

# 《数据库概论》实验二 高级 SQL 实验报告

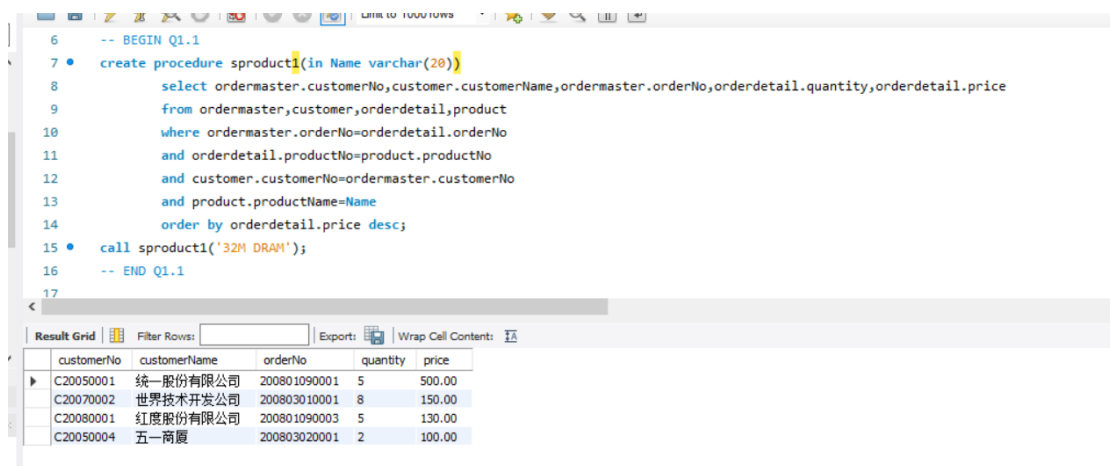
姓名 学号 联系方式

## 实验环境

[一句话介绍你使用的操作系统、软件版本]

## 实验过程

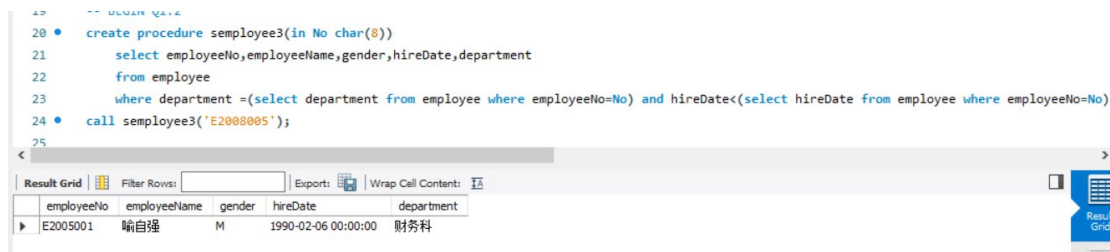
1-1、



```
6 -- BEGIN Q1.1
7 • create procedure sproduct1(in Name varchar(20))
8     select ordermaster.customerNo,customer.customerName,ordermaster.orderNo,orderdetail.quantity,orderdetail.price
9     from ordermaster,customer,orderdetail,product
10    where ordermaster.orderNo=orderdetail.orderNo
11    and orderdetail.productNo=product.productNo
12    and customer.customerNo=ordermaster.customerNo
13    and product.productName=Name
14    order by orderdetail.price desc;
15 • call sproduct1('32M DRAM');
16 -- END Q1.1
17
```

customerNo	customerName	orderNo	quantity	price
C20050001	统一股份有限公司	200801090001	5	500.00
C20070002	世界技术开发公司	200803010001	8	150.00
C20080001	红度股份有限公司	200801090003	5	130.00
C20050004	五一商厦	200803020001	2	100.00

1-2、



```
17 -- BEGIN Q1.2
18 • create procedure semployee3(in No char(8))
19     select employeeNo,employeeName,gender,hireDate,department
20     from employee
21    where department =(select department from employee where employeeNo=No) and hireDate<(select hireDate from employee where employeeNo=No);
22 • call semployee3('E2008005');
23
```

employeeNo	employeeName	gender	hireDate	department
E2005001	喻自强	M	1990-02-06 00:00:00	财务科

2-1、

```

35 • create function saverage (name varchar(20))
36 returns integer
37 begin
38 declare res integer;
39 select avg(orderdetail.price) into res
40 from product,orderdetail
41 where productName=name
42 and product.productNo=orderdetail.productNo;
43 return res;
44 end$$
45 delimiter ;
46 • select distinct product.productName,saverage(product.productName)
47 from product,orderdetail;

```

productName	saverage(product.productName)
龙基777F8纯平显示器	244
硕泰克SL-KBAN-RL主板	265
Pentium主板	220
9600bits/s调制解调	340
计算机字典	283

2-2、

```

55 • create function ssum (no char(9))
56 returns integer
57 begin
58 declare res integer;
59 select sum(orderdetail.quantity) into res
60 from orderdetail
61 where productNo=no;
62 return res;
63 end$$
64 delimiter ;
65 • select product.productNo,product.productName,ssum(product.productNo)
66 from product
67 where ssum(product.productNo)>4;

```

productNo	productName	ssum(product.productNo)
P20050001	32M DRAM	20
P20050003	120GB硬盘	8
P20050004	3.5寸软驱	7
P20050005	键盘	6
P20060002	网卡	7

3-1、

```

72 delimiter $$
73 • create trigger trigger1
74 after insert
75 on product for each row
76 begin
77 if (new.productPrice>1000)
78 then set new.productPrice=1000;
79 end if;
80 end$$
81 delimiter ;
82 • insert into product values('P20080004', '龙基777F8纯平显示器', '显示器', '1800.00'
83 )
84 -- END Q3.1

```

productNo	productName	productClass	productPrice
P20070003	计算机字典	图书	100.00
P20070004	9600bits/s调...	设备	320.00
P20080001	Pentium主板	主板	890.00
P20080002	硕泰克SL-K8...	主板	1100.00
P20080003	龙基777FT纯...	显示器	900.00
P20080004	龙基777F8纯...	显示器	1000.00
NULL	NULL	NULL	NULL

3-2、

```

88 delimiter $$
89 • create trigger trigger2
90 before insert
91 on ordermaster for each row
92 begin
93 if (select employee.hireDate from employee where new.employeeNo=employee.employeeNo < '1992-01-01 00:00:00')
94 then set employee.salary=employee.salary*1.08;
95 else set employee.salary=employee.salary*1.05;
96 end if;
97 end$$
98 delimiter ;
99 • insert into ordermaster values('200806120002', 'C20050002', 'E2005002', '2008-06-12 00:00:00', '0.00', 'I000000010')
100 )

```

4-1、

The screenshot shows a Python IDE with a project named 'dbexp2'. The main.py file contains the following code:

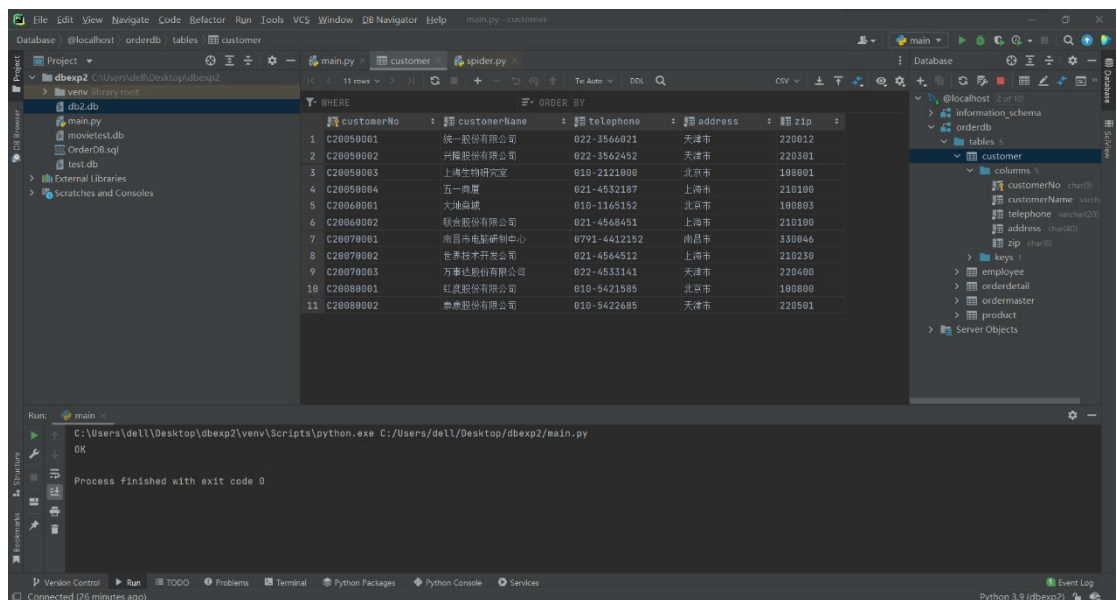
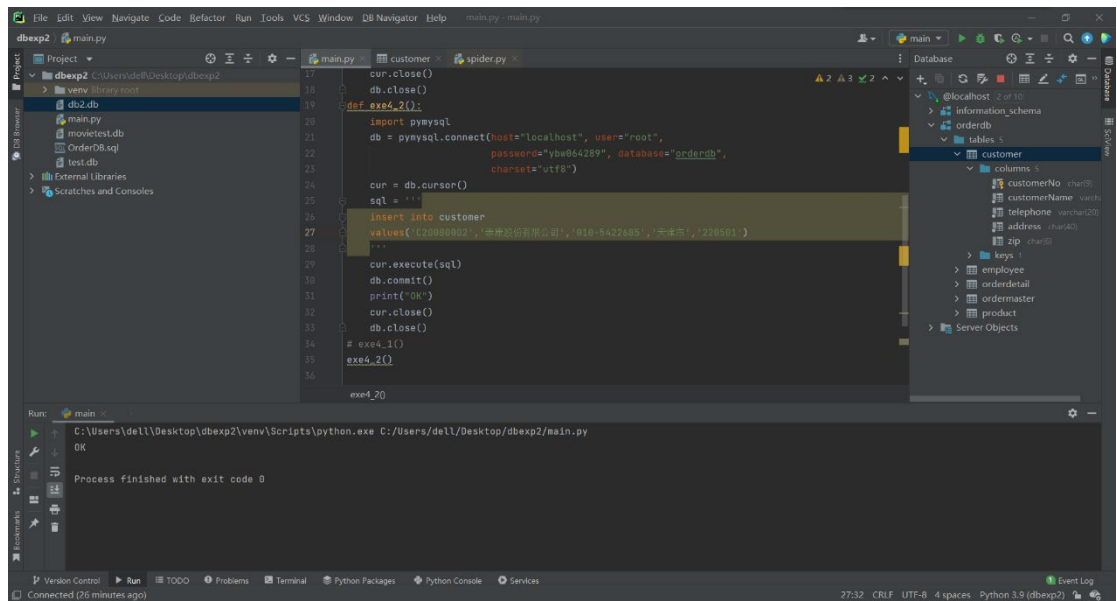
```

1 def exe4_1():
2     import pymysql
3     db = pymysql.connect(host='localhost', user='root',
4                          password='yba064289', database='orderdb',
5                          charset='utf8')
6     cur = db.cursor()
7     sql = '''
8     select employeeNo, employeeName, salary
9     from employee
10    order by salary desc ;
11    '''
12    cur.execute(sql)
13    for i in cur.fetchmany(20):
14        print(i)
15    db.commit()
16    exe4_10 = for i in cur.fetchmany(20)

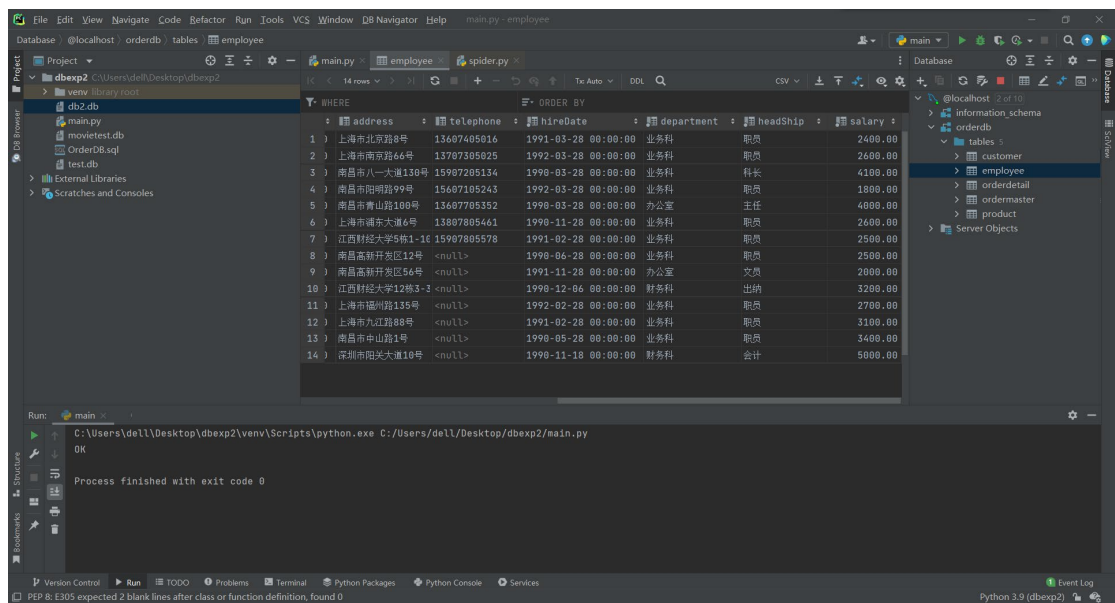
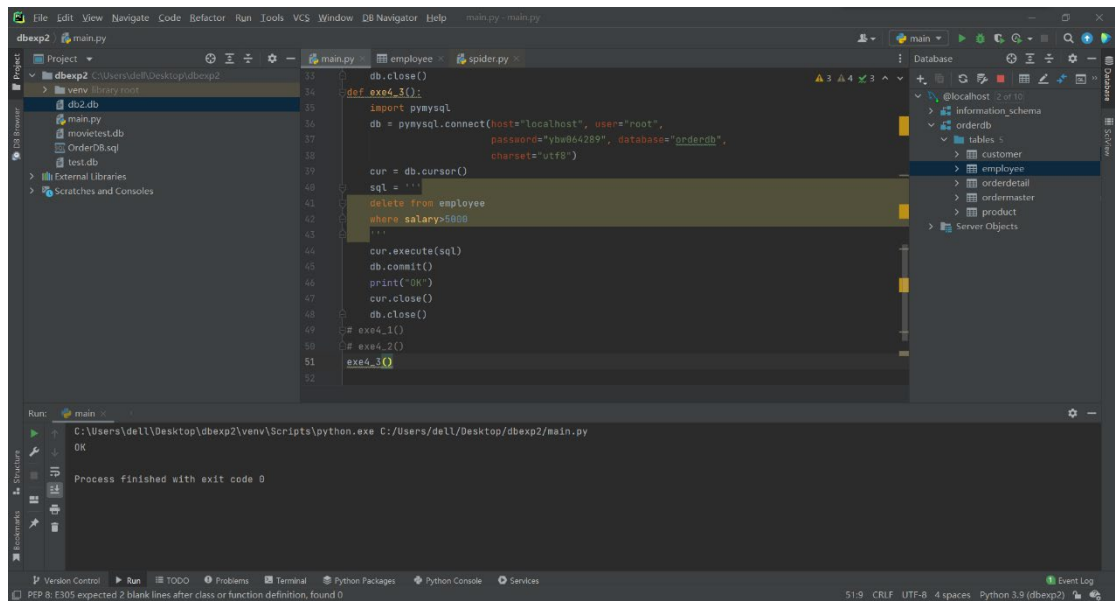
```

The Run console shows the output of the script, displaying a list of employee records with their IDs, names, and salaries, sorted by salary in descending order.

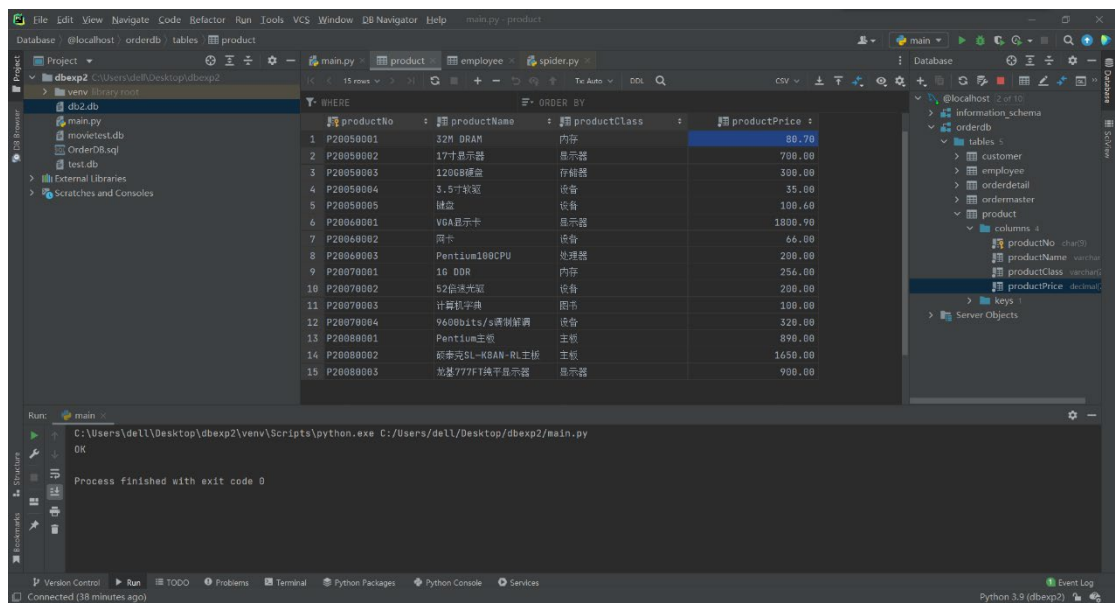
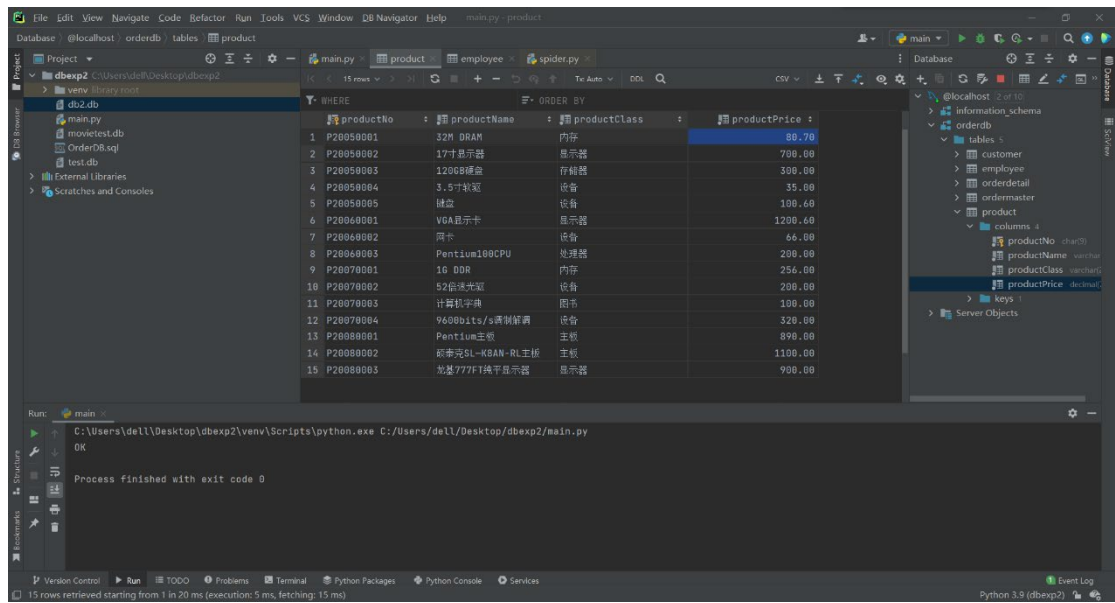
4-2、



4-3、

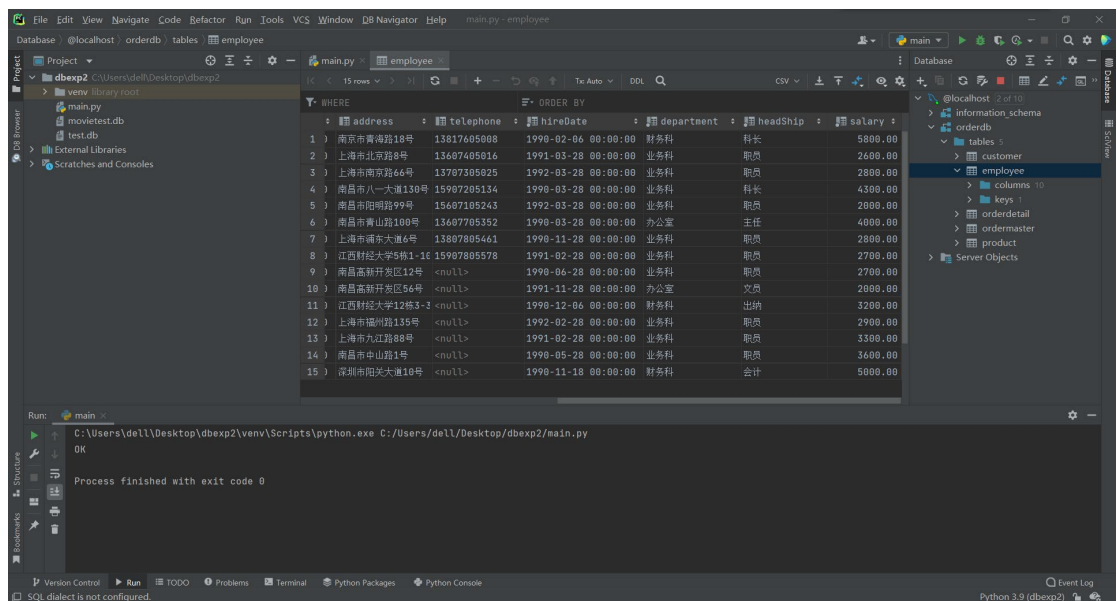
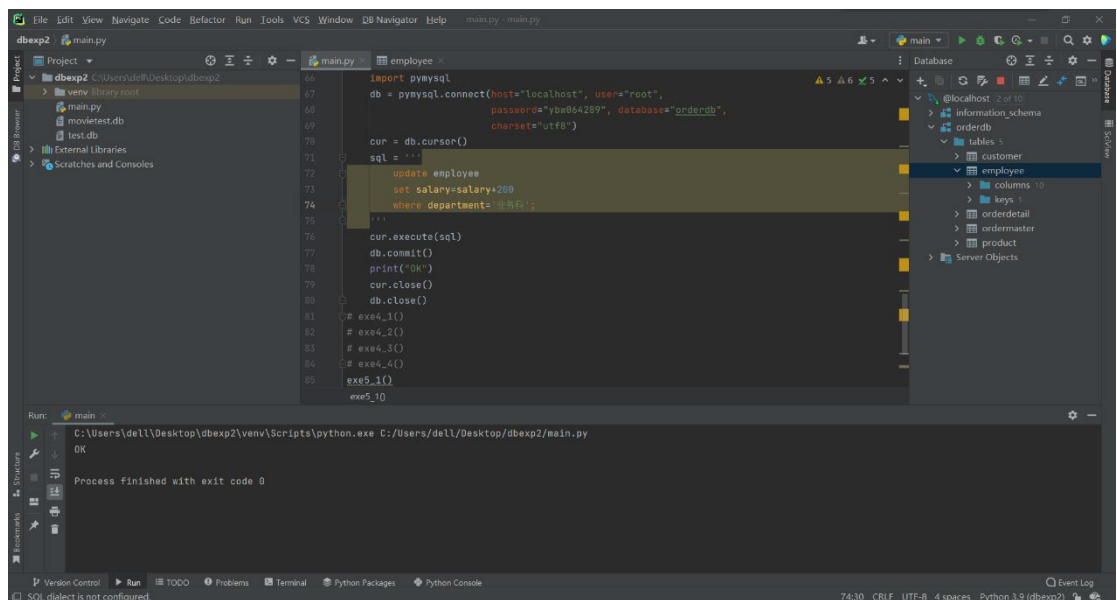
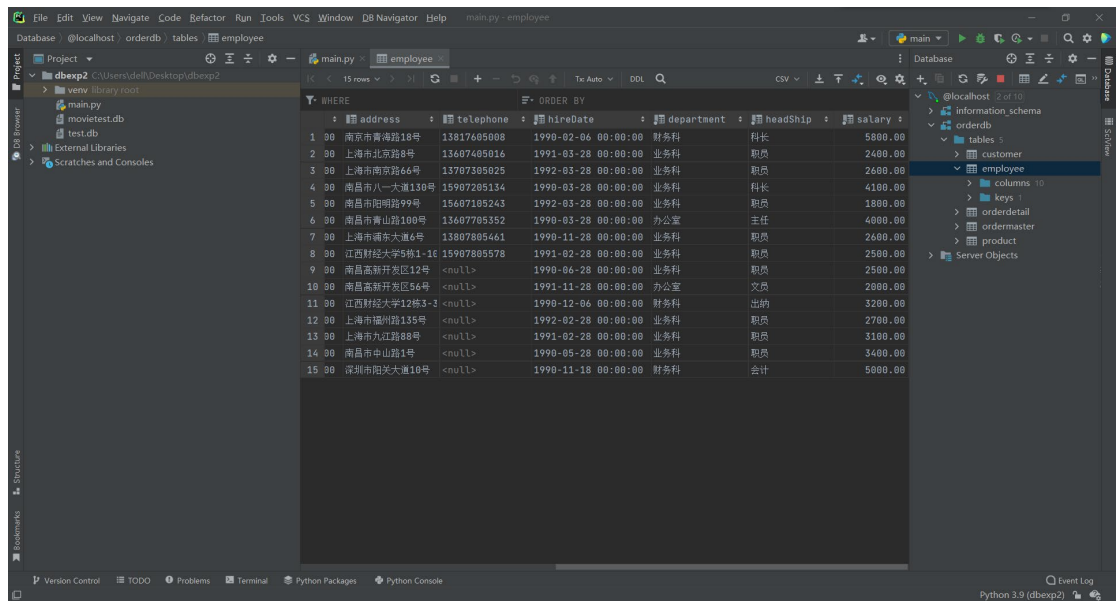


4-4、

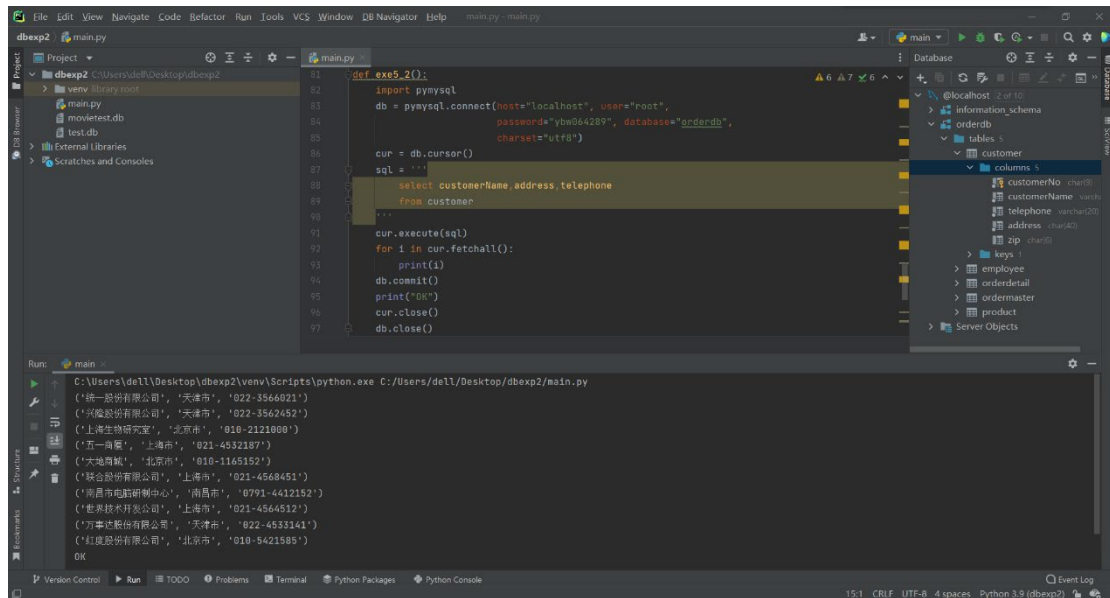


5-1、





5-2、



## 实验中遇到的困难及解决办法

由于对使用高级语言 IDE 访问数据库不了解，我浪费了许多时间。这里使用 Python 语言和 Pycharm IDE。Pycharm Ultimate 版自带的 database django 我个人认为比 MySQL 要好用很多。

首先我尝试了我先前学过的 sqlite3 包进行导入。但是 sqlite3 的导入数据库方式是将一个 .db 文件导入作为数据源，而本次实验是运行一个 sql 脚本来导入数据和模式。于是我想到用 sqlite3 的 execute () 方法来执行 sql 语句。具体操作是用文件读操作将 .sql 脚本中的语句读成一个字符串，将其作为参数传入 execute () 方法中执行。但这样会报错，提示 syntax error. sqlite3 的方法似乎无法正确识别 .sql 脚本中的语句。右键点击 sqlite3 呼出菜单出现 run sql script, 运行实验自带脚本后，虽然能成功创建表，但是表中没有数据。在经历了多次尝试后，使用 sqlite3 包的方式失败。

接着我想到，可能是 MySQL 和 sqlite3 是不同的数据库，所以使用的 sql 语言规范不一样，导致二者并不兼容。于是我在 database Django 中选择了 MySQL 作为数据源。但这又导致了新的问题。由于我并不清楚 MySQL 中，为什么运行了 .sql 脚本就会创建一个库，这导致我面对该界面完全无法操作。但是我注意到，右键点击 @localhost 服务器呼出菜单后，有个选项是 run sql script。我想这应该可以模拟 MySQL 上的导入数据和模式操作。运行完实验自带的脚本后，



再选择可视化 orderdb 模式，这样就成功完成导入。

关于之后的用高级语言处理 sql 语句，由于我先前自学爬虫时，有学到将爬取数据保存在 database 中有用到相关模块，因此对我来说并不是本次实验难点。

## 参考文献及致谢

感谢周心同同学，我与他讨论了有关存储过程和存储函数在 MySQL 中的格式问题。