



数据库系统实验报告

作业名称: SQL 数据定义和操作

姓 名: 汪珉凯

学 号: 3220100975

电子邮箱: 3220100975@zju.edu.cn

联系电话: 18157421318

指导老师: 孙建伶

2024 年 3 月 12 日

实验名称

一、实验目的

1. 掌握关系数据库语言SQL 的使用。
2. 使所有的 SQL 作业都能上机通过。

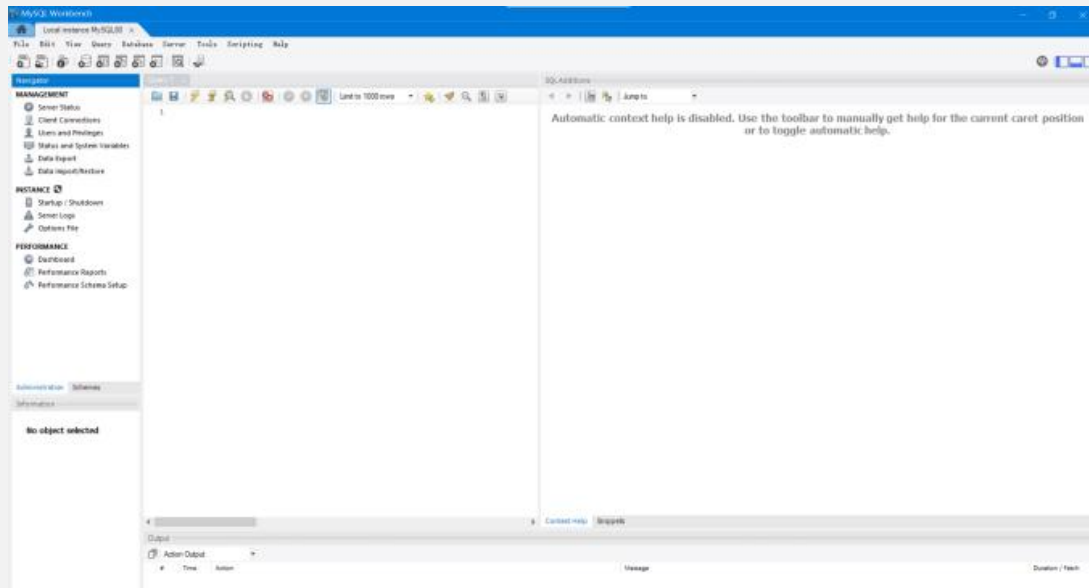
二、实验环境

MySQL Workbench

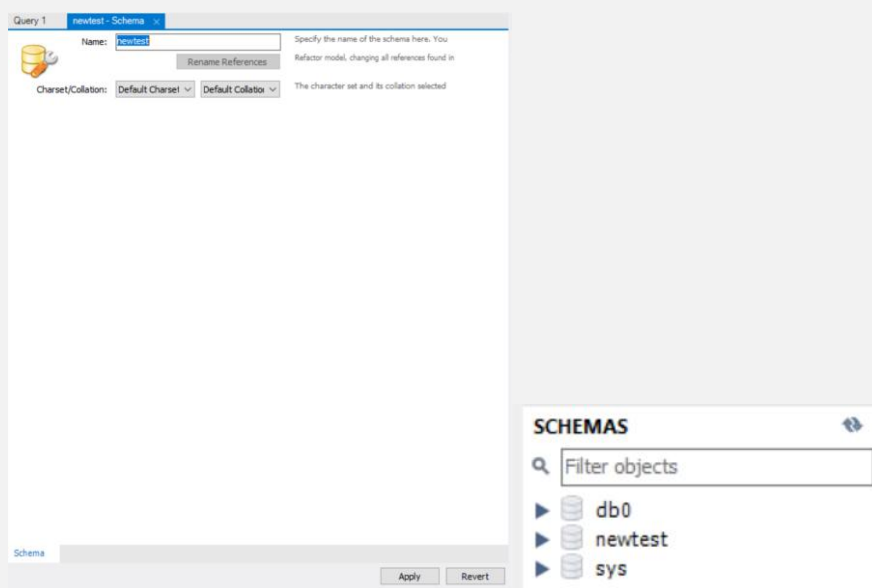
三、实验流程

1.建立数据库

1.1 打开 MySQL Workbench，本地登录。结果如图：



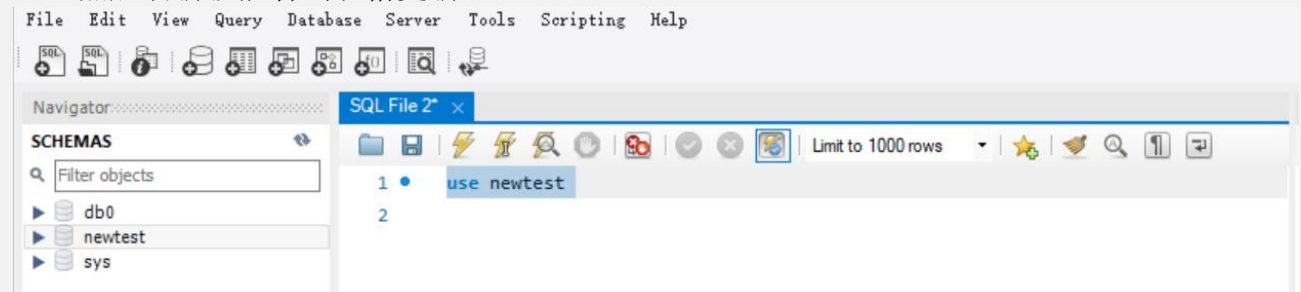
1.2 建立一个名为 newtest 的新数据库，点击 apply 按钮完成操作。



2.数据定义

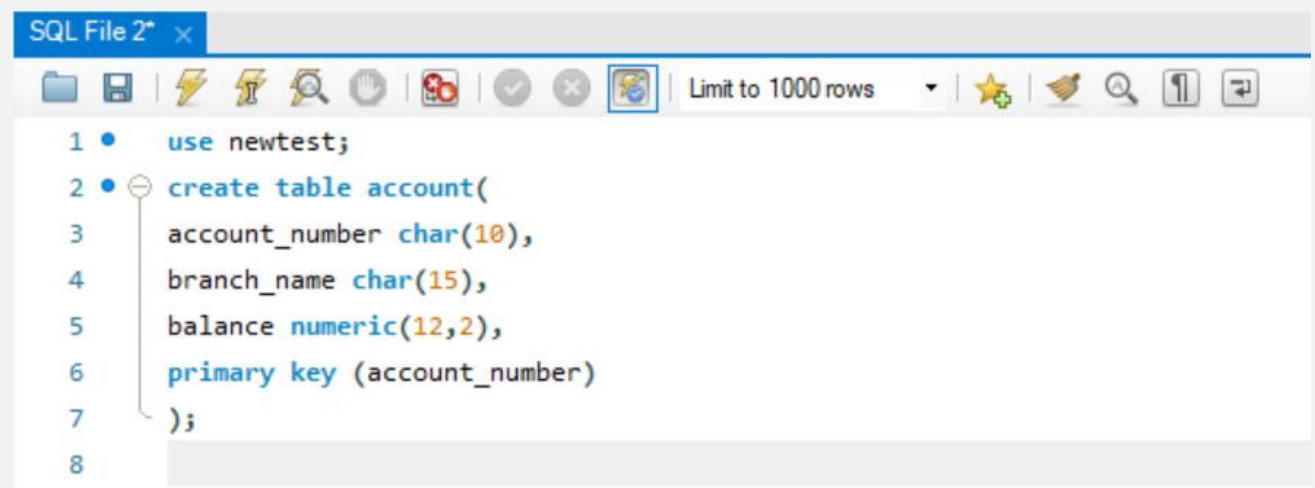
2.1 表

SQL 语言的运用：点击 SQL+按钮，写入 SQL 语言，光标选中后点击“闪电”的按钮，然后刷新就能看到表格更新。



2.1.1 表的建立

输入 use newtest ,代表在 newtest 这个数据库上进行操作。然后输入以下代码：

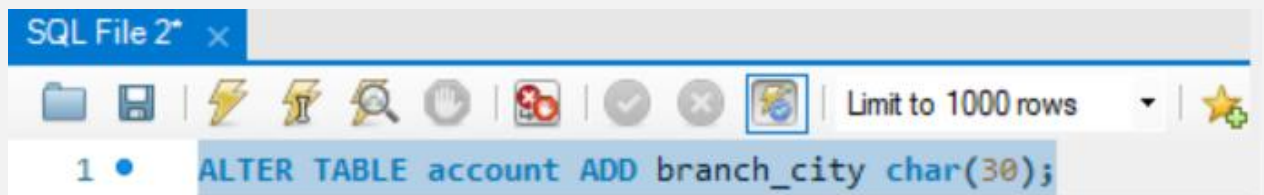


可以看到：在 newtest 数据库中多出了一张名为 account 的新表：

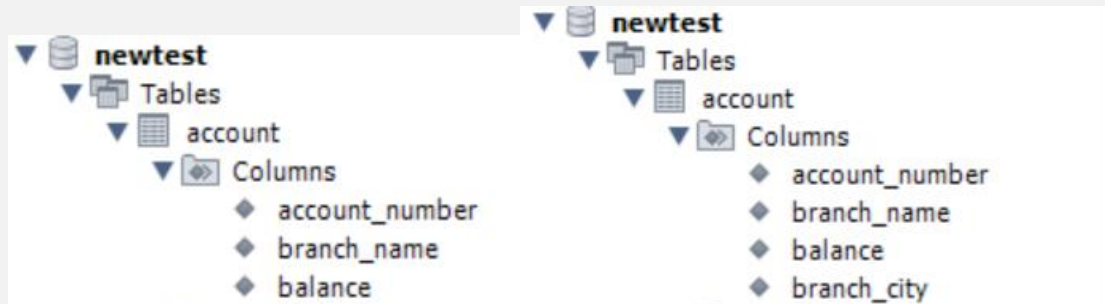


2.1.2 表的修改

(1) 执行以下代码：



(2) 可以看到 account 表的 column 中比原来多了一列 branch_city。

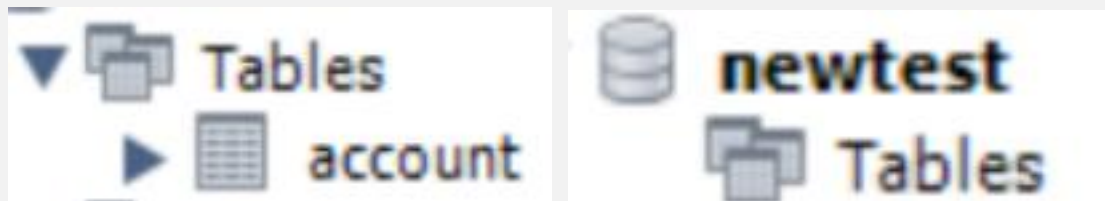


2.1.3 表的删除

(1) 执行以下代码：

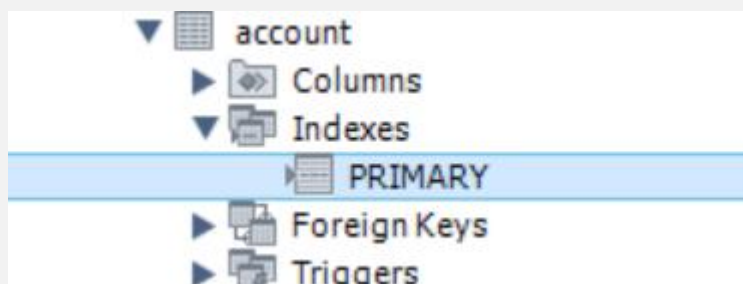


(2) 可以看到，table 中的 account 表被删除了。



2.2 索引

在此之前先重新建表，此时的唯一索引是系统自动为 primary key 建立的索引，如图所示：

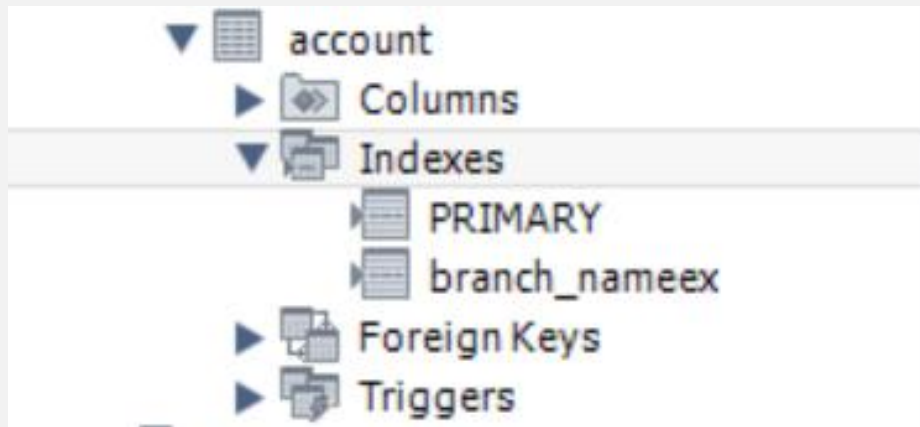


2.2.1 索引的建立

(1) 执行代码：

```
SQL File 2* x
1 CREATE INDEX branch_nameex ON account (branch_name);
```

(2) 可以看到，现在 index 列表里多出了一列 branch_nameex

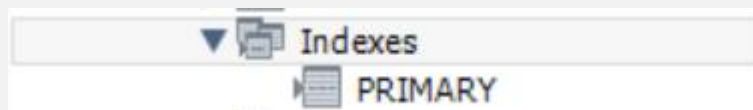


2.2.2 索引的删除

(1) 执行以下代码：

```
1 DROP INDEX branch_nameex on account;
```

(2) 可以看到，index 列中少了对应的一项：



2.3 视图

2.3.1 视图的建立

(1) 执行以下代码：

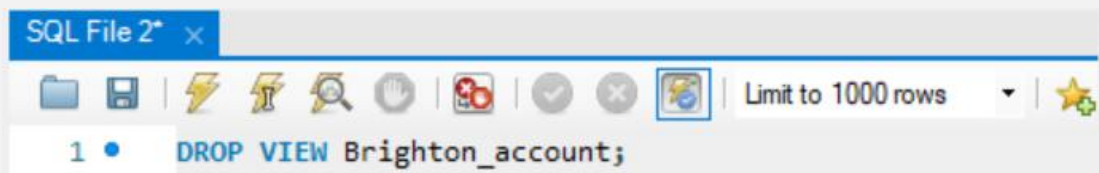
```
SQL File 2* x
1 CREATE VIEW Brighton_account
2 AS
3 SELECT account_number,balance
4 FROM account
5 WHERE branch_name='Brighton'
6 ;
```

(2) 可以看到，views 列表中出现了对应的视图。



2.3.2 视图的删除

(1) 执行以下代码：

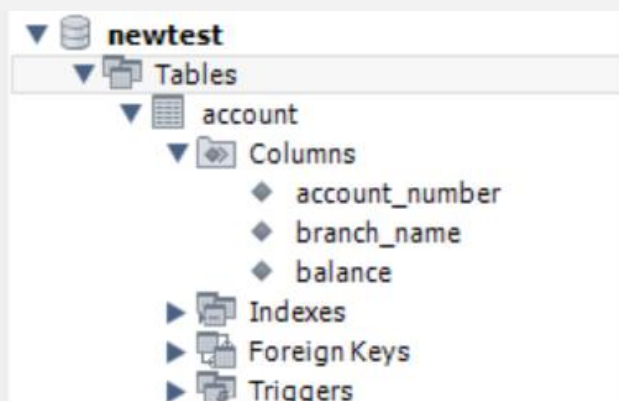
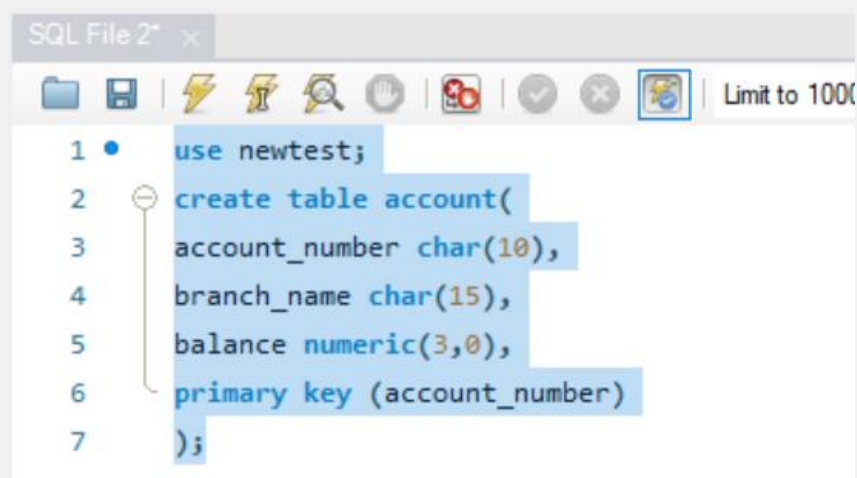


(2) 可以看到，view 视图重新变为空。



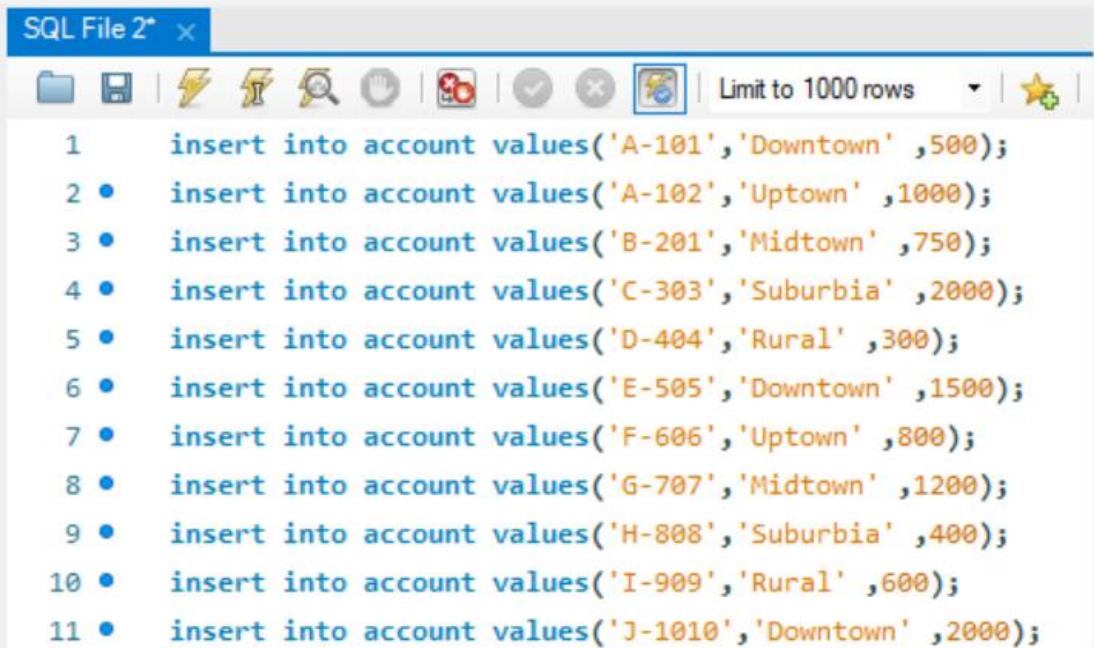
3 数据更新

先删除原表格，重新建一张新的表格 account



3.1 insert 插入表数据

(1) 执行以下代码完成插入：



```
SQL File 2* x
Limit to 1000 rows
1 insert into account values('A-101','Downtown',500);
2 • insert into account values('A-102','Uptown',1000);
3 • insert into account values('B-201','Midtown',750);
4 • insert into account values('C-303','Suburbia',2000);
5 • insert into account values('D-404','Rural',300);
6 • insert into account values('E-505','Downtown',1500);
7 • insert into account values('F-606','Uptown',800);
8 • insert into account values('G-707','Midtown',1200);
9 • insert into account values('H-808','Suburbia',400);
10 • insert into account values('I-909','Rural',600);
11 • insert into account values('J-1010','Downtown',2000);
```

(2) 执行以下代码：



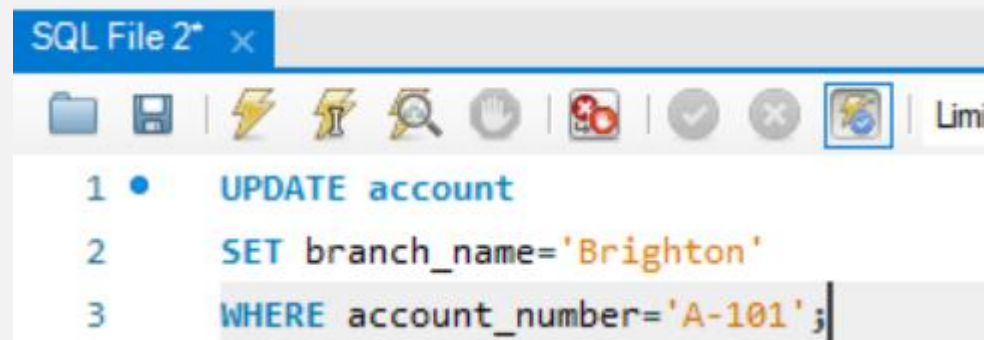
```
SQL File 2* x
1 • select * from account;
```

(3) 可以看到结果如下：

	account_number	branch_name	balance
▶	A-101	Downtown	500
	A-102	Uptown	100
	B-201	Midtown	750
	C-303	Suburbia	200
	D-404	Rural	300
	E-505	Downtown	150
	F-606	Uptown	800
	G-707	Midtown	120
	H-808	Suburbia	400
	I-909	Rural	600
	J-1010	Downtown	200

3.2update 更新表数据

(1) 执行以下代码：



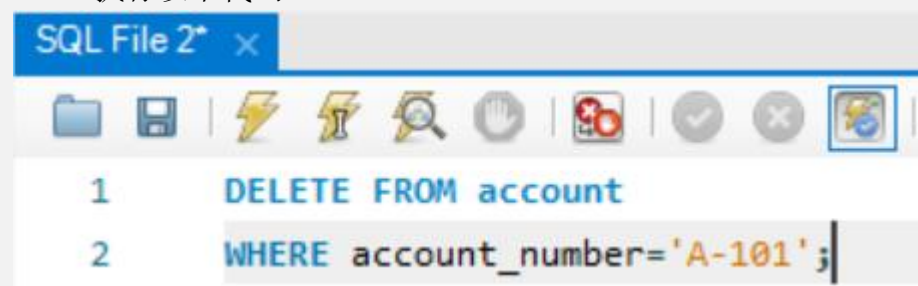
```
1 • UPDATE account
2   SET branch_name='Brighton'
3   WHERE account_number='A-101';
```

(2) 可以看到 A-101 的 branch_name 值已发生了改变。

	account_number	branch_name	balance
▶	A-101	Brighton	500

3.3delete 删除表数据

(1) 执行以下代码：



```
1 DELETE FROM account
2   WHERE account_number='A-101';
```

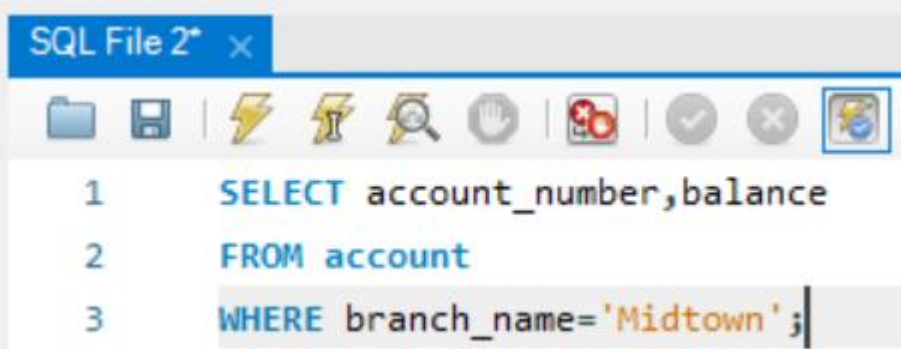
(2) 可以看到 account_name 为 A-101 的一组数据已被删除：

	account_number	branch_name	balance
▶	A-102	Uptown	100
	B-201	Midtown	750
	C-303	Suburbia	200
	D-404	Rural	300
	E-505	Downtown	150
	F-606	Uptown	800
	G-707	Midtown	120
	H-808	Suburbia	400
	I-909	Rural	600
	J-1010	Downtown	200
•	NULL	NULL	NULL

4 数据查询

4.1 单表查询

(1) 执行以下代码，统计所有 Midtown 的 balance:



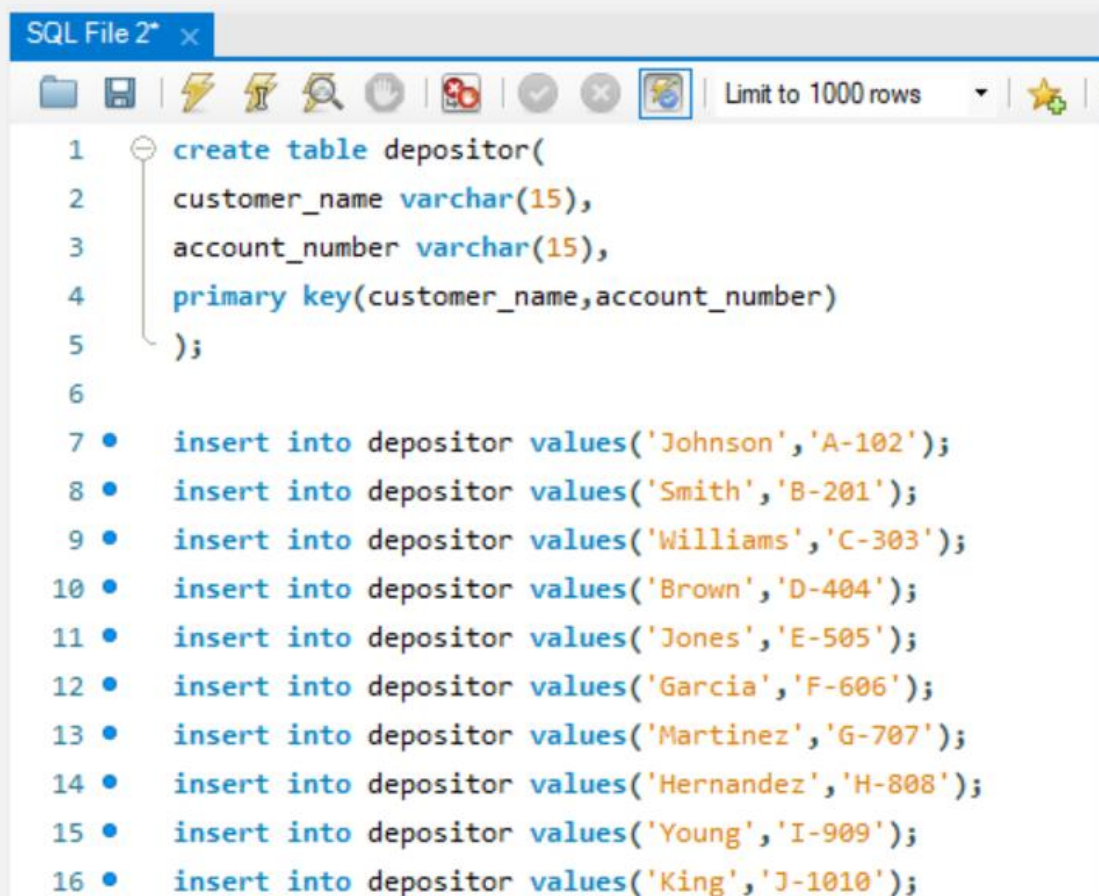
```
1  SELECT account_number,balance
2  FROM account
3  WHERE branch_name='Midtown';
```

(2) 可以看到结果如下:

	account_number	balance
▶	B-201	750
	G-707	120
●	NULL	NULL

4.2 多表查询

首先插入一张名为 depositor 的新表，代码和结果分别如图所示:



```
1  create table depositor(
2      customer_name varchar(15),
3      account_number varchar(15),
4      primary key(customer_name,account_number)
5  );
6
7  • insert into depositor values('Johnson','A-102');
8  • insert into depositor values('Smith','B-201');
9  • insert into depositor values('Williams','C-303');
10 • insert into depositor values('Brown','D-404');
11 • insert into depositor values('Jones','E-505');
12 • insert into depositor values('Garcia','F-606');
13 • insert into depositor values('Martinez','G-707');
14 • insert into depositor values('Hernandez','H-808');
15 • insert into depositor values('Young','I-909');
16 • insert into depositor values('King','J-1010');
```

Tables	
▶	account
▶	depositor

	customer_name	account_number
▶	Brown	D-404
	Garcia	F-606
	Hernandez	H-808
	Johnson	A-102
	Jones	E-505
	King	J-1010
	Martinez	G-707
	Smith	B-201
	Williams	C-303
	Young	I-909

(1) 执行以下多表查询代码：

```

SQL File 2* x
[Icons] | Limit to 1000 rows
1  SELECT customer_name,balance
2  FROM account,depositor
3  WHERE account.account_number=depositor.account_number;

```

(2) 可以看到结果如下：

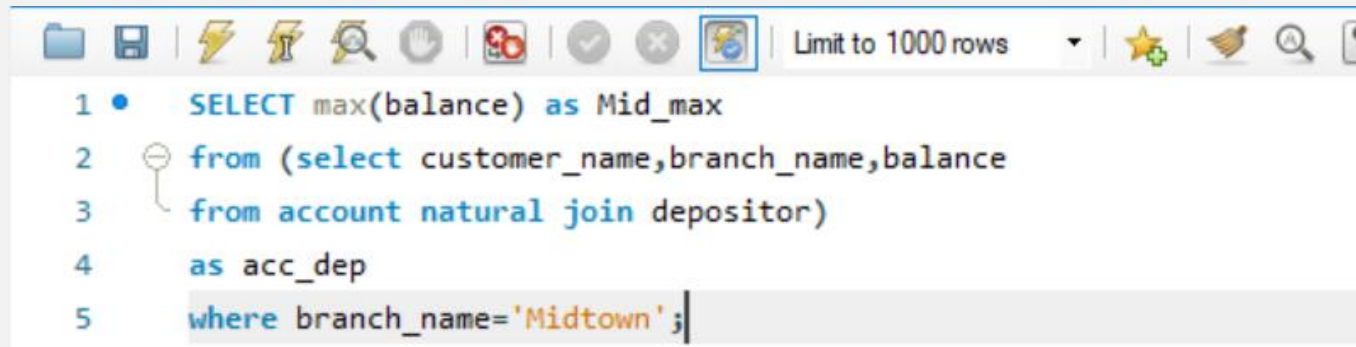
	customer_name	balance
▶	Brown	300
	Garcia	800
	Hernandez	400
	Johnson	100
	Jones	150
	King	200
	Martinez	120
	Smith	750
	Williams	200
	Young	600

(本身我在设计数据的时候，就使得每个客户都有自己的唯一账户。)

4.3 嵌套子查询

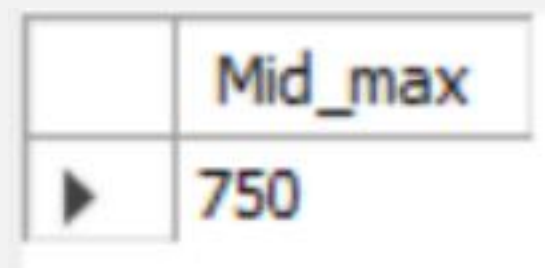
(1) 执行以下代码:

(代码的含义是通过 natural join 将两张表自然连接后, 选择 Midtown 的最大 balance)



```
1 • SELECT max(balance) as Mid_max
2   from (select customer_name,branch_name,balance
3         from account natural join depositor)
4   as acc_dep
5   where branch_name='Midtown';
```

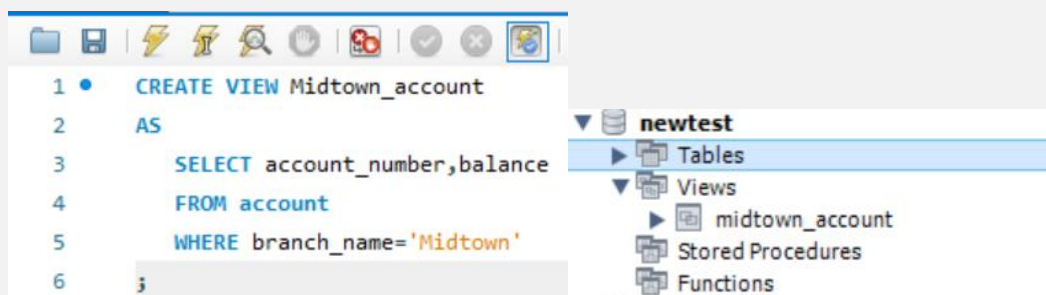
(2) 可以看到结果如下, 成功筛选出 Midtown 里的最大 salary。



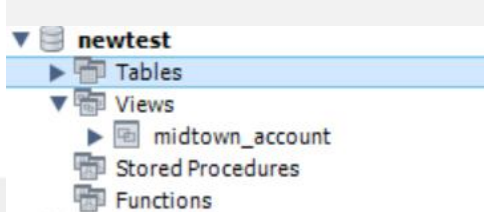
Mid_max
750

5.视图操作

首先重新建立名为的视图, 代码和结果如下所示:



```
1 • CREATE VIEW Midtown_account
2   AS
3     SELECT account_number,balance
4     FROM account
5     WHERE branch_name='Midtown'
6   ;
```

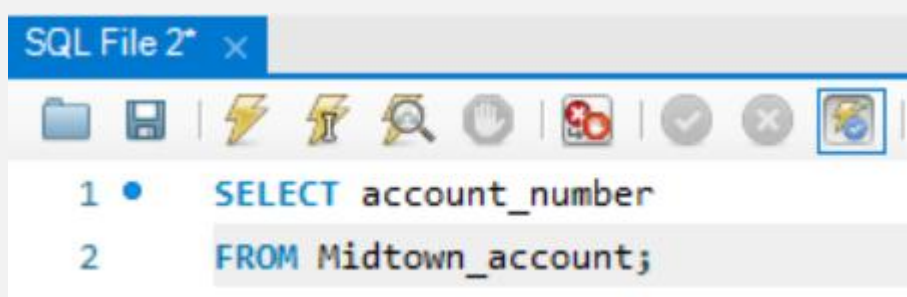


newtest

- Tables
- Views
 - midtown_account
- Stored Procedures
- Functions

5.1 视图数据查询:

(1) 执行如下代码:



```
1 • SELECT account_number
2   FROM Midtown_account;
```

(2) 结果如图所示：

	account_number
▶	B-201
	G-707

5.2 视图修改数据：

(1) 执行代码如下：

```
SQL File 2* x
[Icons]
1 • UPDATE Midtown_account
2   SET balance=balance*1.1
3   WHERE account_number='G-707';
```

(2) 结果如下：

```
select * from Midtown_account
```

	account_number	balance
▶	B-201	750
	G-707	132

```
select * from account
```

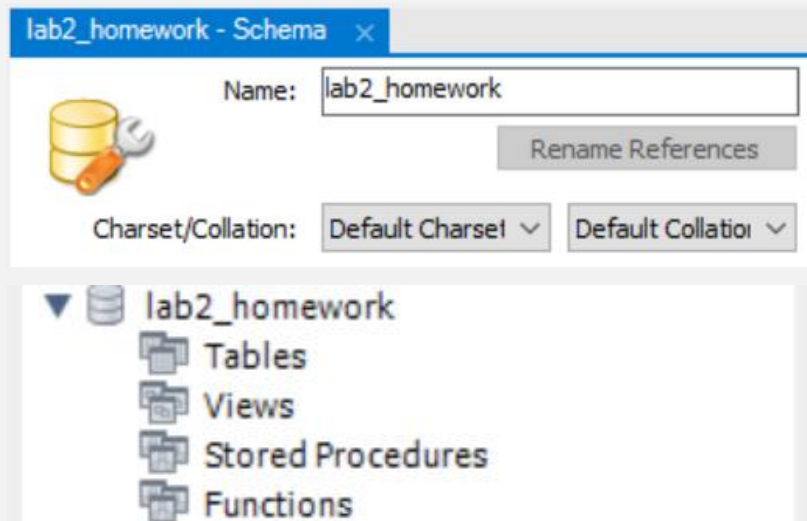
	account_number	branch_name	balance
▶	A-102	Uptown	100
	B-201	Midtown	750
	C-303	Suburbia	200
	D-404	Rural	300
	E-505	Downtown	150
	F-606	Uptown	800
	G-707	Midtown	132
	H-808	Suburbia	400
	I-909	Rural	600
	J-1010	Downtown	200
•	NULL	NULL	NULL

可以看到，编号为 G-707 的 balance 从 120 变成了 132.

6 作业中的 SQL 问题上机

Q3.8

1. 建立一个新的数据库，名叫 lab2_homework



2. 创建 6 张不同的相关表，并规定其属性。

```
1 • use lab2_homework;
2
3 ○ create table branch(
4     branch_name varchar(15),
5     branch_city varchar(15),
6     assets int,
7     primary key (branch_name)
8 );
9
10 ○ create table customer(
11     ID int ,
12     customer_name varchar (15),
13     customer_street varchar(15),
14     customer_city varchar(15),
15     primary key (ID)
16 );
17
18 ○ create table loan(
19     loan_number int,
20     branch_name varchar(15),
21     amount int,
22     primary key (loan_number)
23 );
24
25 ○ create table borrower(
26     ID int,
27     loan_number int,
28     primary key (ID,loan_number)
29 );
30
31 • ○ create table account(
32     account_number int,
33     branch_name varchar(15),
34     balance int,
35     primary key (account_number)
36 );
37
38 • ○ create table depositor(
39     ID int,
40     account_number int,
41     primary key (ID,account_number)
42 );
```

结果如图：



3.为各个表格插入数据。

```
1  -- 向 branch 表插入五组数据
2  • INSERT INTO branch (branch_name, branch_city, assets) VALUES
3    ('Branch A', 'City X', 100000),
4    ('Branch B', 'City Y', 150000),
5    ('Branch C', 'City Z', 200000),
6    ('Branch D', 'City W', 180000),
7    ('Branch E', 'City P', 120000);
8
9  -- 向 customer 表插入五组数据
10 INSERT INTO customer (ID, customer_name, customer_street, customer_city) VALUES
11 (1, 'John Doe', '123 Main St', 'City X'),
12 (2, 'Jane Smith', '456 Elm St', 'City Y'),
13 (3, 'David Brown', '789 Oak St', 'City Z'),
14 (4, 'Emily Johnson', '321 Pine St', 'City W'),
15 (5, 'Michael Lee', '654 Maple St', 'City P');
16
17 -- 向 loan 表插入五组数据
18 INSERT INTO loan (loan_number, branch_name, amount) VALUES
19 (101, 'Branch A', 50000),
20 (102, 'Branch B', 75000),
21 (103, 'Branch C', 100000),
22 (104, 'Branch D', 80000),
23 (105, 'Branch E', 60000);
```

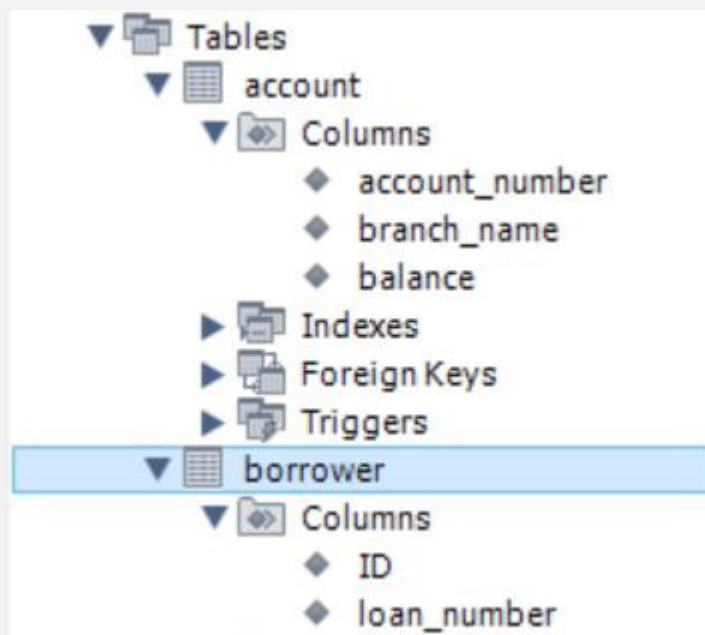


```

25      -- 向 borrower 表插入五组数据
26  •   INSERT INTO borrower (ID, loan_number) VALUES
27      (1, 101),
28      (2, 102),
29      (3, 103),
30      (4, 104),
31      (5, 105);
32
33      -- 向 account 表插入五组数据
34  •   INSERT INTO account (account_number, branch_name, balance) VALUES
35      (1001, 'Branch A', 10000),
36      (1002, 'Branch B', 15000),
37      (1003, 'Branch C', 20000),
38      (1004, 'Branch D', 18000),
39      (1005, 'Branch E', 12000);
40
41      -- 向 depositor 表插入五组数据
42  •   INSERT INTO depositor (ID, account_number) VALUES
43      (1, 1001),
44      (2, 1002),
45      (3, 1003),
46      (4, 1004),
47      (5, 1005);

```

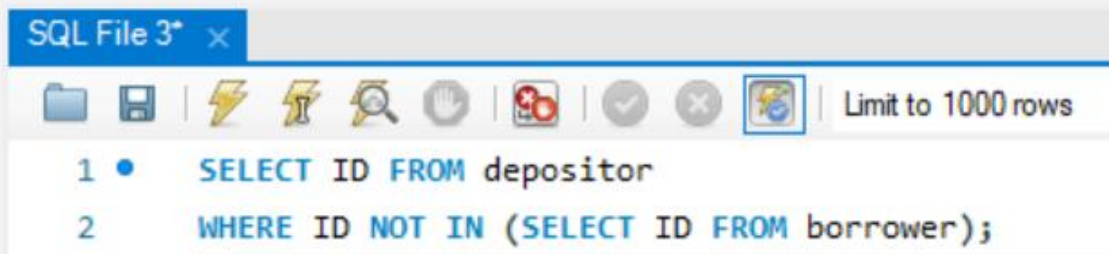
结果如图所示，说明插入成功：



下面执行作业中的 3 个问题。

A.Find the ID of each customer of the bank who has an account but not a loan.

执行以下代码：



```
1 • SELECT ID FROM depositor
2   WHERE ID NOT IN (SELECT ID FROM borrower);
```

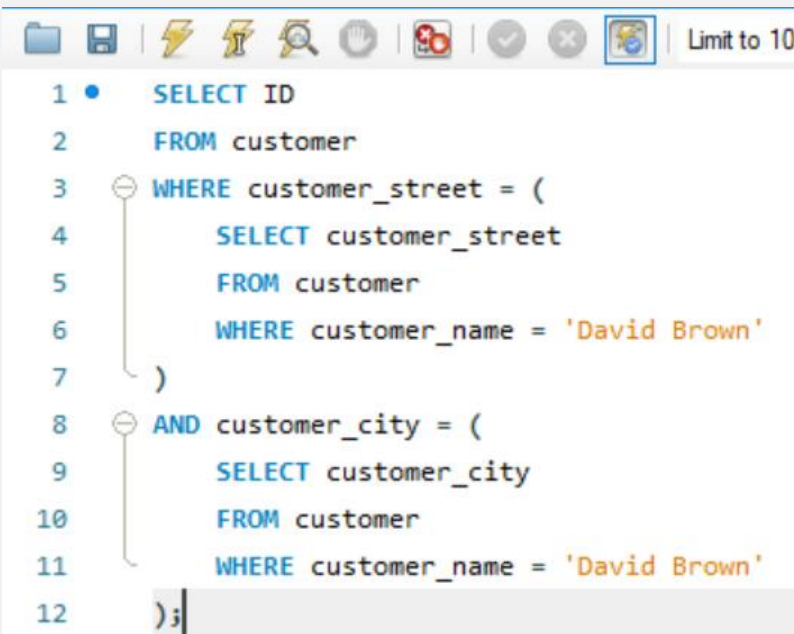
得到结果如图所示：

	ID
▶	2

成功选出了在银行有账户但无贷款用户的 ID。

B.Find the ID of each customer who lives on the same street and in the same city as customer 'David Brown'.

执行以下代码：



```
1 • SELECT ID
2   FROM customer
3   WHERE customer_street = (
4       SELECT customer_street
5       FROM customer
6       WHERE customer_name = 'David Brown'
7   )
8   AND customer_city = (
9       SELECT customer_city
10      FROM customer
11      WHERE customer_name = 'David Brown'
12  );
```

得到结果如图所示：

	ID
▶	3
	4
•	NULL

成功选出了和 David Brown 住在同一个城市同一条街上的用户的 ID。

C.Find the name of each branch that has at least one customer who has an account in the bank and who lives in “City P”.

执行以下代码：

```
1 • SELECT DISTINCT branch_name
2 FROM customer
3 NATURAL JOIN account
4 NATURAL JOIN depositor
5 WHERE customer_city = 'City P';
```

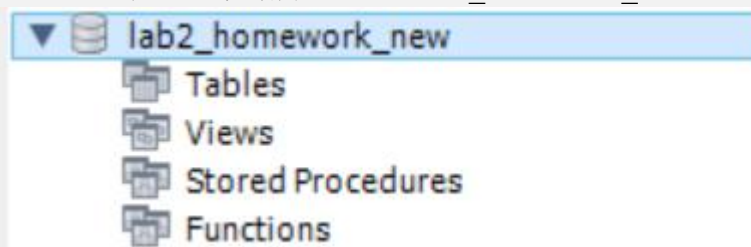
得到结果如图所示：

	branch_name
▶	Branch E

成功找到了有住在 City P 用户的银行名字。

Q3.9

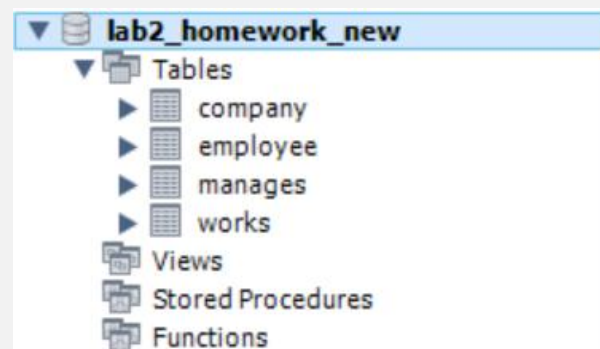
1.建立一个新的数据库，名叫 lab2_homework_new.



2.创建 6 张不同的相关表，并规定其属性。

```
1 use lab2_homework_new;
2
3 • create table employee(
4     ID int,
5     person_name varchar(30),
6     street varchar(30),
7     city varchar(30),
8     primary key(ID)
9 );
10
11 • create table works(
12     ID int,
13     company_name varchar(30),
14     salary int,
15     primary key(ID)
16 );
17
18 • create table company(
19     company_name varchar(30),
20     city varchar(30),
21     primary key(company_name)
22 );
23
24 • create table manages(
25     ID int,
26     manager_id int,
27     primary key(ID)
28 );
29
```

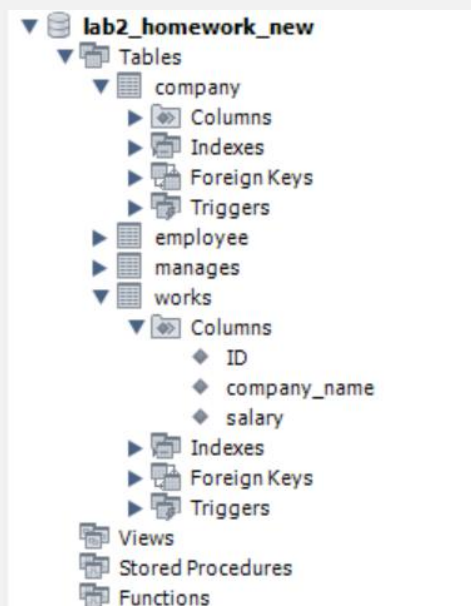
结果如图：



3.为各个表格插入数据。

```
1  -- 插入 employee 表数据
2  • INSERT INTO employee (ID, person_name, street, city) VALUES
3    (1, 'John', '123 Main St', 'New York'),
4    (2, 'Alice', '456 Elm St', 'Los Angeles'),
5    (3, 'Bob', '789 Oak St', 'Chicago'),
6    (4, 'Emily', '101 Pine St', 'Houston'),
7    (5, 'Michael', '202 Maple St', 'Miami');
8
9  -- 插入 works 表数据
10 • INSERT INTO works (ID, company_name, salary) VALUES
11    (1, 'First Bank Corporation', 60000),
12    (2, 'First Bank Corporation', 5500),
13    (3, 'Small Bank Corporation', 6500),
14    (4, 'Small Bank Corporation', 58000),
15    (5, 'GlobalTech', 62000);
16
17  -- 插入 company 表数据
18  INSERT INTO company (company_name, city) VALUES
19    ('First Bank Corporation', 'New York'),
20    ('TechCorp', 'Los Angeles'),
21    ('Small Bank Corporation', 'Chicago'),
22    ('DataSolutions', 'Houston'),
23    ('GlobalTech', 'Miami');
24
25  -- 插入 manages 表数据
26 • INSERT INTO manages (ID, manager_id) VALUES
27    (1, 2),
28    (2, 3),
29    (3, 1),
30    (4, 5),
31    (5, 4);
```

结果如图所示，说明插入成功：



下面执行作业中的 7 个问题。

A. Find the ID, name, and city of residence of each employee who works for “First Bank Corporation”

执行以下代码：

```
1 • SELECT e.ID, e.person_name, e.city
2 FROM employee AS e
3 NATURAL JOIN works AS w
4 WHERE w.company_name = 'First Bank Corporation';
```

得到结果如图所示，成功选出了为“First Bank Corporation”工作的雇员的三项信息：

	ID	person_name	city
▶	1	John	New York
	2	Alice	Los Angeles

B. Find the ID, name, and city of residence of each employee who works for “First Bank Corporation” and earns more than \$10000.

执行以下代码：

```
1 • SELECT e.ID, e.person_name, e.city
2 FROM employee AS e
3 NATURAL JOIN works AS w
4 WHERE w.company_name = 'First Bank Corporation' AND w.salary > 10000;
```


得到结果如图所示，成功筛选出了“First Bank Corporation”薪水超过 10000 美元员工的三项信息：

	ID	person_name	city
▶	1	John	New York

C.Find the ID of each employee who does not work for “First Bank Corporation”

执行以下代码：

```
1 • SELECT ID
2 FROM works
3 WHERE company_name <> 'First Bank Corporation';
```

得到结果如图所示，成功筛选了“First Bank Corporation”雇员的 ID：

	ID
▶	3
	4
	5
•	NULL

D.Find the ID of each employee who earns more than every employee of “Small Bank Corporation”.

执行以下代码：

```
1 • SELECT ID
2 FROM works
3 WHERE salary > (SELECT MAX(salary) FROM works WHERE company_name = 'Small Bank Corporation');
```

得到结果如图所示，成功筛选得到了薪水比“Small Bank Corporation”雇员最高薪水还高的员工的 ID：

	ID
▶	1
	5
•	NULL

E.Assume that companies may be located in several cities. Find the name of each company that is located in every city in which “Small Bank Corporation” is located.

执行以下代码：


```

1 • SELECT W.company_name
2 FROM company AS W
3 WHERE NOT EXISTS (
4     SELECT city FROM company WHERE company_name = 'Small Bank Corporation'
5     and city NOT IN
6     (SELECT city FROM company AS R WHERE R.company_name = W.company_name)
7 );

```

得到结果如图所示，可见除了“Small Bank Corporation”本身以外，没有符合条件的公司：

	company_name
▶	Small Bank Corporation
●	NULL

F.Find the name of the company that has the most employees (or companies,in the case where there is a tie for the most).

执行以下代码：

```

1 • SELECT company_name
2 FROM works
3 GROUP BY company_name
4 HAVING COUNT(DISTINCT ID) >= ALL (
5     SELECT COUNT(DISTINCT ID) FROM works GROUP BY company_name
6 );

```

得到结果如图所示，成功找到了雇员人数最多的公司：

	company_name
▶	First Bank Corporation
	Small Bank Corporation

G.Find the name of each company whose employees earn a higher salary.on average, than the average salary at “First Bank Corporation”

执行以下代码：

```

1 • SELECT company_name
2 FROM works
3 GROUP BY company_name
4 HAVING AVG(salary) > (SELECT AVG(salary) FROM works WHERE company_name = 'First Bank Corporation');

```

得到结果如图所示，成功找到了平均薪资高于“First Bank Corporation”的公司：

	company_name
▶	GlobalTech

四、遇到的问题及解决方法

- 1.一开始没有找到新建的 Schema,后来在最左边 Navigator 列下放找到了 Schema 按钮。
- 2.做嵌套子查询的时候没有把临时的新表格命名,出现了“Every derived table must have its own alias”的错误,用 as 重新命名就得到了解决。

五、总结

本次实验的主要目的是熟悉数据库语言的使用。由于是刚开始接触,一开始操作起来还是比较陌生,但慢慢地开始熟练。希望以后能多多练习,争取早日脱离搜索引擎的帮助能够自主完成相应操作。