01. 查找最晚入职员工的所有信息

2020年5月18日 8:43

查找最晚入职员工的所有信息，为了减轻入门难度，目前所有的数据里员工入职的日期都不是同一天

```
CREATE TABLE `employees` (
`emp_no` int(11) NOT NULL  comment '员工编号',
`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`));

select *
from employees
where hire_date = (select max(hire_date) from employees);
```

02.查找入职员工时间排名倒数第三的员工所有信息

2020年5月18日 8:44

题目描述

查找入职员工时间排名倒数第三的员工所有信息，为了减轻入门难度，目前所有的数据里员工入职的日期都不是同一天

```
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`));
```

```
select *
from employees
where hire_date = (
   select distinct hire_date
   from employees
   order by hire_date desc
   limit 2,1
);
```

03.查找各个部门当前领导当前薪水详情以及其对应部门编号dept_no

2020年5月18日 8:44

题目描述

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(请注意输出结果，dept_no列是最后一列)

```
CREATE TABLE `dept_manager` (
`dept_no` char(4) NOT NULL comment '部门编号',
`emp_no` int(11) NOT NULL comment '员工编号',
`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 comment '员工编号',
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`from_date`));
```

```
select s.*, d.dept_no
from salaries s
inner join dept_manager d
on s.emp_no = d.emp_no
where s.to_date='9999-01-01'
and d.to_date='9999-01-01';
```

04.查找所有已经分配部门的员工的last_name和first_name

2020年5月18日 8:44

题目描述

查找所有已经分配部门的员工的last_name和first_name以及dept_no(请注意输出描述里各个列的前后顺序)

```
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`));

select last_name,first_name,dept_no
from employees e
inner join dept_emp d
on e.emp_no = d.emp_no;
```

05.查找所有员工的last_name和first_name以及对应部门编号dept_no

2020年5月18日 8:44

题目描述

查找所有员工的last_name和first_name以及对应部门编号dept_no，也包括暂时没有分配具体部门的员工(请注意输出描述里各个列的前后顺序)

```
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`));

select last_name,first_name,dept_no
from employees e
left join dept_emp d
on e.emp_no = d.emp_no;
```

注意：

INNER JOIN 两边表同时有对应的数据，即任何一边缺失数据就不显示。

主 LEFT JOIN 从 会读取左边数据表的全部数据，即便右边表无对应数据。

从 RIGHT JOIN 主 会读取右边数据表的全部数据，即便左边表无对应数据。

06.查找所有员工入职时候的薪水情况

2020年5月18日 8:44

题目描述

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(请注意，一个员工可能有多次涨薪的情况)

```
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,#入职
PRIMARY KEY (`emp_no`));

CREATE TABLE `salaries` (
`emp_no` int(11) NOT NULL,
`salary` int(11) NOT NULL,
`from_date` date NOT NULL,#入职日期=from_date
`to_date` date NOT NULL,
PRIMARY KEY (`emp_no`,`from_date`));


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;
```

07.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

2020年5月18日 8:44

题目描述

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`));

select s.emp_no, count(*) t
from salaries s
group by s.emp_no
having count(*) > 15;
```

08.找出所有员工当前具体的薪水salary情况

2020年5月18日 8:44

题目描述

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`));

select distinct salary
from salaries s
where s.to_date='9999-01-01'
order by salary desc;

select salary
from salaries s
where s.to_date='9999-01-01'
group by salary
order by salary desc;
```

09.获取所有部门当前manager的当前薪水情况

2020年5月18日 8:44

题目描述

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`));

select d.dept_no, d.emp_no, s.salary
from dept_manager d
inner join salaries s
on d.emp_no = s.emp_no
where d.to_date='9999-01-01'
and s.to_date='9999-01-01';
```

10.获取所有非manager的员工emp_no

2020年5月18日 8:44

题目描述

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`));

select emp_no
from employees
where emp_no not in(
   select e.emp_no
   from employees e
   inner join dept_manager d
   on e.emp_no = d.emp_no
);
```

11.获取所有员工当前的manager

2020年5月18日 8:44

题目描述

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`));

select e.emp_no, m.emp_no
from dept_emp e
inner join dept_manager m
on e.dept_no = m.dept_no
where e.emp_no <> m.emp_no
and e.to_date='9999-01-01'
and m.to_date='9999-01-01';
```

12.获取所有部门中当前员工薪水最高的相关信息

2020年5月18日 8:44

题目描述

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`));

select d.dept_no, d.emp_no, max(salary)
from dept_emp d
inner 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;
```

13.从titles表获取按照title进行分组

2020年5月18日 8:44

题目描述

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);

select title, count(*) t
from titles
group by title
having count(*) >= 2;
```

14.从titles表获取按照title进行分组，注意对于重复的emp_no进行忽略

2020年5月18日 8:47

题目描述

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);
```

```
select title, count(distinct emp_no) t
from titles
group by title
having t >= 2;
```

15.查找employees表所有emp_no为奇数

2020年5月18日 8:47

题目描述

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`));

select *
from employees e
where e.emp_no & 1 = 1
and e.last_name != 'Mary'
order by  e.hire_date desc;
```

16.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

2020年5月18日 8:47

题目描述

统计出当前各个title类型对应的员工当前（to_date='9999-01-01'）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);

select t.title, avg(s.salary) avg
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;
```

17.获取当前薪水第二多的员工的emp_no以及其对应的薪水salary

2020年5月18日 8:47

题目描述

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`));

select emp_no,salary
from salaries
where to_date='9999-01-01'
and salary = (
   select distinct salary
   from salaries
   order by salary desc
   limit 1,1
);
```

18.查找当前薪水排名第二多的员工编号emp_no

2020年5月18日 8:47

题目描述

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`));
```

```
select e.emp_no, max(s.salary), e.last_name, e.first_name
from employees e
inner join salaries 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'
);
```

19.查找所有员工的last_name和first_name以及对应的dept_name

2020年5月18日 8:47

题目描述

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 left join

```
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`));


select last_name,first_name,dept_name
from employees e
left join dept_emp de on e.emp_no = de.emp_no
left join departments d on d.dept_no = de.dept_no;
```

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

2020年5月18日 8:47

题目描述

查找员工编号emp_no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`));

select
(
   #最近的 - 刚入职的
   (select salary from salaries where emp_no = 10001 order by salary desc)
   - (select salary from salaries where emp_no = 10001 order by salary)
) growth;
```

21.查找所有员工自入职以来的薪水涨幅情况

2020年5月18日 8:47

题目描述

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_no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`));

select e.emp_no, (s1.salary - s2.salary) as growth
from employees e
inner join salaries s1
on e.emp_no = s1.emp_no and s1.to_date = '9999-01-01'#当前薪资
inner join salaries s2
on e.emp_no = s2.emp_no and e.hire_date = s2.from_date#入职薪资
order by growth asc;
```

22.统计各个部门对应员工涨幅的次数总和

2020年5月18日 8:47

题目描述

统计各个部门的工资记录数，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`));

select d.dept_no, d.dept_name,count(s.salary) 'sum'
from departments d
inner join dept_emp de on d.dept_no = de.dept_no
inner join salaries s on de.emp_no = s.emp_no
group by d.dept_no;
```

23.对所有员工的当前薪水按照salary进行按照1-N的排名

2020年5月18日 8:47

题目描述

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`));
```

输出描述:

emp_no	salary	rank
10005	94692	1
10009	94409	2
10010	94409	2
10001	88958	3
10007	88070	4
10004	74057	5
10002	72527	6
10003	43311	7
10006	43311	7
10011	25828	8

来自 <https://www.nowcoder.com/practice/b9068bfe5df74276bd015b9729eec4bf?tpId=82&tqId=29775&tPage=2&rp=&ru=/ta/sql&qru=/ta/sql/question-ranking>

```
select s1.emp_no, s1.salary, count(distinct s2.salary) rank
from salaries s1, salaries 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
```

24.获取所有非manager员工当前的薪水情况

2020年5月18日 8:47

题目描述

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`));
```

1.查询所有员工的情况

2.筛选掉manager员工

```
select de.dept_no,e.emp_no,s.salary
from employees e
inner join  salaries s
on e.emp_no = s.emp_no and s.to_date='9999-01-01'
inner join dept_emp de
on e.emp_no = de.emp_no
where de.emp_no not in(
   select emp_no
   from dept_manager dm
   where dm.to_date='9999-01-01'
);
```

25.获取员工其当前的薪水比其manager当前薪水还高的相关信息

2020年5月18日 8:47

题目描述

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`));

select de.emp_no emp_no,dm.emp_no manager_no,s1.salary emp_salary,s2.salary manager_salary
from dept_emp de, salaries s1, dept_manager dm, salaries s2
where de.emp_no = s1.emp_no
and s1.to_date='9999-01-01'
and de.to_date='9999-01-01'
and dm.emp_no = s2.emp_no
and s2.to_date='9999-01-01'
and dm.to_date='9999-01-01'
and de.dept_no = dm.dept_no
and s1.salary > s2.salary;
```

26.汇总各个部门当前员工的title类型的分配数目

2020年5月18日 8:47

题目描述

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);
```

输出描述:

dept_no	dept_name	title	count
d001	Marketing	Senior Engineer	1
d001	Marketing	Staff	1
d002	Finance	Senior Engineer	1
d003	Human Resources	Senior Staff	1
d004	Production	Senior Engineer	2
d005	Development	Senior Staff	1
d006	Quality Management	Engineer	2
d006	Quality Management	Senior Engineer	1

来自 <https://www.nowcoder.com/practice/4bcb6a7d3e39423291d2f7bdbbff87f8?tpId=82&tqId=29778&tPage=2&rp=&ru=%2Fta%2Fsql&qru=%2Fta%2Fsql%2Fquestion-ranking>

```
select de.dept_no,d.dept_name,t.title,count(t.title) 'count'
from dept_emp de
inner join titles t
on de.emp_no = t.emp_no
and de.to_date='9999-01-01'
and t.to_date='9999-01-01'
```

```
inner join departments d
on de.dept_no = d.dept_no
group by de.dept_no, t.title;
```

27.给出每个员工每年薪水涨幅超过5000的员工编号emp_no

2020年5月18日 8:47

题目描述

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`));

select s2.emp_no,s2.from_date,(s2.salary - s1.salary) salary_growth
from salaries s1, salaries s2
where s1.emp_no = s2.emp_no
and salary_growth > 5000
and (strftime('%Y', s2.to_date) - strftime('%Y', s1.to_date) = 1
    or strftime('%Y', s2.from_date) - strftime('%Y', s1.from_date) = 1)
order by salary_growth desc;
```

28.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

2020年5月18日 8:47

题目描述

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

来自 <https://www.nowcoder.com/practice/3a303a39cc40489b99a7e1867e6507c5?tpId=82&tqId=29780&tPage=2

&rp=&ru=%2Fta%2Fsql&qru=%2Fta%2Fsql%2Fquestion-ranking>

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

```
CREATE TABLE IF NOT EXISTS film (
film_id smallint(5)  NOT NULL DEFAULT '0',
title varchar(255) NOT NULL,
description text, #描述信息 like '%robot%'
PRIMARY KEY (film_id));

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

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

select c.name,count(fc.film_id) num
from category c
inner join film_category fc
on c.category_id = fc.category_id
inner join(
   select *
   from film f
   where f.description like '%robot%'
   ) f_robot on f_robot.film_id = fc.film_id
inner join(
   select *,count(fc1.film_id) num1
   from film_category fc1
   group by category_id
   having num1 >= 5
) f5 on f5.category_id = fc.category_id;
```

29.使用join查询方式找出没有分类的电影id以及名称

2020年5月18日 8:47

使用join查询方式找出没有分类的电影id以及名称

```
CREATE TABLE IF NOT EXISTS film (
film_id smallint(5)  NOT NULL DEFAULT '0',
title varchar(255) NOT NULL,
description text, #描述信息 like '%robot%'
PRIMARY KEY (film_id));

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

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

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;
```

30.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

2020年5月18日 8:47

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

```
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));

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

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

select f.title, f.description
from film f
where f.film_id in(
   select film_id
   from film_category fc
   where fc.category_id in(
      select c.category_id
      from category c
      where c.name='Action'
   )
);
```

31.获取select

2020年5月18日 18:14

题目描述

获取select * from employees对应的执行计划

explain select * from employees;

32.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

2020年5月18日 18:14

题目描述

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`));

select last_name||' '||first_name as name
from employees;
```

33.创建一个actor表，包含如下列信息

2020年5月18日 18:14

题目描述

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

```
create table 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'))
);
```

34.批量插入数据

2020年5月18日 18:14

题目描述

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

```
insert into actor
values (1, 'PENELOPE', 'GUINESS', '2006-02-15 12:34:33'),
(2, 'NICK', 'WAHLBERG', '2006-02-15 12:34:33')
```

```
insert into actor
select 1,'PENELOPE','GUINESS','2006-02-15 12:34:33'
union select 2,'NICK','WAHLBERG','2006-02-15 12:34:33';
```

35.批量插入数据,如果数据已经存在，请忽略，不使用replace操作

2020年5月18日 18:14

题目描述

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'

insert or ignore into actor
values (3,'ED','CHASE','2006-02-15 12:34:33');
```

36.创建一个actor_name表，将actor表中的所有first_name以及last_name导入改表

2020年5月18日 18:14

题目描述

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

```
create table actor_name as
select first_name,last_name from actor;
```

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

2020年5月18日 18:14

题目描述

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

```
create unique index uniq_idx_firstname on actor(first_name);
create index idx_lastname on actor(last_name);
```

38.针对actor表创建视图actor_name_view

2020年5月18日 18:14

题目描述

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')))

create view actor_name_view (first_name_v,last_name_v) as
select first_name, last_name from actor;
```

39.针对上面的salaries表emp_no字段创建索引idx_emp_no，查询emp_no为10005,

2020年5月18日 18:14

题目描述

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);

select *
from salaries
indexed by idx_emp_no
where emp_no = 10005;
```

MYSQL中强制索引查询使用：FORCE INDEX(indexname);

SQLite中强制索引查询使用：INDEXED BY indexname;

来自 <https://www.nowcoder.com/profile/340487413/codeBookDetail?submissionId=36255256>

SQLite中，使用 INDEXED BY 语句进行强制索引查询，可参考：

http://www.runoob.com/sqlite/sqlite-indexed-by.html

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

MySQL中，使用 FORCE INDEX 语句进行强制索引查询，可参考：

http://www.jb51.net/article/49807.htm

```
1 SELECT * FROM salaries FORCE INDEX idx_emp_no WHERE emp_no = 10005
```

来自 <https://www.nowcoder.com/profile/340487413/codeBookDetail?submissionId=36255256>

40.在last_update后面新增加一列名字为create_date

2020年5月18日 18:14

题目描述

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'

```
alter table actor add column create_date datetime not null default '0000-00-00 00:00:00';
```

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

2020年5月18日 18:18

题目描述

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
);

create trigger audit_log after insert on employees_test
begin
   insert into audit values (new.ID, new.NAME);
end;
```

42.删除emp_no重复的记录，只保留最小的id对应的记录。

2020年5月18日 18:18

题目描述

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);

insert into titles_test values
('1', '10001', 'Senior Engineer', '1986-06-26', '9999-01-01'),
('2', '10002', 'Staff', '1996-08-03', '9999-01-01'),
('3', '10003', 'Senior Engineer', '1995-12-03', '9999-01-01'),
('4', '10004', 'Senior Engineer', '1995-12-03', '9999-01-01'),
('5', '10001', 'Senior Engineer', '1986-06-26', '9999-01-01'),
('6', '10002', 'Staff', '1996-08-03', '9999-01-01'),
('7', '10003', 'Senior Engineer', '1995-12-03', '9999-01-01');
```

1.group by和 min 找到每个emp_no中 最小的 id

2.删除 不是这个最小id 数据

```
delete from titles_test
where id not in(
   select min(id)
   from titles_test
   group by emp_no
);
```

43.将所有to_date为9999-01-01的全部更新为NULL,且

2020年5月18日 18:18

题目描述

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);

insert into titles_test values ('1', '10001', 'Senior Engineer', '1986-06-26', '9999-01-01'),
('2', '10002', 'Staff', '1996-08-03', '9999-01-01'),
('3', '10003', 'Senior Engineer', '1995-12-03', '9999-01-01'),
('4', '10004', 'Senior Engineer', '1995-12-03', '9999-01-01'),
('5', '10001', 'Senior Engineer', '1986-06-26', '9999-01-01'),
('6', '10002', 'Staff', '1996-08-03', '9999-01-01'),
('7', '10003', 'Senior Engineer', '1995-12-03', '9999-01-01');


update titles_test set to_date = null, from_date = '2001-01-01'
where to_date = '9999-01-01';
```

44.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。

2020年5月18日 18:18

题目描述

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);

insert into titles_test values ('1', '10001', 'Senior Engineer', '1986-06-26', '9999-01-01'),
('2', '10002', 'Staff', '1996-08-03', '9999-01-01'),
('3', '10003', 'Senior Engineer', '1995-12-03', '9999-01-01'),
('4', '10004', 'Senior Engineer', '1995-12-03', '9999-01-01'),
('5', '10001', 'Senior Engineer', '1986-06-26', '9999-01-01'),
('6', '10002', 'Staff', '1996-08-03', '9999-01-01'),
('7', '10003', 'Senior Engineer', '1995-12-03', '9999-01-01');

replace into titles_test select 5,10005,title,from_date,to_date
from titles_test
where id = 5
and emp_no = 10001;
```

45.将titles_test表名修改为titles_2017

2020年5月18日 18:19

题目描述

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);

insert into titles_test values ('1', '10001', 'Senior Engineer', '1986-06-26', '9999-01-01'),
('2', '10002', 'Staff', '1996-08-03', '9999-01-01'),
('3', '10003', 'Senior Engineer', '1995-12-03', '9999-01-01'),
('4', '10004', 'Senior Engineer', '1995-12-03', '9999-01-01'),
('5', '10001', 'Senior Engineer', '1986-06-26', '9999-01-01'),
('6', '10002', 'Staff', '1996-08-03', '9999-01-01'),
('7', '10003', 'Senior Engineer', '1995-12-03', '9999-01-01');

alter table titles_test rename to titles_2017;
```

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

2020年5月18日 18:19

题目描述

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
);

CREATE TABLE audit(
EMP_no INT NOT NULL,
create_date datetime NOT NULL
);

drop table audit;
create table audit(
   EMP_no int not null,
   create_date datetime NOT NULL,
   foreign key(EMP_no) references employees_test(ID));
```

47.如何获取emp_v和employees有相同的数据no

2020年5月18日 18:19

题目描述

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`));

select * from emp_v;

select e.*
from employees e, emp_v ev
where e.emp_no = ev.emp_no;

select e.*
from employees e
intersect select ev.* from emp_v ev;
```

48.将所有获取奖金的员工当前的薪水增加10%

2020年5月18日 18:19

题目描述

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`));

update salaries set salary = salary * 1.1
where emp_no in(
   select s.emp_no
   from salaries s
   inner join emp_bonus eb
   on s.emp_no = eb.emp_no
   and s.to_date = '9999-01-01'
);
```

49.针对库中的所有表生成select

2020年5月18日 18:19

题目描述

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`));
```

输出格式:

```
cnts
select count(*) from employees;
select count(*) from departments;
select count(*) from dept_emp;
select count(*) from dept_manager;
select count(*) from salaries;
select count(*) from titles;
select count(*) from emp_bonus;

select 'select count(*) from '||name||';' cnts
from sqlite_master where type = 'table';
```

50.将employees表中的所有员工的last_name和first_name通过(')连接起来。

2020年5月18日 18:19

题目描述

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`));
```

输出格式:

```
name
Facello'Georgi
Simmel'Bezalel
Bamford'Parto
Koblick'Chirstian
Maliniak'Kyoichi
Preusig'Anneke
Zielinski'Tzvetan
Kalloufi'Saniya
Peac'Sumant
Piveteau'Duangkaew
Sluis'Mary
```

```
select last_name||"'"||first_name name
from employees;
```

51.查找字符串'10,A,B'

2020年5月18日 18:19

题目描述

查找字符串'10,A,B' 中逗号','出现的次数cnt。

```
select (length("10,A,B") - length(replace("10,A,B",",","")))/length(",") as cnt;
```

52.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

2020年5月18日 18:19

题目描述

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`));
```

输出格式:

```
first_name
Chirstian
Tzvetan
Bezalel
Duangkaew
Georgi
Kyoichi
Anneke
Sumant
Mary
Parto
Saniya
```

```
select first_name
from employees
order by substr(first_name,-2,2) asc;
```

53.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

2020年5月18日 18:19

题目描述

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`));
```

输出格式:

```
dept_no         employees
d001    10001,10002
d002    10006
d003    10005
d004    10003,10004
d005    10007,10008,10010
d006    10009,10010
```

```
select dept_no, group_concat(emp_no,",") employees
from dept_emp
group by dept_no;
```

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

2020年5月18日 18:19

题目描述

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`));
```

输出格式:

```
avg_salary
69462.5555555556
```

```
select avg(salary) avg_salary
from salaries
where to_date = '9999-01-01'
and salary not in(
   select max(salary)
   from salaries
)
and salary not in(
   select min(salary)
   from salaries
);
```

55.分页查询employees表，每5行一页，返回第2页的数据

2020年5月18日 18:19

题目描述

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`));
```

公式：limit (页数-1)*每一页的数量，每一页的数量

```
select *
from employees
limit (2-1)*5,5;
```

56.获取所有员工的emp_no

2020年5月18日 18:19

题目描述

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received，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,
received datetime not null,
btype smallint not null);
```

输出格式:

```
e.emp_no        dept_no          btype    received
10001   d001    1        2010-01-01
10002   d001    2        2010-10-01
10003   d004    3        2011-12-03
10004   d004    1        2010-01-01
10005   d003
10006   d002
10007   d005
10008   d005
10009   d006
10010   d005
10010   d006
```

```
select e.emp_no,de.dept_no,eb.btype,eb.recevied
from employees e
inner join dept_emp de
```

```
on e.emp_no = de.emp_no
left join emp_bonus eb
on e.emp_no = eb.emp_no;
```

57.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。

2020年5月18日 18:19

题目描述

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`));
```

输出格式:

```
emp_no          birth_date       first_name       last_name       gender  hire_date
10011   1953-11-07      Mary    Sluis     F       1990-01-22
```

```
select *
from employees e
where not exists(
   select *
   from dept_emp de
   where de.emp_no = e.emp_no
);
```

58.获取employees中的行数据，且这些行也存在于emp_v中

2020年5月18日 18:19

题目描述

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。

输出格式:

```
emp_no       birth_date      first_name      last_name       gender  hire_date
10006   1953-04-20      Anneke Preusig F       1989-06-02
10007   1957-05-23      Tzvetan Zielinski F       1989-02-10
10008   1958-02-19      Saniya  Kalloufi M       1994-09-15
10009   1952-04-19      Sumant Peac     F       1985-02-18
10010   1963-06-01      Duangkaew       Piveteau        F       1989-08-24
10011   1953-11-07      Mary    Sluis   F       1990-01-22
```

```
select * from emp_v;

select e.*
from employees e, emp_v ev
where e.emp_no = ev.emp_no;
```

59.获取有奖金的员工相关信息。

2020年5月18日 18:19

题目描述

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'

输出格式:

```
emp_no          first_name      last_name       btype   salary  bonus
10001   Georgi  Facello 1       88958   8895.8
10002   Bezalel Simmel 2        72527   14505.4
10003   Parto   Bamford         3       43311   12993.3
10004   Chirstian       Koblick 1       74057   7405.7
```

```
select e.emp_no, e.first_name, e.last_name, b.btype, s.salary,(
   case b.btype
   when 1 then s.salary*0.1
   when 2 then s.salary*0.2
   else s.salary*0.3
   end
) bonus
from employees e
inner join emp_bonus b
on e.emp_no = b.emp_no
inner join salaries s
on s.emp_no = e.emp_no
and s.to_date='9999-01-01';
```

60.统计salary的累计和running_total

2020年5月18日 18:19

题目描述

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`));
```

输出格式:

```
emp_no          salary    running_total
10001   88958   88958
10002   72527   161485
10003   43311   204796
10004   74057   278853
10005   94692   373545
10006   43311   416856
10007   88070   504926
10009   95409   600335
10010   94409   694744
10011   25828   720572
```

```
select s.emp_no, s.salary,(
   select sum(s2.salary)
   from salaries s2
   where s2.emp_no <= s.emp_no
   and s2.to_date = '9999-01-01'
) running_total
from salaries s
where s.to_date = '9999-01-01'
order by s.emp_no asc;
```

61.对于employees表中，给出奇数行的first_name

2020年5月18日 18:19

题目描述

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`));
```

输出格式:

```
first_name
Georgi
Chirstian
Anneke
Tzvetan
Saniya
Mary
```

题目描述错误:

更改为： 对first_name 先排序 后给出奇数行

```
select e1.first_name
from employees e1
where(
   select count(*)
   from employees e2
   where e1.first_name <= e2.first_name
) % 2 = 1;
```