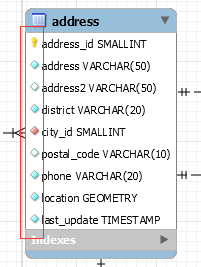
实验一报告

# 回答问题

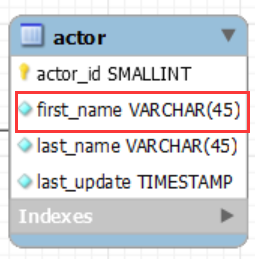
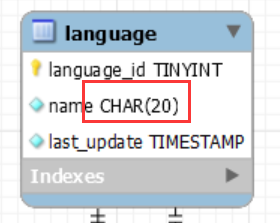
请一边熟悉sakila数据库，一边回答以下问题：

1. sakila.mwb模型中，表结构里每个字段前面的小标记分别表示什么意思？ （观察字段的属性）



|  |  |
| --- | --- |
| 标记 | 意义 |
|  | Primary key |
|  | not null |
|  | 其他 |
|  | 说明该 attribute 和其他表共享 |

2. char和varchar类型的区别是什么？



1. CHAR 的长度是固定的，无论其内部的存储了多少个字符，例 CHAR(4) 的存储大小一直是 4 个 byte

2. VARCHAR 的长度不是固定的，和 CHAR 不同。

3. VARCHAR 的长度上限比 CHAR 更大，VARCHAR 最大有 65535，CHAR 最大有 255。

3. 图中哪部分体现影片-演员关系？换句话说，如果要找出演某个影片的演员名字，访问哪几张表可以获得信息？

表 film\_actor 直接体现了影片和演员关系。其内部存储的是 film 的 id 和演员的 id。

film 名称和演员名称分别存储在 film 表和 actor 表内。

若要根据影片名得到演员的名字，需要访问 film, film\_actor, actor 这三张表。

4. 如果已知某个顾客姓名，要找到他租借的所有影片名，需要访问哪几张表？

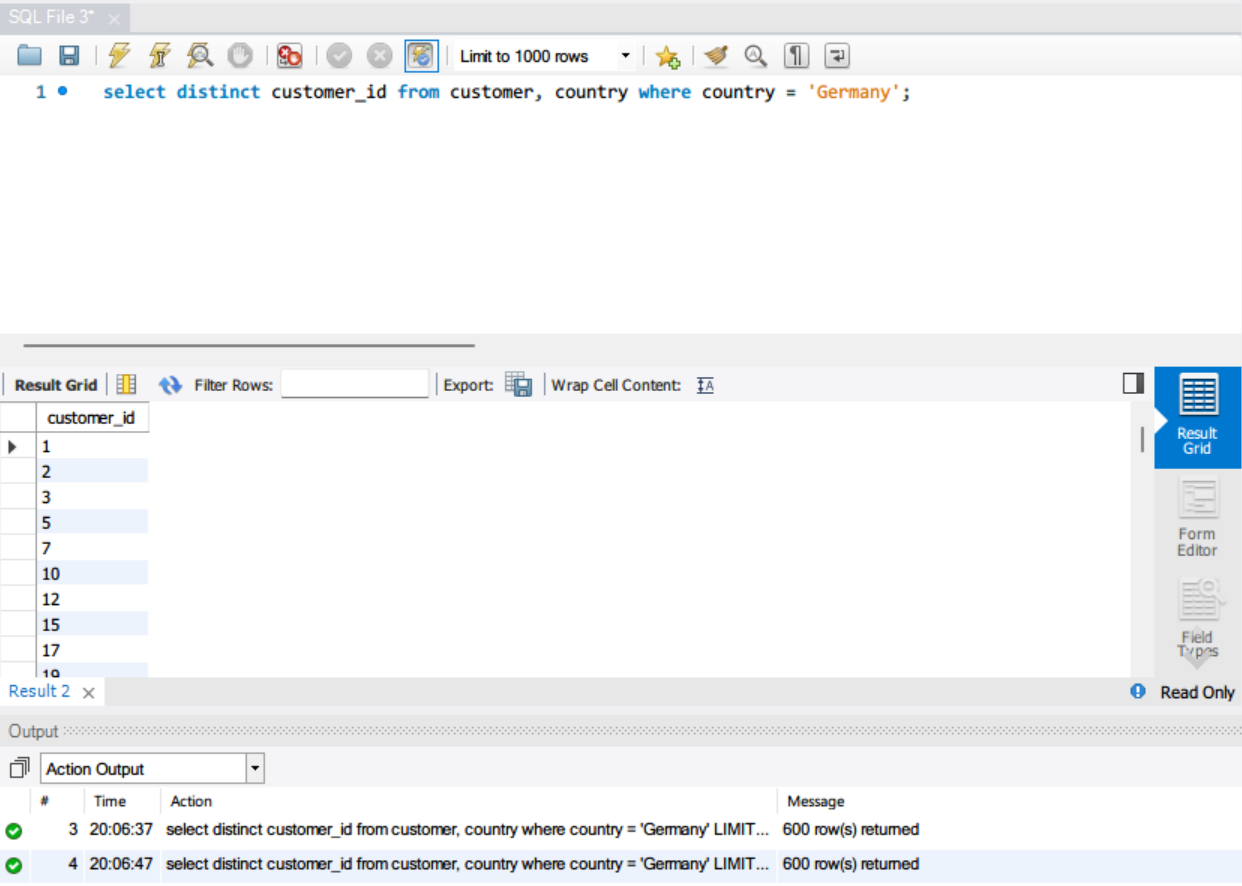
查找以下几个表：customer, rental, inventory, film

# 实验截图

*（注意截图清晰，截图时需要体现SQL语句、执行结果、Output窗口）*

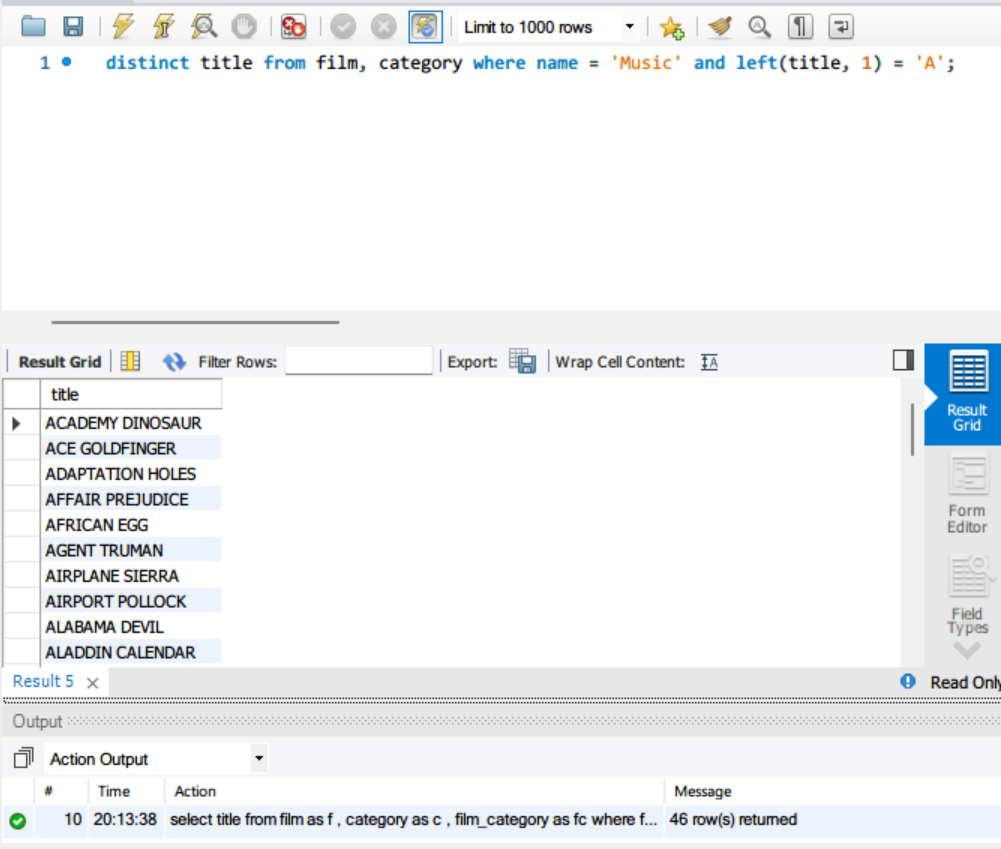
1. 请列出所有country是“Germany”的客户的信息，显示customer\_id、客户姓名、地址、所在区域，所在城市（注意：客户姓名请以first\_name+空格+last\_name的格式，例如：SISSY SOBIESKI）；

select distinct customer\_id from customer, country where country = 'Germany';



1. 列出属于“Music”类型并以“A”开头的电影名；

select distinct film\_id from film, category where name = 'Music' and left(title, 1) = 'A';



1. 找出租DVD花费的总费用在160至170之间的客户，列出他们的first\_name, last\_name和每个人花费的金额；

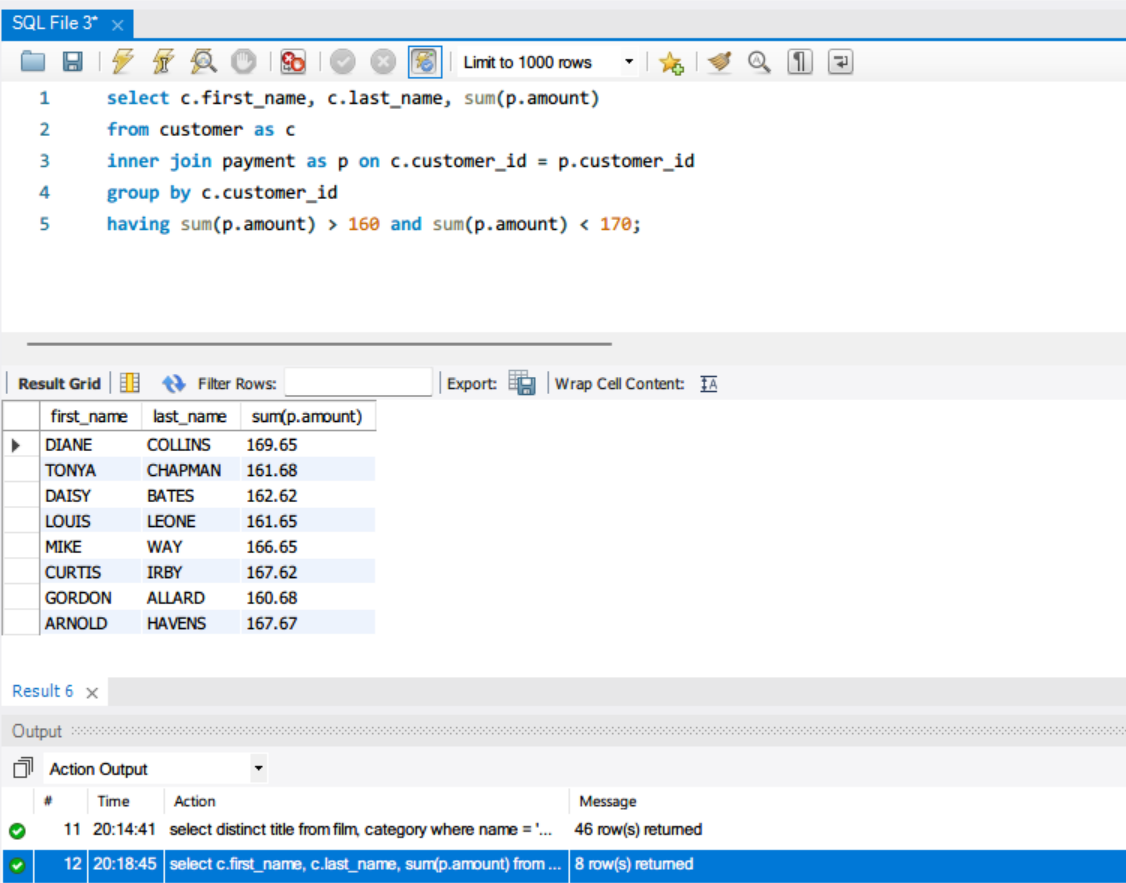
select c.first\_name, c.last\_name, sum(p.amount)

from customer as c

inner join payment as p on c.customer\_id = p.customer\_id

group by c.customer\_id

having sum(p.amount) > 160 and sum(p.amount) < 170;



1. 哪个影片获得了总体最高的租金？请列出影片id、影片名、总租金；

select f.film\_id AS id, f.title AS title, sum(p.amount) AS total\_sales

from payment AS p

inner join rental AS r ON p.rental\_id = r.rental\_id

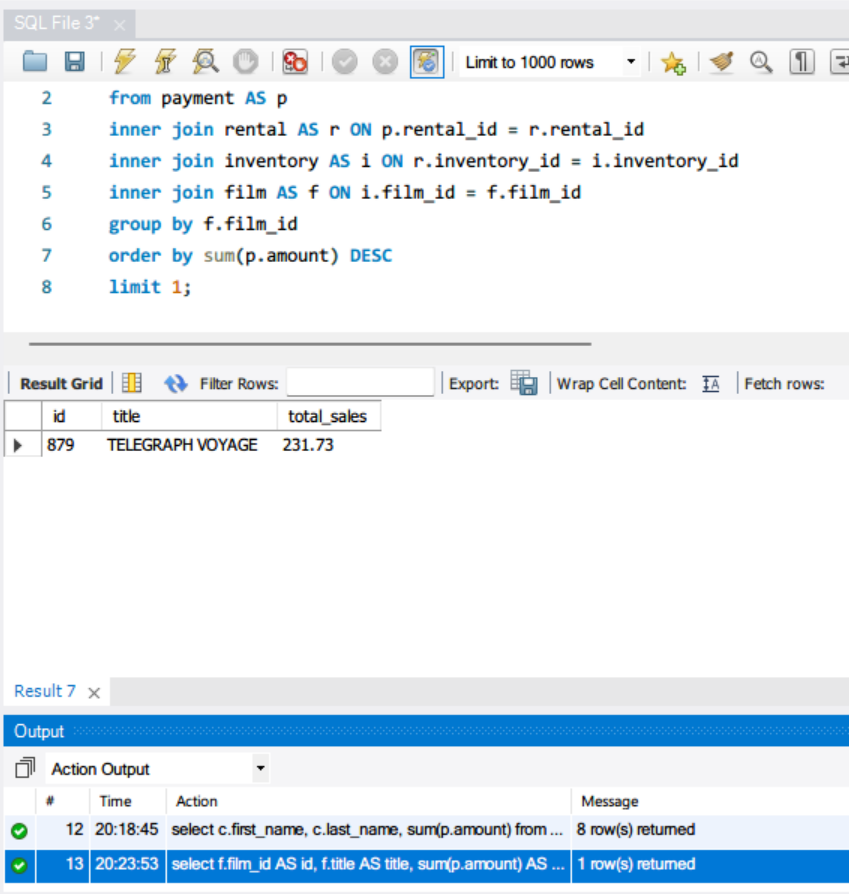
inner join inventory AS i ON r.inventory\_id = i.inventory\_id

inner join film AS f ON i.film\_id = f.film\_id

group by f.film\_id

order by sum(p.amount) DESC

limit 1;



1. 哪些演员出演的电影超过40部？ 请列出演员名、出演的电影数；

select a.first\_name AS first\_name

, a.last\_name AS last\_name

, count(f.film\_id) AS total\_films

from actor AS a

inner join film\_actor AS fa ON a.actor\_id = fa.actor\_id

inner join film AS f ON fa.film\_id = f.film\_id

group by a.actor\_id

having count(f.film\_id) > 40

order by count(f.film\_id) DESC;

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

1. 请找出没有租借过电影《NATURAL STOCK》的顾客姓名；

select c.first\_name AS first\_name

, c.last\_name AS last\_name

from customer AS c

where not exists (

select \*

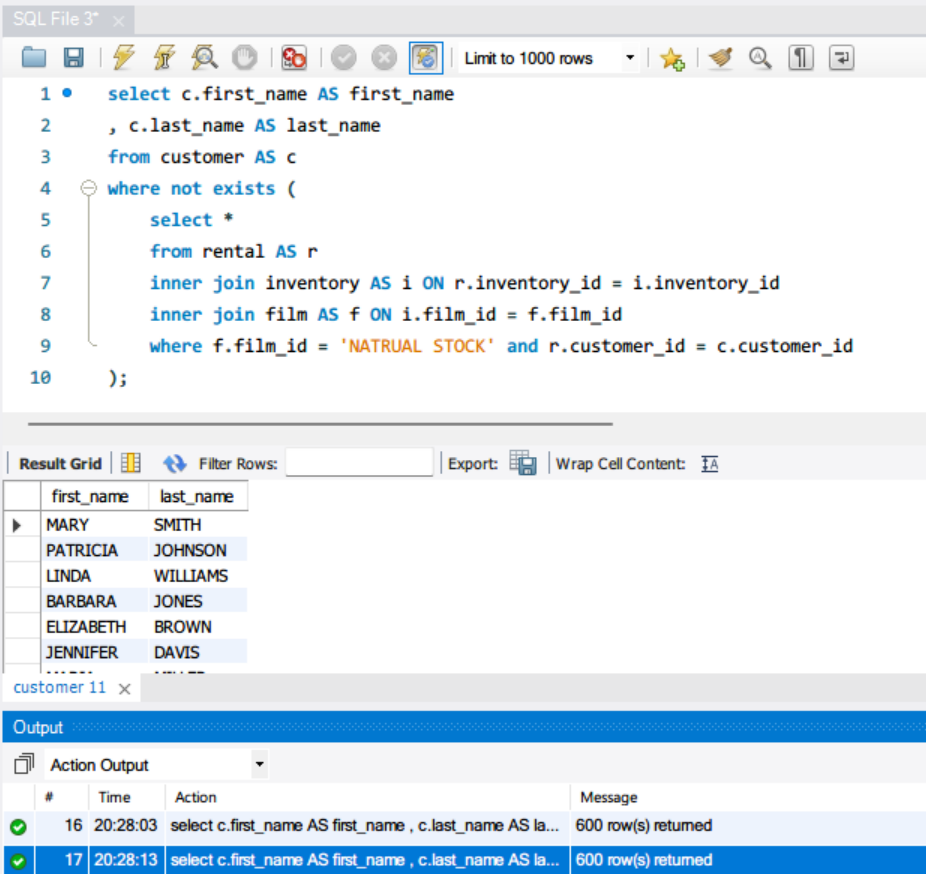
from rental AS r

inner join inventory AS i ON r.inventory\_id = i.inventory\_id

inner join film AS f ON i.film\_id = f.film\_id

where f.film\_id = 'NATRUAL STOCK' and r.customer\_id = c.customer\_id

);



1. 查询既演过《ELEPHANT TROJAN》又演过《DOGMA FAMILY》的演员，列出其姓名；

select a.first\_name AS first\_name

, a.last\_name AS last\_name

from actor AS a

inner join film\_actor AS fa ON a.actor\_id = fa.actor\_id

inner join film AS f ON fa.film\_id = f.film\_id

where f.title = 'ELEPHANT TROJAN' or f.title = 'DOGMA FAMILY'

group by a.actor\_id

having count(f.film\_id) = 2;

图形用户界面, 文本, 应用程序

描述已自动生成

1. 统计每种类型的影片数，显示类型编号、类型名称、该类型影片数；

select c.name AS category\_name

, c.category\_id AS category\_id

, count(f.film\_id) AS total\_films

from film AS f

inner join film\_category AS fc ON f.film\_id = fc.film\_id

inner join category AS c ON fc.category\_id = c.category\_id

group by c.category\_id;

图形用户界面

中度可信度描述已自动生成

1. 有哪些影片是2个商店都有库存的？显示影片名。

select f.film\_id AS film\_id

from film AS f

inner join inventory AS i ON f.film\_id = i.film\_id

where not exists (

select \*

from rental AS r

where r.return\_date is null and i.inventory\_id = r.inventory\_id

)

or not exists(

select \*

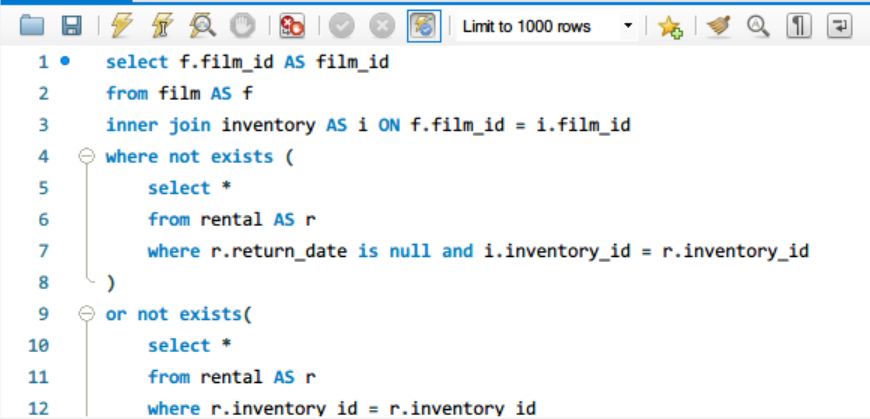
from rental AS r

where r.inventory\_id = r.inventory\_id

)

group by f.film\_id

having count(distinct i.store\_id) = 2;



图形用户界面, 文本, 应用程序

描述已自动生成

1. 查询单次租借影片时间最长的2位客户，列出其first\_name、last\_name和当次租借时长（单位秒）；

select c.first\_name AS first\_name

, c.last\_name AS last\_name

, max(UNIX\_TIMESTAMP(r.return\_date) - UNIX\_TIMESTAMP(r.rental\_date)) AS max\_rent\_time

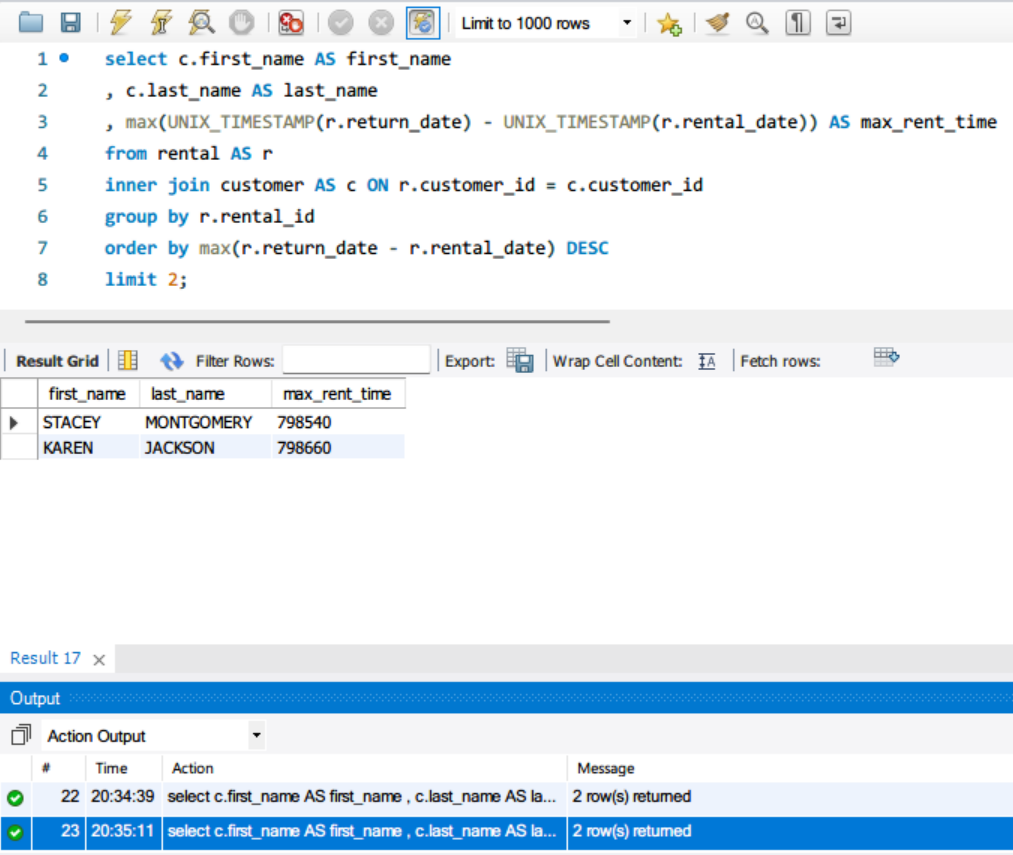
from rental AS r

inner join customer AS c ON r.customer\_id = c.customer\_id

group by r.rental\_id

order by max(r.return\_date - r.rental\_date) DESC

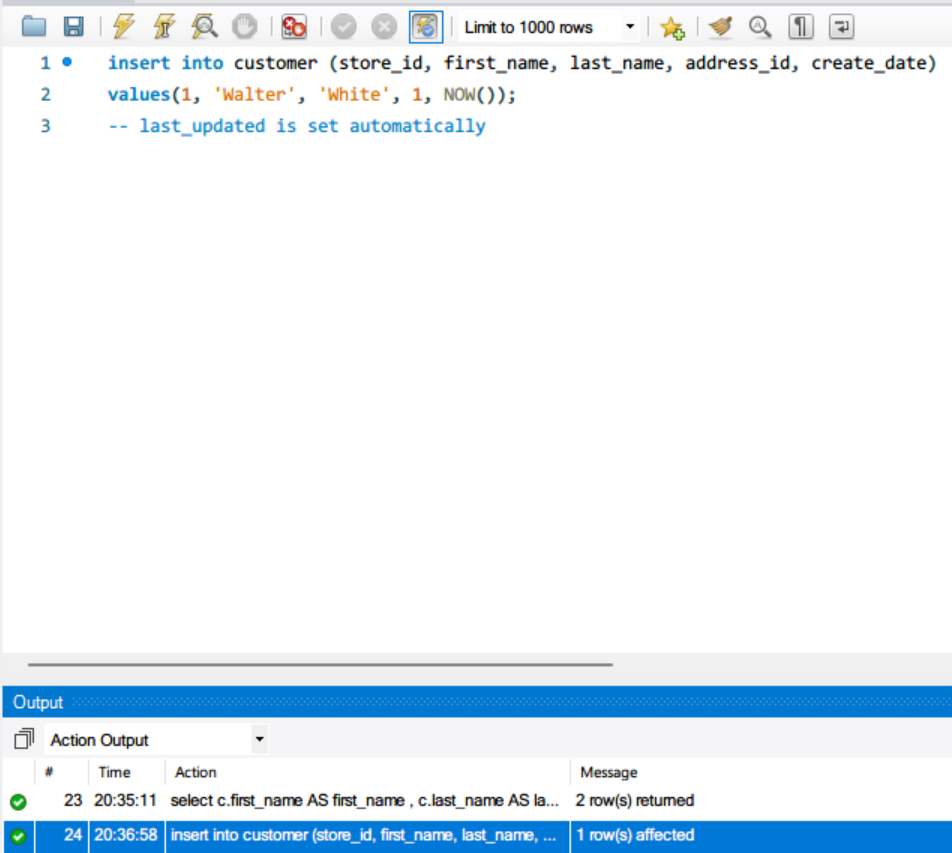
limit 2;



1. 在customer表中新增一条数据，注意customer表与其他表的关系；

insert into customer (store\_id, first\_name, last\_name, address\_id, create\_date, last\_update)

values(1, 'walter', 'white', 1, NOW(), NOW());



1. 修改刚才在customer表中新增的那条数据；

update customer

set last\_update = NOW()

where customer\_id = 600;

insert rental (rental\_date, inventory\_id, customer\_id, return\_date, staff\_id)

values (now(), 1, 600, null, 1);

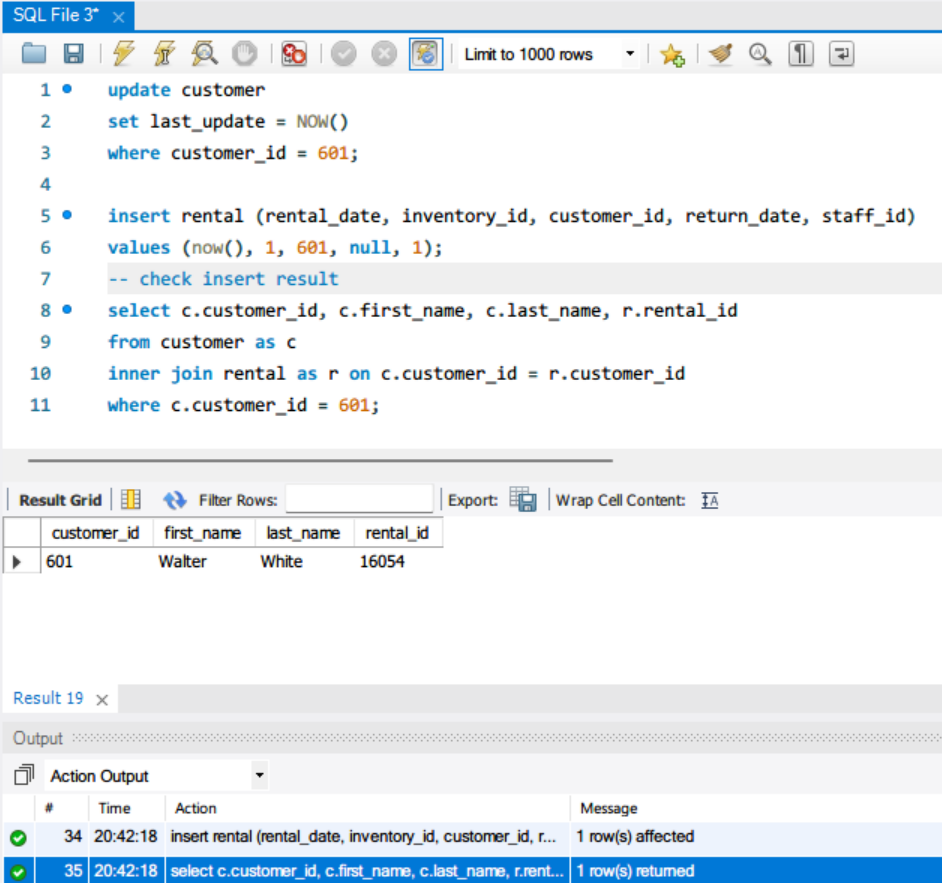
-- check insert result

select c.customer\_id, c.first\_name, c.last\_name, r.rental\_id

from customer as c

inner join rental as r on c.customer\_id = r.customer\_id

where c.customer\_id = 601;



1. 删除第11步新增的那条数据。

delete from rental

where customer\_id = 601;

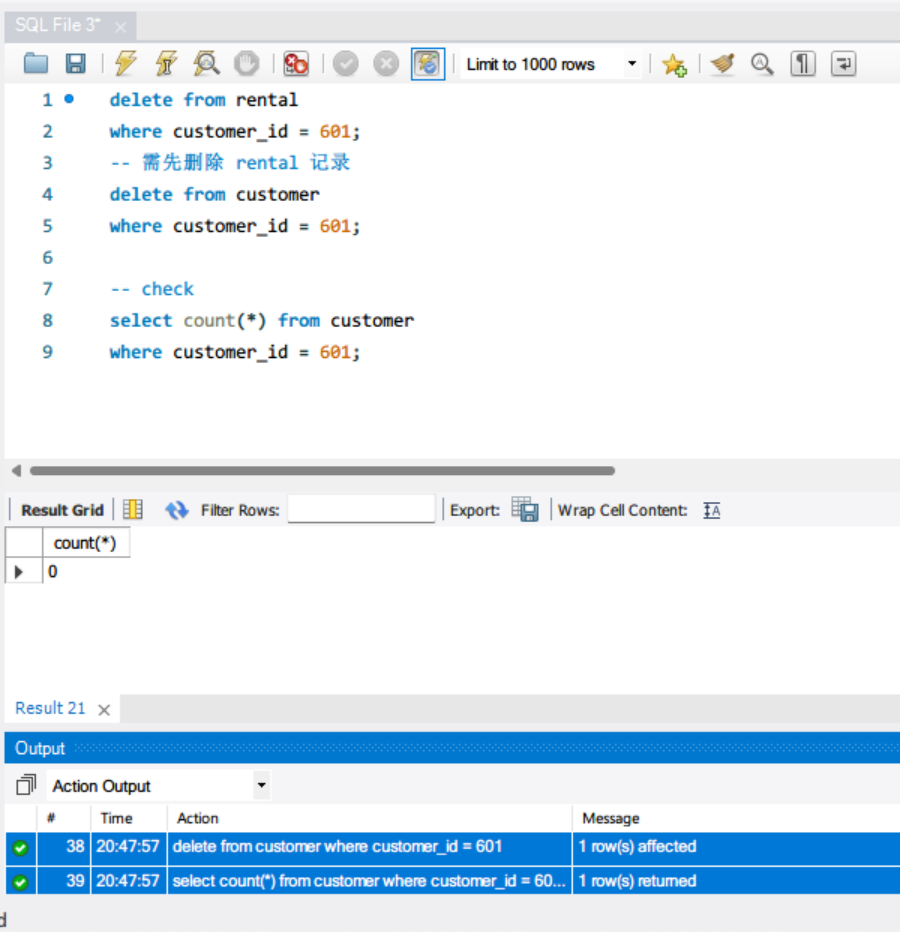
delete from customer

where customer\_id = 601;

-- check

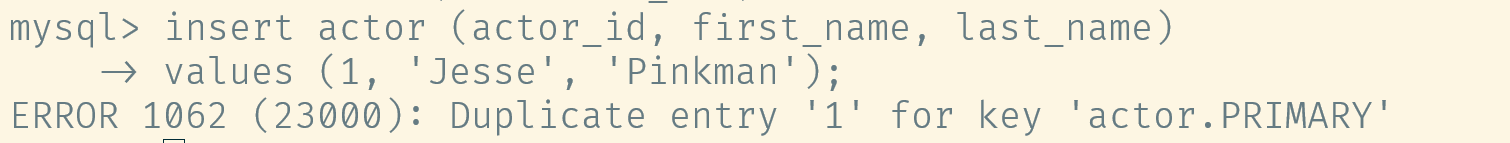
select count(\*) from customer

where customer\_id = 601;

****

# 思考题

1. 如果insert一条数据到actor表，但actor\_id和已有数据重复，会发生什么？同学们请自己尝试一下，截图并分析原因。



Primary Key 不能有重复值。

1. insert语句还用了一个函数NOW()，是做什么的呢？

更新 last\_updated 的值。