action = 'show',1,0))
    from SurveyLog
    group by question_id
)
```

```
order by question_id
limit 1
```

579. 查询员工的累计薪水【困难】

难度：困难

`Employee` 表保存了一年内的薪水信息。

请你编写 SQL 语句，对于每个员工，查询他除最近一个月（即最大月）之外，剩下每个月的近三个月的累计薪水（不足三个月也要计算）。

结果请按 Id 升序，然后按 Month 降序显示。

示例：

输入：

Id	Month	Salary
1	1	20
2	1	20
1	2	30
2	2	30
3	2	40
1	3	40
3	3	60
1	4	60
3	4	70

输出：

Id	Month	Salary
1	3	90
1	2	50
1	1	20
2	1	20
3	3	100
3	2	40

解释：
员工 '1' 除去最近一个月（月份 '4'），有三个月的薪水记录：月份 '3' 薪水为 40，月份 '2' 薪水为 30，月份 '1' 薪水为 20。
所以近 3 个月的薪水累计分别为 (40 + 30 + 20) = 90，(30 + 20) = 50 和 20。

Id	Month	Salary
1	3	90
1	2	50
1	1	20

员工 '2' 除去最近的一个月（月份 '2'）的话，只有月份 '1' 这一个月的薪水记录。

Id	Month	Salary
2	1	20

员工 '3' 除去最近一个月（月份 '4'）后有两个月，分别为：月份 '3' 薪水为 60 和 月份 '2' 薪水为 40。所以各月的累计情况如下：

Id	Month	Salary
3	3	100
3	2	40

知识点：

rows 和range的区别

rows指的物理位置上的连续，即使数字之间不连续也没事。比如数字串3.4.7.8中求3.4.7的和，三者相加。

range指的是数字之间的连续，[n,n+1,n+2],比如1,2,3或5,6,7

思路

首次用where (id,month) not in...max(month)...的方式排除最大月

然后有两种方法，注意order：

1. 用窗口函数range去计算和
2. 用自连接（笛卡尔积）的方式，找到每个月的最近的三个月的薪水之和。

代码

```
# 方法1:
select
    Id
    , Month
    , sum(Salary) over(partition by Id order by month range 2 preceding) as Salary
```

```
from Employee
where (id,month) not in(
    select
        id
        ,max(month)
    from Employee
    group by Id
)
order by Id asc, Month desc

# 方法2：（自连接）
select
    a.Id
    , a.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 asc, a.Month desc
```