**每日作业卷答案**

**javaweb第20天mysql**

传智播客.黑马程序员

# 关卡1

## 训练案例1

### 训练描述

常用聚合函数使用说明

* count：统计指定列不为NULL的记录行数；
* sum：计算指定列的数值和，如果指定列类型不是数值类型，那么计算结果为0；
* max：计算指定列的最大值，如果指定列是字符串类型，那么使用字符串排序运算；
* min：计算指定列的最小值，如果指定列是字符串类型，那么使用字符串排序运算；
* avg：计算指定列的平均值，如果指定列类型不是数值类型，那么计算结果为0；

### 操作步骤答案

1. 基于products表的数据,进行聚合函数查询.
2. 商品汇总,总价等查询。

#1 查询商品的总条数

SELECT COUNT(\*) FROM product;

#2 查询价格大于200商品的总条数

SELECT COUNT(\*) FROM product WHERE price > 200;

#3 查询分类为'c001'的所有商品的总和

SELECT SUM(price) FROM product WHERE category\_id = 'c001';

#4 查询分类为'c002'所有商品的平均价格

SELECT AVG(price) FROM product WHERE category\_id = 'c002';

#5 查询商品的最大价格和最小价格

SELECT MAX(price),MIN(price) FROM product;

## 训练案例2

### 训练描述

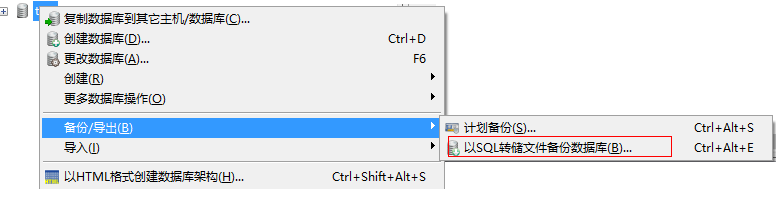
能完成sqlyog实现数据库备份和恢复

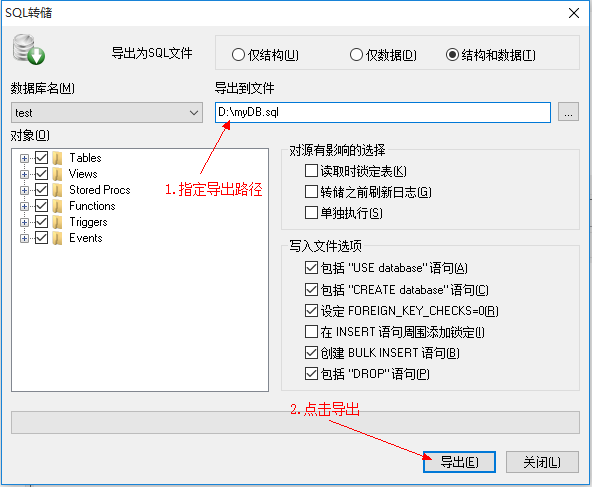
### 操作步骤答案

1:使用sqlyog连接数据库

2:操作备份和恢复数据库

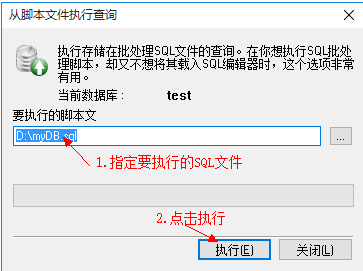
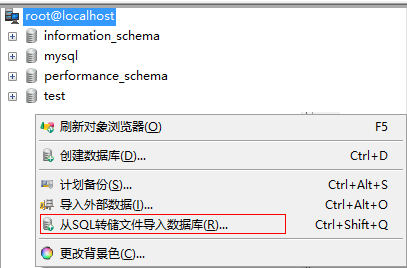
选中数据库，右键 ”备份/导出” ， 指定导出路径，保存成.sql文件即可。





恢复

数据库列表区域右键“从SQL转储文件导入数据库”， 指定要执行的SQL文件，执行即可。



## 训练案例3

### 训练描述

使用group by 完成商品分类查询

分组查询是指使用group by字句对查询信息进行分组。

* 格式：

SELECT 字段1,字段2… FROM 表名 GROUP BY分组字段 HAVING 分组条件;

分组操作中的having子语句，是用于在分组后对数据进行过滤的，作用类似于where条件。

* having与where的区别:
  + having是在分组后对数据进行过滤.

where是在分组前对数据进行过滤

* + having后面可以使用分组函数(统计函数)

where后面不可以使用分组函数。

### 操作步骤答案

#1 统计各个分类商品的个数

SELECT category\_id ,COUNT(\*) FROM product GROUP BY category\_id ;

#2 统计各个分类商品的个数,且只显示个数大于1的信息

SELECT category\_id ,COUNT(\*) FROM product GROUP BY category\_id HAVING COUNT(\*) > 1;

## 训练案例4

### 训练描述

实现表字段的主键约束,自动增长,非空约束,唯一性约束,外键约束.

### 操作步骤答案

1:创建表的时候,添加主键

* 方式一：创建表时，在字段描述处，声明指定字段为主键：

CREATE TABLE Persons

(

Id\_P int PRIMARY KEY,

LastName varchar(255),

FirstName varchar(255),

Address varchar(255),

City varchar(255)

)

2:创建表完成之后添加主键

* 方式二：创建表之后，通过修改表结构，声明指定字段为主键：

ALTER TABLE Persons ADD [CONSTRAINT 名称] PRIMARY KEY (字段列表)

CREATE TABLE Persons

(

FirstName varchar(255),

LastName varchar(255),

Address varchar(255),

City varchar(255)

)

ALTER TABLE Persons ADD PRIMARY KEY (FirstName,LastName)

自动增长:

我们通常希望在每次插入新记录时，数据库自动生成字段的值。

我们可以在表中使用 auto\_increment（自动增长列）关键字，自动增长列类型必须是整形，自动增长列必须为键(一般是主键)。

* 下列 SQL 语句把 "Persons" 表中的 "P\_Id" 列定义为 auto\_increment 主键

CREATE TABLE Persons

(

P\_Id int PRIMARY KEY AUTO\_INCREMENT,

LastName varchar(255),

FirstName varchar(255),

Address varchar(255),

City varchar(255)

)

* 向persons添加数据时，可以不为P\_Id字段设置值，也可以设置成null，数据库将自动维护主键值：

INSERT INTO Persons (FirstName,LastName) VALUES ('Bill','Gates')

INSERT INTO Persons (P\_Id,FirstName,LastName) VALUES (NULL,'Bill','Gates')

* 面试：delete和truncate的区别
  + Delete删除表中的数据，但不重置auto-increment记录数。
  + Truncate删除表中的数据，auto-increment记录数将重置。Truncate其实先删除表然后再创建表。
* 扩展：默认AUTO\_INCREMENT 的开始值是 1，如果希望修改起始值，请使用下列 SQL 语法：

ALTER TABLE Persons AUTO\_INCREMENT=100

3:唯一性约束unique

UNIQUE 约束唯一标识数据库表中的每条记录。

UNIQUE 和 PRIMARY KEY 约束均为列或列集合提供了唯一性的保证。

PRIMARY KEY 拥有自动定义的 UNIQUE 约束。

请注意，每个表可以有多个 UNIQUE 约束，但是每个表只能有一个 PRIMARY KEY 约束。

* 方式1：创建表时，在字段描述处，声明唯一：

CREATE TABLE Persons

(

Id\_P int UNIQUE,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255)

)

* 方式2：创建表时，在约束区域，声明唯一：

CREATE TABLE Persons

(

Id\_P int,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255),

CONSTRAINT 名称UNIQUE (Id\_P)

)

* 方式3：创建表后，修改表结构，声明字段唯一：

ALTER TABLE Persons ADD [CONSTRAINT 名称] UNIQUE (Id\_P)

4:非空约束 not null

NOT NULL 约束强制列不接受 NULL 值。

NOT NULL 约束强制字段始终包含值。这意味着，如果不向字段添加值，就无法插入新记录或者更新记录。

* 下面的 SQL 语句强制 "Id\_P" 列和 "LastName" 列不接受 NULL 值：

CREATE TABLE Persons

(

Id\_P int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255)

)

5:外键约束: foreign key

FOREIGN KEY 表示外键约束，将在多表中学习。

## 训练案例

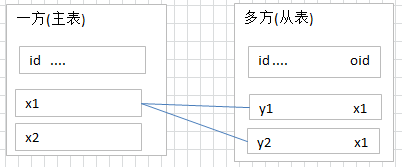
### 训练描述

多表设计原则

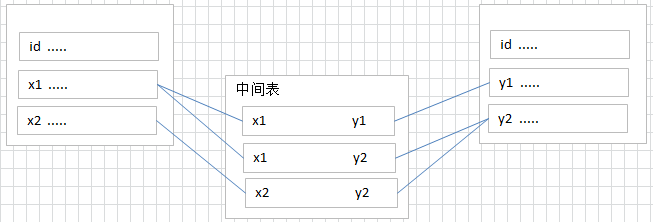
一对多表设计以及数据录入.

多对多表设计以及数据录入.

* 一对多关系：
  + 常见实例：客户和订单，分类和商品，部门和员工.
  + 一对多建表原则：在从表(多方)创建一个字段，字段作为外键指向主表(一方)的主键.



* 多对多关系：
  + 常见实例：学生和课程、用户和角色
  + 多对多关系建表原则：需要创建第三张表,中间表中至少两个字段，这两个字段分别作为外键指向各自一方的主键.

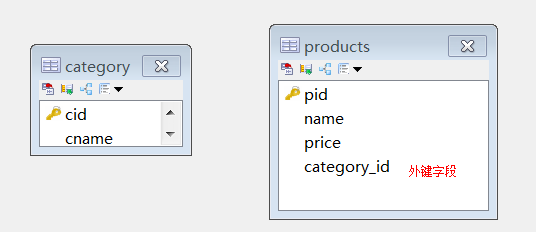


* 一对一关系：(了解)
  + 在实际的开发中应用不多.因为一对一可以创建成一张表.
  + 两种建表原则：
    - 外键唯一：主表的主键和从表的外键（唯一），形成主外键关系，外键唯一unique。
    - 外键是主键：主表的主键和从表的主键，形成主外键关系。

外键约束:

现在我们有两张表“分类表”和“商品表”，为了表明商品属于哪个分类，通常情况下，我们将在商品表上添加一列，用于存放分类cid的信息，此列称为：外键





此时“分类表category”称为：主表，“cid”我们称为主键。“商品表products”称为：从表，category\_id称为外键。我们通过主表的主键和从表的外键来描述主外键关系，呈现就是一对多关系。

外键特点：

* + - 从表外键的值是对主表主键的引用。
    - 从表外键类型，必须与主表主键类型一致。
* 声明外键约束

语法：alter table 从表 add [constraint] [外键名称] foreign key (从表外键字段名) references 主表 (主表的主键);

[外键名称] 用于删除外键约束的，一般建议“\_fk”结尾

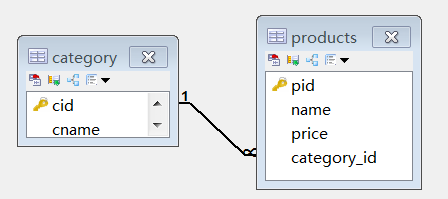
alter table 从表 drop foreign key 外键名称

* 使用外键目的：
  + 保证数据完整性

### 操作步骤答案

1:完成1:n表设计,外键约束.

分析1:n表设计



* category分类表，为一方，也就是主表，必须提供主键cid
* products商品表，为多方，也就是从表，必须提供外键category\_id

分类和商品表设计实现

###创建分类表

create table category(

cid varchar(32) PRIMARY KEY ,

cname varchar(100) #分类名称

);

# 商品表

CREATE TABLE `products` (

`pid` varchar(32) PRIMARY KEY ,

`name` VARCHAR(40) ,

`price` DOUBLE

);

#添加外键字段

alter table products add column category\_id varchar(32);

#添加约束

alter table products add constraint product\_fk foreign key (category\_id) references category (cid);

操作 1:n数据的录入

#1 向分类表中添加数据

INSERT INTO category (cid ,cname) VALUES('c001','服装');

#2 向商品表添加普通数据,没有外键数据，默认为null

INSERT INTO products (pid,pname) VALUES('p001','商品名称');

#3 向商品表添加普通数据，含有外键信息(category表中存在这条数据)

INSERT INTO products (pid ,pname ,category\_id) VALUES('p002','商品名称2','c001');

#4 向商品表添加普通数据，含有外键信息(category表中不存在这条数据) -- 失败,异常

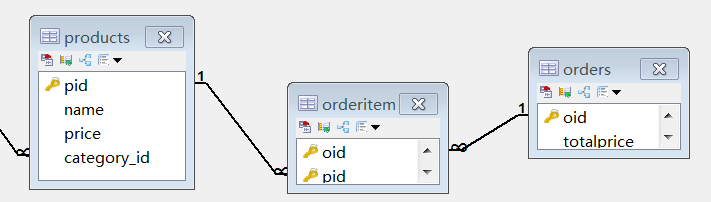
INSERT INTO products (pid ,pname ,category\_id) VALUES('p003','商品名称2','c999');

#5 删除指定分类(分类被商品使用) -- 执行异常

DELETE FROM category WHERE cid = 'c001';

2:完成n:n表设计,中间表设计原则以及联合主键的使用.

1: 分析:



* 商品和订单多对多关系，将拆分成两个一对多。
* products商品表，为其中一个一对多的主表，需要提供主键pid
* orders 订单表，为另一个一对多的主表，需要提供主键oid
* orderitem中间表，为另外添加的第三张表，需要提供两个外键oid和pid

2:实现表设计

### 商品表[已存在]

### 订单表

create table `orders`(

`oid` varchar(32) PRIMARY KEY ,

`totalprice` double #总计

);

### 订单项表

create table orderitem(

oid varchar(50),-- 订单id

pid varchar(50)-- 商品id

);

###---- 订单表和订单项表的主外键关系

alter table `orderitem` add constraint orderitem\_orders\_fk foreign key (oid) references orders(oid);

###---- 商品表和订单项表的主外键关系

alter table `orderitem` add constraint orderitem\_product\_fk foreign key (pid) references products(pid);

### 联合主键

alter table `orderitem` add primary key (oid,pid);

3:多对多数据录入

#1 向商品表中添加数据

INSERT INTO products (pid,pname) VALUES('p003','商品名称');

#2 向订单表中添加数据

INSERT INTO orders (oid ,totalprice) VALUES('x001','998');

INSERT INTO orders (oid ,totalprice) VALUES('x002','100');

#3向中间表添加数据(数据存在)

INSERT INTO orderitem(pid,oid) VALUES('p001','x001');

INSERT INTO orderitem(pid,oid) VALUES('p001','x002');

INSERT INTO orderitem(pid,oid) VALUES('p002','x002');

#4删除中间表的数据

DELETE FROM orderitem WHERE pid='p002' AND oid = 'x002';

#5向中间表添加数据(数据不存在) -- 执行异常

INSERT INTO orderitem(pid,oid) VALUES('p002','x003');

#6删除商品表的数据 -- 执行异常

DELETE FROM products WHERE pid = 'p001';

# 关卡2

## 训练案例1

### 训练描述

1:设计员工表emp和部门表dept

要求: emp表的基本字段 empno int 主键 自动自增长,ename 字符类型 salary 数值类型 bonus 奖金数值型……deptno int 员工在的部门 (外键)

dept表的基本字段 deptno int 主键 自动增长 , dname 部门名称 字符类型 dlocation 地理位置 字符类型

2: 设计表结构 在emp表中设计deptno外键.

### 操作步骤答案

1:完成emp和dept 1:n设计

2:录入初始化数据

create table dept(

deptno int primary key auto\_increment,

dname varchar(14) ,

dlocation varchar(13)

) ;

create table emp(

empno int primary key auto\_increment,

ename varchar(10),

JOB varchar(9),

MGR int,

HIREDATE date,

SAL int,

COMM int,

DEPTNO int not null,

foreign key (DEPTNO) references dept(deptno)

);

-- 录入相关数据

INSERT INTO dept VALUES(10,'ACCOUNTING','NEW YORK');

INSERT INTO dept VALUES(20,'RESEARCH','DALLAS');

INSERT INTO dept VALUES(30,'SALES','CHICAGO');

INSERT INTO dept VALUES(40,'OPERATIONS','BOSTON');

INSERT INTO emp VALUES(7369,'SMITH','CLERK',7902,'1980-12-17',800,NULL,20);

INSERT INTO emp VALUES(7499,'ALLEN','SALESMAN',7698,'1981-02-20',1600,300,30);

INSERT INTO emp VALUES(7521,'WARD','SALESMAN',7698,'1981-02-22',1250,500,30);

INSERT INTO emp VALUES(7566,'JONES','MANAGER',7839,'1981-04-02',2975,NULL,20);

INSERT INTO emp VALUES(7654,'MARTIN','SALESMAN',7698,'1981-09-28',1250,1400,30);

INSERT INTO emp VALUES(7698,'BLAKE','MANAGER',7839,'1981-05-01',2850,NULL,30);

INSERT INTO emp VALUES(7782,'CLARK','MANAGER',7839,'1981-06-09',2450,NULL,10);

INSERT INTO emp VALUES(7788,'SCOTT','ANALYST',7566,'1987-06-13',3000,NULL,20);

INSERT INTO emp VALUES(7839,'KING','PRESIDENT',NULL,'1981-11-17',5000,NULL,10);

INSERT INTO emp VALUES(7844,'TURNER','SALESMAN',7698,'1981-09-08',1500,0,30);

INSERT INTO emp VALUES(7876,'ADAMS','CLERK',7788,'1987-06-13',1100,NULL,20);

INSERT INTO emp VALUES(7900,'JAMES','CLERK',7698,'1981-12-03',950,NULL,30);

INSERT INTO emp VALUES(7902,'FORD','ANALYST',7566,'1981-12-03',3000,NULL,20);

INSERT INTO emp VALUES(7934,'MILLER','CLERK',7782,'1983-01-23',1300,NULL,10);

## 训练案例2

### 训练描述

1:完成学员student表,老师 teacher 表,课程表course表的设计.

2:多对多设计原则,引入中间表.

### 操作步骤答案

1: 完成学员和老师表设计

create table student(

id int primary key auto\_increment,

name varchar(20),

city varchar(10),

age int

);

create table teacher(

id int primary key auto\_increment,

name varchar(20)

);

create table course(

id int primary key auto\_increment,

name varchar(20),

teacher\_id int,

foreign key (teacher\_id) references teacher(id)

);

create table student\_course(

student\_id int,

course\_id int,

score int,

foreign key (student\_id) references student(id),

foreign key (course\_id) references course(id)

);

2: 录入相关数据

insert into teacher values(null,'关羽');

insert into teacher values(null,'张飞');

insert into teacher values(null,'赵云');

insert into student values(null,'小王','北京',20);

insert into student values(null,'小李','上海',18);

insert into student values(null,'小周','北京',22);

insert into student values(null,'小刘','北京',21);

insert into student values(null,'小张','上海',22);

insert into student values(null,'小赵','北京',17);

insert into student values(null,'小蒋','上海',23);

insert into student values(null,'小韩','北京',25);

insert into student values(null,'小魏','上海',25);

insert into student values(null,'小明','北京',20);

insert into course values(null,'语文',1);

insert into course values(null,'数学',1);

insert into course values(null,'生物',2);

insert into course values(null,'化学',2);

insert into course values(null,'物理',2);

insert into course values(null,'英语',3);

insert into student\_course values(1,1,80);

insert into student\_course values(1,2,90);

insert into student\_course values(1,3,85);

insert into student\_course values(1,4,78);

insert into student\_course values(2,2,53);

insert into student\_course values(2,3,77);

insert into student\_course values(2,5,80);

insert into student\_course values(3,1,71);

insert into student\_course values(3,2,70);

insert into student\_course values(3,4,80);

insert into student\_course values(3,5,65);

insert into student\_course values(3,6,75);

insert into student\_course values(4,2,90);

insert into student\_course values(4,3,80);

insert into student\_course values(4,4,70);

insert into student\_course values(4,6,95);

insert into student\_course values(5,1,60);

insert into student\_course values(5,2,70);

insert into student\_course values(5,5,80);

insert into student\_course values(5,6,69);

insert into student\_course values(6,1,76);

insert into student\_course values(6,2,88);

insert into student\_course values(6,3,87);

insert into student\_course values(7,4,80);

insert into student\_course values(8,2,71);

insert into student\_course values(8,3,58);

insert into student\_course values(8,5,68);

insert into student\_course values(9,2,88);

insert into student\_course values(10,1,77);

insert into student\_course values(10,2,76);

insert into student\_course values(10,3,80);

insert into student\_course values(10,4,85);

insert into student\_course values(10,5,83);

# 关卡3

## 训练案例1

### 训练描述

1:基于设计和学员表,老师表,课程表以及中间表的数据,完成相关查询

### 操作步骤答案

-- 1、查询平均成绩大于70分的同学的学号和平均成绩

-- 查询每个学生的平均分

select student\_id, avg(score) from studentcourse

group by student\_id;

-- 增加条件：平均成绩大于70

select student\_id, avg(score) from studentcourse

group by student\_id

having avg(score)>70;

-- 2、查询所有同学的学号、姓名、选课数、总成绩

-- 2.1从学生课程中间表 查询学号，课程id,成绩

select student\_id,course\_id,score from studentcourse;

-- 2.2 根据学号分组 求出每个学生的选课数和总成绩

select student\_id,count(course\_id),sum(score) from studentcourse

group by student\_id;

-- 2.3 将2.2作为临时表 和 学生表 关联查询

select student.id,student.name,temp.countCourse,temp.sumScore

from student, (select student\_id,count(course\_id) countCourse,sum(score) sumScore from studentcourse

group by student\_id) temp

where student.id=temp.student\_id;

-- 3、查询学过赵云老师所教课的同学的学号、姓名

-- 3.1 查询赵云老师教了那些课

select course.id from teacher,course

where teacher.id=course.teacher\_id

and teacher.name='赵云';

-- 3.2 查询那些学生上了赵云老师的课

select student\_id from studentcourse

where course\_id in (select course.id from teacher,course

where teacher.id=course.teacher\_id

and teacher.name='赵云');

-- 3.3 根据学号查询学生信息

select id, name from student

where id in (select student\_id from studentcourse

where course\_id in (select course.id from teacher,course

where teacher.id=course.teacher\_id

and teacher.name='赵云'));

-- 4、查询没学过关羽老师课的同学的学号、姓名

-- 4.1 找到关羽的id

select id from teacher where name='关羽';

-- 4.2 查询关羽教授的课程

select id from course

where teacher\_id=(select id from teacher where name='关羽');

-- 4.3 查询那些学生上了关羽老师的课程

select distinct student\_id from studentcourse

where course\_id in (select id from course

where teacher\_id=(select id from teacher where name='关羽'));

-- 4.4 反向查询学号和姓名

select id,name from student

where id not in (select student\_id from studentcourse

where course\_id in (select id from course

where teacher\_id=(select id from teacher where name='关羽'))

);

-- 5、查询没有学三门课以上的同学的学号、姓名

-- 5.1 查询每个学生学了几门课

select student\_id, count(course\_id) from studentcourse

group by student\_id

;

-- 5.2 查询每个学生学了几门课 条件1：小于等于三门

select student\_id, count(course\_id) from studentcourse

group by student\_id

having count(course\_id)<=3

;

-- 5.3 查询 学号和姓名， 将5.2 作为临时表

select id,name from student, (select student\_id, count(course\_id) from studentcourse

group by student\_id

having count(course\_id)<=3) temp

where student.id=temp.student\_id;

-- 6、查询各科成绩最高和最低的分

-- 根据学科分组 最高分和最低分

select course\_id,max(score),min(score)

from studentcourse

group by course\_id;

## 训练案例2

### 训练描述

1:基于设计和员工表和部门表的数据完成1:n复杂查询

### 操作步骤答案

1．列出至少有4个员工的所有部门。

分析：每个部门有多少员工，

select count(\*) from emp group by deptno

select \* from dept where deptno in (

select deptno from emp group by deptno having count(\*) > 4

);

2．列出薪金比“SMITH”多的所有员工。

分析：先查出SMITH的工资

select \* from emp where sal > (select sal from emp where ename = 'SMITH');

3．\*\*\*\*\* 列出所有员工的姓名及其直接上级的姓名。

分析:把一张表当成两张表

select e1.ename , e2.ename as mgr\_name from emp e1 ,emp e2 where e1.empno = e2.mgr;

4．列出受雇日期早于其直接上级的所有员工。

SELECT t1.ename,t2.ename AS mgr\_name FROM emp t1, emp t2 WHERE t1.mgr = t2.empno AND t1.hiredate < t2.hiredate;

5．列出部门名称和这些部门的员工信息，同时列出那些没有员工的部门。

select \* from dept d left join emp e on d.deptno = e.deptno;

6．列出所有“CLERK”（办事员）的姓名及其部门名称。

---分析数据来源

select \* from emp e ,dept d where e.deptno = d.deptno and e.job = 'CLERK';

7．列出最低薪金大于1500的各种工作。

分析：工作的最低薪金

select min(sal) from emp group by job;

select job from emp group by job having min(sal) > 1500;