数据库

175、组合两个表

表1：`Person`

```
+-------------+---------+
| 列名         | 类型     |
+-------------+---------+
| PersonId    | int     |
| FirstName   | varchar |
| LastName    | varchar |
+-------------+---------+
PersonId 是上表主键
```

表2：`Address`

```
+-------------+---------+
| 列名         | 类型    |
+-------------+---------+
| AddressId   | int     |
| PersonId    | int     |
| City        | varchar |
| State       | varchar |
+-------------+---------+
AddressId 是上表主键
```

> 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
> ```

题解

1、左连接表

```
SELECT FirstName,LastName,City,State
FROM Person p LEFT JOIN Address a
ON p.PersonID=a.PersonID
```

176、第二高的薪水

编写一个 SQL 查询，获取 `Employee` 表中第二高的薪水（Salary） 。

```
+----+--------+
| Id | Salary |
+----+--------+
| 1  | 100    |
| 2  | 200    |
| 3  | 300    |
+----+--------+
```

> 例如上述 `Employee` 表，SQL查询应该返回 `200` 作为第二高的薪水。如果不存在第二高的薪水，那么查询应返回 `null`。
>
> ```
> +---------------------+
> | SecondHighestSalary |
> +---------------------+
> | 200                 |
> +---------------------+
> ```

题解

1、使用临时表

```
SELECT
    (SELECT DISTINCT
            Salary
        FROM
            Employee
        ORDER BY Salary DESC
        LIMIT 1 OFFSET 1) AS SecondHighestSalary;
```

2、使用IFNULL

```
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

编写一个 SQL 查询，获取 `Employee` 表中第 *n* 高的薪水（Salary）。

```
+----+--------+
| Id | Salary |
+----+--------+
| 1  | 100    |
| 2  | 200    |
| 3  | 300    |
+----+--------+
```

> 例如上述 `Employee` 表，*n* = *2* 时，应返回第二高的薪水 `200`。如果不存在第 *n* 高的薪水，那么查询应返回 `null`。
>
> ```
> +------------------------+
> | getNthHighestSalary(2) |
> +------------------------+
> | 200                    |
> +------------------------+
> ```

题解

1、单表查询

- 同薪同名且不跳级的问题，解决办法是用group by按薪水分组后再order by
- 排名第N高意味着要跳过N-1个薪水，由于无法直接用limit N-1，所以需先在函数开头处理N为N=N-1。
  注：这里不能直接用limit N-1是因为limit和offset字段后面只接受正整数（意味着0、负数、小数都不行）或者单一变量（意味着不能用表达式），也就是说想取一条，limit 2-1、limit 1.1这类的写法都是报错的。

```
CREATE FUNCTION getNthHighestSalary(N INT) RETURNS INT
BEGIN
    SET N := N-1;
  RETURN (
      # Write your MySQL query statement below.
      SELECT
            salary
      FROM
            employee
      GROUP BY
            salary
      ORDER BY
            salary DESC
      LIMIT N, 1
  );
END
```

2、子查询

1. 排名第N的薪水意味着该表中存在N-1个比其更高的薪水
2. 注意这里的N-1个更高的薪水是指去重后的N-1个，实际对应人数可能不止N-1个
3. 最后返回的薪水也应该去重，因为可能不止一个薪水排名第N
4. 由于对于每个薪水的where条件都要执行一遍子查询，注定其效率低下

```
CREATE FUNCTION getNthHighestSalary(N INT) RETURNS INT
BEGIN
  RETURN (
      # Write your MySQL query statement below.
      SELECT
          DISTINCT e.salary
      FROM
          employee e
      WHERE
          (SELECT count(DISTINCT salary) FROM employee WHERE salary>e.salary) =
N-1
  );
END
```

3、自连接

1. 两表自连接，连接条件设定为表1的salary小于表2的salary
2. 以表1的salary分组，统计表1中每个salary分组后对应表2中salary唯一值个数，即去重
3. 限定步骤2中having 计数个数为N-1，即实现了该分组中表1salary排名为第N个
4. 考虑N=1的特殊情形(特殊是因为N-1=0，计数要求为0)，此时不存在满足条件的记录数，但仍需返回结果，所以连接用left join
5. 如果仅查询薪水这一项值，那么不用left join当然也是可以的，只需把连接条件放宽至小于等于、同时查询个数设置为N即可。因为连接条件含等号，所以一定不为空，用join即可。

```
CREATE FUNCTION getNthHighestSalary(N INT) RETURNS INT
BEGIN
  RETURN (
      # Write your MySQL query statement below.
      SELECT
          DISTINCT e1.salary
      FROM
          employee e1 LEFT JOIN employee e2 ON e1.salary < e2.salary
      GROUP BY
          e1.salary
      HAVING
          count(DISTINCT e2.salary) = N-1
  );
END
```

4、笛卡尔积

- 将思路2中的代码改为笛卡尔积连接形式，其执行过程实际上一致的，甚至MySQL执行时可能会优化成相同的查询语句。

```
CREATE FUNCTION getNthHighestSalary(N INT) RETURNS INT
BEGIN
  RETURN (
      # Write your MySQL query statement below.
      SELECT
          e1.salary
      FROM
          employee e1, employee e2
      WHERE
          e1.salary <= e2.salary
      GROUP BY
          e1.salary
```

```
        HAVING
            count(DISTINCT e2.salary) = N
    );
END
```

5、自定义变量

- 以上方法2-4中均存在两表关联的问题，表中记录数少时尚可接受，当记录数量较大且无法建立合适索引时，实测速度会比较慢，用算法复杂度来形容大概是O(n^2)量级（实际还与索引有关）。那么，用下面的自定义变量的方法可实现O(2*n)量级，速度会快得多，且与索引无关。
    1. 自定义变量实现按薪水降序后的数据排名，同薪同名不跳级，即3000、2000、2000、1000排名后为1、2、2、3；
    2. 对带有排名信息的临时表二次筛选，得到排名为N的薪水；
    3. 因为薪水排名为N的记录可能不止1个，用distinct去重

```
CREATE FUNCTION getNthHighestSalary(N INT) RETURNS INT
BEGIN
  RETURN (
      # Write your MySQL query statement below.
      SELECT
          DISTINCT salary
      FROM
          (SELECT
                salary, @r:=IF(@p=salary, @r, @r+1) AS rnk,  @p:= salary
            FROM
                employee, (SELECT @r:=0, @p:=NULL)init
            ORDER BY
                salary DESC) tmp
      WHERE rnk = N
  );
END
```

6、窗口函数

实际上，在mysql8.0中有相关的内置函数，而且考虑了各种排名问题：

- row_number(): 同薪不同名，相当于行号，例如3000、2000、2000、1000排名后为1、2、3、4
- rank(): 同薪同名，有跳级，例如3000、2000、2000、1000排名后为1、2、2、4
- dense_rank(): 同薪同名，无跳级，例如3000、2000、2000、1000排名后为1、2、2、3
- ntile(): 分桶排名，即首先按桶的个数分出第一二三桶，然后各桶内从1排名，实际不是很常用

显然，本题是要用第三个函数。
另外这三个函数必须要要与其搭档over()配套使用，over()中的参数常见的有两个，分别是

- partition by，按某字段切分
- order by，与常规order by用法一致，也区分ASC(默认)和DESC，因为排名总得有个依据

```
CREATE FUNCTION getNthHighestSalary(N INT) RETURNS INT
BEGIN
  RETURN (
      # Write your MySQL query statement below.
        SELECT
            DISTINCT salary
        FROM
            (SELECT
                salary, dense_rank() over(ORDER BY salary DESC) AS rnk
```

```
            FROM
                employee) tmp
        WHERE rnk = N
  );
END
```

178、分数排名

编写一个 SQL 查询来实现分数排名。

如果两个分数相同，则两个分数排名（Rank）相同。请注意，平分后的下一个名次应该是下一个连续的整数值。换句话说，名次之间不应该有“间隔”。

```
+----+-------+
| Id | Score |
+----+-------+
| 1  | 3.50  |
| 2  | 3.65  |
| 3  | 4.00  |
| 4  | 3.85  |
| 5  | 4.00  |
| 6  | 3.65  |
+----+-------+
```

> 例如，根据上述给定的 `Scores` 表，你的查询应该返回（按分数从高到低排列）：
>
> ```
> +-------+------+
> | Score | Rank |
> +-------+------+
> | 4.00  | 1    |
> | 4.00  | 1    |
> | 3.85  | 2    |
> | 3.65  | 3    |
> | 3.65  | 3    |
> | 3.50  | 4    |
> +-------+------+
> ```

题解

1、四大排名函数

- row_number(): 同薪不同名，相当于行号，例如3000、2000、2000、1000排名后为1、2、3、4
- rank(): 同薪同名，有跳级，例如3000、2000、2000、1000排名后为1、2、2、4
- dense_rank(): 同薪同名，无跳级，例如3000、2000、2000、1000排名后为1、2、2、3
- ntile(): 分桶排名，即首先按桶的个数分出第一二三桶，然后各桶内从1排名，实际不是很常用

```
SELECT SCORE, DENSE_RANK()OVER(ORDER BY Score DESC) AS `Rank`
FROM Scores
```

180、连续出现的数字

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| num         | varchar |
+-------------+---------+
id 是这个表的主键。
```

编写一个 SQL 查询，查找所有至少连续出现三次的数字。

返回的结果表中的数据可以按 **任意顺序** 排列。

> ```
> Logs 表：
> +----+-----+
> | Id | Num |
> +----+-----+
> | 1  | 1   |
> | 2  | 1   |
> | 3  | 1   |
> | 4  | 2   |
> | 5  | 1   |
> | 6  | 2   |
> | 7  | 2   |
> +----+-----+
>
> Result 表：
> +-----------------+
> | ConsecutiveNums |
> +-----------------+
> | 1               |
> +-----------------+
> 1 是唯一连续出现至少三次的数字。
> ```

题解

1、用 `DISTINCT` 和 `WHERE` 语句

- 连续出现的意味着相同数字的 Id 是连着的，由于这题问的是至少连续出现 3 次，我们使用 **Logs** 并检查是否有 3 个连续的相同数字。

```
SELECT *
FROM
    Logs l1,
    Logs l2,
    Logs l3
WHERE
    l1.Id = l2.Id - 1
    AND l2.Id = l3.Id - 1
    AND l1.Num = l2.Num
    AND l2.Num = l3.Num;
```

- 然后我们从上表中选择任意的 *Num* 获得想要的答案。同时我们需要添加关键字 `DISTINCT` ，因为如果一个数字连续出现超过 3 次，会返回重复元素。

```
SELECT DISTINCT
    l1.Num AS ConsecutiveNums
FROM
    Logs l1,
    Logs l2,
    Logs l3
WHERE
    l1.Id = l2.Id - 1
    AND l2.Id = l3.Id - 1
    AND l1.Num = l2.Num
    AND l2.Num = l3.Num;
```

2、定义两个变量

- 定义2个变量，@last_num:用于存放上一行记录的num，@cnt:用于计数。
- 上一行num等于当前num时@cnt+1，即@last_num=num时@num=@num+1；
- 之后将本行num赋值给@last_num，最后筛选出@cnt>=3的记录去重即可。

```
select distinct Num ConsecutiveNums
from (select Id, Num,
case when @last_num=Num then @cnt:=@cnt+1 else @cnt:=1 end cnt,
@last_num:=Num
from Logs x, (select @last_num:=0, @cnt:=0) y) w
where cnt >= 3;
```

3、编写查询函数

```
#①首先遍历一遍整张表，找出每个数字的连续重复次数
#具体方法为：
    #初始化两个变量，一个为pre，记录上一个数字；一个为count，记录上一个数字已经连续出现的次数。
    #然后调用if()函数，如果pre和当前行数字相同，count加1极为连续出现的次数；如果不同，意味着重新开始一个数字，count重新从1开始。
    #最后，将当前的Num数字赋值给pre，开始下一行扫描。
    select
        Num,    #当前的Num 数字
        if(@pre=Num,@count := @count+1,@count := 1) as nums, #判断 和 计数
        @pre:=Num   #将当前Num赋值给pre
    from Logs as l ,
        (select @pre:= null,@count:=1) as pc #这里需要别名
    #上面这段代码执行结果就是一张三列为Num,count as nums,pre的表。

#②将上面表的结果中，重复次数大于等于3的数字选出，再去重即为连续至少出现三次的数字。
    select
        distinct Num as ConsecutiveNums
    from
        (select Num,
                if(@pre=Num,@count := @count+1,@count := 1) as nums,
                @pre:=Num
            from Logs as l ,
                (select @pre:= null,@count:=1) as pc
        ) as n
    where nums >=3;
```

```
select
    distinct Num as ConsecutiveNums
from
    (select Num,
            if(@pre=Num,@count := @count+1,@count := 1) as nums,
            @pre:=Num
        from Logs as l ,
            (select @pre:= null,@count:=1) as pc
     ) as n
where nums >=3;
```

181、超过经理收入的员工

`Employee` 表包含所有员工，他们的经理也属于员工。每个员工都有一个 Id，此外还有一列对应员工的经理的 Id。

```
+----+-------+--------+-----------+
| Id | Name  | Salary | ManagerId |
+----+-------+--------+-----------+
| 1  | Joe   | 70000  | 3         |
| 2  | Henry | 80000  | 4         |
| 3  | Sam   | 60000  | NULL      |
| 4  | Max   | 90000  | NULL      |
+----+-------+--------+-----------+
```

> 给定 `Employee` 表，编写一个 SQL 查询，该查询可以获取收入超过他们经理的员工的姓名。在上面的表格中，Joe 是唯一一个收入超过他的经理的员工。
>
> ```
> +----------+
> | Employee |
> +----------+
> | Joe      |
> +----------+
> ```

题解

1、左连接表，使用where判断

```
select
    l1.Name AS Employee
from Employee l1 left join Employee l2 on l1.ManagerId=l2.Id
where l1.Salary>l2.Salary;
```

```
SELECT
     a.NAME AS Employee
FROM Employee AS a JOIN Employee AS b
     ON a.ManagerId = b.Id
     AND a.Salary > b.Salary;
```

2、仅使用where

```
SELECT
    a.Name AS 'Employee'
FROM
    Employee AS a,
    Employee AS b
WHERE
    a.ManagerId = b.Id
        AND a.Salary > b.Salary;
```

182、查找重复的电子邮箱

编写一个 SQL 查询，查找 `Person` 表中所有重复的电子邮箱。

```
+----+---------+
| Id | Email   |
+----+---------+
| 1  | a@b.com |
| 2  | c@d.com |
| 3  | a@b.com |
+----+---------+
```

> 根据以上输入，你的查询应返回以下结果：
>
> ```
> +---------+
> | Email   |
> +---------+
> | a@b.com |
> +---------+
> ```

题解

1、使用where判断

```
select DISTINCT
    a.Email AS 'Email'
from
    Person AS a,
    Person AS b
where a.Id!=b.Id
    AND a.Email=b.Email;
```

2、使用GROUP BY和临时表

- 重复的电子邮箱存在多次。要计算每封电子邮件的存在次数，我们可以使用以下代码。

```
select Email, count(Email) as num
from Person
group by Email;
```

```sql
select Email from
(
  select Email, count(Email) as num
  from Person
  group by Email
) as statistic
where num > 1;
```

3、使用`GROUP BY`和`HAVING`条件

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

4、表连接

```sql
SELECT DISTINCT L1.Email
    FROM Person L1 JOIN Person L2
    ON L1.Email = L2.Email
    WHERE L1.Id <> L2.Id;
```

183、从不订购的客户

某网站包含两个表，`Customers`表和`Orders`表。编写一个 SQL 查询，找出所有从不订购任何东西的客户。

- Customers 表:

```
+----+-------+
| Id | Name  |
+----+-------+
| 1  | Joe   |
| 2  | Henry |
| 3  | Sam   |
| 4  | Max   |
+----+-------+
```

- Orders 表:

```
+----+------------+
| Id | CustomerId |
+----+------------+
| 1  | 3          |
| 2  | 1          |
+----+------------+
```

> 例如给定上述表格，你的查询应返回:

```
+-----------+
| Customers |
+-----------+
| Henry     |
| Max       |
+-----------+
```

题解

1、使用子查询和 `NOT IN` 子句

```sql
select customers.name as 'Customers'
from customers
where customers.id not in
(
    select customerid from orders
);
```

2、左连接

```sql
select c.Name as Customers
from Customers as c
left join Orders as o on c.Id = o.CustomerId
where o.Id is null;
```

184、部门工资最高的员工

`Employee` 表包含所有员工信息，每个员工有其对应的 Id, salary 和 department Id。

```
+----+-------+--------+--------------+
| Id | Name  | Salary | DepartmentId |
+----+-------+--------+--------------+
| 1  | Joe   | 70000  | 1            |
| 2  | Jim   | 90000  | 1            |
| 3  | Henry | 80000  | 2            |
| 4  | Sam   | 60000  | 2            |
| 5  | Max   | 90000  | 1            |
+----+-------+--------+--------------+
```

`Department` 表包含公司所有部门的信息。

```
+----+----------+
| Id | Name     |
+----+----------+
| 1  | IT       |
| 2  | Sales    |
+----+----------+
```

> 编写一个 SQL 查询，找出每个部门工资最高的员工。对于上述表，您的 SQL 查询应返回以下行（行的顺序无关紧要）。

```
+------------+----------+--------+
| Department | Employee | Salary |
+------------+----------+--------+
| IT         | Max      | 90000  |
| IT         | Jim      | 90000  |
| Sales      | Henry    | 80000  |
+------------+----------+--------+
```

题解

1、表连接加where

```
SELECT
    Department.name AS 'Department',
    Employee.name AS 'Employee',
    Salary
FROM
    Employee
        JOIN
    Department ON Employee.DepartmentId = Department.Id
WHERE
    (Employee.DepartmentId , Salary) IN
    (   SELECT
            DepartmentId, MAX(Salary)
        FROM
            Employee
        GROUP BY DepartmentId
    );
```

2、使用where

```
SELECT
    Department.NAME AS Department,
    Employee.NAME AS Employee,
    Salary
FROM
    Employee,
    Department
WHERE
    Employee.DepartmentId = Department.Id
    AND ( Employee.DepartmentId, Salary )
    IN (SELECT DepartmentId, max( Salary )
        FROM Employee
        GROUP BY DepartmentId );
```

3、使用排序

1. 第一个,row_nubmer(),这个排序函数的特点是相同数据,先查出的排名在前,没有重复值。像我们这里Salary相同,先查出来的数据的rank排名优 先,显然不满足本题要求;
2. 第二个，rank()函数，是跳跃排序,相同数据排名相同,比如并列第1,则两行数据(这里为rank列)都标为1,下一位将是第3名.中间的2被直接跳过了。排名存在重复值。
3. 第三个，dense_rank(),这个是连续排序的,比如两条并列第1,则两行数据(这里为rank列)都标为1,下一个排名将是第2名。

```sql
select Department, Employee, Salary
from
(
select
    D.Name as Department,
    E.Name as Employee,
    E.Salary as Salary,
    rank() over(partition by D.Name order by E.Salary desc) as rank
from Employee E join Department D on E.DepartmentId = D.Id
) as tmp
where rank= 1;
```

185、部门工资前三高的所有员工

`Employee` 表包含所有员工信息，每个员工有其对应的工号 `Id`，姓名 `Name`，工资 `Salary` 和部门编号 `DepartmentId` 。

```
+----+-------+--------+--------------+
| Id | Name  | Salary | DepartmentId |
+----+-------+--------+--------------+
| 1  | Joe   | 85000  | 1            |
| 2  | Henry | 80000  | 2            |
| 3  | Sam   | 60000  | 2            |
| 4  | Max   | 90000  | 1            |
| 5  | Janet | 69000  | 1            |
| 6  | Randy | 85000  | 1            |
| 7  | Will  | 70000  | 1            |
+----+-------+--------+--------------+
```

`Department` 表包含公司所有部门的信息。

```
+----+----------+
| Id | Name     |
+----+----------+
| 1  | IT       |
| 2  | Sales    |
+----+----------+
```

> 编写一个 SQL 查询，找出每个部门获得前三高工资的所有员工。例如，根据上述给定的表，查询结果应返回：
>
> ```
> +------------+----------+--------+
> | Department | Employee | Salary |
> +------------+----------+--------+
> | IT         | Max      | 90000  |
> | IT         | Randy    | 85000  |
> | IT         | Joe      | 85000  |
> | IT         | Will     | 70000  |
> | Sales      | Henry    | 80000  |
> | Sales      | Sam      | 60000  |
> +------------+----------+--------+
> ```

题解

1、JOIN和子查询

先找出公司里前 3 高的薪水，意思是不超过三个值比这些值大

```
SELECT e1.Salary
FROM Employee AS e1
WHERE 3 >
        (SELECT  count(DISTINCT e2.Salary)
         FROM   Employee AS e2
         WHERE  e1.Salary < e2.Salary   AND e1.DepartmentId = e2.DepartmentId) ;
```

- 当 e1 = e2 = [4,5,6,7,8]
- e1.Salary = 4，e2.Salary 可以取值 [5,6,7,8]，count(DISTINCT e2.Salary) = 4
- e1.Salary = 5，e2.Salary 可以取值 [6,7,8]，count(DISTINCT e2.Salary) = 3
- e1.Salary = 6，e2.Salary 可以取值 [7,8]，count(DISTINCT e2.Salary) = 2
- e1.Salary = 7，e2.Salary 可以取值 [8]，count(DISTINCT e2.Salary) = 1
- e1.Salary = 8，e2.Salary 可以取值 []，count(DISTINCT e2.Salary) = 0

最后 3 > count(DISTINCT e2.Salary)，所以 e1.Salary 可取值为 [6,7,8]，即集合前 3 高的薪水

```
SELECT
    d.Name AS 'Department', e1.Name AS 'Employee', e1.Salary
FROM
    Employee e1
        JOIN
    Department d ON e1.DepartmentId = d.Id
WHERE
    3 > (SELECT
            COUNT(DISTINCT e2.Salary)
        FROM
            Employee e2
        WHERE
            e2.Salary > e1.Salary
                AND e1.DepartmentId = e2.DepartmentId
        );
```

2、自定义变量解法

- 根据 **部门 (升)**，**薪水 (降)** 顺序查询出每个部门的员工 **(Department, Employee, Salary)**

```
SELECT dep.Name Department, emp.Name Employee, emp.Salary
FROM Employee emp
INNER JOIN Department dep ON emp.DepartmentId = dep.Id
ORDER BY emp.DepartmentId, emp.Salary DESC
```

- 每个部门的员工根据薪水进行排序

```
## 先(部门,薪水)去重,再 部门(升),薪水(降) 排序
SELECT te.DepartmentId, te.Salary,
       CASE
            WHEN @pre = DepartmentId THEN @rank:= @rank + 1
            WHEN @pre := DepartmentId THEN @rank:= 1
       END AS RANK
FROM (SELECT @pre:=null, @rank:=0)tt,
     (## (部门,薪水)去重,根据 部门(升),薪水(降) 排序
         SELECT DepartmentId,Salary
         FROM Employee
         GROUP BY DepartmentId,Salary
         ORDER BY DepartmentId,Salary DESC
     )te
```

- 组合步骤时,尽量将每个步骤变成一个 **结果集（不存在二次查询）**再将所有步骤的 **结果集进行关联**，从而提高性能

```
SELECT dep.Name Department, emp.Name Employee, emp.Salary
FROM (## 自定义变量RANK, 查找出 每个部门工资前三的排名
        SELECT te.DepartmentId, te.Salary,
               CASE
                    WHEN @pre = DepartmentId THEN @rank:= @rank + 1
                    WHEN @pre := DepartmentId THEN @rank:= 1
               END AS RANK
        FROM (SELECT @pre:=null, @rank:=0)tt,
             (## (部门,薪水)去重,根据 部门(升),薪水(降) 排序
                 SELECT DepartmentId,Salary
                 FROM Employee
                 GROUP BY DepartmentId,Salary
                 ORDER BY DepartmentId,Salary DESC
             )te
       )t
INNER JOIN Department dep ON t.DepartmentId = dep.Id
INNER JOIN Employee emp ON t.DepartmentId = emp.DepartmentId and t.Salary =
emp.Salary and t.RANK <= 3
ORDER BY t.DepartmentId, t.Salary DESC ## t 结果集已有序,根据该集合排序
```

3、开窗

1. 第一步 在子查询里面关联employee 和department
   取出员工姓名，和所属部门，使用dense_rank开窗，每个员工工资按部门分组打上标记
2. 第二步 利用dense_rank标记取出工资排名前三高员工，所属部门和工资
   类似于构造列的解法，在where中对构建起来的rk进行筛选，符合rk<=3即可

```
select Department,Employee,Salary
from(
    select d.name Department,e.name Employee,Salary,
    dense_rank() over(partition by DepartmentId order by Salary desc) rk
    from Employee e inner join Department d
    on e.DepartmentId = d.Id
    )t
where rk<=3
```

196、删除重复的电子邮箱

编写一个 SQL 查询，来删除 `Person` 表中所有重复的电子邮箱，重复的邮箱里只保留 **Id** *最小* 的那个。

```
+----+------------------+
| Id | Email            |
+----+------------------+
| 1  | john@example.com |
| 2  | bob@example.com  |
| 3  | john@example.com |
+----+------------------+
Id 是这个表的主键。
```

> 例如，在运行你的查询语句之后，上面的 `Person` 表应返回以下几行:
>
> ```
> +----+------------------+
> | Id | Email            |
> +----+------------------+
> | 1  | john@example.com |
> | 2  | bob@example.com  |
> +----+------------------+
> ```

题解

1、使用 `DELETE` 和 `WHERE` 子句

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

2、开窗函数

- 针对同一邮箱，对其Id序号进行排序rn；
- 依据题意，只保留排序rn=1的数据，其余数据均需删除

```
delete
from Person
where Id in
    (
        select Id
        from
            (
                select Id,
                    row_number() over(partition by Email order by Id) rn
                from Person
            ) t1
        where rn>1
    )
```

197、上升的温度

表 `Weather`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| recordDate    | date    |
| temperature   | int     |
+---------------+---------+
id 是这个表的主键
该表包含特定日期的温度信息
```

> 编写一个 SQL 查询，来查找与之前（昨天的）日期相比温度更高的所有日期的 `id` 。
>
> 返回结果 **不要求顺序** 。
>
> 查询结果格式如下例：
>
> ```
> Weather
> +----+------------+-------------+
> | id | recordDate | Temperature |
> +----+------------+-------------+
> | 1  | 2015-01-01 | 10          |
> | 2  | 2015-01-02 | 25          |
> | 3  | 2015-01-03 | 20          |
> | 4  | 2015-01-04 | 30          |
> +----+------------+-------------+
>
> Result table:
> +----+
> | id |
> +----+
> | 2  |
> | 4  |
> +----+
> 2015-01-02 的温度比前一天高（10 -> 25）
> 2015-01-04 的温度比前一天高（20 -> 30）
> ```

题解

1、使用 `JOIN` 和 `DATEDIFF()` 子句

```
SELECT
    weather.id AS 'Id'
FROM
    weather
        JOIN
    weather w ON DATEDIFF(weather.date, w.date) = 1
        AND weather.Temperature > w.Temperature;
```

2、adddate（）函数方法

```sql
select a.id
    from weather a join weather b
    on (a.recorddate = adddate(b.recorddate,INTERVAL 1 day))
where a.temperature > b.temperature
```

262、行程和用户

表：`Trips`

```
+-------------+----------+
| Column Name | Type     |
+-------------+----------+
| Id          | int      |
| Client_Id   | int      |
| Driver_Id   | int      |
| City_Id     | int      |
| Status      | enum     |
| Request_at  | date     |
+-------------+----------+
Id 是这张表的主键。
这张表中存所有出租车的行程信息。每段行程有唯一 Id ，其中 Client_Id 和 Driver_Id 是 Users 表中 Users_Id 的外键。
Status 是一个表示行程状态的枚举类型，枚举成员为(‘completed’, ‘cancelled_by_driver’, ‘cancelled_by_client’) 。
```

表：`Users`

```
+-------------+----------+
| Column Name | Type     |
+-------------+----------+
| Users_Id    | int      |
| Banned      | enum     |
| Role        | enum     |
+-------------+----------+
Users_Id 是这张表的主键。
这张表中存所有用户，每个用户都有一个唯一的 Users_Id ，Role 是一个表示用户身份的枚举类型，枚举成员为 (‘client’, ‘driver’, ‘partner’) 。
Banned 是一个表示用户是否被禁止的枚举类型，枚举成员为 (‘Yes’, ‘No’) 。
```

> 写一段 SQL 语句查出 "2013-10-01" 至 "2013-10-03" 期间非禁止用户（乘客和司机都必须未被禁止）的取消率。非禁止用户即 Banned 为 No 的用户，禁止用户即 Banned 为 Yes 的用户。
>
> 取消率 的计算方式如下：(被司机或乘客取消的非禁止用户生成的订单数量) / (非禁止用户生成的订单总数)。
>
> 返回结果表中的数据可以按任意顺序组织。其中取消率 Cancellation Rate 需要四舍五入保留 两位小数
>
> ```
> Trips 表：
> +----+-----------+-----------+---------+---------------------+------------+
> | Id | Client_Id | Driver_Id | City_Id | Status              | Request_at |
> +----+-----------+-----------+---------+---------------------+------------+
> ```

```
| 1  | 1         | 10        | 1       | completed           | 2013-10-01 |
| 2  | 2         | 11        | 1       | cancelled_by_driver | 2013-10-01 |
| 3  | 3         | 12        | 6       | completed           | 2013-10-01 |
| 4  | 4         | 13        | 6       | cancelled_by_client | 2013-10-01 |
| 5  | 1         | 10        | 1       | completed           | 2013-10-02 |
| 6  | 2         | 11        | 6       | completed           | 2013-10-02 |
| 7  | 3         | 12        | 6       | completed           | 2013-10-02 |
| 8  | 2         | 12        | 12      | completed           | 2013-10-03 |
| 9  | 3         | 10        | 12      | completed           | 2013-10-03 |
| 10 | 4         | 13        | 12      | cancelled_by_driver | 2013-10-03 |
+----+-----------+-----------+---------+---------------------+------------+

Users 表：
+----------+--------+--------+
| Users_Id | Banned | Role   |
+----------+--------+--------+
| 1        | No     | client |
| 2        | Yes    | client |
| 3        | No     | client |
| 4        | No     | client |
| 10       | No     | driver |
| 11       | No     | driver |
| 12       | No     | driver |
| 13       | No     | driver |
+----------+--------+--------+

Result 表：
+------------+-------------------+
| Day        | Cancellation Rate |
+------------+-------------------+
| 2013-10-01 | 0.33              |
| 2013-10-02 | 0.00              |
| 2013-10-03 | 0.50              |
+------------+-------------------+

2013-10-01：
  - 共有 4 条请求，其中 2 条取消。
  - 然而，Id=2 的请求是由禁止用户（User_Id=2）发出的，所以计算时应当忽略它。
  - 因此，总共有 3 条非禁止请求参与计算，其中 1 条取消。
  - 取消率为 (1 / 3) = 0.33
2013-10-02：
  - 共有 3 条请求，其中 0 条取消。
  - 然而，Id=6 的请求是由禁止用户发出的，所以计算时应当忽略它。
  - 因此，总共有 2 条非禁止请求参与计算，其中 0 条取消。
  - 取消率为 (0 / 2) = 0.00
2013-10-03：
  - 共有 3 条请求，其中 1 条取消。
  - 然而，Id=8 的请求是由禁止用户发出的，所以计算时应当忽略它。
  - 因此，总共有 2 条非禁止请求参与计算，其中 1 条取消。
  - 取消率为 (1 / 2) = 0.50
```

题解

1、首先确定被禁止用户的行程记录，再剔除这些行程记录。

- 行程表中，字段 client_id 和 driver_id，都与用户表中的 users_id 关联。因此只要 client_id 和 driver_id 中有一个被禁止了，此条行程记录要被剔除。

```
SELECT T.request_at AS `Day`,
    ROUND(
            SUM(
                IF(T.STATUS = 'completed',0,1)
            )
            /
            COUNT(T.STATUS),
            2
    ) AS `Cancellation Rate`
FROM Trips AS T
JOIN Users AS U1 ON (T.client_id = U1.users_id AND U1.banned ='No')
JOIN Users AS U2 ON (T.driver_id = U2.users_id AND U2.banned ='No')
WHERE T.request_at BETWEEN '2013-10-01' AND '2013-10-03'
GROUP BY T.request_at
```

2、采用不同的方法排除掉被禁止用户的行程记录

- 想到排除，就联想到集合差。client_id 和 driver_id 的全部为集合 U。被禁止的 users_id 集合为 A。U 减去 A 的结果为没被禁止的用户。

```
SELECT T.request_at AS `Day`,
    ROUND(
            SUM(
                IF(T.STATUS = 'completed',0,1)
            )
            /
            COUNT(T.STATUS),
            2
    ) AS `Cancellation Rate`
FROM trips AS T LEFT JOIN
(
    SELECT users_id
    FROM users
    WHERE banned = 'Yes'
) AS A ON (T.Client_Id = A.users_id)
LEFT JOIN (
    SELECT users_id
    FROM users
    WHERE banned = 'Yes'
) AS A1
ON (T.Driver_Id = A1.users_id)
WHERE A.users_id IS NULL AND A1.users_id IS NULL AND T.request_at BETWEEN '2013-
10-01' AND '2013-10-03'
GROUP BY T.request_at
```

3、找出被禁止的用户后，不再连接行程表和用户表，直接从行程表中排除掉被被禁止用户的行程记录。

```
SELECT T.request_at AS `Day`,
    ROUND(
            SUM(
                IF(T.STATUS = 'completed',0,1)
            )
            /
            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
```

595、大的国家

这里有张 `World` 表

```
+-----------------+------------+------------+--------------+---------------+
| name            | continent  | area       | population   | gdp           |
+-----------------+------------+------------+--------------+---------------+
| Afghanistan     | Asia       | 652230     | 25500100     | 20343000      |
| Albania         | Europe     | 28748      | 2831741      | 12960000      |
| Algeria         | Africa     | 2381741    | 37100000     | 188681000     |
| Andorra         | Europe     | 468        | 78115        | 3712000       |
| Angola          | Africa     | 1246700    | 20609294     | 100990000     |
+-----------------+------------+------------+--------------+---------------+
```

> 如果一个国家的面积超过 300 万平方公里，或者人口超过 2500 万，那么这个国家就是大国家。
>
> 编写一个 SQL 查询，输出表中所有大国家的名称、人口和面积。
>
> 例如，根据上表，我们应该输出:
>
> ```
> +--------------+-------------+--------------+
> | name         | population  | area         |
> +--------------+-------------+--------------+
> | Afghanistan  | 25500100    | 652230       |
> | Algeria      | 37100000    | 2381741      |
> +--------------+-------------+--------------+
> ```

题解

1、where语句

```
select name,population,area
from World
where area>3000000 or population>25000000;
```

2、使用 `WHERE` 子句和 `UNION`

```sql
SELECT
    name, population, area
FROM
    world
WHERE
    area > 3000000

UNION

SELECT
    name, population, area
FROM
    world
WHERE
    population > 25000000;
```

596、超过5名学生的课

有一个`courses`表，有: **student (学生)** 和 **class (课程)**。

请列出所有超过或等于5名学生的课。

```
+---------+------------+
| student | class      |
+---------+------------+
| A       | Math       |
| B       | English    |
| C       | Math       |
| D       | Biology    |
| E       | Math       |
| F       | Computer   |
| G       | Math       |
| H       | Math       |
| I       | Math       |
+---------+------------+
```

```
+---------+
| class   |
+---------+
| Math    |
+---------+
```

题解

1、使用`GROUP BY`子句和子查询

- 先统计每门课程的学生数量，再从中选择超过 5 名学生的课程。

```sql
SELECT
    class
FROM
    (SELECT
        class, COUNT(DISTINCT student) AS num
    FROM
        courses
    GROUP BY class) AS temp_table
WHERE
    num >= 5;
```

2、使用 `GROUP BY` 和 `HAVING` 条件

```sql
SELECT
    class
FROM
    courses
GROUP BY class
HAVING COUNT(DISTINCT student) >= 5;
```

601、体育馆的人流量

表：`Stadium`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| visit_date    | date    |
| people        | int     |
+---------------+---------+
visit_date 是表的主键
每日人流量信息被记录在这三列信息中：序号 (id)、日期 (visit_date)、 人流量 (people)
每天只有一行记录，日期随着 id 的增加而增加
```

> 编写一个 SQL 查询以找出每行的人数大于或等于 100 且 id 连续的三行或更多行记录。
>
> 返回按 visit_date 升序排列的结果表。
>
> 查询结果格式如下所示。
>
> ```
> Stadium table:
> +------+------------+-----------+
> | id   | visit_date | people    |
> +------+------------+-----------+
> | 1    | 2017-01-01 | 10        |
> | 2    | 2017-01-02 | 109       |
> | 3    | 2017-01-03 | 150       |
> | 4    | 2017-01-04 | 99        |
> | 5    | 2017-01-05 | 145       |
> | 6    | 2017-01-06 | 1455      |
> | 7    | 2017-01-07 | 199       |
> ```

```
| 8    | 2017-01-09 | 188       |
+------+------------+-----------+

Result table:
+------+------------+-----------+
| id   | visit_date | people    |
+------+------------+-----------+
| 5    | 2017-01-05 | 145       |
| 6    | 2017-01-06 | 1455      |
| 7    | 2017-01-07 | 199       |
| 8    | 2017-01-09 | 188       |
+------+------------+-----------+
id 为 5、6、7、8 的四行 id 连续，并且每行都有 >= 100 的人数记录。
请注意，即使第 7 行和第 8 行的 visit_date 不是连续的，输出也应当包含第 8 行，因为我们只需要考虑 id 连续的记录。
不输出 id 为 2 和 3 的行，因为至少需要三条 id 连续的记录。
```

题解

1、使用 `JOIN` 和 `WHERE` 子句

1. 查询人流量超过 100 的记录，然后将结果与其自身的临时表连接。

```
select distinct t1.*
from stadium t1, stadium t2, stadium t3
where t1.people >= 100 and t2.people >= 100 and t3.people >= 100;
```

2. 表 t1，t2 和 t3 相同，需要考虑添加哪些条件能够得到想要的结果。以 t1 为例，它有可能是高峰期的第 1 天，第 2 天，或第 3 天。
   - t1 是高峰期第 1 天：`(t1.id - t2.id = 1 and t1.id - t3.id = 2 and t2.id - t3.id =1) -- t1, t2, t3`
   - t1 是高峰期第 2 天：`(t2.id - t1.id = 1 and t2.id - t3.id = 2 and t1.id - t3.id =1) -- t2, t1, t3`
   - t1 是高峰期第 3 天：`(t3.id - t2.id = 1 and t2.id - t1.id =1 and t3.id - t1.id = 2) -- t3, t2, t1`

```
select 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
);
```

3. 可以看到查询结果中存在重复的记录，再使用 `DISTINCT` 去重。

```
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;
```

2、窗口函数

1. 找到>100的行并给不同的连续id打标
2. 用count(*)over(partition by t_rank)计算每个标签的个数，筛选出>2的标签

```
select id,visit_date,people
from(
    select *,count(*)over(partition by t_rank) t2_rank
    from(
        select *,id-row_number()over(order by id) t_rank
        from stadium
        where people>99
        )t
    )t2
where t2.t2_rank>2
```

3、with临时空间

- 提前用With临时空间，是因为where子句中需要再次调用from中选取的表
- 对我们得出的唯一的rk进行group by 和having 筛选（注意having是SQL中唯一的剪裁表的action）

```
with t1 as(
    select *,id - row_number() over(order by id) as rk
    from stadium
    where people >= 100
)

select id,visit_date,people
from t1
where rk in(
    select rk
    from t1
    group by rk
    having count(rk) >= 3
)
```

4、加了两列用来做标记，先用一列来算连续出现的情况，针对新加的上一列倒过来再算一个分组标记。

1. 先用查询算出连续不小于 100 出现的统计，记为countt（小于 100 的值为0，不小于的值在上一次的基础上加一）。

2. 对第1步的结果增加一个标记位flag，倒叙看countt，不小于3或上一flag为1，并且countt不等于0的，标记flag为1
3. 对第2步的结构查询，找出flag为1的就好，排序倒回来

```
SELECT id, visit_date, people
FROM (
    SELECT r1.*, @flag := if((r1.countt >= 3 OR @flag = 1) AND r1.countt != 0,
1, 0) AS flag
    FROM (
        SELECT s.*, @count := if(s.people >= 100, @count + 1, 0) AS `countt`
        FROM stadium s, (SELECT @count := 0) b
    ) r1, (SELECT @flag := 0) c
    ORDER BY id DESC
) result
WHERE flag = 1 ORDER BY id;
```

620、有趣的电影

某城市开了一家新的电影院，吸引了很多人过来看电影。该电影院特别注意用户体验，专门有个 LED显示板做电影推荐，上面公布着影评和相关电影描述。

作为该电影院的信息部主管，您需要编写一个 SQL查询，找出所有影片描述为非 boring (不无聊) 的并且 id 为奇数 的影片，结果请按等级 rating 排列。

> 下表 `cinema`:
>
> ```
> +---------+-----------+--------------+-----------+
> |   id    | movie     |  description |  rating   |
> +---------+-----------+--------------+-----------+
> |   1     | War       |   great 3D   |   8.9     |
> |   2     | Science   |   fiction    |   8.5     |
> |   3     | irish     |   boring     |   6.2     |
> |   4     | Ice song  |   Fantacy    |   8.6     |
> |   5     | House card|   Interesting|   9.1     |
> +---------+-----------+--------------+-----------+
> ```
>
> 对于上面的例子，则正确的输出是为:
>
> ```
> +---------+-----------+--------------+-----------+
> |   id    | movie     |  description |  rating   |
> +---------+-----------+--------------+-----------+
> |   5     | House card|   Interesting|   9.1     |
> |   1     | War       |   great 3D   |   8.9     |
> +---------+-----------+--------------+-----------+
> ```

题解

1、使用 `MOD()`

```
select *
from cinema
where mod(id, 2) = 1 and description != 'boring'
order by rating DESC;
```

2、位运算

```
# Write your MySQL query statement below
select id, movie, description, rating
from cinema
where description <> 'boring' and id & 1
order by rating desc;
```

626、换座位

小美是一所中学的信息科技老师，她有一张 seat 座位表，平时用来储存学生名字和与他们相对应的座位 id。

其中纵列的 id 是连续递增的

小美想改变相邻俩学生的座位。

如果学生人数是奇数，则不需要改变最后一个同学的座位。

> ```
> +---------+---------+
> |    id   | student |
> +---------+---------+
> |    1    | Abbot   |
> |    2    | Doris   |
> |    3    | Emerson |
> |    4    | Green   |
> |    5    | Jeames  |
> +---------+---------+
> ```
>
> 假如数据输入的是上表，则输出结果如下：
>
> ```
> +---------+---------+
> |    id   | student |
> +---------+---------+
> |    1    | Doris   |
> |    2    | Abbot   |
> |    3    | Green   |
> |    4    | Emerson |
> |    5    | Jeames  |
> +---------+---------+
> ```

题解

1、使用 `CASE`

- 对于所有座位 id 是奇数的学生，修改其 id 为 id+1，如果最后一个座位 id 也是奇数，则最后一个座位 id 不修改。对于所有座位 id 是偶数的学生，修改其 id 为 id-1。

```
SELECT
    (CASE
        WHEN MOD(id, 2) != 0 AND counts != id THEN id + 1
        WHEN MOD(id, 2) != 0 AND counts = id THEN id
        ELSE id - 1
    END) AS id,
    student
FROM
    seat,
    (SELECT
        COUNT(*) AS counts
    FROM
        seat) AS seat_counts
ORDER BY id ASC;
```

2、使用位操作和 `COALESCE()`

- 使用 `(id+1)^1-1` 计算交换后每个学生的座位 id。

```
SELECT id, (id+1)^1-1, student FROM seat;
```

```
| id | (id+1)^1-1 | student |
|----|------------|---------|
| 1  | 2          | Abbot   |
| 2  | 1          | Doris   |
| 3  | 4          | Emerson |
| 4  | 3          | Green   |
| 5  | 6          | Jeames  |
```

```
SELECT
    s1.id, COALESCE(s2.student, s1.student) AS student
FROM
    seat s1
        LEFT JOIN
    seat s2 ON ((s1.id + 1) ^ 1) - 1 = s2.id
ORDER BY s1.id;
```

3、同1，另一种写法

```
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;
```

627、变更性别

给定一个 salary 表，如下所示，有 m = 男性 和 f = 女性 的值。交换所有的 f 和 m 值（例如，将所有 f 值更改为 m，反之亦然）。要求只使用一个更新（Update）语句，并且没有中间的临时表。

注意，您必只能写一个 Update 语句，请不要编写任何 Select 语句。

> ```
> | id | name | sex | salary |
> |----|------|-----|--------|
> | 1  | A    | m   | 2500   |
> | 2  | B    | f   | 1500   |
> | 3  | C    | m   | 5500   |
> | 4  | D    | f   | 500    |
> ```
>
> 运行你所编写的更新语句之后，将会得到以下表:
>
> ```
> | id | name | sex | salary |
> |----|------|-----|--------|
> | 1  | A    | f   | 2500   |
> | 2  | B    | m   | 1500   |
> | 3  | C    | f   | 5500   |
> | 4  | D    | m   | 500    |
> ```

题解

1、使用 `UPDATE` 和 `CASE...WHEN`

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

2、IF函数

```
UPDATE salary SET sex = IF(sex = 'm', 'f', 'm');
```

3、ASCII码

```
UPDATE salary
SET sex = char(ascii('f') + ascii('m') - ascii(sex))
```

1179、重新格式化部门表

部门表 `Department`：

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| revenue       | int     |
| month         | varchar |
+---------------+---------+
```
(id, month) 是表的联合主键。
这个表格有关于每个部门每月收入的信息。
月份（month）可以取下列值
["Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"]。

> 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：
>
> Department 表：
> ```
> +------+---------+-------+
> | id   | revenue | month |
> +------+---------+-------+
> | 1    | 8000    | Jan   |
> | 2    | 9000    | Jan   |
> | 3    | 10000   | Feb   |
> | 1    | 7000    | Feb   |
> | 1    | 6000    | Mar   |
> +------+---------+-------+
> ```
>
> 查询得到的结果表：
> ```
> +------+-------------+-------------+-------------+-----+-------------+
> | id   | Jan_Revenue | Feb_Revenue | Mar_Revenue | ... | Dec_Revenue |
> +------+-------------+-------------+-------------+-----+-------------+
> | 1    | 8000        | 7000        | 6000        | ... | null        |
> | 2    | 9000        | null        | null        | ... | null        |
> | 3    | null        | 10000       | null        | ... | null        |
> +------+-------------+-------------+-------------+-----+-------------+
> ```
>
> 注意，结果表有 13 列 (1个部门 id 列 + 12个月份的收入列)。

题解

1、行合并

```sql
select id,
    sum(case month when 'Jan' then revenue end) as Jan_Revenue,
    sum(case month when 'Feb' then revenue end) as Feb_Revenue,
    sum(case month when 'Mar' then revenue end) as Mar_Revenue,
    sum(case month when 'Apr' then revenue end) as Apr_Revenue,
    sum(case month when 'May' then revenue end) as May_Revenue,
    sum(case month when 'Jun' then revenue end) as Jun_Revenue,
    sum(case month when 'Jul' then revenue end) as Jul_Revenue,
    sum(case month when 'Aug' then revenue end) as Aug_Revenue,
    sum(case month when 'Sep' then revenue end) as Sep_Revenue,
    sum(case month when 'Oct' then revenue end) as Oct_Revenue,
    sum(case month when 'Nov' then revenue end) as Nov_Revenue,
```

```
    sum(case month when 'Dec' then revenue end) as Dec_Revenue
from Department
group by id
```

2、列拆成行

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

```
SELECT id,
IF(`month`='Jan',revenue,NULL) Jan_Revenue,
IF(`month`='Feb',revenue,NULL) Feb_Revenue,
IF(`month`='Mar',revenue,NULL) Mar_Revenue,
IF(`month`='Apr',revenue,NULL) Apr_Revenue,
IF(`month`='May',revenue,NULL) May_Revenue,
IF(`month`='Jun',revenue,NULL) Jun_Revenue,
IF(`month`='Jul',revenue,NULL) Jul_Revenue,
IF(`month`='Aug',revenue,NULL) Aug_Revenue,
IF(`month`='Sep',revenue,NULL) Sep_Revenue,
IF(`month`='Oct',revenue,NULL) Oct_Revenue,
IF(`month`='Nov',revenue,NULL) Nov_Revenue,
IF(`month`='Dec',revenue,NULL) Dec_Revenue
FROM Department;
```