8-12练习答案.sql

第8章 子查询

1.编写一个查询，显示和Zlotkey在同一个部门的所有员工的姓氏和入职日期，但不包括Zlotkey

```
select last_name,hire_date
from employees
where department_id=(
                     select department_id
                     from employees
                     where last_name='Zlotkey')
and last_name <> 'Zlotkey';
```

2.创建一个查询，显示年薪（工资*12加奖金）超过平均年薪的所有员工的员工编号、姓氏、部门名称和部门所在地。

```
select employee_id,last_name,department_name,city
from employees join departments using(department_id)
join locations using(location_id)
where salary*12*(1+nvl(commission_pct,0))>(
                                           select
avg(salary*12*(1+nvl(commission_pct,0)))
                                           from
employees);
```

3.编写一个查询，显示所有员工的员工编号，姓氏，部门ID，条件是他们所工作的部门里有员工姓氏包含且只包含一个u，
输出结果里，排除这些姓氏里包含且只包含一个u的结果。

```
select employee_id,last_name,department_id
from employees
where department_id in(
                       select department_id
                       from employees
                       where
instr(last_name,'u',1,1)>0
                       and instr(last_name,'u',1,2)=0)
and (last_name not like '%u%'
or instr(last_name,'u',1,2)>0);
```

4.显示部门的location ID为1700的所有员工姓氏、部门编号和职务。

```
select last_name,department_id,job_id
from employees join departments using(department_id)
where location_id=1700;
select last_name,department_id,job_id
from employees
where department_id in(
                        select department_id
                        from departments
                        where location_id=1700);
```

5.显示King的每个下属员工的姓氏、工资、经理ID和经理姓氏。

```
select a.last_name,a.salary,a.manager_id,b.last_name
from employees a join employees b
on(a.manager_id=b.employee_id)
where b.last_name='King';
select a.last_name,a.salary,a.manager_id,b.last_name
from employees a join employees b
on (a.manager_id=b.employee_id)
where a.manager_id in(
                    select employee_id
                    from employees
                    where last_name='King')
```

6.查询部门工作地点在Seattle，且职务与111号员工相同的所有员工的员工编号、姓氏、部门编号和职务。

```
select employee_id,last_name,department_id,job_id
from employees
where department_id in (
                         select department_id
                         from departments join
locations
                         using(location_id)
                         where city='Seattle')
and job_id=(
            select job_id
            from employees
            where employee_id=111)
```

8-12练习答案.sql

7.查询工资超过平均工资，且部门里有员工姓氏中包含字母u的所有员工的员工编号，姓氏和工资。

```
select employee_id,last_name,salary
from employees
where salary>(
               select avg(salary)
               from employees)
and department_id in(
                     select department_id
                     from employees
                     where last_name like '%u%')
```

第9章 集合

1.列出不包含职务ST_CLERK的所有部门ID

```
select department_id from employees
minus
select department_id from employees where
job_id='ST_CLERK'
```

2.显示没有设立部门的国家代码和国家名称

```
select country_id,country_name
from countries join locations using(country_id)
where location_id in(
select location_id from locations
minus
select location_id from departments);

select country_id,country_name
from countries join locations using(country_id)
where location_id in(
select location_id from locations
minus
select location_id from departments)
minus
select country_id,country_name
from countries join locations using(country_id)
```

```
where location_id in(
                     select location_id
                     from departments);
```

3.不使用DISTINCT，输出设立有部门的国家代码

```
select country_id,country_name
from countries join locations using(country_id)
where location_id in(
                     select location_id
                     from departments)
intersect
select country_id,country_name
from countries;
```

4.创建一个查询，显示部门10，50，20部门所包含的职务和部门ID，要求部门显示的顺序以10，50，20显示。

```
select job_id,department_id
from employees
where department_id=10
union all
select job_id,department_id
from employees
where department_id=50
union all
select job_id,department_id
from employees
where department_id=20
```

5.列出哪些员工，从事当前职务之前，还在公司任职过其它职务，输出该员工的ID，以及以前任职的其它职务的ID

```
select employee_id,job_id
from job_history
minus
select employee_id,job_id
from employees;
```

6.整合以下两个查询，合并成一个输出。

1）显示所有员工的姓氏和部门ID，不管该员工是否有部门。

2）显示所有部门的ID和部门名称，不管该部门是否有员工。

```
select last_name,department_id,to_char(null)
```

```
department_name from employees
union
select to_char(null),department_id,department_name
from departments;
```

第10章 处理数据

1.执行下发的脚本create_my_emp.sql创建本练习所需要的表。

```
create table my_employee
(id number primary key,
last_name varchar2(40),
first_name varchar2(40),
userid varchar2(40),
salary number);
```

2.描述该表的表结构

```
desc my_employee
```

3.将下面数据表中的第一行添加到MY_EMPLOYEE表中，不要在INSERT子句中，出现这些列名。

```
insert into my_employee
values(1,'Patel','Ralph','rpatel',895);
```

4.使用上述表的第二行数据添加到MY_EMPLOYEE表中，要求在INSERT子句中显式的列出这些列名。

```
insert into my_employee
(id,last_name,first_name,userid,salary)
values(2,'Dancs','Betty','bdancs',860);
```

5.确认添加到MY_EMPLOYEE表中的内容。

```
select * from my_employee;
```

6.要求，编写一个脚本，名为loademp.sql，功能是在MY_EMPLOYEE表中插入示例表中的数据，要求使用替代变量，使该脚本可以被重复调用，并且USERID列，使用表达式构造出来，构造方法，将名字的第一个字母与姓氏的前7个字母连接起来，并且小写。使用该脚本将示例数据中的第三、四行插入MY_EMPLOYEE表。

```
undefine last_name;
undefine first_name;
insert into my_employee
```

```
values(&id,'&&last_name','&&first_name',lower(substr('&first_name',1,1)||substr('&last_name',1,7)),&salary);
```
7.确认MY_EMPLOYEE表中的结果
```
select * from my_employee;
```
8.将刚才添加的数据永久化。
```
commit;
```
9.将员工3的姓氏改为Drexler
```
update my_employee set last_name='Drexler' where id=3;
```
10.将工资低于900的员工工资改为1000
```
update my_employee set salary=1000 where salary<900;
```
11.验证对表所做的更改
```
select * from my_employee;
```
13.将Betty Dancs从MY_EMPLOYEE表中删除
```
delete my_employee where first_name||last_name='BettyDancs';
```
14.确认对表的修改
```
select * from my_employee;
```
15.提交所有更改
```
commit;
```
16.使用前面创建的loademp.sql添加示例数据的第五行。
```
undefine last_name;
undefine first_name;
insert into my_employee values(&id,'&&last_name','&&first_name',lower(substr('&first_name',1,1)||substr('&last_name',1,7)),&salary);
```
17.确认表中的内容
```
select * from my_employee;
```
以下题目需要标记事务处理过程中的记录点
18.要求修改员工工资为1550的员工工资为3000
```
update my_employee set salary=3000 where salary=1550;
savepoint sp1;
```
19.删除Ben Biri所在行
```
delete my_employee where id=3;
```

```
savepoint sp2;
```
20.使用loademp.sql脚本将删除的Betty Dancs添加回来
```
undefine last_name;
undefine first_name;
insert into my_employee
values(&id,'&&last_name','&&first_name',lower(substr(
'&first_name',1,1)||substr('&last_name',1,7)),&salary
);
savepoint sp3;
```
21.要求回退到插入Betty Dancs之前，要求要保留之前18，19题的对表的修改。
```
rollback to sp2;
```
22.将18题以后的对数据的操作永久化。
```
commit;
select * from my_employee;
```

第11章 创建和管理表

1.基于以下表格创建表DEPT，将创建表的语句保存到lab10_1.sql中，执行该脚本完成本题
```
create table dept
(id number(7),
name varchar2(25));
```
2.用DEPARTMENTS表的数据填充DEPT表（department_id对应id，department_name对就name）。
```
insert into dept
select department_id,department_name from
departments;
commit;
```
3.基于以下表格创建表EMP。将创建语句保存到lab10_3.sql中，执行脚本完成本题
```
create table emp
(id number(7),
last_name varchar2(25),
first_name varchar2(25),
dept_id varchar2(7)
);
```

4.修改EMP表，从而允许输入更长的员工姓氏，由原来25个字符长度改为50
```
alter table emp modify(last_name varchar2(50));
```
5.确认DEPT和EMP表都在数据字典USER_TABLES中。
```
select * from user_tables where table_name
in('EMP','DEPT');
```
6.根据EMPLOYEES表中的数据填充EMP表，对应关系
employee_id对应id，last_name对应last_name，
first_name对应first_name，depart_id对应dept_id。
```
insert into emp
select
employee_id,last_name,first_name,to_char(department_id) from employees;
select * from emp;
```
7.删除EMP表。
```
drop table emp;
```
8.根据表EMPLOYEES表创建表EMPLOYEES2。仅包含
EMPLOYEE_ID,FIRST_NAME,LAST_NAME,SALARY和
DEPARTMENT_ID。
将新表的名列分别命名为ID、FIRST_NAME、LAST_NAME、
SALARY和DEPT_ID。
```
create table employees2
(id,first_name,last_name,salary,dept_id)
as
select
employee_id,first_name,last_name,salary,department_id
from employees;
```
9.将EMPLOYEES2重命名为EMP。
```
alter table employees2 rename to emp;
```
10.给DEPT和EMP两表添加备注，'create from departments'和'create from employees'
```
comment on table dept is 'create from departments';
comment on table emp is 'create from employees';
```
11.给EMP表的ID列添加PRIMARY KEY约束，给该约束命名为
my_emp_id_pk。
```
alter table emp add constraint my_emp_id_pk primary
```

key(id);
12.为DEPT表的ID列添加PRIMARY KEY约束，命名为my_dept_id_pk。
alter table dept add constraint my_dept_id_pk primary key(id);
13.将DEPT表的ID列作为EMP表的引用列，创建相关约束，确保不会将员工分配到不存在的部门。
将约束命名为my_emp_dept_id_fk。
alter table emp add constraint my_emp_dept_id_fk foreign key (dept_id) references dept(id);
14.通过查询数据字典user_constraints，确认前几题添加的约束。
select * from user_constraints
where table_name in('EMP','DEPT');
15.为EMP表，增加一个列COMMISSION，该列不允许为空值，默认值为0，该列数据类型为NUMBER，小数位2位，
数位长度为2。
alter table emp add (commission number(2,2) default 0 not null);

第12章 创建和管理其它数据库对象
1.基于EMPLOYEES表中的员工编号、员工姓名和部门编号，创建一个名为EMPLOYEES_VU的视图，
要求视图中员工姓名的列命名为EMPLOYEE。
create or replace view employees_vu
as
select employee_id,last_name||first_name employee,department_id
from employees;
2.显示EMPLOYEES_VU视图的内容及视图的结构。
select * from employees_vu;
desc employees_vu;
3.从USER_VIEWS数据字典中查看我们刚才创建的视图。
select * from user_views
where view_name='EMPLOYEES_VU';
4.使用EMPLOYEES_VU视图，创建一个查询，使其显示所有员

工的员工姓名和部门编号。
```
select employee,department_id from employees_vu;
```
5.创建一个名为DEPT50的视图，其中包含部门50中所有员工的员工编号、员工姓氏和部门编号以及部门名称，
将视图的各列命名为EMPNO、EMPLOYEE、DEPTNO和DEPTNAME。
不允许通过该视图将员工重新分配给其它部门。
```
create or replace view dept50
as
select employee_id empno,last_name
employee,department_id deptno,department_name
deptname
from employees join departments using(department_id)
where department_id=50
with check option;
```
6.显示DEPT50视图的结构和内容。
```
select * from dept50;
desc dept50;
```
7.尝试将Matos重新分配给部门80.
```
update dept50 set deptno=80 where employee='Matos';
```
8.基于所有员工的员工姓氏、部门名称、薪金和薪金等级，
创建一个名为SALARY_VU的视图，
各列命名为Employee、Department、Salary和Grade。
```
create or replace view salary_vu
as
select a.last_name "Employee",b.department_name
"Department",
a.salary*(1+nvl(a.commission_pct,0)) "Salary",c.grade
"Grade"
from employees a,departments b,job_grade c
where a.department_id=b.department_id
and a.salary between c.low_sal and c.high_sal;
```
9.创建一个用于DEPT表的主键列的序列。该序列从200开始，
最大值为1000，序号增量为10.将该序列命名为DEPT_ID_SEQ.
```
create sequence dept_id_seq
start with 200
increment by 10
```

```
maxvalue 1000
nocycle;
```

10.编写一个查询，使其显示有关序列的以下信息：序列名、最大值、增量值和最后编号。

```
desc user_sequences
select
sequence_name,max_value,increment_by,last_number from
user_sequences
where sequence_name='DEPT_ID_SEQ';
```

11.准备DEPT表，准备的语句

```
drop table dept;
Create table dept2 as select * from departments;
```

，使其在表DEPT中插入两行，要求插入时使用我们为ID列创建的序列。

添加两个名为Education和Administration的部门。确认添加的内容。

```
 desc dept2;
 insert into dept2 (department_id,department_name)
 values(dept_id_seq.nextval,'&name');
commit;
select * from dept2;
```