链滴

业务场景 SQL 练习

作者：Hefery

原文链接：https://ld246.com/article/1641870768056

来源网站：链滴

公司场景

176.第二高的薪水

题目

获取 Employee 表中第二高的薪水（Salary）

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

上述 Employee 表，SQL应该返回 200 作为第二高的薪水。如果不存在第二高的薪水，那么查询应回 null

```
+---------------------+
| SecondHighestSalary |
+---------------------+
| 200                 |
+---------------------+
```

题解

ORDER BY DESC + DISTINCT + LIMIT

薪资降序排序，然后使用 LIMIT 子句获得第二高的薪资

PS

- 去重（DISTINCT）
- 考虑第二高的薪资不存在的情况

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

# 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

题目

获取 Employee 表中第n高的薪水（Salary）

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

上述 Employee 表，n = 2时，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 nu
l

```
+----------------------+
| getNthHighestSalary(2) |
+----------------------+
| 200                  |
+----------------------+
```

题解

ORDER BY DESC + DISTINCT + LIMIT

薪资降序排序，然后使用 LIMIT 子句获得第二高的薪资

PS

- 去重（DISTINCT）
- 考虑第二高的薪资不存在的情况
- LIMIT里面不能做运算，所以要处理下N的值

```
CREATE FUNCTION getNthHighestSalary(N INT) RETURNS INT
BEGIN
  SET n = N-1;
  RETURN (
      SELECT (
          IFNULL(
              (SELECT DISTINCT Salary FROM Employee ORDER BY Salary DESC LIMIT n,1), NULL
          )
      )
  );
END
```

181. 超过经理收入的员工

题目

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

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

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员工

```
+----------+
| Employee |
+----------+
| Joe      |
+----------+
```

题解

自查询

```
# WHERE
SELECT
    a.name AS Employee
FROM
    Employee AS a,
    Employee AS b
WHERE
    a.ManagerId = b.Id AND a.Salary > b.Salary

# JOIN，更常用也更有效
SELECT a.name AS Employee
FROM Employee AS a
JOIN Employee AS b
ON a.ManagerId = b.Id AND a.Salary > b.Salary
```

184. 部门工资最高的员工

题述

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SQL，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（行的顺序无紧要）

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

PS：Max 和 Jim 在 IT 部门的工资都是最高的，Henry 在销售部的工资最高

题解

使用 JOIN 和 IN 语句

```
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
    )
```

185.部门工资前三高的所有员工

题述

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 epartmentId

```
+----+-------+--------+--------------+
| 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    |
+----+----------+
```

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  |
+------------+----------+--------+
```

PS：IT 部门中，Max 获得了最高的工资，Randy 和 Joe 都拿到了第二高的工资，Will 的工资排第三
销售部门（Sales）只有两名员工，Henry 的工资最高，Sam 的工资排第二

题解

使用 JOIN 和子查询

```
# JOIN + 子查询
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 )

# 子查询
```

```
SELECT
    Department.NAME AS Department,
    e1.NAME AS Employee,
    e1.Salary AS Salary
FROM
    Employee AS e1,
        Department
WHERE
    e1.DepartmentId = Department.Id
    AND 3 > (SELECT count( DISTINCT e2.Salary )
         FROM Employee AS e2
         WHERE    e1.Salary < e2.Salary AND e1.DepartmentId = e2.DepartmentId)
ORDER BY Department.NAME,Salary DESC;

# dense_rank函数，找到每个部门最高，然后取dense_rank<=3的结果
SELECT
    B.name AS Department,
    A.Employee,
    A.Salary
FROM
    (
        SELECT
            DepartmentId,
            name AS employee,
            salary,
            dense_rank() over (partition by departmentid order by salary desc) as rk
        FROM employee
    ) AS A
LEFT JOIN department B ON A.departmentid = B.id
WHERE A.rk <= 3
```

183.从不订购的客户

题述

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客户

Customers 表:

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

Orders 表:

```
+----+------------+
| Id | CustomerId |
```

```
+----+------------+
| 1  | 3          |
| 2  | 1          |
+----+------------+
```

查询应返回：

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

题解

使用子查询和 NOT IN 子句

```
SELECT c.name AS Customers
FROM Customers AS c
WHERE c.id NOT IN
    (
        SELECT CustomerId FROM Orders
    )
```

182.查找重复的电子邮箱

题述

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 |
+----+---------+
```

查询应返回以下结果：

```
+---------+
| Email   |
+---------+
| a@b.com |
+---------+
PS：所有电子邮箱都是小写字母
```

题解

```
# GROUP BY + HAVING
SELECT Email
```

```
FROM Person
GROUP BY Email HAVING COUNT(Email) > 1

# 子查询
SELECT DISTINCT a.Email
FROM Person a, Person b
WHERE a.Email=b.Email AND a.Id!=b.Id
```

196.删除重复的电子邮箱

题述

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 |
+----+------------------+
```

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

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

学校场景

至少连续出现3次的字段

```
-- user_info(userId, username)  连续出现3次的username
SELECT DISTINCT
    user_info
FROM
    table_name a
WHERE
    a.username = (SELECT username FROM a.userId = userId-1)
    AND
    a.username = (SELECT username FROM a.userId = userId-2)
```