SQL 练习

笔记本：	数据分析资料		
创建时间：	2018/4/11 23:08	**更新时间：**	2018/4/27 23:17
作者：	Parus		
URL：	https://www.nowcoder.com/practice/ec1ca44c62c14ceb990c3c40def1ec6c?tpl...		

查找最晚入职员工的所有信息

```
CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,
`hire_date` date NOT NULL,
PRIMARY KEY (`emp_no`));
```

代码：

```
SELECT * FROM employees
WHERE hire_date = (SELECT MAX(hire_date) FROM employees);
```

用inner join的方法：

```
SELECT * FROM employees as A
INNER JOIN (SELECT MAX(hire_date) as hire_date FROM employees) as B on A.hire_date
= B.hire_date
```

(待验证，W3School例子尝试成功，牛客尝试失败)

如果确定记录只有一条：

```
SELECT * FROM employees
ORDER BY hire_date DESC
LIMIT 1
```

查找入职员工时间排名倒数第三的员工所有信息

```
CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,
`hire_date` date NOT NULL,
PRIMARY KEY (`emp_no`));
```

代码：

```
WITH top3 AS (SELECT * FROM employees
            ORDER BY hire_date DESC
            LIMIT 3)
```

```
SELECT * FROM top3
WHERE hire_date = (SELECT MIN(hire_date) FROM top3);
```

（ROWNUM是Oracle的）

简单version：

```
SELECT * FROM employees
ORDER BY hire_date DESC
LIMIT 2,1
```

(LIMIT m,n : 表示从第m+1条开始，取n条数据；
LIMIT n ： 表示从第0条开始，取n条数据，是limit(0,n)的缩写。
本题limit 2,1 表示从第（2+1）条数据开始，取一条数据，即入职员工时间排名倒数第三的员工。）

查找各个部门当前(to_date='9999-01-01')领导当前薪水详情以及其对应部门编号dept_no

```
CREATE TABLE `dept_manager` (
`dept_no` char(4) NOT NULL,
`emp_no` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`dept_no`));
CREATE TABLE `salaries` (
`emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`from_date`));
```

代码：

```
SELECT salaries.*, dept_manager.dept_no
FROM salaries
LEFT JOIN dept_manager
ON salaries.emp_no = dept_manager.emp_no
WHERE salaries.to_date = '9999-01-01'
AND dept_manager.to_date = '9999-01-01';
```

更简单的方法：

```
SELECT s.*, d.dept_no
FROM salaries s， dept_manager d
WHERE s.to_date='9999-01-01'
AND d.to_date='9999-01-01'
AND s.emp_no = d.emp_no;
```

查找所有已经分配部门的员工的last_name和first_name

```
CREATE TABLE `dept_emp` (
`emp_no` int(11) NOT NULL,
`dept_no` char(4) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`dept_no`));
CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,
`hire_date` date NOT NULL,
PRIMARY KEY (`emp_no`));
```

代码:

```
SELECT  employees.last_name, employees.first_name, dept_emp.dept_no
FROM employees
INNER JOIN dept_emp
ON employees.emp_no = dept_emp.emp_no;
```

查找所有员工的last_name和first_name以及对应部门编号dept_no，也包括展示没有分配具体部门的员工

```
CREATE TABLE `dept_emp` (
`emp_no` int(11) NOT NULL,
`dept_no` char(4) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`dept_no`));
CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,
`hire_date` date NOT NULL,
PRIMARY KEY (`emp_no`));
```

代码:

```
SELECT  employees.last_name, employees.first_name, dept_emp.dept_no
FROM employees
LEFT JOIN dept_emp
ON employees.emp_no = dept_emp.emp_no;
```

查找所有员工入职时候的薪水情况，给出emp_no以及salary， 并按照emp_no进行逆序

```
CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,
`hire_date` date NOT NULL,
PRIMARY KEY (`emp_no`));
CREATE TABLE `salaries` (
`emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`from_date`));
```

代码:

```
SELECT e.emp_no, s.salary
FROM employees e, salaries s
WHERE e.emp_no = s.emp_no
AND e.hire_date = s.from_date
ORDER BY e.emp_no DESC;
```

用INNER JOIN的方法（运行时间更快，占用内存更少）：

```
SELECT e.emp_no, s.salary
FROM employees e
INNER JOIN salaries s
ON e.emp_no = s.emp_no
WHERE e.hire_date = s.from_date
ORDER BY e.emp_no DESC;
```

查找薪水涨幅超过15次的员工号emp_no以及其对应的涨幅次数t

```
CREATE TABLE `salaries` (
`emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,

PRIMARY KEY (`emp_no`,`from_date`));
```

代码:

```
SELECT emp_no, COUNT(from_date) AS t
FROM salaries
GROUP BY emp_no
HAVING COUNT(from_date) > 15;
```

找出所有员工当前(to_date='9999-01-01')具体的薪水salary情况，对于相同的薪水只显示一次,并按照逆序显示

```
CREATE TABLE `salaries` (
`emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,

PRIMARY KEY (`emp_no`,`from_date`));
```

代码：

```
SELECT salary
FROM salaries
WHERE to_date = '9999-01-01'
GROUP BY salary
ORDER BY salary DESC;
```

获取所有部门当前manager的当前薪水情况，给出dept_no, emp_no以及salary，当前表示to_date='9999-01-01'

```
CREATE TABLE `dept_manager` (
`dept_no` char(4) NOT NULL,
`emp_no` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`dept_no`));
CREATE TABLE `salaries` (
`emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`from_date`));
```

代码：

```
SELECT d.dept_no, d.emp_no, s.salary
FROM salaries AS s
INNER JOIN dept_manager AS d
ON d.emp_no = s.emp_no
WHERE d.to_date = '9999-01-01'
AND s.to_date = '9999-01-01';
```

注：此题如果两表调换顺序会报错，是系统问题
当连接语句为 FROM dept_manager AS d INNER JOIN salaries AS s 时，在最后面加上 ORDER BY d.emp_no即可通过。
原因分析可能如下：连接后按照前面的第一个 KEY 值排序，若 salaries 在前，则按照 s.emp_no 排序（因为限制条件为 d.emp_no = s.emp_no，所以对 s.emp_no 排序就是对 d.emp_no 排序），输出跟参考答案一致，没问题；若 dept_manager 在前，则按照 d.dept_no排序，此时与参考答案不同，所以需要在末尾手动用 ORDER BY 对d.emp_no进行排序。

获取所有非manager的员工emp_no

```
CREATE TABLE `dept_manager` (
`dept_no` char(4) NOT NULL,
`emp_no` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`dept_no`));
CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,
`hire_date` date NOT NULL,

PRIMARY KEY (`emp_no`));
```

代码：

```
SELECT e.emp_no
FROM employees e
LEFT JOIN dept_manager d
ON e.emp_no = d.emp_no
WHERE d.emp_no IS NULL
```

获取所有员工当前的manager，如果当前的manager是自己的话结果不显示，当前表示to_date='9999-01-01'。
结果第一列给出当前员工的emp_no,第二列给出其manager对应的manager_no。

```
CREATE TABLE `dept_emp` (
`emp_no` int(11) NOT NULL,
`dept_no` char(4) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`dept_no`));
CREATE TABLE `dept_manager` (
`dept_no` char(4) NOT NULL,
`emp_no` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,

PRIMARY KEY (`emp_no`,`dept_no`));
```

代码：

```
SELECT e.emp_no, m.emp_no AS manager_no
FROM dept_emp e
LEFT JOIN dept_manager m
ON e.dept_no = m.dept_no
WHERE m.to_date = '9999-01-01'
AND e.emp_no IS NOT m.emp_no;
```

获取所有部门中当前员工薪水最高的相关信息，给出dept_no, emp_no以及其对应的salary

```
CREATE TABLE `dept_emp` (
`emp_no` int(11) NOT NULL,
`dept_no` char(4) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`dept_no`));
CREATE TABLE `salaries` (
`emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,

PRIMARY KEY (`emp_no`,`from_date`));
```

代码：

```
SELECT d.dept_no, d.emp_no, MAX(s.salary) AS salary
FROM dept_emp d
LEFT JOIN salaries s
ON d.emp_no = s.emp_no
WHERE d.to_date = '9999-01-01'
AND s.to_date = '9999-01-01'
GROUP BY d.dept_no;
```

注：

此题如考虑多条最大记录，则可

1、创建两张表，一张为maxsalary，用于存放当前每个部门薪水的最大值；另一张为currentsalary，用于存放当前每个部门所有员工的编号和薪水；

2、限定条件为两张表的 dept_no 和 salary 相等，这样就可以找出当前每个部门所有薪水等于最大值的员工的相关信息了；

3、最后记得根据 currentsalary.dept_no 升序排列，输出与参考答案相同的记录表。

```
SELECT currentsalary.dept_no, currentsalary.emp_no, currentsalary.salary AS salary
FROM
//创建maxsalary表用于存放当前每个部门薪水的最大值
(SELECT d.dept_no, MAX(s.salary) AS salary
FROM salaries AS s INNER JOIN dept_emp As d
ON d.emp_no = s.emp_no
WHERE d.to_date = '9999-01-01' AND s.to_date = '9999-01-01'
GROUP BY d.dept_no) AS maxsalary,
//创建currentsalary表用于存放当前每个部门所有员工的编号和薪水
(SELECT d.dept_no, s.emp_no, s.salary
FROM salaries AS s INNER JOIN dept_emp As d
ON d.emp_no = s.emp_no
WHERE d.to_date = '9999-01-01' AND s.to_date = '9999-01-01'
) AS currentsalary
//限定条件为两表的dept_no和salary均相等
WHERE currentsalary.dept_no = maxsalary.dept_no
AND currentsalary.salary = maxsalary.salary
//最后以currentsalary.dept_no排序输出符合要求的记录表
ORDER BY currentsalary.dept_no
```

从titles表获取按照title进行分组，每组个数大于等于2，给出title以及对应的数目t。

```
CREATE TABLE IF NOT EXISTS "titles" (
`emp_no` int(11) NOT NULL,
`title` varchar(50) NOT NULL,
`from_date` date NOT NULL,

`to_date` date DEFAULT NULL);
```

代码：

```
SELECT title, COUNT(emp_no)
FROM titles
GROUP BY title
HAVING COUNT(title) >= 2;
```

从titles表获取按照title进行分组，每组个数大于等于2，给出title以及对应的数目t。注意对于重复的emp_no进行忽略。

```
CREATE TABLE IF NOT EXISTS "titles" (
`emp_no` int(11) NOT NULL,
`title` varchar(50) NOT NULL,
`from_date` date NOT NULL,

`to_date` date DEFAULT NULL);
```

代码：

```
SELECT title, COUNT(DISTINCT emp_no) AS t
FROM titles
GROUP BY title
HAVING t >= 2;
```

查找employees表所有emp_no为奇数，且last_name不为Mary的员工信息，并按照hire_date逆序排列

```
CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,
`hire_date` date NOT NULL,
PRIMARY KEY (`emp_no`));
```

代码：

```
SELECT *
FROM employees
WHERE emp_no %2=1
AND last_name NOT LIKE '%Mary'
ORDER BY hire_date DESC;
```

统计出当前各个title类型对应的员工当前薪水对应的平均工资。结果给出title以及平均工资avg。

```
CREATE TABLE `salaries` (
`emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`from_date`));
CREATE TABLE IF NOT EXISTS "titles" (
`emp_no` int(11) NOT NULL,
`title` varchar(50) NOT NULL,
`from_date` date NOT NULL,
`to_date` date DEFAULT NULL);
```

代码:

```
SELECT t.title, AVG(s.salary)
FROM titles t
INNER JOIN salaries s
ON t.emp_no = s.emp_no
WHERE t.to_date = '9999-01-01'
AND s.to_date = '9999-01-01'
GROUP BY t.title;
```

获取当前（to_date='9999-01-01'）薪水第二多的员工的emp_no以及其对应的薪水salary

```
CREATE TABLE `salaries` (
`emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`from_date`));
```

代码:

```
SELECT emp_no, salary
FROM salaries
WHERE to_date = '9999-01-01'
ORDER BY salary DESC
LIMIT 1,1;
```

查找当前薪水(to_date='9999-01-01')排名第二多的员工编号emp_no、薪水salary、last_name以及first_name，不准使用order by

```
CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,
`hire_date` date NOT NULL,
PRIMARY KEY (`emp_no`));
CREATE TABLE `salaries` (
`emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`from_date`));
```

代码:

```
WITH exceptMax AS (SELECT e.*, s.*
               FROM employees e
```

```
                LEFT JOIN salaries s
                ON e.emp_no = s.emp_no
                WHERE s.salary IS NOT (SELECT MAX(salary) FROM salaries))
SELECT emp_no, salary, last_name, first_name
FROM exceptMax
WHERE salary = (SELECT MAX(salary) FROM exceptMax)
AND to_date = '9999-01-01';
```

网友version (用到了IN/NOT IN)：

```
SELECT e.emp_no, MAX(s.salary) AS salary, e.last_name, e.first_name
FROM employees AS e INNER JOIN salaries AS s
ON e.emp_no = s.emp_no
WHERE s.to_date = '9999-01-01'
AND s.salary NOT IN (SELECT MAX(salary) FROM salaries WHERE to_date =
'9999-01-01')
```

查找所有员工的last_name和first_name以及对应的dept_name，也包括暂时没有分配部门的员工

```
CREATE TABLE `departments` (
`dept_no` char(4) NOT NULL,
`dept_name` varchar(40) NOT NULL,
PRIMARY KEY (`dept_no`));
CREATE TABLE `dept_emp` (
`emp_no` int(11) NOT NULL,
`dept_no` char(4) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`dept_no`));
CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,
`hire_date` date NOT NULL,

PRIMARY KEY (`emp_no`));
```

代码：

```
SELECT e.last_name, e.first_name, d.dept_name
FROM employees e
LEFT JOIN (SELECT departments.dept_name, dept_emp.emp_no
        FROM departments
        LEFT JOIN dept_emp
        ON departments.dept_no = dept_emp.dept_no) AS d

ON e.emp_no = d.emp_no;
```

查找员工编号emp_now为10001其自入职以来的薪水salary涨幅值growth

```
CREATE TABLE `salaries` (
`emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,

PRIMARY KEY (`emp_no`,`from_date`));
```

代码：

```
SELECT MAX(salary)-MIN(salary)
FROM salaries

WHERE emp_no = 10001;
```

（有点难）
查找所有员工自入职以来的薪水涨幅情况，给出员工编号emp_noy以及其对应的薪水涨幅growth，并按照growth进行升序

```
CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,
`hire_date` date NOT NULL,
PRIMARY KEY (`emp_no`));
CREATE TABLE `salaries` (
`emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,

PRIMARY KEY (`emp_no`,`from_date`));
```

代码：

```
SELECT hire.emp_no, cur.salary-hire.salary AS growth
FROM (SELECT e.emp_no, s.salary FROM employees e LEFT JOIN salaries s
    ON e.emp_no=s.emp_no AND e.hire_date = s.from_date) AS hire
INNER JOIN (SELECT emp_no, salary FROM salaries WHERE to_date = '9999-01-
01') AS cur
ON hire.emp_no = cur.emp_no

ORDER BY growth ASC;
```

统计各个部门对应员工涨幅的次数总和，给出部门编码dept_no、部门名称dept_name以及次数sum

```
CREATE TABLE `departments` (
`dept_no` char(4) NOT NULL,
`dept_name` varchar(40) NOT NULL,
PRIMARY KEY (`dept_no`));
CREATE TABLE `dept_emp` (
`emp_no` int(11) NOT NULL,
`dept_no` char(4) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`dept_no`));
CREATE TABLE `salaries` (
`emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,

PRIMARY KEY (`emp_no`,`from_date`));
```

代码：

```
SELECT d.dept_no, d.dept_name, COUNT(ds.emp_no)
FROM departments d
INNER JOIN (salaries s INNER JOIN dept_emp de
                    ON de.emp_no = s.emp_no) AS ds
ON d.dept_no = ds.dept_no

GROUP BY ds.dept_no;
```

（没必要在连接两个表的时候每次都考虑取哪些列，全并在一起更简单）

（难）
对所有员工的当前(to_date='9999-01-01')薪水按照salary进行按照1-N的排名，相同salary并列且按照emp_no升序排列

```
CREATE TABLE `salaries` (
`emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,

PRIMARY KEY (`emp_no`,`from_date`));
```

代码：

```
SELECT s1.emp_no, s1.salary, COUNT(DISTINCT s2.salary) AS rank
FROM salaries AS s1, salaries AS s2
WHERE s1.to_date = '9999-01-01'
AND s2.to_date = '9999-01-01'
AND s1.salary <= s2.salary
GROUP BY s1.emp_no

ORDER BY s1.salary DESC, s1.emp_no ASC
```

解析：
本题的精髓在于 **s1.salary <= s2.salary，意思是在输出s1.salary的情况下，有多少个s2.salary大于等于s1.salary**，比如当s1.salary=94409时，有3个s2.salary（分别为94692,94409,94409）大于等于它，但由于94409重复，利用COUNT(DISTINCT s2.salary)去重可得工资为94409的rank等于2。其余排名以此类推。

最后在支持ROW_NUMBER、RANK、DENSE_RANK等函数的SQL Server数据库中，有以下参考代码，可惜在本题的SQLite数据库中不支持。

```
SELECT emp_no, salaries, DENSE_RANK() OVER(ORDER BY salary DESC) AS
rank
WHERE to_date = '9999-01-01' ORDER BY salary DESC, emp_no ASC
```

获取所有非manager员工当前的薪水情况，给出dept_no、emp_no以及salary ，当前表示to_date='9999-01-01'

```
CREATE TABLE `dept_emp` (
`emp_no` int(11) NOT NULL,
`dept_no` char(4) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`dept_no`));
CREATE TABLE `dept_manager` (
`dept_no` char(4) NOT NULL,
`emp_no` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`dept_no`));
CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,
`hire_date` date NOT NULL,
PRIMARY KEY (`emp_no`));
CREATE TABLE `salaries` (
`emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,

PRIMARY KEY (`emp_no`,`from_date`));
```

代码:

```
SELECT ds.dept_no, nm.emp_no, ds.salary
FROM (SELECT e.*, m.* FROM employees e
    LEFT JOIN dept_manager m
    ON e.emp_no = m.emp_no
    WHERE m.emp_no IS NULL) AS nm
INNER JOIN (SELECT d.*, s.* FROM dept_emp d
        INNER JOIN salaries s
        ON d.emp_no = s.emp_no
        WHERE s.to_date = '9999-01-01') AS ds
ON nm.emp_no = ds.emp_no;
```

获取员工其当前的薪水比其manager当前薪水还高的相关信息，当前表示to_date='9999-01-01',
结果第一列给出员工的emp_no,
第二列给出其manager的manager_no,
第三列给出该员工当前的薪水emp_salary,
第四列给该员工对应的manager当前的薪水manager_salary

```
CREATE TABLE `dept_emp` (
`emp_no` int(11) NOT NULL,
`dept_no` char(4) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`dept_no`));
CREATE TABLE `dept_manager` (
`dept_no` char(4) NOT NULL,
`emp_no` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`dept_no`));
CREATE TABLE `salaries` (
`emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`from_date`));
```

代码：

```
SELECT es.emp_no AS emp_no, ms.emp_no AS manager_no,
       es.salary AS emp_salart, ms.salary AS manager_salary
FROM (SELECT e.emp_no, s.salary, e.dept_no FROM dept_emp e
      INNER JOIN salaries s
      ON e.emp_no = s.emp_no
      WHERE s.to_date = '9999-01-01') AS es,
     (SELECT m.emp_no, s.salary, m.dept_no FROM dept_manager m
      INNER JOIN salaries s
      ON m.emp_no = s.emp_no
      WHERE s.to_date = '9999-01-01') AS ms
WHERE es.dept_no = ms.dept_no
AND es.salary > ms.salary
```

汇总各个部门当前员工的title类型的分配数目，结果给出部门编号dept_no、
dept_name、其当前员工所有的title以及该类型title对应的数目count

```
CREATE TABLE `departments` (
`dept_no` char(4) NOT NULL,
`dept_name` varchar(40) NOT NULL,
PRIMARY KEY (`dept_no`));
CREATE TABLE `dept_emp` (
`emp_no` int(11) NOT NULL,
`dept_no` char(4) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`dept_no`));
CREATE TABLE IF NOT EXISTS "titles" (
`emp_no` int(11) NOT NULL,
`title` varchar(50) NOT NULL,
`from_date` date NOT NULL,

`to_date` date DEFAULT NULL);
```

代码：

```
SELECT d.dept_no, d.dept_name, et.title, COUNT(et.emp_no)
FROM departments d
LEFT JOIN (SELECT e.*, t.* FROM dept_emp e
           LEFT JOIN titles t
           ON e.emp_no = t.emp_no
```

```
        WHERE e.to_date = '9999-01-01' AND t.to_date = '9999-01-01') AS et
ON d.dept_no = et.dept_no

GROUP BY et.dept_no,et.title;
```

给出每个员工每年薪水涨幅超过5000的员工编号emp_no、薪水变更开始日期from_date以及薪水涨幅值salary_growth，并按照salary_growth逆序排列。
提示：在sqlite中获取datetime时间对应的年份函数为strftime('%Y', to_date)

```
CREATE TABLE `salaries` (
`emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,

PRIMARY KEY (`emp_no`,`from_date`));
```

代码：

```
SELECT a.emp_no, a.from_date, a.salary-b.salary AS growth
FROM salaries a, salaries b
WHERE strftime('%Y', a.to_date)-strftime('%Y', b.to_date) = 1
AND a.emp_no = b.emp_no
AND a.salary-b.salary > 5000

ORDER BY growth DESC;
```

---

film表

字段	说明
film_id	电影id
title	电影名称
description	电影描述信息

```
CREATE TABLE IF NOT EXISTS film (
film_id smallint(5)  NOT NULL DEFAULT '0',
title varchar(255) NOT NULL,
description text,
PRIMARY KEY (film_id));
```

category表

字段	说明

category_id	电影分类id
name	电影分类名称
last_update	电影分类最后更新时间

```
CREATE TABLE category  (
category_id  tinyint(3)  NOT NULL ,
name  varchar(25) NOT NULL, `last_update` timestamp,
PRIMARY KEY ( category_id ));
```

film_category表

字段	说明
film_id	电影id
category_id	电影分类id
last_update	电影id和分类id对应关系的最后更新时间

```
CREATE TABLE film_category  (
film_id  smallint(5)  NOT NULL,
category_id  tinyint(3)  NOT NULL, `last_update` timestamp);
```

查找描述信息中包括robot的电影对应的分类名称以及电影数目，而且还需要该分类对应电影数量>=5部

代码：

```
SELECT c.name, COUNT(fs.film_id)
FROM (SELECT f.*,fc.* FROM film f
          INNER JOIN film_category fc
          ON f.film_id = fc.film_id
          ) AS fs
LEFT JOIN category c
ON fs.category_id = c.category_id
WHERE fs.description LIKE '%robot%'
AND fs.category_id IN (SELECT category_id FROM film_category GROUP BY
category_id
```

HAVING COUNT(film_id) >= 5)

GROUP BY fs.category_id;

---

film表

字段	说明
film_id	电影id
title	电影名称
description	电影描述信息

> CREATE TABLE IF NOT EXISTS film (
> film_id smallint(5) NOT NULL DEFAULT '0',
> title varchar(255) NOT NULL,
> description text,
> PRIMARY KEY (film_id));

category表

字段	说明
category_id	电影分类id
name	电影分类名称
last_update	电影分类最后更新时间

> CREATE TABLE category (
> category_id tinyint(3) NOT NULL ,
> name varchar(25) NOT NULL, `last_update` timestamp,
> PRIMARY KEY (category_id));

film_category表

字段	说明
film_id	电影id

category_id	电影分类id
last_update	电影id和分类id对应关系的最后更新时间

> CREATE TABLE film_category (
> film_id smallint(5) NOT NULL,
> category_id tinyint(3) NOT NULL, `last_update` timestamp);

使用join查询方式找出没有分类的电影id以及名称

代码：
SELECT f.film_id, f.title
FROM film f
LEFT JOIN film_category fc
ON f.film_id = fc.film_id

WHERE fc.category_id IS NULL;

---

film表

字段	说明
film_id	电影id
title	电影名称
description	电影描述信息

> CREATE TABLE IF NOT EXISTS film (
> film_id smallint(5) NOT NULL DEFAULT '0',
> title varchar(255) NOT NULL,
> description text,
> PRIMARY KEY (film_id));

category表

字段	说明
category_id	电影分类id
name	电影分类名称
last_update	电影分类最后更新时间

> CREATE TABLE category （
> category_id tinyint(3) NOT NULL ,
> name varchar(25) NOT NULL, `last_update` timestamp,
> PRIMARY KEY (category_id));

film_category表

字段	说明
film_id	电影id
category_id	电影分类id
last_update	电影id和分类id对应关系的最后更新时间

> CREATE TABLE film_category （
> film_id smallint(5) NOT NULL,
> category_id tinyint(3) NOT NULL, `last_update` timestamp);

使用子查询的方式找出属于Action分类的所有电影对应的title,description

代码：

```
SELECT fs.title, fs.description
FROM (SELECT f.*, c.* FROM film f
    INNER JOIN film_category c
    ON f.film_id = c.film_id) AS fs
WHERE fs.category_id IS (SELECT category_id FROM category
                WHERE name == 'Action');
```

将employees表的所有员工的last_name和first_name拼接起来作为Name，中间以一个空格区分

```
CREATE TABLE `employees` ( `emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,
`hire_date` date NOT NULL,

PRIMARY KEY (`emp_no`));
```

代码：

```
SELECT last_name || ' ' || first_name AS name
FROM employees
```

创建一个actor表，包含如下列信息

列表	类型	是否为NULL	含义
actor_id	smallint(5)	not null	主键id
first_name	varchar(45)	not null	名字
last_name	varchar(45)	not null	姓氏
last_update	timestamp	not null	最后更新时间，默认是系统的当前时间

代码：

```
CREATE TABLE 'actor'('actor_id' smallint(5) NOT NULL,
              'first_name' varchar(45) NOT NULL,
              'last_name' varchar(45) NOT NULL,
              'last_update' timestamp NOT NULL DEFAULT (datetime('now','localtime')),

              PRIMARY KEY('actor_id'))
```

对于表actor批量插入如下数据

```
CREATE TABLE IF NOT EXISTS actor (
actor_id smallint(5) NOT NULL PRIMARY KEY,
first_name varchar(45) NOT NULL,
```

last_name varchar(45) NOT NULL,
last_update timestamp NOT NULL DEFAULT (datetime('now','localtime')))

actor_id	first_name	last_name	last_update
1	PENELOPE	GUINESS	2006-02-15 12:34:33
2	NICK	WAHLBERG	2006-02-15 12:34:33

代码:

```
INSERT INTO actor(actor_id,first_name,last_name,last_update)
VALUES (1, 'PENELOPE','GUINESS', '2006-02-15 12:34:33'),

      (2,'NICK','WAHLBERG', '2006-02-15 12:34:33')
```

对于表actor批量插入如下数据,如果数据已经存在，请忽略，不使用replace操作
CREATE TABLE IF NOT EXISTS actor (
actor_id smallint(5) NOT NULL PRIMARY KEY,
first_name varchar(45) NOT NULL,
last_name varchar(45) NOT NULL,
last_update timestamp NOT NULL DEFAULT (datetime('now','localtime')))

actor_id	first_name	last_name	last_update
'3'	'ED'	'CHASE'	'2006-02-15 12:34:33'

代码:

```
INSERT OR IGNORE INTO actor(actor_id, first_name, last_name, last_update)
VALUES (3, 'ED','CHASE','2006-02-15 12:34:33')
```

（此题用replace的话就是把ignore换成replace，INSERT OR REPLACE INTO actor）

对于如下表actor，其对应的数据为:

actor_id	first_name	last_name	last_update
1	PENELOPE	GUINESS	2006-02-15 12:34:33

2	NICK	WAHLBERG	2006-02-15 12:34:33

创建一个actor name表，将actor表中的所有first_name以及last_name导入改表。
actor_name表结构如下：

列表	类型	是否为NULL	含义
first_name	varchar(45)	not null	名字
last_name	varchar(45)	not null	姓氏

代码：

```
CREATE TABLE actor_name AS
SELECT first_name, last_name

FROM actor
```

（W3School上这个应该也行，但是编程平台未通过）

```
SELECT first_name, last_name INTO actor_name

FROM actor;
```

针对如下表actor结构创建索引：

```
CREATE TABLE IF NOT EXISTS actor (
actor  id smallint(5) NOT NULL PRIMARY KEY,
first  name varchar(45) NOT NULL,
last  name varchar(45) NOT NULL,
last_update timestamp NOT NULL DEFAULT (datetime('now','localtime')))
```

对first_name创建唯一索引uniq_idx_firstname，对last_name创建普通索引idx_lastname

代码：

```
CREATE UNIQUE INDEX uniq  idx  firstname ON actor(first_name);
CREATE INDEX idx_lastname ON actor(last_name);
```

针对actor表创建视图actor name view，只包含first name以及last name两列，并对这两列重新命名，first name为first name v，last_name修改为last_name_v：

```
CREATE TABLE IF NOT EXISTS actor (
actor  id smallint(5) NOT NULL PRIMARY KEY,
first  name varchar(45) NOT NULL,
last_name varchar(45) NOT NULL,
last_update timestamp NOT NULL DEFAULT (datetime('now','localtime')))
```

代码：

```
CREATE VIEW actor_name_view AS
SELECT first_name AS first_name_v, last_name AS last_name_v
FROM actor
```

针对salaries表emp_no字段创建索引idx_emp_no，查询emp_no为10005, 使用强制索引。

```
CREATE TABLE `salaries` (
`emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`from_date`));
create index idx_emp_no on salaries(emp_no);
```

代码：

```
SELECT * FROM salaries
INDEXED BY idx_emp_no
WHERE emp_no = 10005
```

（SQLite中，使用 INDEXED BY 语句进行强制索引查询；

MySQL中，使用 FORCE INDEX 语句进行强制索引查询
）

存在actor表，包含如下列信息：

```
CREATE TABLE IF NOT EXISTS actor (
actor_id smallint(5) NOT NULL PRIMARY KEY,
first_name varchar(45) NOT NULL,
last_name varchar(45) NOT NULL,
last_update timestamp NOT NULL DEFAULT (datetime('now','localtime')));
```

现在在last_update后面新增加一列名字为create_date, 类型为datetime, NOT NULL，默认值为'0000 00:00:00'

代码：

```
ALTER TABLE actor
ADD create_date datetime NOT NULL DEFAULT '0000-00-00 00:00:00';
```

构造一个触发器audit_log，在向employees_test表中插入一条数据的时候，触发插入相关的数据到audit中。

```
CREATE TABLE employees_test(
ID INT PRIMARY KEY NOT NULL,
NAME TEXT NOT NULL,
AGE INT NOT NULL,
ADDRESS CHAR(50),
SALARY REAL
);
CREATE TABLE audit(
EMP_no INT NOT NULL,
NAME TEXT NOT NULL
);
```

代码:

```
CREATE TRIGGER audit_log AFTER INSERT
ON employees_test
BEGIN
INSERT INTO audit(EMP_no, NAME)
VALUES (new.ID, new.NAME);
END;
```

删除emp_no重复的记录，只保留最小的id对应的记录。

```
CREATE TABLE IF NOT EXISTS titles_test (
id int(11) not null primary key,
emp_no int(11) NOT NULL,
title varchar(50) NOT NULL,
from_date date NOT NULL,

to_date date DEFAULT NULL);
```

代码:

```
DELETE FROM titles_test
WHERE id NOT IN (SELECT MIN(id) FROM titles_test GROUP BY emp_no);
```

将所有to_date为9999-01-01的全部更新为NULL,且 from_date更新为2001-01-01。

```
CREATE TABLE IF NOT EXISTS titles_test (
id int(11) not null primary key,
emp_no int(11) NOT NULL,
title varchar(50) NOT NULL,
from_date date NOT NULL,
to_date date DEFAULT NULL);
```

代码:

```
UPDATE titles_test
SET to_date = NULL, from_date = '2001-01-01'
WHERE to_date = '9999-01-01';
```

将id=5以及emp_no=10001的行数据替换成id=5以及emp_no=10005,其他数据保持不变，使用replace实现。

```
CREATE TABLE IF NOT EXISTS titles_test (
id int(11) not null primary key,
emp_no int(11) NOT NULL,
title varchar(50) NOT NULL,
from_date date NOT NULL,

to_date date DEFAULT NULL);
```

代码:

```
INSERT OR REPLACE INTO titles_test(id,emp_no,title,from_date,to_date)
VALUES (5,10005,'Senior Engineer','1986-06-26','9999-01-01');
```

（用update则是： UPDATE titles_test
SET emp_no = 10005

WHERE id = 5;）

将titles_test表名修改为titles_2017。

```
CREATE TABLE IF NOT EXISTS titles_test (
id int(11) not null primary key,
emp_no int(11) NOT NULL,
title varchar(50) NOT NULL,
from_date date NOT NULL,
to_date date DEFAULT NULL);
```

代码：

```
ALTER TABLE titles_test RENAME TO titles_2017;
```

（rename列名则是：
首先rename表格： ALTER TABLE table RENAME TO table_tmp;
然后以原表的名字创建表格，用想要替换的列名：
CREATE TABLE table (newcol1 INT, newcol2 INT);
然后把临时表中的数据拷贝过来：
INSERT INTO table
SELECT col1, col2
FROM table_tmp;
最后删除临时表：DROP TABLE table_tmp）

在audit表上创建外键约束，其emp_no对应employees_test表的主键id。

```
CREATE TABLE employees_test(
ID INT PRIMARY KEY NOT NULL,
NAME TEXT NOT NULL,
AGE INT NOT NULL,
ADDRESS CHAR(50),
SALARY REAL
);
```

代码：

```
DROP TABLE audit;
CREATE TABLE audit(
   EMP_no INT NOT NULL,
   create_date datetime NOT NULL,
   FOREIGN KEY(EMP_no) REFERENCES employees_test(ID));
```

（SQLite只能先删除表再重新创建表来做，MySQL可以用：
ALTER TABLE audit
ADD FOREIGN KEY (EMP_no) REFERENCES employees_test(ID)
）

存在如下的视图：
create view emp_v as select * from employees where emp_no >10005;
如何获取emp_v和employees有相同的数据？

```
CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,
`hire_date` date NOT NULL,

PRIMARY KEY (`emp_no`));
```

代码：

```
SELECT * FROM employees
WHERE emp_no IN (SELECT emp_no FROM emp_v);
```

将所有获取奖金的员工当前的薪水增加10%。

```
create table emp_bonus(
emp_no int not null,
recevied datetime not null,
btype smallint not null);
CREATE TABLE `salaries` (
`emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,

`to_date` date NOT NULL, PRIMARY KEY (`emp_no`,`from_date`));
```

代码：

```
UPDATE salaries
SET salary = salary*1.1

WHERE emp_no IN (SELECT emp_no FROM emp_bonus);
```

针对库中的所有表生成select count(*)对应的SQL语句

```
CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,
`hire_date` date NOT NULL,
PRIMARY KEY (`emp_no`));
create table emp_bonus(
emp_no int not null,
recevied datetime not null,
btype smallint not null);
CREATE TABLE `dept_emp` (
`emp_no` int(11) NOT NULL,
`dept_no` char(4) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`dept_no`));
CREATE TABLE `dept_manager` (
`dept_no` char(4) NOT NULL,
`emp_no` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`dept_no`));
```

CREATE TABLE `salaries` (
`emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,

PRIMARY KEY (`emp_no`,`from_date`));

代码:
SELECT "select count(*) from " || name || ";" AS cnts
FROM sqlite_master WHERE type = 'table'

（1、在 SQLite 系统表 sqlite_master 中可以获得所有表的索引，其中字段 name 是所有表的名字，而且对于自己创建的表而言，字段 type 永远是 'table'，详情可参考：

http://blog.csdn.net/xingfeng0501/article/details/7804378

2、在 SQLite 中用 "||" 符号连接字符串）

将employees表中的所有员工的last_name和first_name通过(')连接起来。
CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,
`hire_date` date NOT NULL,

PRIMARY KEY (`emp_no`));

代码:
SELECT last_name || "'" || first_name AS name

FROM employees;

（参考资料：https://blog.csdn.net/ameyume/article/details/8007149）

查找字符串'10,A,B' 中逗号','出现的次数cnt。

代码:
SELECT (length("10,A,B")-length(replace("10,A,B",",","")))/length(",") AS cnt

---

获取Employees中的first_name，查询按照first_name最后两个字母，按照升序进行排列
CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,

```
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,
`hire_date` date NOT NULL,

PRIMARY KEY (`emp_no`));
```

代码：

```
SELECT first_name
FROM employees
ORDER BY substr(first_name,-2) ASC;
```

按照dept_no进行汇总，属于同一个部门的emp_no按照逗号进行连接，结果给出dept_no以及连接出的结果employees

```
CREATE TABLE `dept_emp` (
`emp_no` int(11) NOT NULL,
`dept_no` char(4) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,

PRIMARY KEY (`emp_no`,`dept_no`));
```

代码：

```
SELECT dept_no, group_concat(emp_no) AS employees
FROM dept_emp

GROUP BY dept_no;
```

（group_cancat函数参考资料：
https://blog.csdn.net/langzxz/article/details/16807859）

查找排除当前最大、最小salary之后的员工的平均工资avg_salary。

```
CREATE TABLE `salaries` ( `emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,

PRIMARY KEY (`emp_no`,`from_date`));
```

代码：

```
SELECT AVG(salary) AS avg_salary
FROM salaries
WHERE salary <> (SELECT MAX(salary) FROM salaries)
AND salary <> (SELECT MIN(salary) FROM salaries)
AND to_date = '9999-01-01';
```

分页查询employees表，每5行一页，返回第2页的数据

```
CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,
`hire_date` date NOT NULL,

PRIMARY KEY (`emp_no`));
```

代码：

```
SELECT * FROM employees
LIMIT 5 offset 5;
```

（参考：http://www.cnblogs.com/chris1943/archive/2008/02/03/1063279.html）

获取所有员工的emp_no、部门编号dept_no以及对应的bonus类型btype和recevied，没有分配具体的员工不显示

```
CREATE TABLE `dept_emp` ( `emp_no` int(11) NOT NULL,
`dept_no` char(4) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`dept_no`));

CREATE TABLE `dept_manager` (
`dept_no` char(4) NOT NULL,
`emp_no` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`dept_no`));

CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,
`hire_date` date NOT NULL,
PRIMARY KEY (`emp_no`));

CREATE TABLE `salaries` (
`emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`from_date`));

create table emp_bonus(
emp_no int not null,
recevied datetime not null,
btype smallint not null);
```

代码：

```
SELECT de.emp_no, de.dept_no, eb.btype, eb.recevied
FROM dept_emp AS de LEFT JOIN emp_bonus AS eb
```

ON de.emp_no = eb.emp_no

使用含有关键字exists查找未分配具体部门的员工的所有信息。
CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,
`hire_date` date NOT NULL,
PRIMARY KEY (`emp_no`));
CREATE TABLE `dept_emp` (
`emp_no` int(11) NOT NULL,
`dept_no` char(4) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,

PRIMARY KEY (`emp_no`,`dept_no`));

代码:
SELECT *
FROM employees e
WHERE NOT EXISTS
(SELECT d.emp_no FROM dept_emp d
WHERE d.emp_no = e.emp_no);

存在如下的视图:
create view emp_v as select * from employees where emp_no >10005;
CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,
`hire_date` date NOT NULL,
PRIMARY KEY (`emp_no`));
获取employees中的行数据，且这些行也存在于emp_v中。注意不能使用intersect关键字。

代码:
SELECT * FROM employees
WHERE emp_no IN (SELECT emp_no FROM emp_v);

（或者：`SELECT em.* FROM employees AS em, emp_v AS ev WHERE em.emp_no = ev.emp_no`）

获取有奖金的员工相关信息。
CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,

```
`hire_date` date NOT NULL,
PRIMARY KEY (`emp_no`));
CREATE TABLE `dept_emp` (
`emp_no` int(11) NOT NULL,
`dept_no` char(4) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`dept_no`));
create table emp_bonus(
emp_no int not null,
recevied datetime not null,
btype smallint not null);
CREATE TABLE `salaries` (
`emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL, PRIMARY KEY (`emp_no`,`from_date`));
```

给出emp_no、first_name、last_name、奖金类型btype、对应的当前薪水情况salary以及奖金金额bonus。 bonus类型btype为1其奖金为薪水salary的10%，btype为2其奖金为薪水的20%，其他类型均为薪水的30%。 当前薪水表示to_date='9999-01-01'

代码：

```
SELECT e.emp_no, e.first_name, e.last_name, b.btype, s.salary,
      CASE WHEN b.btype = 1 THEN s.salary*0.1
          WHEN b.btype = 2 THEN s.salary*0.2
          ELSE s.salary*0.3
      END AS bonus
FROM employees e
INNER JOIN emp_bonus b ON e.emp_no = b.emp_no
INNER JOIN salaries s ON e.emp_no = s.emp_no
WHERE s.to_date = '9999-01-01';
```

按照salary的累计和running_total，其中running_total为前两个员工的salary累计和，其他以此类推。 具体结果如下Demo展示。。

```
CREATE TABLE `salaries` ( `emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`from_date`));
```

代码：

```
SELECT a.emp_no, a.salary, SUM(b.salary) AS running_total
FROM salaries a, salaries b
WHERE a.to_date = '9999-01-01'
AND b.to_date = '9999-01-01'
AND a.emp_no >= b.emp_no
GROUP BY a.emp_no;
```

（网友version：

```
SELECT s1.emp_no, s1.salary,
(SELECT SUM(s2.salary) FROM salaries AS s2
 WHERE s2.emp_no <= s1.emp_no AND s2.to_date = '9999-01-01') AS
running_total
FROM salaries AS s1 WHERE s1.to_date = '9999-01-01' ORDER BY s1.emp_no）
```

对于employees表中，给出奇数行的first_name

```
CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL,
`birth_date` date NOT NULL,
`first_name` varchar(14) NOT NULL,
`last_name` varchar(16) NOT NULL,
`gender` char(1) NOT NULL,
`hire_date` date NOT NULL,
PRIMARY KEY (`emp_no`));
```

代码：

```
SELECT e1.first_name
FROM employees e1
WHERE (SELECT COUNT(*) FROM employees e2
      WHERE e1.first_name <= e2.first_name) %2 =1
```