DAY 1

1068. 产品销售分析 I

难度：简单
销售表 `Sales`：

Column Name	Type
sale_id	int
product_id	int
year	int
quantity	int
price	int

`(sale_id, year)`是销售表 `Sales` 的主键.
`product_id`是关联到产品表 `Product` 的外键.
注意: price 表示每单位价格

产品表 `Product`：

Column Name	Type
product_id	int
product_name	varchar

`product_id`是表的主键
写一条SQL 查询语句获取 `Sales` 表中所有产品对应的 **产品名称 product_name** 以及该产品的所有 **售卖年份 year** 和 **价格 price** 。

`Sales`表:

sale_id	product_id	year	quantity	price
1	100	2008	10	5000
2	100	2009	12	5000
7	200	2011	15	9000

`Product` 表:

product_id	product_name
100	Nokia
200	Apple
300	Samsung

`Result`表:

product_name	year	price
Nokia	2008	5000
Nokia	2009	5000
Apple	2011	9000

思路

看见需要取两个表的数据，想到表的连接。

需要注意的是，能够想到如果按照Product表join就会出现null的问题，虽然此题不需要。

这里有主键，若是没有主键最好+distinct。

算法

left join... on ....

代码

```
select
    b.product_name
    , a.year
    , a.price
from Sales a
left join Product b
on a.product_id = b.product_id
```

1069. 产品销售分析 II

难度：简单
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。
查询结果格式如下面例子所示:
`Result` 表：

product_id	total_quantity
100	22
200	15

基础知识

注意group by 和 partition by 的区别

思路

按照`product_id`分组，对`quantity`进行求和

注意对列名重命名

算法

(1) group by（group by 1的意思是select的出来的表的第一个字段）

(2) 聚合函数 sum()

代码

```
select
    product_id
    ,sum(quantity) as total_quantity
from Sales
group by product_id
```

577. 员工奖金

难度：简单

选出所有 bonus < 1000 的员工的 name 及其 bonus。

`Employee` 表单
empId 是这张表单的主关键字

empId	name	supervisor	salary
1	John	3	1000
2	Dan	3	2000
3	Brad	null	4000
4	Thomas	3	4000

`Bonus` 表单
empId 是这张表单的主关键字

empId	bonus
2	500
4	2000

输出示例：

name	bonus
John	null
Dan	500
Brad	null

思路

1. ifnull的用法：ifnull(字段, 如果是null则变成的值)
2. left join：观察输出示例可以知道，John和Brad没有出现在Bonus表单中，但仍算为null，因此直接join两张表是无法join到这两个人的，需要用**左连接**

算法

left join table1 as a table2 as b on a.key = b.key

ifnull(字段,0)

代码

```
select
    a.name
    ,b.bonus
from Employee a
left join Bonus b # 存在null
on a.empId = b.empId
# where bonus<1000 or bonus is null # 注意这里是is 不是=
where ifnull(bonus,0) < 1000 # 如果值是null就变成0
```

584. 寻找用户推荐人

难度：简单

给定表 `customer` ，里面保存了所有客户信息和他们的推荐人。

id	name	referee_id
1	Will	NULL
2	Jane	NULL
3	Alex	2
4	Bill	NULL
5	Zack	1
6	Mark	2

写一个查询语句，返回一个客户列表，列表中客户的推荐人的编号都**不是 2**。
对于上面的示例数据，结果为：

name
Will
Jane
Bill
Zack

思路

null不可以和任何东西比较，如果比较的结果也是null

算法

ifnull(referee_id,0) <> 2

代码

```
# 错误写法:
select name
from customer
where referee_id <> 2
# 注意:null不能与任何值比较。在sql中,null与任何其他值比较(即使是null)永远不是true。包含null的表达是总是会导出null值。因此上述只能筛选出zack而不是四个人。
# 正确写法:
select name
from customer
where ifnull(referee_id,0) <> 2
```

511. 游戏玩法分析 I

难度：简单

活动表 `Activity`：

Column Name	Type
player_id	int
device_id	int
event_date	date
games_played	int

表的主键是 (player_id, event_date)。
这张表展示了一些游戏玩家在游戏平台上的行为活动。
每行数据记录了一名玩家在退出平台之前，当天使用同一台设备登录平台后打开的游戏的数目（可能是 0 个）。
写一条 SQL 查询语句获取每位玩家**第一次登陆平台的日期**。
查询结果的格式如下所示
`Activity` 表：

player_id	device_id	event_date	games_played
1	2	2016-03-01	5
1	2	2016-05-02	6
2	3	2017-06-25	1
3	1	2016-03-02	0
3	4	2018-07-03	5

`Result` 表：

player_id	first_login
1	2016-03-01
2	2017-06-25
3	2016-03-02

思路

按照player_id分组后按要求取即可

算法

group by、聚合函数min()

代码

```
select
    player_id
    , min(event_date) as first_login
from Activity
group by player_id
# group by 1
```

512. 游戏玩法分析 II

难度：简单

请编写一个 SQL 查询，描述每一个玩家首次登陆的设备名称
查询结果格式在以下示例中：

Result table:

player_id	device_id
1	2
2	3
3	1

思路

注意看主键，表的主键是 (player_id, event_date)

算法

这个题有一个很常用的结构，其实也是从套题的第一题想到的：

```
select
  xx, yy
from table
where (aa通常是主键, bb) in # 从另一个表选做筛选，这里aa的作用是用来避免多筛了，bb就是筛选条件
( select
    aa, bb的某个聚合函数
  from table
  group by aa) # group by、order by、limit…根据情况来
```

代码

```
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)

# 我的错误写法:
select
    player_id
    , device_id
from Activity
group by player_id
having min(event_date) # 这句话是不对的,
# 错误原因（来自leetcode）：
# 1.我查了一下 min（），max（）和group by一起用的时候 此时会发现获取到的数据不是最大/最小的； 问题根源：group by默认返回每一组的第一条数据（每一组的数据排序都是按默认顺序排序的 还挺坑的!
# 2.having子句只能对select中存在的字段进行筛选，很显然你的select子句中并没有event_date字段，所以报错，建议去百度一下having子句和where子句的区别。
```

534. 游戏玩法分析 III

难度：中等

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。

`Result` table:

player_id	event_date	games_played_so_far
1	2016-03-01	5
1	2016-05-02	11
1	2017-06-25	12
3	2016-03-02	0
3	2018-07-03	5

对于 ID 为 1 的玩家，2016-05-02 共玩了 5+6=11 个游戏，2017-06-25 共玩了 5+6+1=12 个游戏。
对于 ID 为 3 的玩家，2018-07-03 共玩了 0+5=5 个游戏。
请注意，对于每个玩家，我们只关心玩家的登录日期。

思路

这个题目主要考察窗口函数的使用，可以学会这种累加的做法，也别忘了order by

算法

窗口函数 partition by order by

代码

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

550. 游戏玩法分析 IV

难度：中等

编写一个 SQL 查询，报告在首次登录的第二天再次登录的玩家的比率，四舍五入到小数点后两位。换句话说，您需要计算从首次登录日期开始至少连续两天登录的玩家的数量，然后除以玩家总数。

`Result` table:

fraction
0.33

只有 ID 为 1 的玩家在第一天登录后才重新登录，所以答案是 1/3 = 0.33

思路

这个题目知识点不少：

1. distinct的用法

2. round的用法
3. MySQL 使用下列数据类型在数据库中存储日期或日期/时间值：

DATE - 格式 YYYY-MM-DD

DATETIME - 格式: YYYY-MM-DD HH:MM:SS

TIMESTAMP - 格式: YYYY-MM-DD HH:MM:SS

YEAR - 格式 YYYY 或 YY

本题需要转换一下

算法

代码

```
# 方法一
select
    ifnull(
        round(count(player_id)/(select count(distinct player_id) from Activity)
        ,2)
        ,0) as fraction
from Activity
where (player_id, event_date) in
(select player_id, DATE_ADD(min(event_date), INTERVAL 1 DAY)
from Activity
group by player_id)


# 方法二:
with cte as
(select
  player_id, min(event_date) first_login
  from activity
  group by player_id)  # 先找到每个player_id的首次登陆时间
select round(count(event_date)/
             (select count(distinct player_id) from activity)
             ,2)
as fraction
from activity, cte
where datediff(event_date, first_login)=1
and cte.player_id=activity.player_id
```

570. 至少有5名直接下属的经理

难度：中等

表: Employee

Column Name	Type
id	int
name	varchar
department	varchar
managerId	int

Id是该表的主键列。
该表的每一行都表示雇员的名字、他们的部门和他们的经理的id。
如果managerId为空，则该员工没有经理。
没有员工会成为自己的管理者。

编写一个SQL查询，查询至少有5名直接下属的经理 。
以 任意顺序 返回结果表。
查询结果格式如下所示

示例 1:
输入:
Employee 表:

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

输出:

name
John

思路

算法

代码

```sql
select
    name
from Employee
where id in
(   select
        distinct managerId # 这里注意加distinct
    from Employee
    group by managerId
    having count(id) >= 5)
```

569. 员工薪水中位数

难度：困难

表: `Employee`

Column Name	Type
id	int
company	varchar
salary	int

id是该表的主键列。
该表的每一行表示公司和一名员工的工资。
写一个SQL查询，找出每个公司的工资中位数。
以 任意顺序 返回结果表。
查询结果格式如下所示。
示例 1:
输入:
`Employee` 表:

id	company	salary
1	A	2341
2	A	341
3	A	15
4	A	15314
5	A	451
6	A	513
7	B	15
8	B	13
9	B	1154
10	B	1345
11	B	1221
12	B	234
13	C	2345
14	C	2645
15	C	2645
16	C	2652
17	C	65

输出:

id	company	salary
5	A	451
6	A	513
12	B	234
9	B	1154
14	C	2645

进阶: 你能在不使用任何内置函数或窗口函数的情况下解决它吗?

知识点：

1. 三种排序方式及区别：

row_number: 直接排序，并列的也排 排出来是123456

dense_rank: 排名是连续的 排出来是123345

rank: 排名是不连续的 排出来是123356

2. **MySQL 的CAST()和CONVERT()函数** 可用来获取一个类型的值，并产生另一个类型的值。两者具体的语法如下：

   （1） CAST(value as type);

   （2） CONVERT(value, type);

   就是CAST(xxx AS 类型), CONVERT(xxx,类型)。

   可以转换的类型是有限制的。这个类型可以是以下值其中的一个：

   二进制，同带binary前缀的效果 : BINARY

   字符型，可带参数 : CHAR()

   日期 : DATE

   时间: TIME

   日期时间型 : DATETIME

   浮点数 : DECIMAL

   整数 : SIGNED

   无符号整数 : UNSIGNED

思路和**算法**见代码注释

代码

```
# 方法一：这道题的巧妙解法
# 巧妙之处在于中位数不需要分奇偶了，因为必在cnt/2, cnt/2+1, (cnt+1)/2 中
select
    id
    ,company
    ,salary
from
(   select
        a.*
        , row_number() over(partition by company order by salary) as rnk
        , count(id) over(partition by company) as cnt
    from Employee a) tmp
where rnk in (cnt/2, cnt/2+1, (cnt+1)/2)

# 方法二：中位数的通用解法1
select id, company, salary
from (
    select
        tmp.*,
        # 小细节，在使用SQL窗口函数按照大小顺序编号的时候，相等的两个数会存在以升序序列来标注问题
# 此时可以利用SQL表的主键id，在待求序列相等时，正向编号按id升序排序编号，反向编号按id降序排序编号，这样就保证了两列编号的走向处处相反，从而使得判断条件生效。
```

```
        row_number() over(partition by company order by salary, id) as rnk1,
        row_number() over(partition by company order by salary desc, id desc) as rnk2,
        count(id) over(partition by company) as cnt
    from Employee as tmp) as a
where rnk1>=cnt/2 and rnk2>=cnt/2

# 方法三：中位数的通用解法2
# 与方法二类似，但是这里必须+cast(… as signed) 即转换为整数
select a.Id, a.Company, a.Salary
from
(select
    Id, Company, Salary,
    cast(row_number() over(partition by Company order by salary desc, Id desc) as
signed) as id1,
    cast(row_number() over(partition by Company order by salary asc, Id asc) as signed)
as id2
from  Employee) a
where abs(a.id1-a.id2) =1 or a.id1=a.id2
```

571. 给定数字的频率查询中位数

难度：困难
Numbers 表：

Column Name	Type
num	int
frequency	int

num 是这张表的主键。这张表的每一行表示某个数字在该数据库中的出现频率。

中位数 是将数据样本中半数较高值和半数较低值分隔开的值。
编写一个 SQL 查询，解压 Numbers 表，报告数据库中所有数字的 中位数 。结果四舍五入至 一位小数 。
查询结果如下例所示。
示例：
输入：
Numbers 表：

num	frequency
0	7
1	1
2	3
3	1

输出：

median
0.0

解释：
如果解压这个 Numbers 表，可以得到 [0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 2, 3] ，所以中位数是 (0 + 0) / 2 = 0 。

思路

与569不同之处：

569是直接给出所有数据，因此需要首先按照公司分组，然后在组内直接排序找到中位数；

571是给出所有数字的频数，相当于整体是一组，而且已经按照数字分组了，那么就需要通过累积求和frequency的方式，来确认中位数所在的位置。然后按照569的思路2的方式（思路1失效，因为只有累积求和，已经没有办法准确定位到中位数的位置了），sql从里到外写法为：

算法

select 中位数均值

from 新创立的表（整个表+正序累积求和（order by num）+倒序累积求和（order by num））

where 正序>总和/2 and 倒序>总和/2

代码

```
select
    round(
        avg(num)
        ,1) as median
from(
    select
        a.*
        , sum(frequency) over(order by num asc) as rnk1
        , sum(frequency) over(order by num desc) as rnk2
        # 如果这里计算总frequency的话，需要加over：
        #, sum(frequency) over() as cnt
    from Numbers a) as tmp
where tmp.rnk1 >= (select sum(frequency)from Numbers)/2
and tmp.rnk2 >= (select sum(frequency)from Numbers)/2
# 如果按照上面注释的写法，那么这里where条件改为：
# where tmp.rnk1>=tmp.cnt/2 and tmp.rnk2>=tmp.cnt/2
```