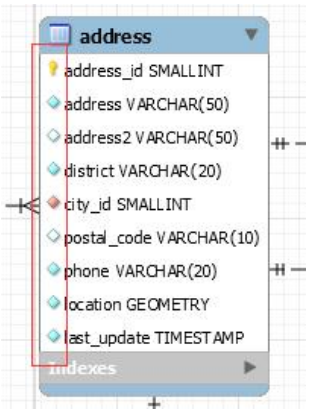
实验一报告

一、 回答问题

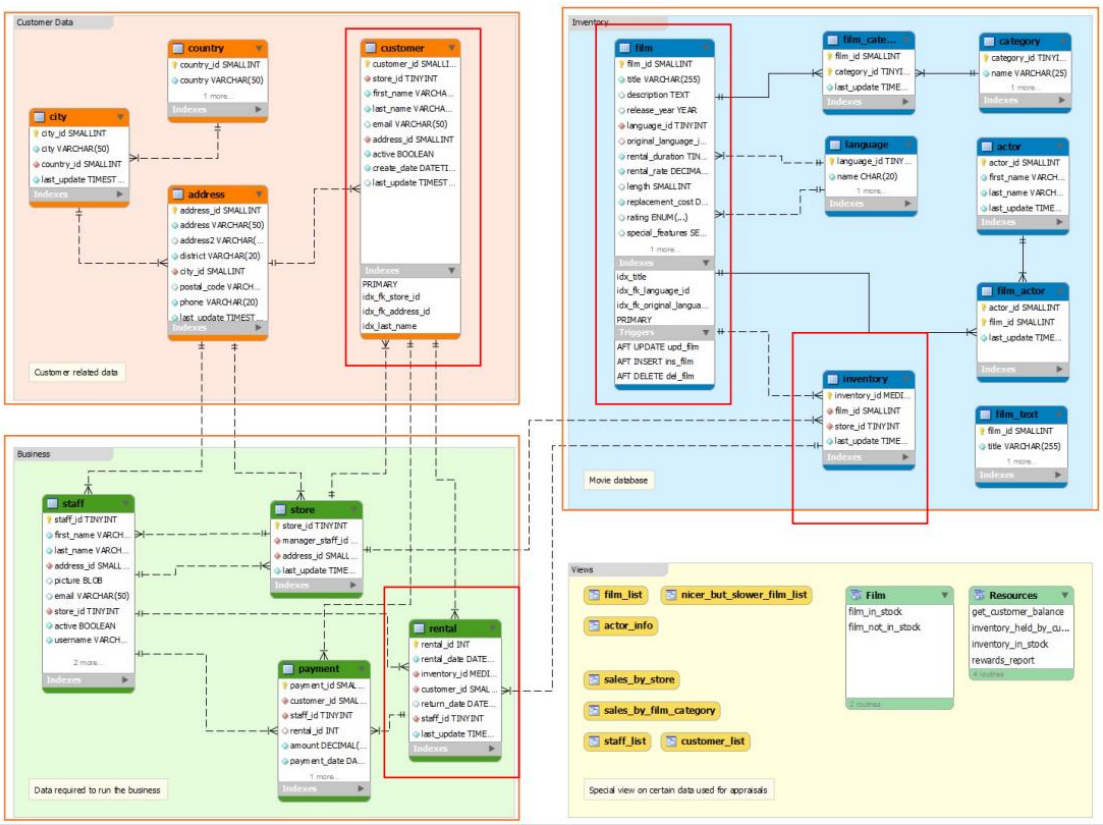
请一边熟悉 sakila 数据库，一边回答以下问题：

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
	主码（Primary Key）/主键，非空，其值能唯一标识一个元组，DBMS 以主码为主要线索管理表中的各个元组。 该标记为为黄色时（如左框），表示它只是一个主键；为红色时（ ），表示它既是一个主键，也是一个外键。
	简单属性，非主码或外码，非空
	简单属性，非主码或外码，可以为空
	外码（Forei Key）/外键，表中的一个属性组，它不是该表的候选码，但是与另一张表的候选码相对应，非空。

2. 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？

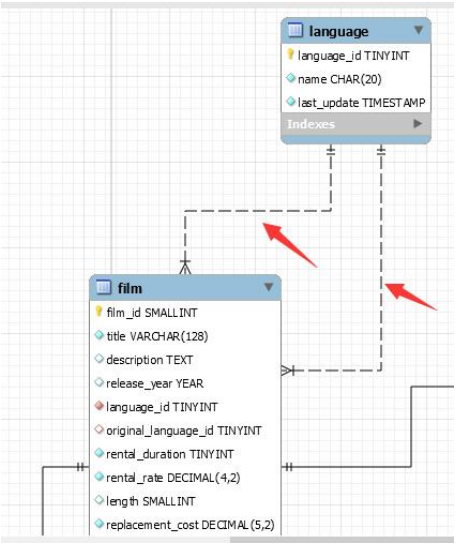


如上图，蓝色框 Movie database 可以体现影片-演员关系，具体来说 film_actor 中记录了影片和演员的具体对应关系。若要找出演某个影片的演员名字，访问 film、film_actor 和 actor 这三张表即可，具体来说，通过 film 表找到某个影片名字 title 对应的影片序号 film_id，然后通过 film_actor 表找到出演该电影的演员序号，即 film_id 对应的 actor_id，然后通过 actor 表找到该演员序号 actor_id 对应的演员名字 first_name 和 last_name。

3. 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？

需要访问 customer、rental、inventory 和 film 这四张表，具体来说，通过 customer 表找到该顾客姓名 first_name last_name 对应的顾客序号 customer_id，然后通过 rental 表找到该 customer_id 对应的所有 inventory_id，然后通过 inventory 表找到这些 inventory_id 对应的电影序号 film_id，最后通过 film 表找到这些 film_id 对应的影片名 title。

4. film 和 language 表间的 2 条虚线表示什么意思？



这两条虚线一条指 film 表中的外键 language_id 对应到 language 表中的主键 language_id，可以唯一确定该电影的语言，而一个 language_id 可以对应 film 表中的多个元组；另一条指 film 表中的外键 original_language_id 对应到 language 表中的主键 language_id，可以唯一确定该电影的原始语言，其中一个 language_id 可以对应 film 表中的多个元组。

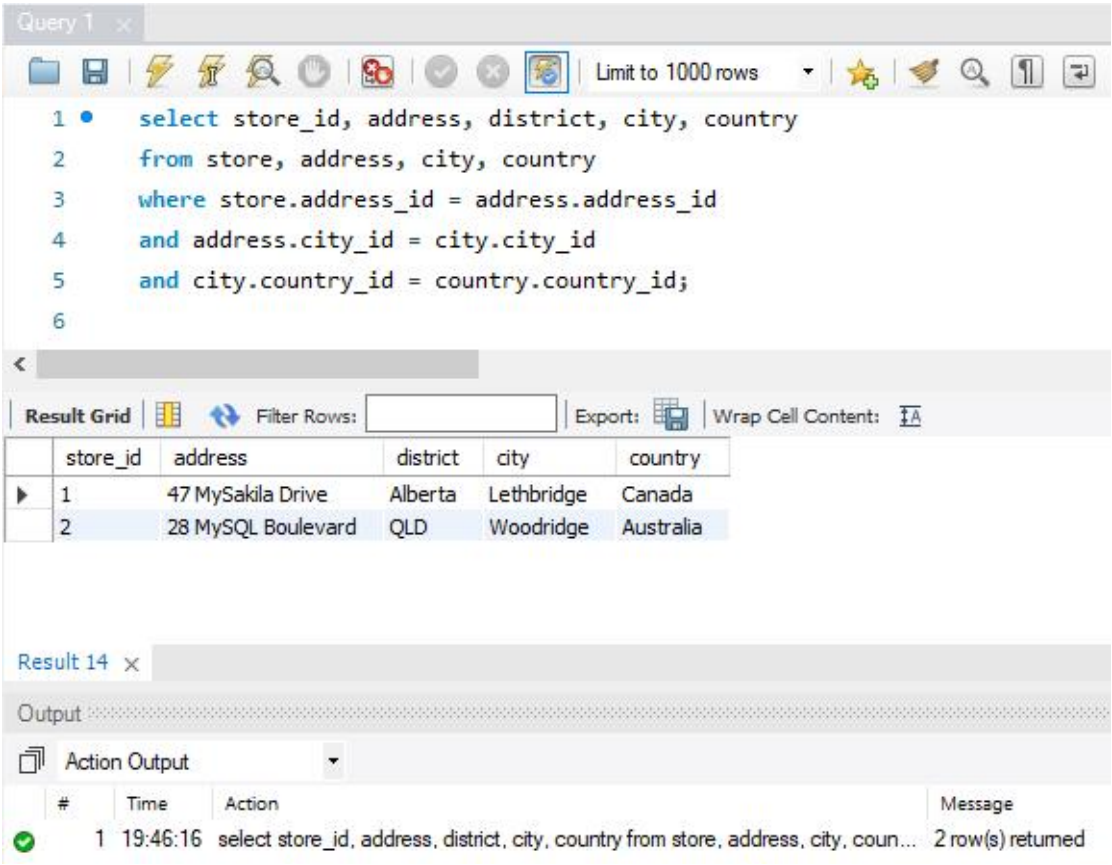
图中的虚线表示非决定关系（non-identifying relationship），指该表中的主键并不是这个外键的一部分，因此该外键在该表中无法对应到唯一的元组，如上例中的 language_id 在 film 表中无法唯一的确定一部电影。若某表的主键是外键的一部分，那么该外键在该表中可以唯一的确定一个元组，此时则要用实线，表示决定关系（identifying relationship）。

二、 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）

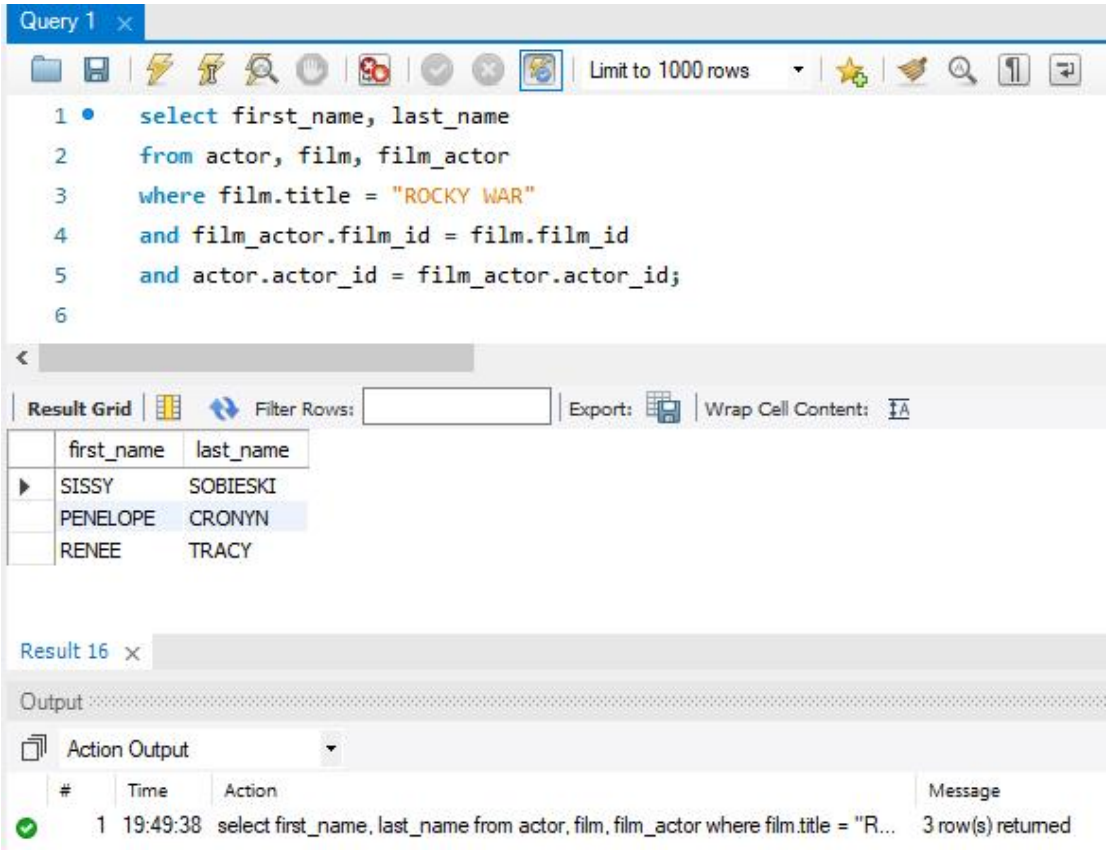
1、 请列出所有商店的详细地址，显示商店 id，商店地址，所在区域，所在城市，所在国家；

```
select store_id, address, district, city, country
from store, address, city, country
where store.address_id = address.address_id
and address.city_id = city.city_id
and city.country_id = country.country_id;
```



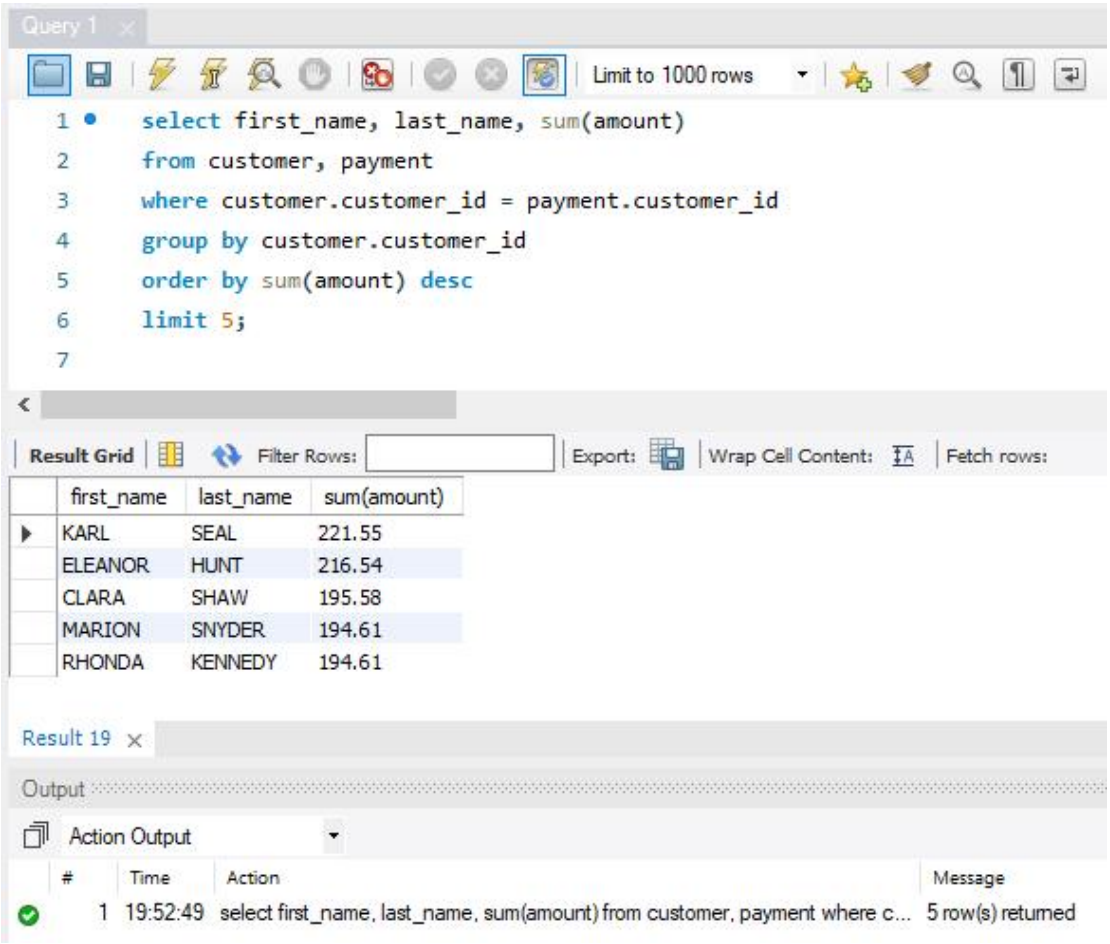
2、 哪些演员出演过影片《ROCKY WAR》？请列出他的 first_name, last_name；

```
select first_name, last_name
from actor, film, film_actor
where film.title = "ROCKY WAR"
and film_actor.film_id = film.film_id
and actor.actor_id = film_actor.actor_id;
```



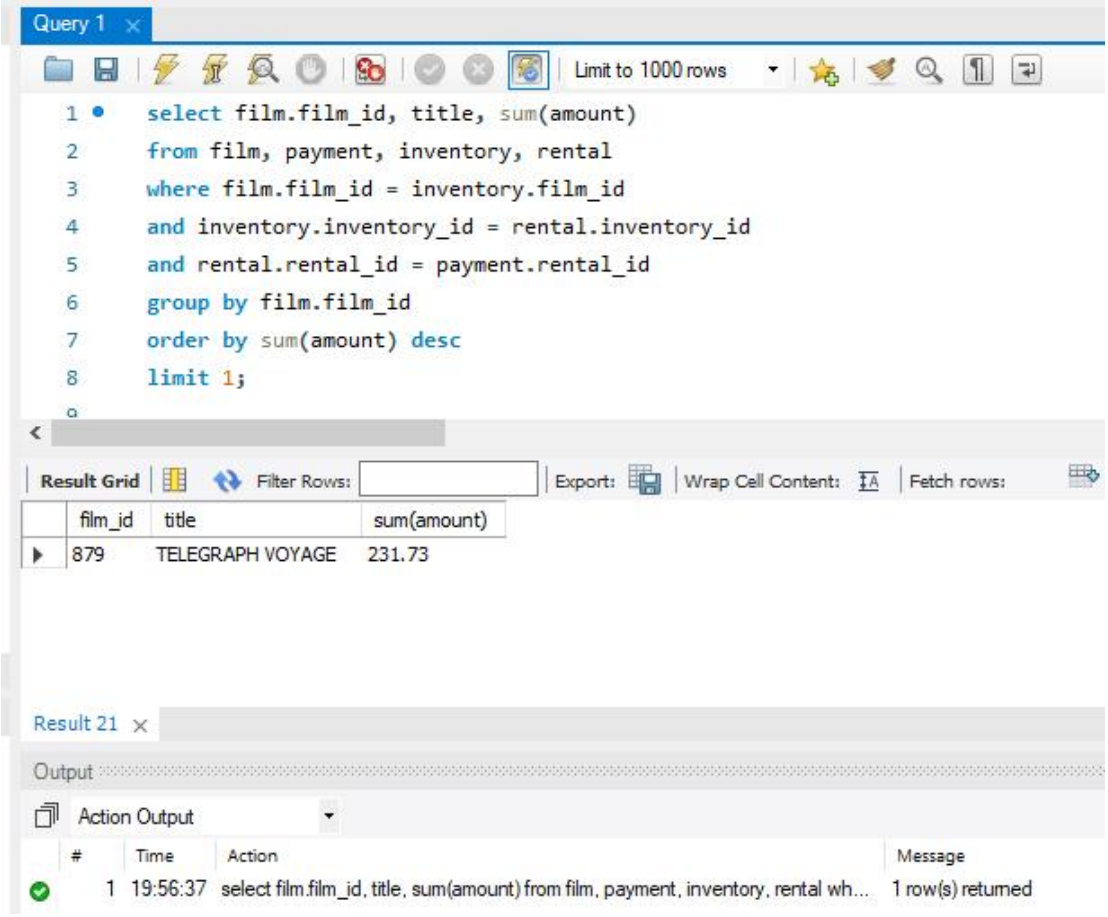
3、 找出租 DVD 花费最高的前 5 名，请列出他们的 first_name, last_name 和每个人花费的金额；

```
select first_name, last_name, sum(amount)
from customer, payment
where customer.customer_id = payment.customer_id
group by customer.customer_id
order by sum(amount) desc
limit 5;
```



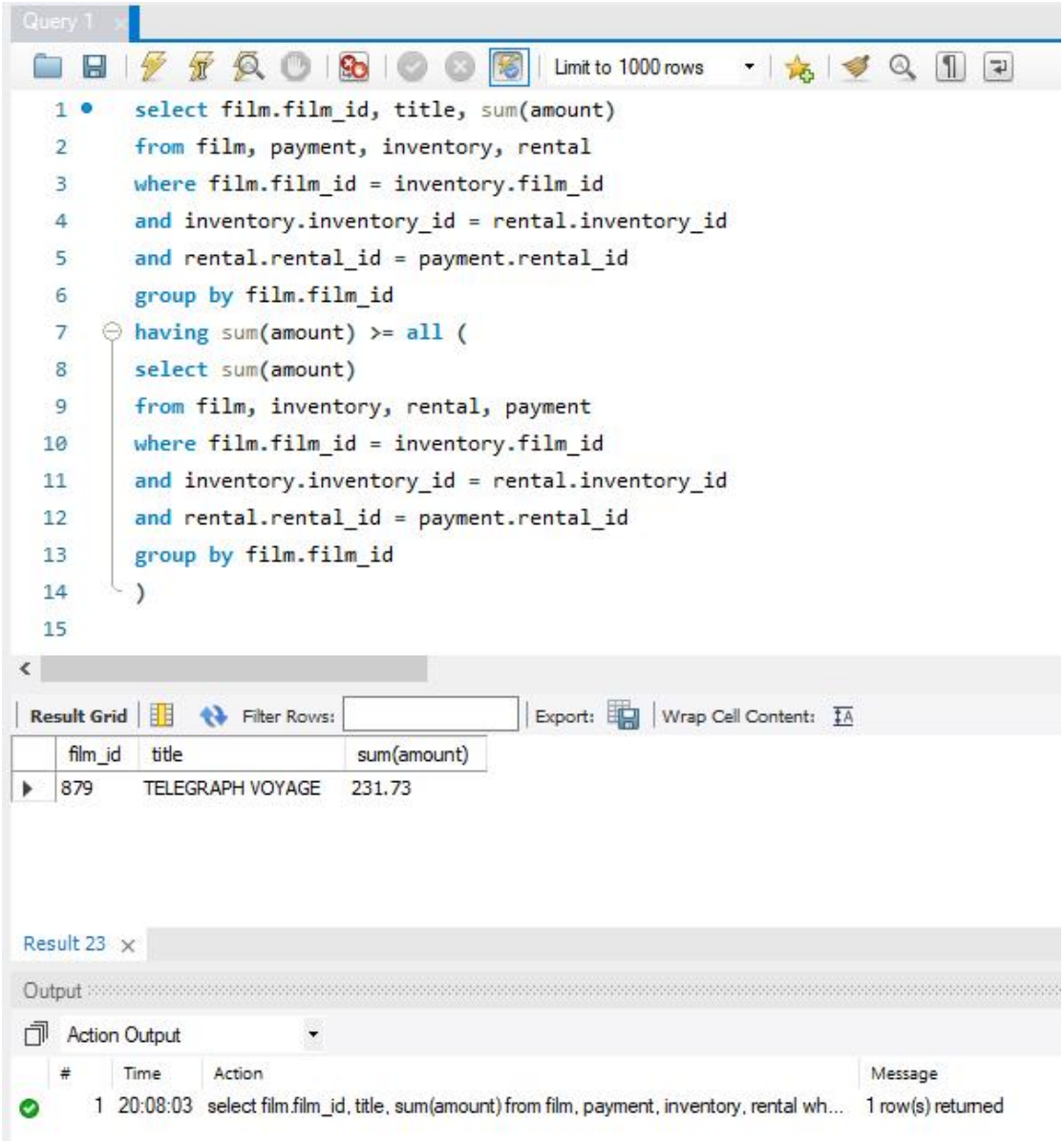
4、 哪个影片获得了总体最高的租金？请列出影片 id、影片名、总租金；

```
select film.film_id, title, sum(amount)
from film, payment, inventory, rental
where film.film_id = inventory.film_id
and inventory.inventory_id = rental.inventory_id
and rental.rental_id = payment.rental_id
group by film.film_id
order by sum(amount) desc
limit 1;
```



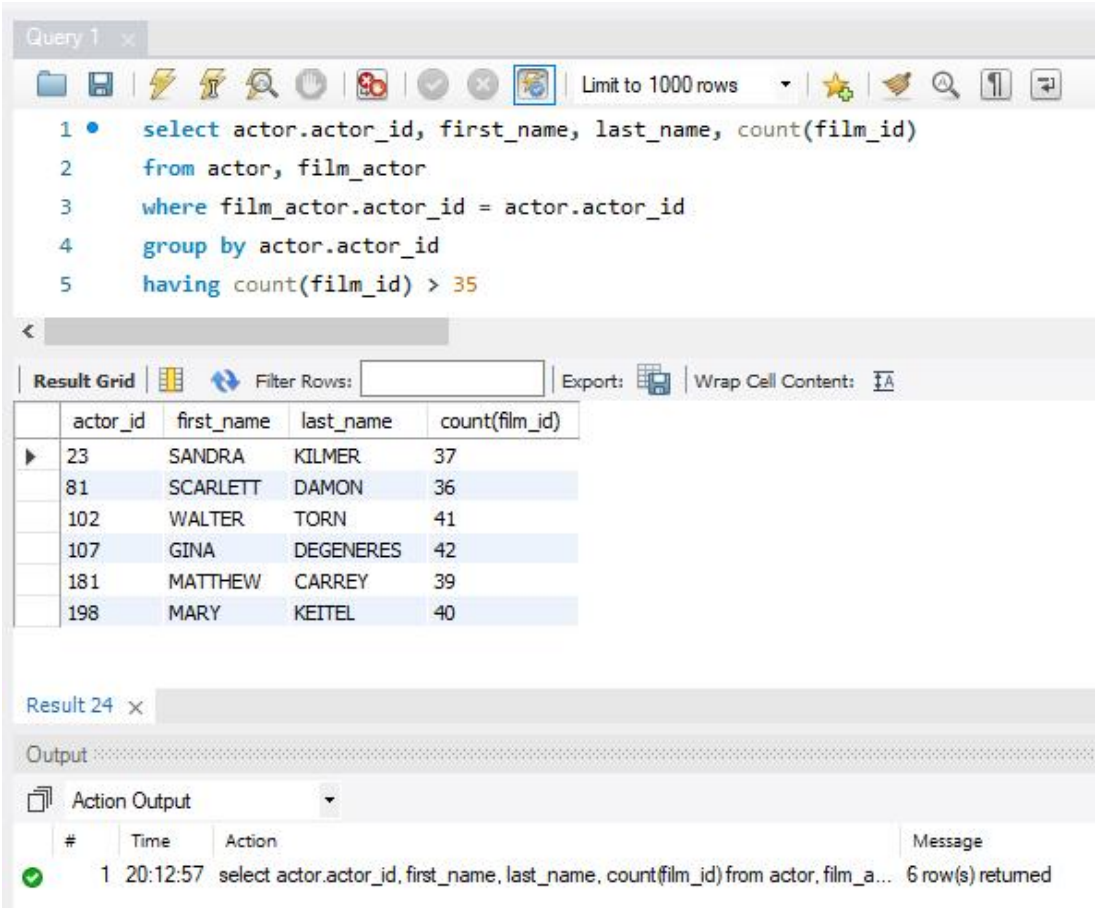
但此种方法无法解决有两部电影同时获得总体最高租金的情况，下面给出一种略复杂但可以解决该问题的做法：

```
select film.film_id, title, sum(amount)
from film, payment, inventory, rental
where film.film_id = inventory.film_id
and inventory.inventory_id = rental.inventory_id
and rental.rental_id = payment.rental_id
group by film.film_id
having sum(amount) >= all (
select sum(amount)
from film, inventory, rental, payment
where film.film_id = inventory.film_id
and inventory.inventory_id = rental.inventory_id
and rental.rental_id = payment.rental_id
group by film.film_id)
```



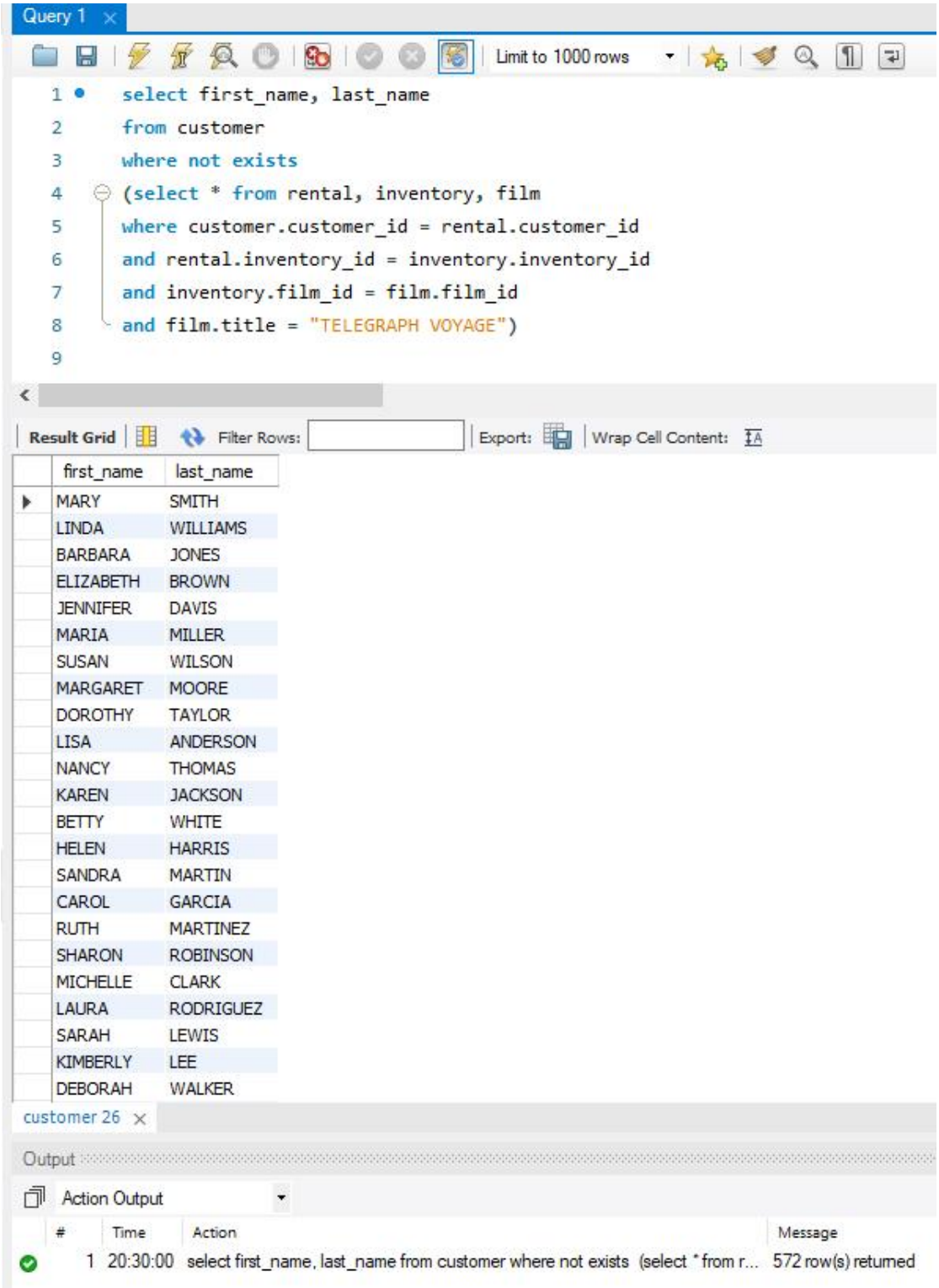
5、 哪个演员出演的电影超过 35 部？ 请列出演员 id、演员名、出演的电影数；

```
select actor.actor_id, first_name, last_name, count(film_id)
from actor, film_actor
where film_actor.actor_id = actor.actor_id
group by actor.actor_id
having count(film_id) > 35
```



6、 请找出没有租借过电影《TELEGRAPH VOYAGE》的顾客姓名；

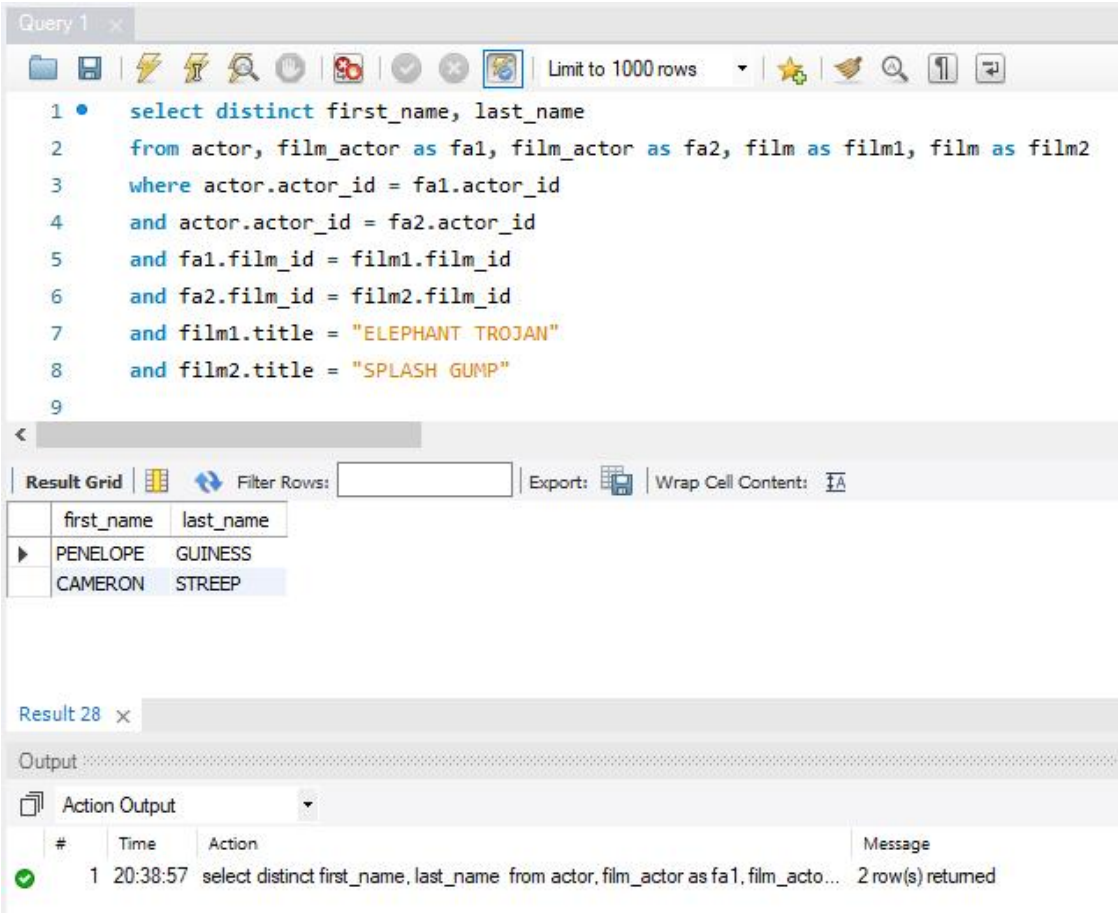
```
select first_name, last_name
from customer
where not exists
(select * from rental, inventory, film
where customer.customer_id = rental.customer_id
and rental.inventory_id = inventory.inventory_id
and inventory.film_id = film.film_id
and film.title = "TELEGRAPH VOYAGE")
```



7、 查询演过《ELEPHANT TROJAN》和《SPLASH GUMP》这两部电影的演员，列出其姓名；

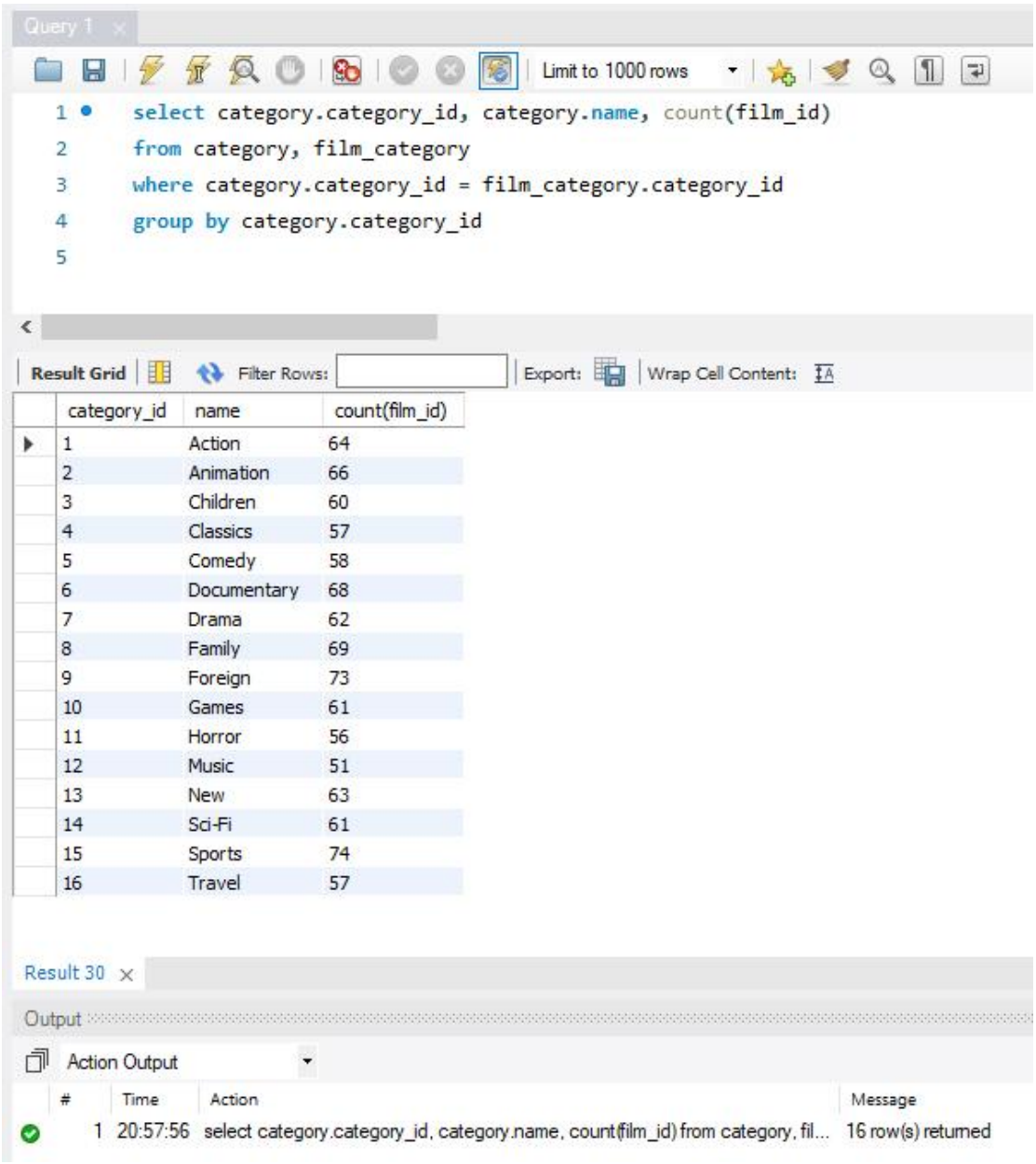
```
select distinct first_name, last_name
from actor, film_actor as fa1, film_actor as fa2, film as film1, film as film2
where actor.actor_id = fa1.actor_id
```

```
and actor.actor_id = fa2.actor_id
and fa1.film_id = film1.film_id
and fa2.film_id = film2.film_id
and film1.title = "ELEPHANT TROJAN"
and film2.title = "SPLASH GUMP"
```



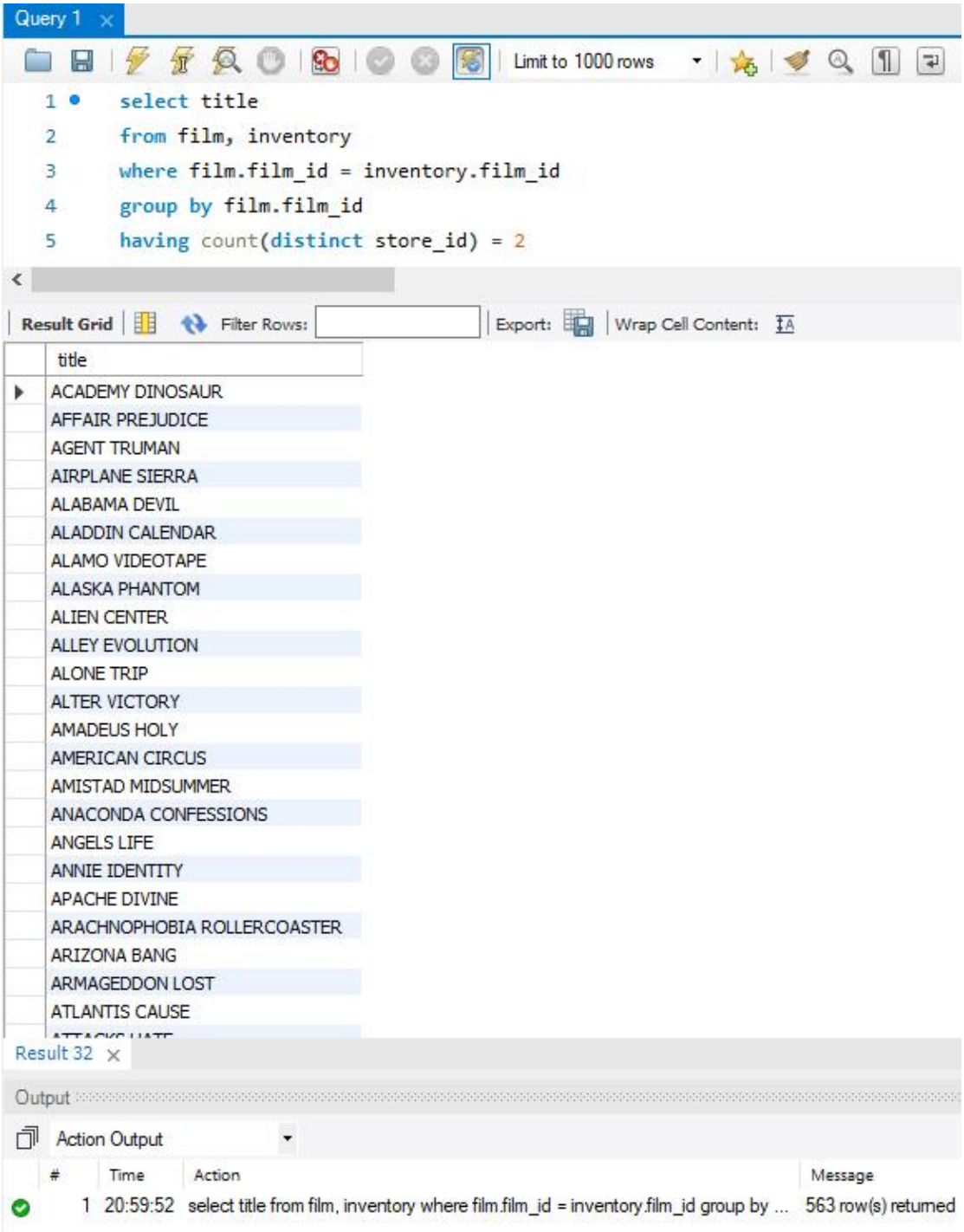
8、 统计每种类型的影片数，显示类型编号、类型名称、该类型影片数；

```
select category.category_id, category.name, count(film_id)
from category, film_category
where category.category_id = film_category.category_id
group by category.category_id
```



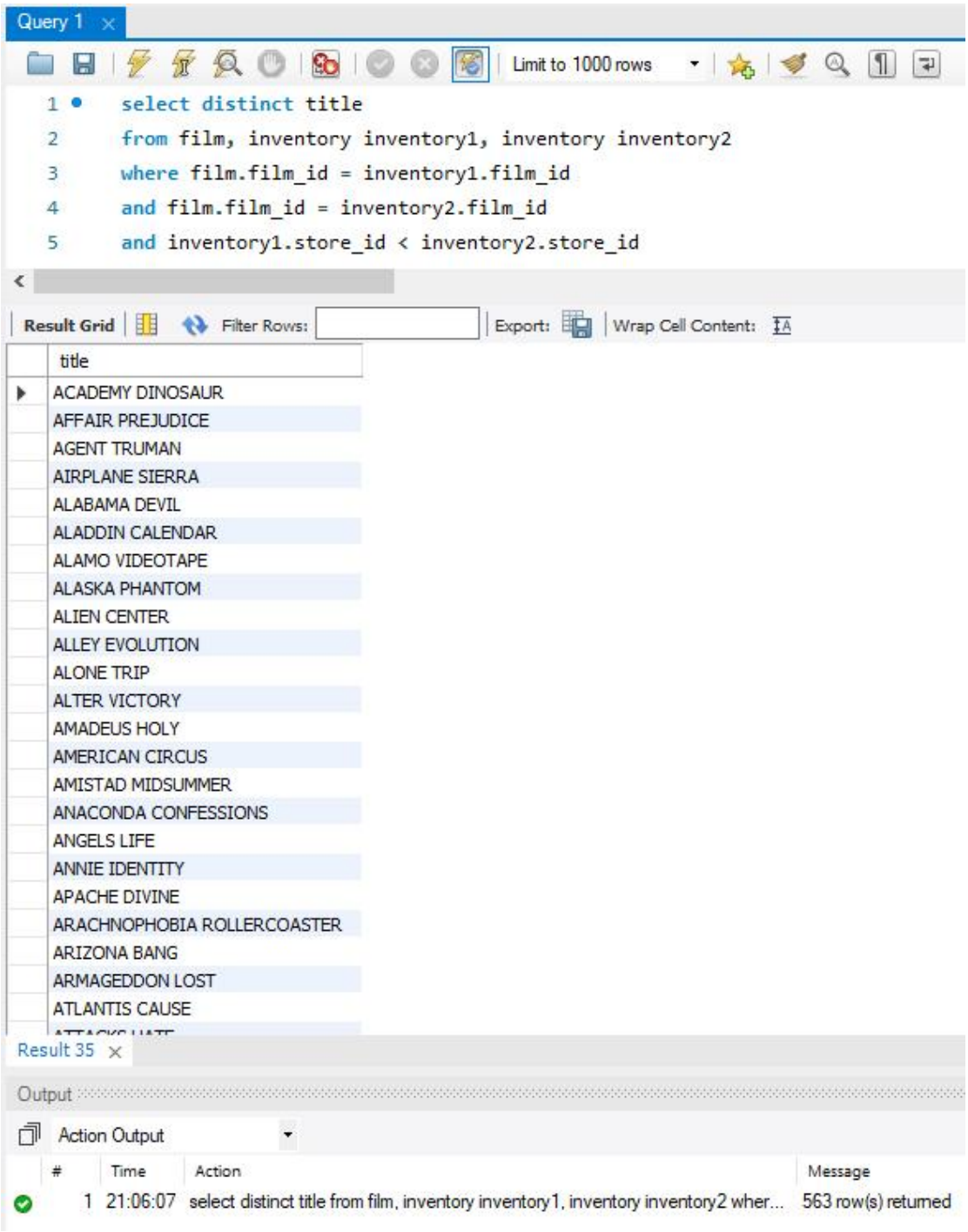
9、 有哪些影片是 2 个商店都有库存的？

```
select title
from film, inventory
where film.film_id = inventory.film_id
group by film.film_id
having count(distinct store_id) = 2
```



使用连接也可以达到相同的效果：

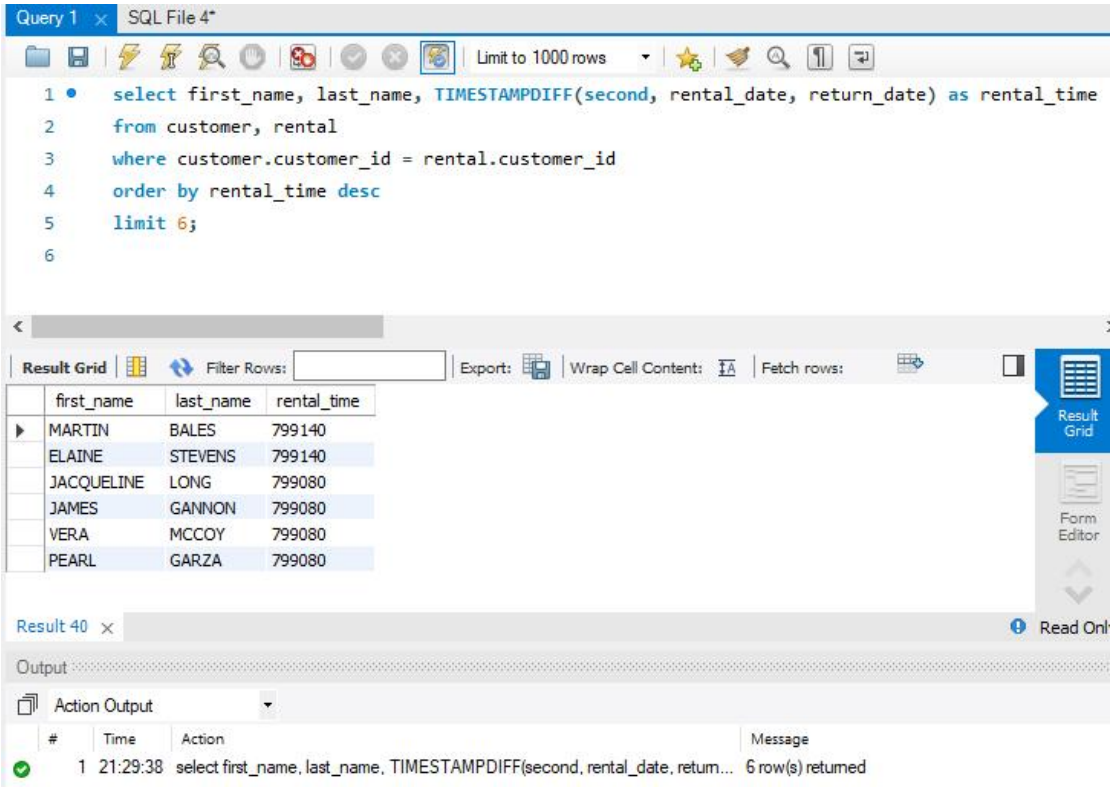
```
select distinct title
from film, inventory inventory1, inventory inventory2
where film.film_id = inventory1.film_id
and film.film_id = inventory2.film_id
and inventory1.store_id < inventory2.store_id
```



10、 查询单次租借影片时间最长的 6 位客户，列出其 first_name、last_name 和当次租借时长；

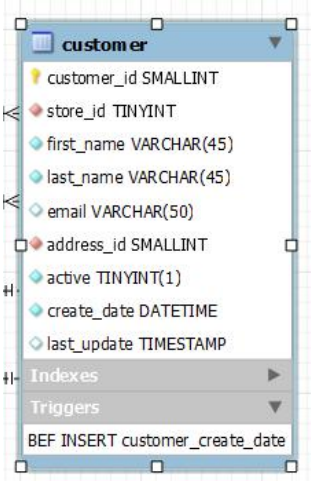
```
select first_name, last_name, TIMESTAMPDIFF(second, rental_date, return_date) as rental_time
from customer, rental
where customer.customer_id = rental.customer_id
order by rental_time desc
```

```
limit 6;
```



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

观察如下 sakila.mwb 数据模型可知，在 customer 表中，属性 store_id 和 address_id 均为外键，因此在插入时只能选择其作为候选码的表中已有的值，如 store_id 取值为 1，address_id 取值为 605。同时，create_date 属性存在 trigger，会设置为 insert 该行时的时间；last_update 属性存在默认值，为 last_update 该行的时间；active 属性存在默认值为 1。因此，在插入时，只需要设置 customer_id, store_id, first_name, last_name, email, address_id 这几个属性值即可。



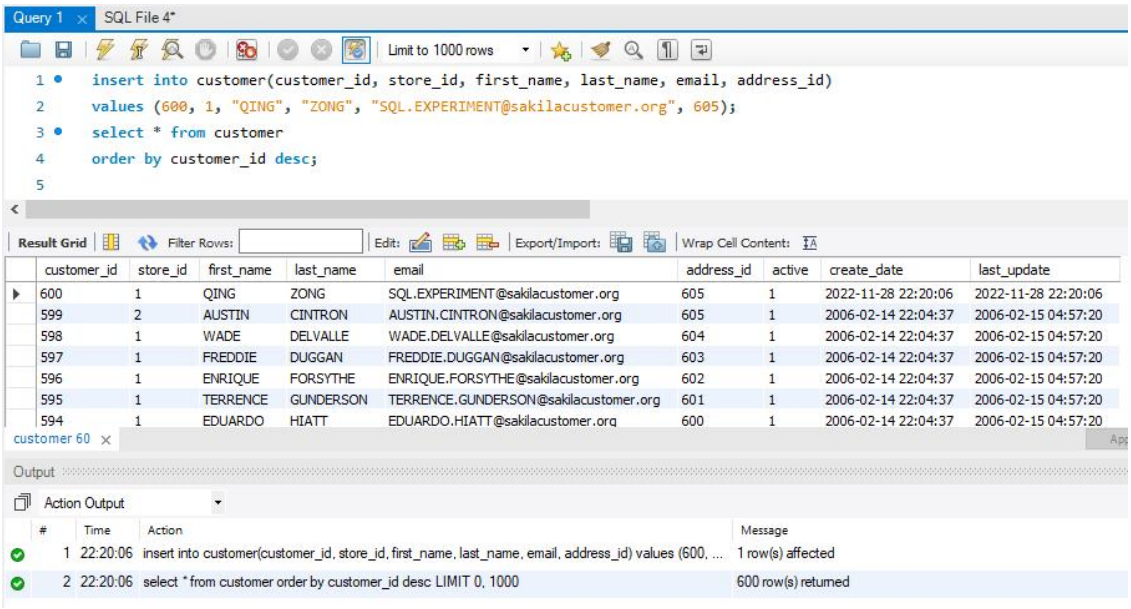
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
customer_id	SMALLINT	☑	☑	☐	☐	☑	☐	☑	☐	
store_id	TINYINT	☐	☑	☐	☐	☑	☐	☐	☐	
first_name	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
last_name	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
email	VARCHAR(50)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
address_id	SMALLINT	☐	☑	☐	☐	☑	☐	☐	☐	
active	TINYINT(1)	☐	☑	☐	☐	☐	☐	☐	☐	'1'
create_date	DATETIME	☐	☑	☐	☐	☐	☐	☐	☐	
last_update	TIMESTAMP	☐	☐	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP ON...
		☐	☐	☐	☐	☐	☐	☐	☐	

可在表中增加数据如下：

```
insert into customer(customer_id, store_id, first_name, last_name, email, address_id)
values (600, 1, "QING", "ZONG", "SQL.EXPERIMENT@sakilacustomer.org", 605);
```

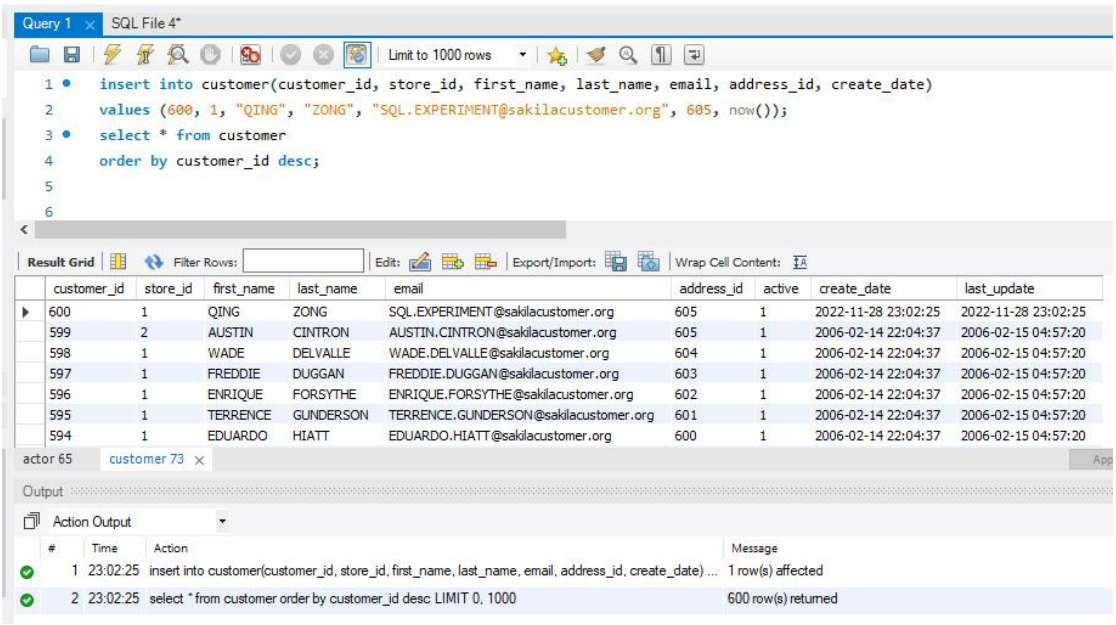
然后查看添加数据之后的表：

```
select * from customer
order by customer_id desc;
```



当然，也可以用 now()函数来设置 create_date 属性：

```
insert into customer(customer_id, store_id, first_name, last_name, email, address_id,
create_date)
values (600, 1, "QING", "ZONG", "SQL.EXPERIMENT@sakilacustomer.org", 605,
now());
```



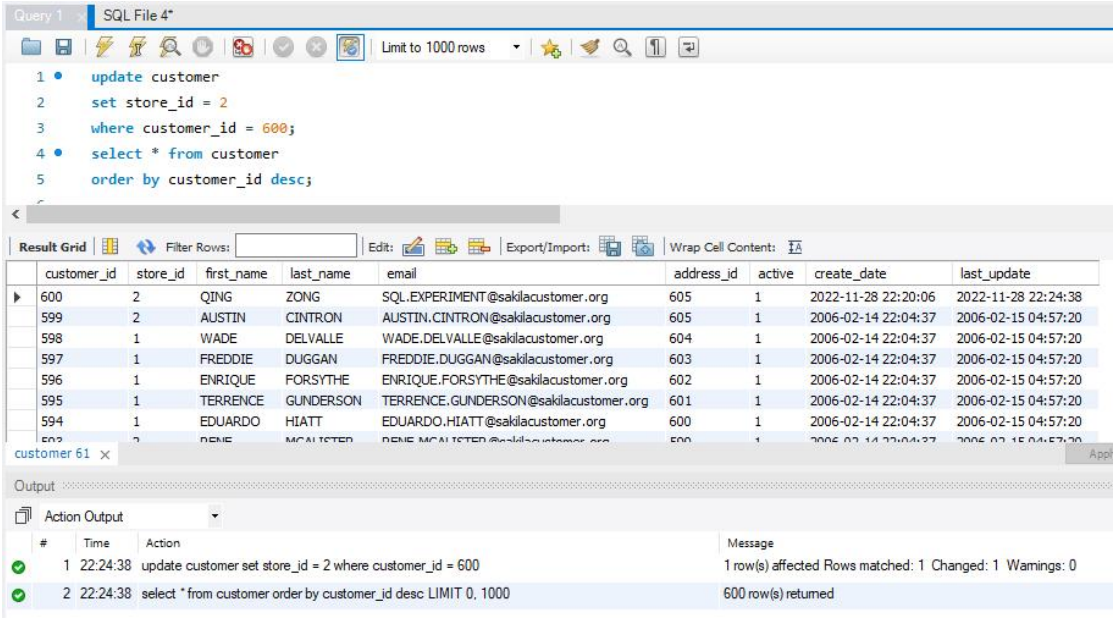
12、 修改刚才在 customer 表中新增的那条数据；

可在表中修改新增的数据如下：

```
update customer
set store_id = 2
where customer_id = 600;
```

然后查看添加数据之后的表：

```
select * from customer
order by customer_id desc;
```

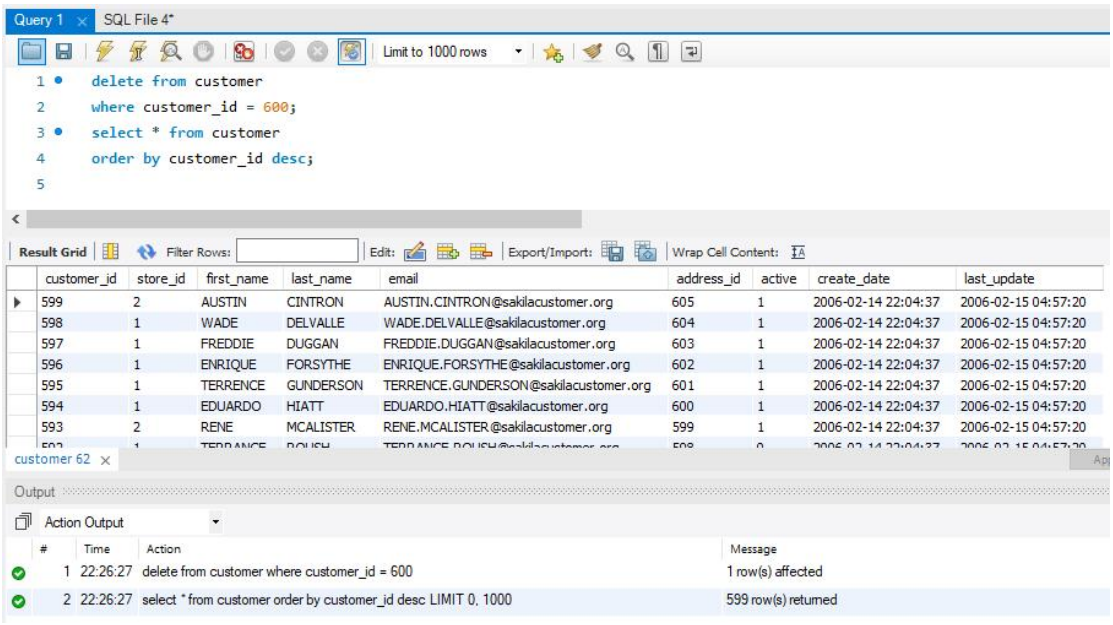


13、 删除第 11 步新增的那条数据。

```
delete from customer
where customer_id = 600;
```

然后查看添加数据之后的表：

```
select * from customer
order by customer_id desc;
```



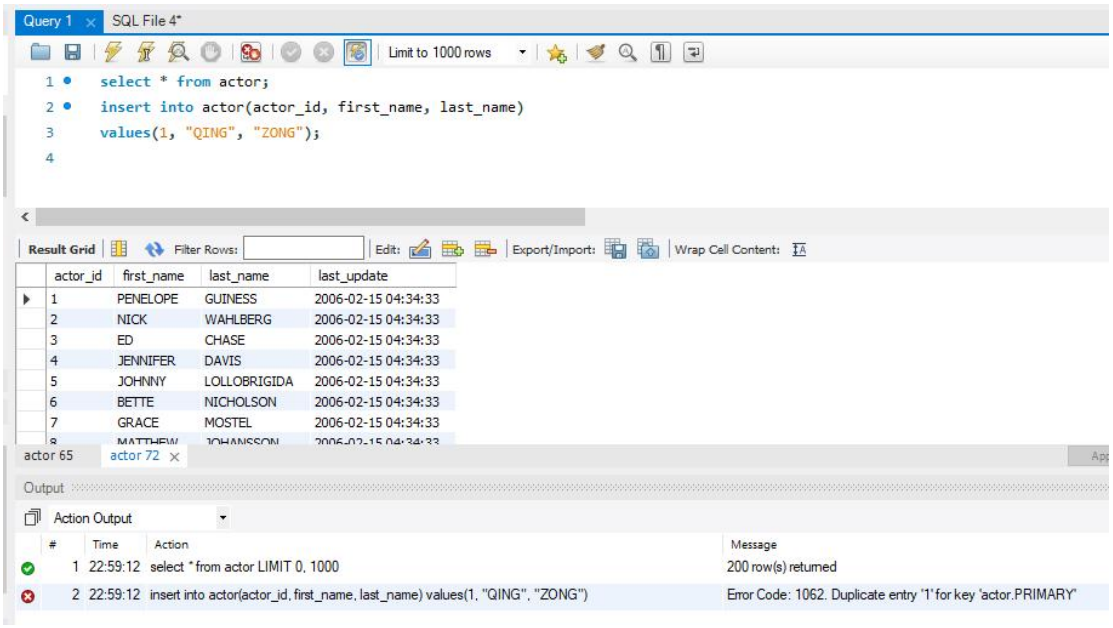
三、 思考题

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。

插入一条 actor_id 为 1 的数据：

```
insert into actor(actor_id, first_name, last_name)
values(1, “QING”, “ZONGQ”);
```

会产生如下错误：



通过 Message，我们发现错误原因是插入了相同主键的元组。因为主键需要满足唯一性，能唯一标识一个元组，因此只要插入了主键与现有元组中的主键相同的数据就会产生这样的错误。

2) insert 语句还用了一个函数 NOW()，是做什么的呢？

函数 now()用于获取当前的日期和时间，以时间戳 timestamp 的格式显示：