浙江大学

数据库系统实验报告

作业名称：	SQL 数据定义和操作
姓　　名：	龙永奇
学　　号：	3220105907
电子邮箱：	3220105907@zju.edu.cn
联系电话：	15393113093
指导老师：	孙建伶

2024 年 03 月 17 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

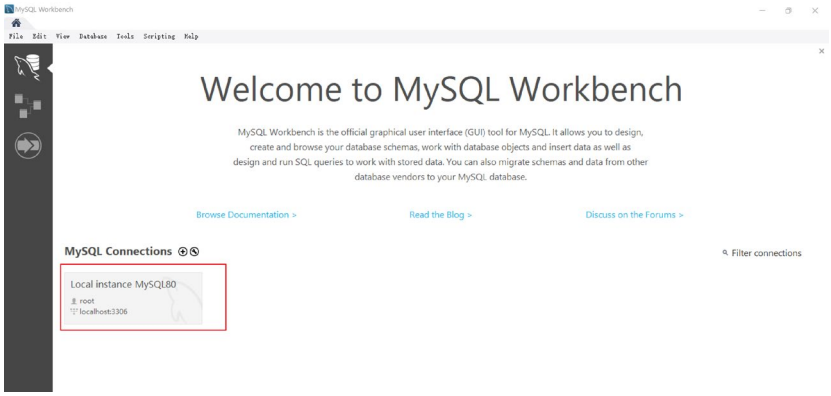
二、实验环境

1. 操作系统：Windows 11
2. 数据库管理系统：MySQL 8.0.36
3. 实验工具：Powershell

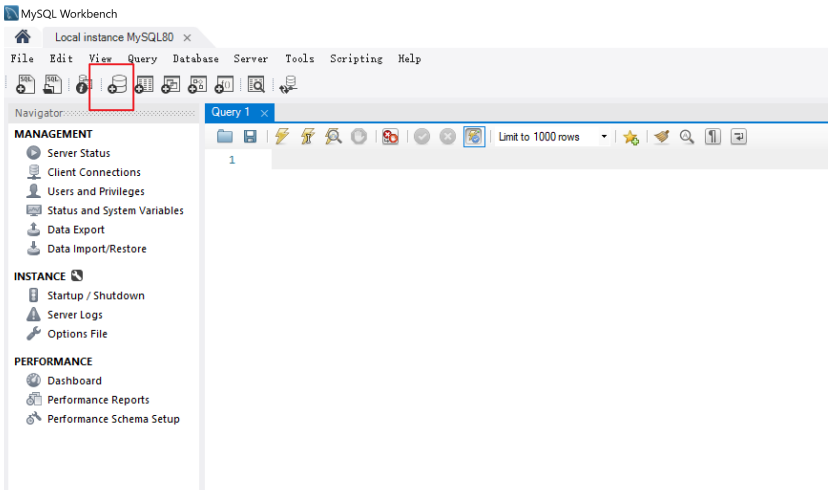
三、实验流程

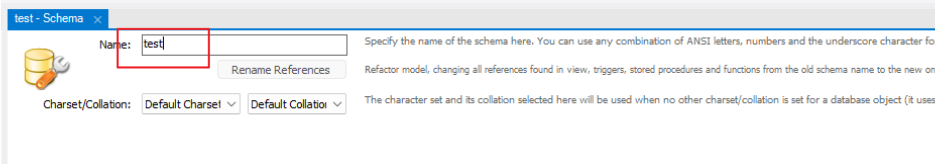
1. **建立数据库**

   打开 MySQL Workbench，使用本地登录



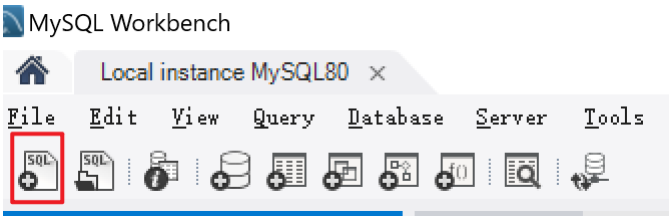
   点击新建数据库，输入数据库名称

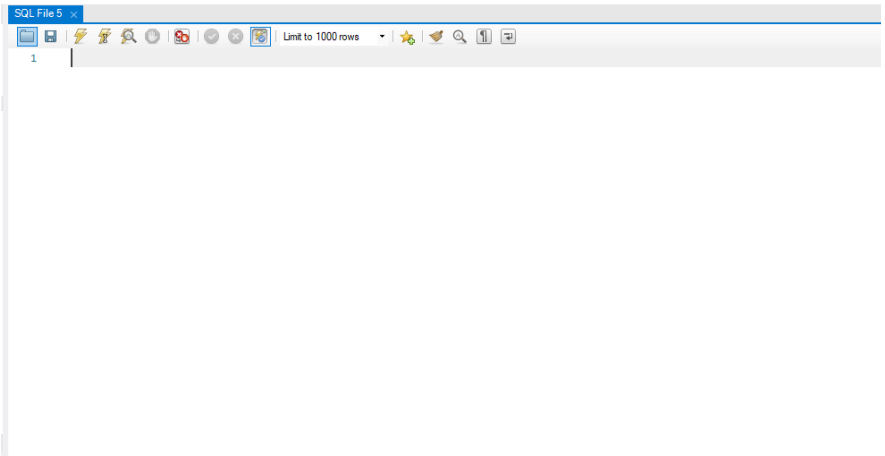




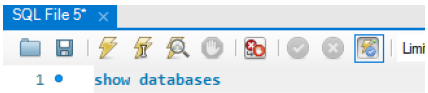
2. **数据定义：表的建立/删除/修改；索引的建立/删除；视图的建立/删除**

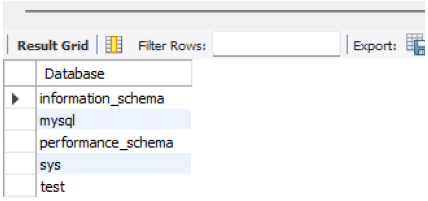
数据库新建成功后打开一个文本框：



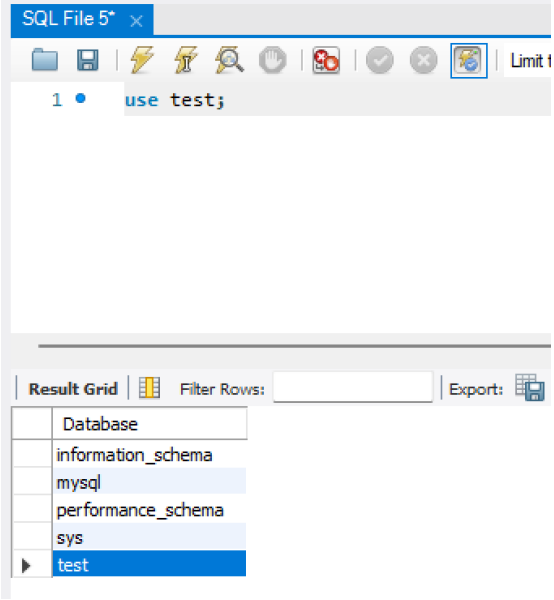


查看当前数据库：

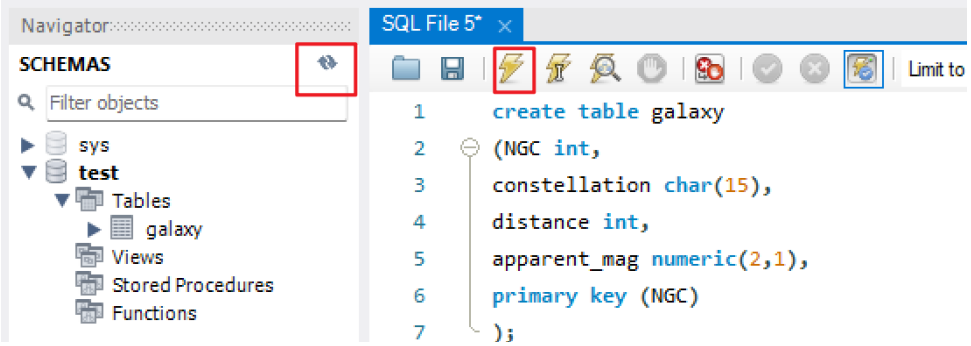




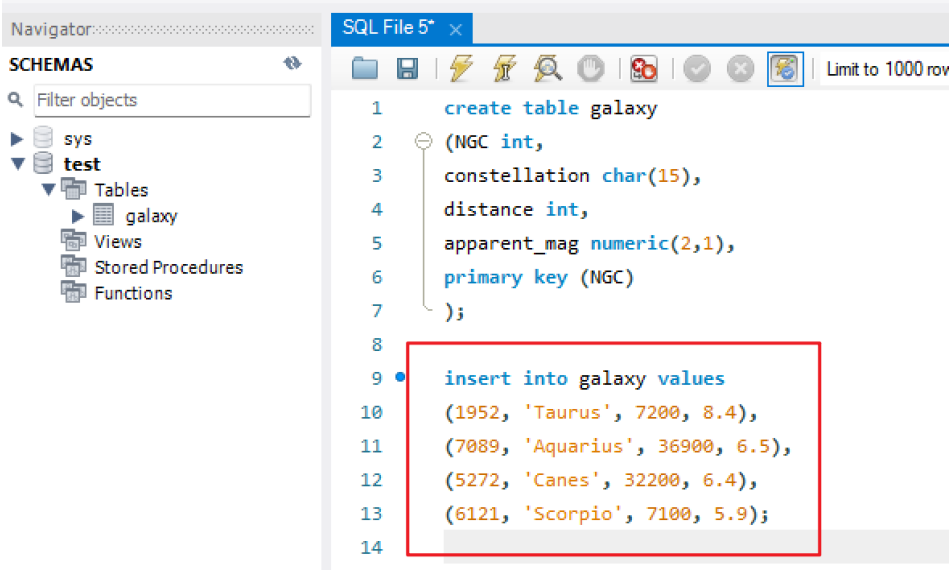
切换至实验二所用数据库：test

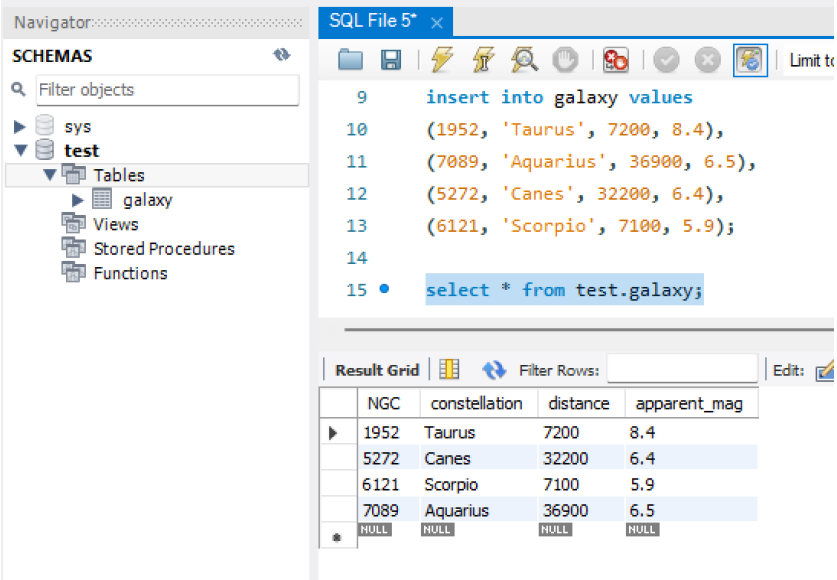


建立新表，输入以下代码，点击刷新可以看见多出一张新表：

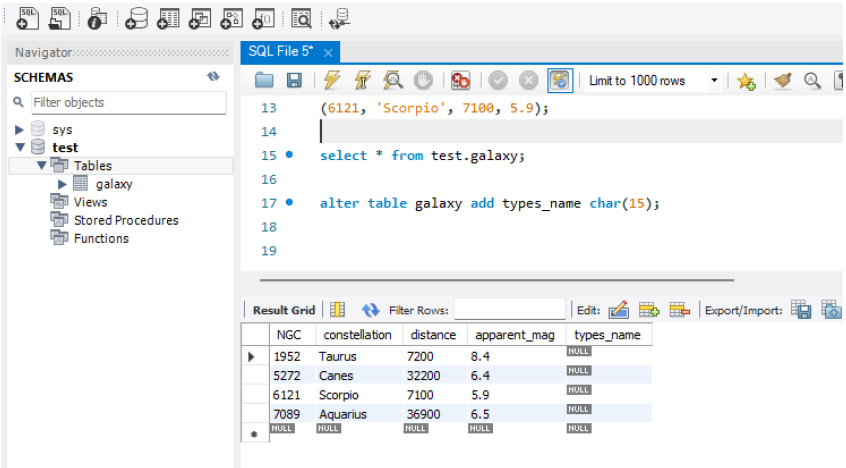


输入数据：

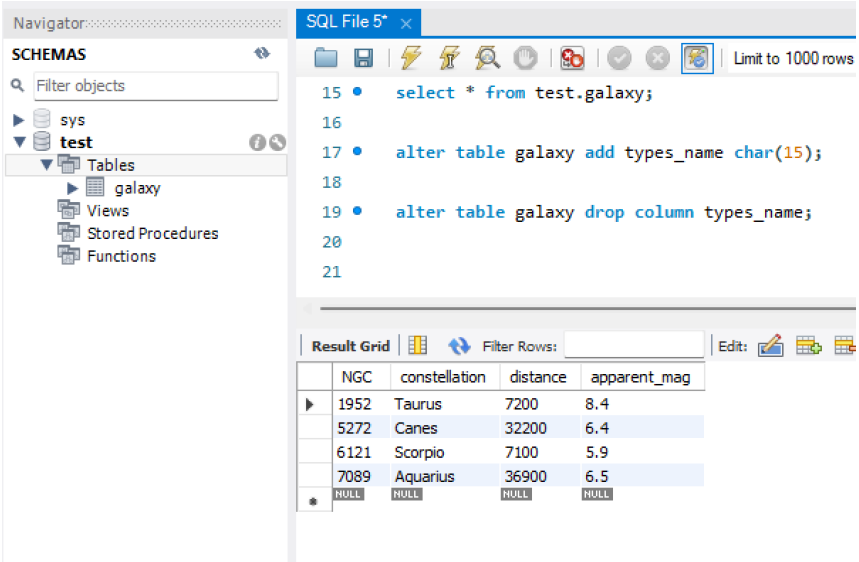




增加一列：



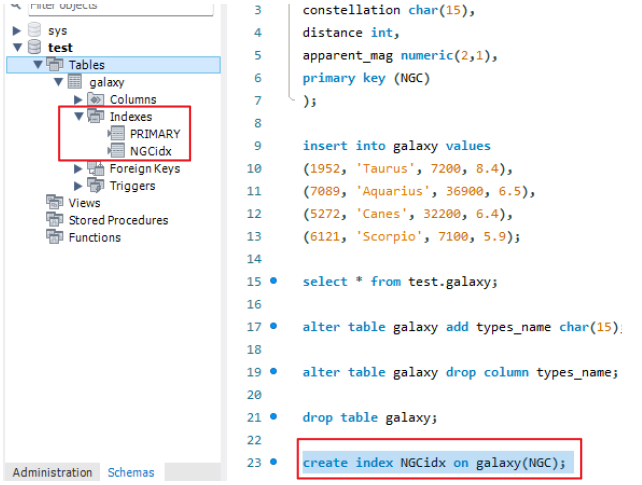
删除一列：



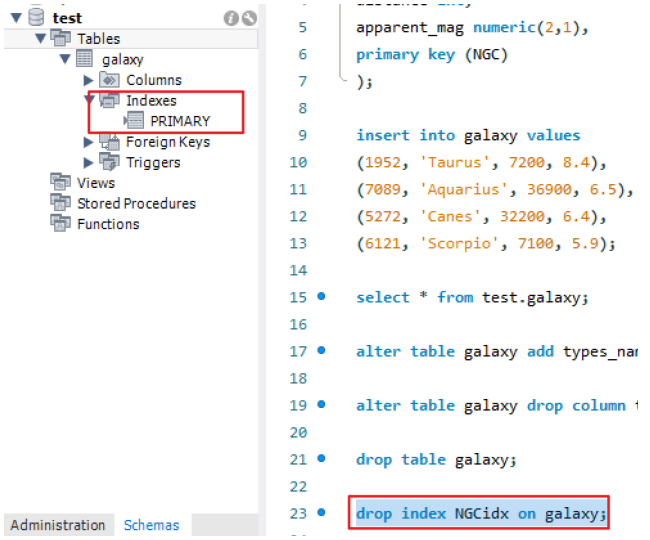
删除整张表：

```
drop table galaxy;
```

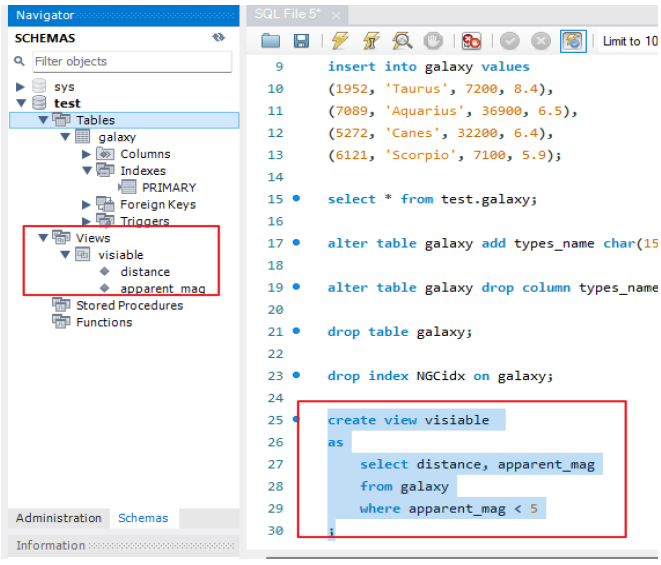
重新建好表，建立索引，注意到系统已经帮我们建立好了一个 PRIMARY 索引



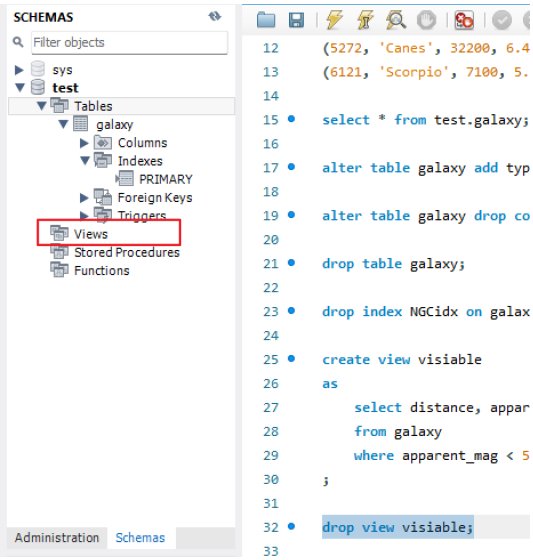
删除索引：



建立视图：

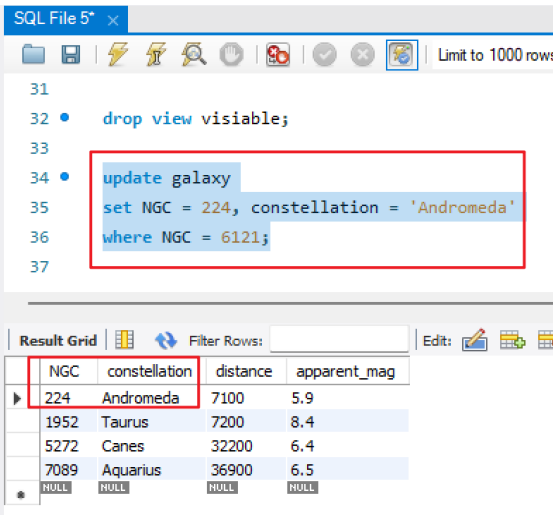


删除视图：



3. **数据更新**

插入数据在第二步已经完成，我们接下来进行数据的修改：

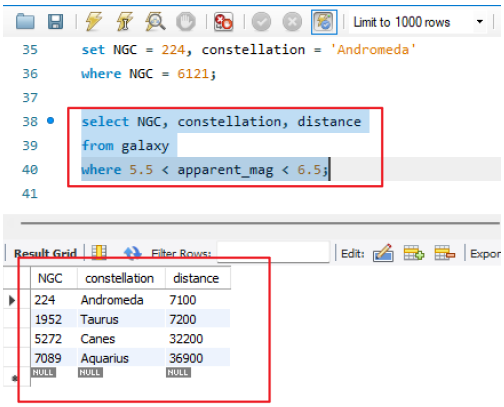


我们将 NGC = 6121 的星系改为了 NGC = 224 的仙女座星系，并更新了名称

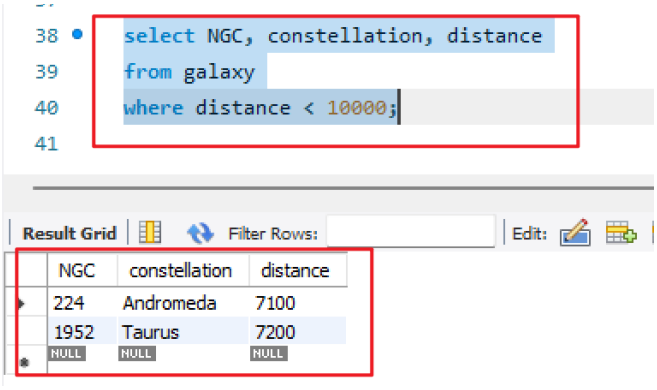
4. **数据查询**

单表查询：

查询视星等在 5.5 至 6.5 之间的星系

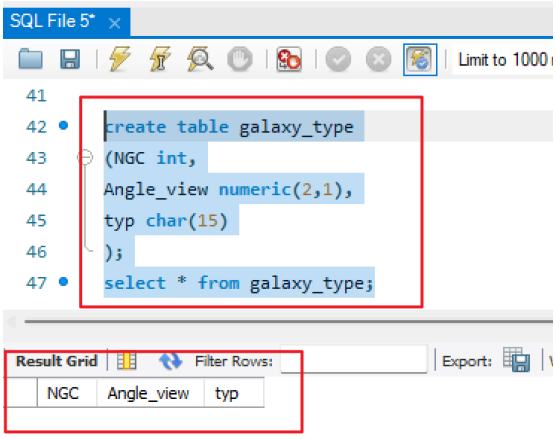


查询距离我们 10000 光年内的星系：



多表查询：

准备另一张表，表示星系的视角大小、类别，使用 NGC 编号作为主键



输入信息：

```sql
insert into galaxy_type value
(224, 180, 'spiral'),
(221, 8, 'elliptical'),
(598, 62, 'spiral'),
(1952, 6, 'Crab Nebula'),
(7089, 12, 'globular'),
(5272, 19, 'globular'),
(6121, 20, 'globular');
select * from galaxy_type;
```

NGC	Angle_view	typ
224	180	spiral
221	8	elliptical
598	62	spiral
1952	6	Crab Nebula
7089	12	globular
5272	19	globular
6121	20	globular

多表查询以及嵌套子查询

视星等大于 6，视角大小大于 10，并且不是球状星团的元组，展示其 NGC 编号以及星座：

```sql
select galaxy.NGC, constellation
from galaxy, galaxy_type
where apparent_mag > 6
      and Angle_view > 10
      and (select galaxy_type.NGC
           from galaxy_type
           where galaxy.NGC = galaxy_type.NGC
           and typ <> 'globular')
group by NGC;
```

NGC	constellation
1952	Taurus

距离我们最远的球状星团：

```
select galaxy.NGC, distance
from galaxy, galaxy_type
where galaxy.NGC in (
    select galaxy_type.NGC
    from galaxy_type
    where typ = 'globular'
)
order by distance desc
limit 1;
```

Result Grid | Filter Rows: | Export: | Wr

NGC	distance
7089	36900

5. **视图操作**

视图数据查询：

重新建立视图可见星系：

SQL File 5* | Limit to 1000 rows

```
drop index NGCidx on galaxy;

create view visiable
as
    select NGC, distance, apparent_mag
    from galaxy
    where apparent_mag < 7
;
select * from visiable;

drop view visiable;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

NGC	distance	apparent_mag
224	7100	5.9
5272	32200	6.4
7089	36900	6.5

通过视图修改数据：

得到 NGC = 224 的绝对星等

```
select * from visiable;
update visiable
set apparent_mag = apparent_mag - 5
where NGC = 224;
select * from visiable;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Co

NGC	distance	apparent_mag
224	7100	0.9
5272	32200	6.4
7089	36900	6.5

可以看到 apparent_mag 的变化

6. **作业验证**

3.8

建立表格，填入测试数据

```sql
create table branch (
    branch_name varchar(50),
    branch_city varchar(50),
    assets decimal(10, 2),
    primary key (branch_name)
);

create table customer (
    id int,
    customer_name varchar(50),
    customer_street varchar(100),
    customer_city varchar(50),
    primary key (id)
);

create table loan (
    loan_number int,
    branch_name varchar(50),
    amount decimal(10, 2),
    primary key (loan_number)
);

create table borrower (
    id int,
    loan_number int,
    primary key (id, loan_number)
);

create table account (
    account_number int,
    branch_name varchar(50),
    balance decimal(10, 2),
    primary key (account_number)
);

create table depositor (
    id int,
    account_number int,
    primary key (id, account_number)
);
```

```sql
insert into branch values
('downtown', 'los angeles', 100000.00),
('uptown', 'los angeles', 200000.00),
('midtown', 'harrison', 150000.00);

insert into customer values
(12345, 'john smith', '123 main st', 'harrison'),
(67890, 'jane doe', '456 oak st', 'los angeles'),
(13579, 'alice johnson', '789 elm st', 'harrison'),
(40216, 'proton long', '123 main st', 'harrison');

insert into loan values
(1, 'downtown', 5000.00),
(2, 'uptown', 7000.00),
(3, 'midtown', 3000.00);

insert into borrower values
(12345, 1),
(67890, 2);

insert into account values
(101, 'downtown', 1000.00),
(102, 'uptown', 2000.00),
(103, 'midtown', 3000.00);

insert into depositor values
(12345, 101),
(67890, 102),
(13579, 103);
```

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

```sql
select id
from depositor
where id not in(select id
                from borrower);
```

```
select id
from depositor
where id not in(select id
                from borrower);
```

Result Grid | Filter Rows: | Exp

id
13579

b. Find the ID of each customer who lives on the same street and in the same city as customer '12345'.

```
select A.ID
from customer as A, customer as B
where A.customer_street = B.customer_stree
      and A.customer_city = B.customer_city
      and B.customer_ID = '12345'
      and A.customer_ID <> '12345'
```

```
select A.ID
from customer as A, customer as B
where A.customer_street = B.customer_street
      and A.customer_city = B.customer_city
      and B.ID = '12345'
      and A.ID <> '12345';
```

Result Grid | Filter Rows: | Export: | Wrap

ID
40216

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

```
select distinct branch_name
from account, depositor, customer
where account.account_number = depositor.account_number
      and customer.id = depositor.id
      and customer_city = 'Harrison'
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

branch_name
downtown
midtown

3.9

建立数据库，填入测试数据

```
create table employee (
    id int,
```

```sql
    person_name varchar(100),
    street varchar(100),
    city varchar(100)
);

create table works (
    id int,
    company_name varchar(100),
    salary decimal(10, 2)
);

create table company (
    company_id int primary key auto_increment,
    company_name varchar(100),
    city varchar(100)
);

create table manages (
    id int,
    manager_id int
);

insert into employee (id, person_name, street, city) values
(123, 'John Doe', '123 Main St', 'New York'),
(456, 'Jane Smith', '456 Elm St', 'Los Angeles'),
(789, 'Bob Johnson', '789 Oak St', 'Chicago');

insert into works (id, company_name, salary) values
(789, 'Small Bank Corporation', 50000),
(123, 'deutsche Bank Corporation', 80000),
(234, 'deutsche Bank Corporation', 90000),
(356, 'deutsche Bank Corporation', 100000),
(456, 'First Bank Corporation', 40000),
(544, 'First Bank Corporation', 60000);

insert into company (company_name, city) values
('Small Bank Corporation', 'New York'),
('Small Bank Corporation', 'Los Angeles'),
('Small Bank Corporation', 'Chicago'),
('deutsche Bank Corporation', 'New York'),
('deutsche Bank Corporation', 'Los Angeles'),
('deutsche Bank Corporation', 'Chicago'),
('Global Corp', 'Chicago');
```

```
insert into manages (id, manager_id) values
(1, 2),
(2, 3),
(3, 1);
```

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

```
select employee.ID, person_name, city
from employee, works
where employee.ID = works.ID
      and works.company_name = 'First Bank Corporation'
```

```
42 select employee.ID, person_name, city
43 from employee, works
44 where employee.ID = works.ID
45       and works.company_name = 'First Bank Corporation'
46
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ID	person_name	city
1	john doe	new york
2	jane smith	new york

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

```
select employee.ID, person_name, city
from employee, works
where employee.ID = works.ID
      and works.company_name = 'First Bank Corporation'
      and salary > 10000
```

```
47 select employee.ID, person_name, city
48 from employee, works
49 where employee.ID = works.ID
50       and works.company_name = 'First Bank Corporation'
51       and salary > 10000;
52
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ID	person_name	city
1	john doe	new york
2	jane smith	new york

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

```
select ID
from employee
where ID not in
     (select ID
      from works
      where company_name = 'First Bank Corporation'
```

```
53  select ID
54  from employee
55  where ID not in
56      (select ID
57       from works
58       where company_name = 'First Bank Corporation');
59
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ID
3

d. Find the ID of each employee who earns more than every employee of "Small Bank Corporation"

```
select ID
from works
where salary > all (select salary
                    from works
                    where company_name = 'Small Bank
Corporation');
```

```
60  select ID
61  from works
62  where salary > all (select salary
63                      from works
64                      where company_name = 'Small Bank Corporation');
65
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ID
1
2

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.

```
select A.company_name
from company as A
where A.company_name <> 'Small Bank Corporation'
        and not exists((select city
                        from company
                        where company_name = 'Small Bank
Corporation')
                    except
                     (select city
                       from company as B
                       where A.company_name = B.company_name))
group by company_name
```

```
select A.company_name
from company as A
where A.company_name <> 'Small Bank Corporation'
        and not exists((select city
                        from company
                        where company_name = 'Small Bank Corporation')
                    except
                     (select city
                       from company as B
                       where A.company_name = B.company_name))
group by company_name
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

company_name
deutsche Bank Corporation

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).

```
select company_name
from works
group by company_name
having count(distinct ID) >= all (select count(distinct ID)
                                  from works
                                  group by company_name)
```

```
select company_name
from works
group by company_name
having count(distinct ID) >= all (select count(distinct ID)
                                  from works
                                  group by company_name)
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

company_name
deutsche 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”.

```
select company_name
from works
group by company_name
having avg(salary) > (select avg(salary)
                      from work
                      where company_name = 'First Bank
Corporation')
```

```
select company_name
from works
group by company_name
having avg(salary) > (select avg(salary)
                      from works
                      where company_name = 'First Bank Corporation')
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

company_name
deutsche Bank Corporation

四、遇到的问题及解决方法

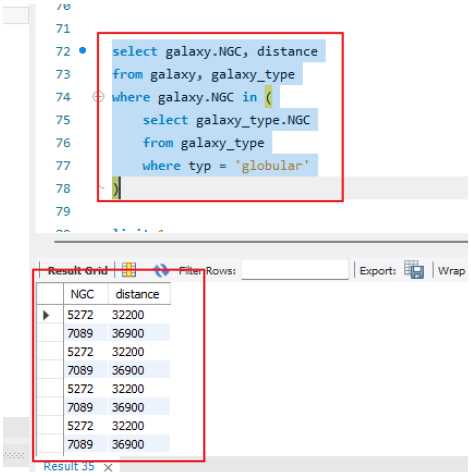
1. 在删除表的过程中出现

   `Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a KEY column. To disable safe mode, toggle the option in Preferences -> SQL Editor and reconnect.`

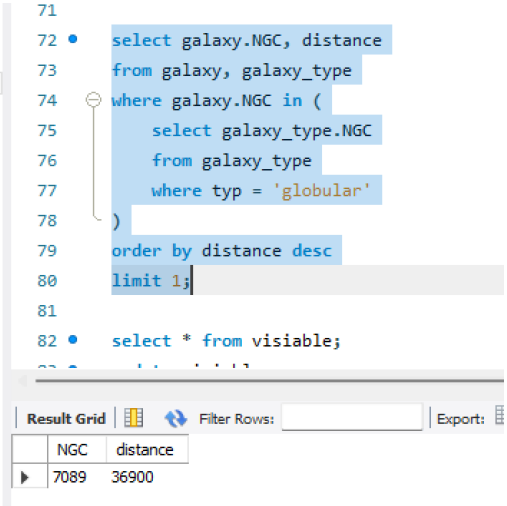
   错误，在查询后发现可以通过修改 SQL_SAFE_UPDATES 为 0 来解决

   `set SQL_SAFE_UPDATES = 0;`

2. 在多表查询的过程中，由于使用的是笛卡尔积对两张表进行了叉乘，因此需要消去重复项



   只显示最大的，使用 limit 1：



3. 在建立测试作业的数据集时，由于 3.9 e 问要求一个银行可以在多个城市分布，因此在建立 company 表格时需要规定 company_id int primary key auto_increment，否则会出现主键不唯一的情况

```
create table company (
    company_id int primary key auto_increment,
    company_name varchar(100),
    city varchar(100) );
```

五、总结

通过本次实验，我掌握了关系数据库语言 SQL 的基本使用，并且完成了以下实验任务：

1. 建立数据库：使用 MySQL Workbench 创建新数据库，并在其中执行了各种 SQL 操作。
2. 数据定义：学习如何创建、删除和修改表，以及如何创建和删除索引和视图。
3. 数据更新：学习如何向表中插入、修改和删除数据。
4. 数据查询：学习如何使用 SQL 查询语句从数据库中检索数据。进行单表查询和多表查询，使用嵌套子查询来解决复杂的查询需求。
5. 视图操作：学习如何创建和删除视图，以及如何通过视图来查询和修改数据。

总的来说，本次实验使我更加熟悉了 SQL 数据库的操作，对课堂中学习的理论知识有了实践经验，掌握更加熟练，本次实验难度较小，仍属于适应、熟悉数据库基本操作，期望后面的实验继续努力！