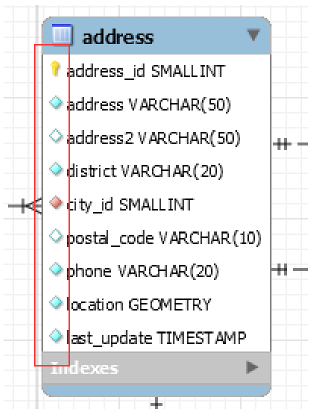
实验一报告

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）

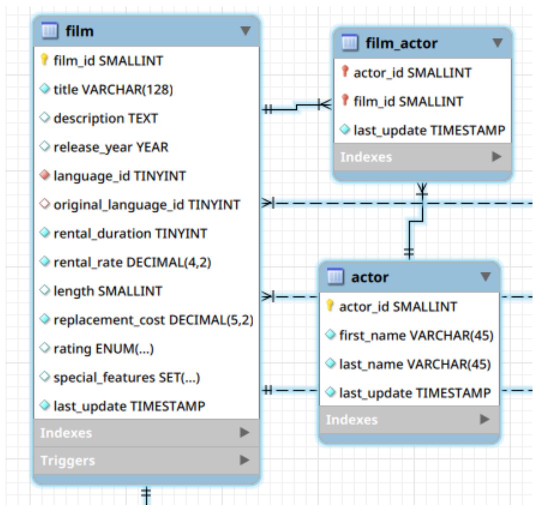


address
- address_id SMALLINT
- address VARCHAR(50)
- address2 VARCHAR(50)
- district VARCHAR(20)
- city_id SMALLINT
- postal_code VARCHAR(10)
- phone VARCHAR(20)
- location GEOMETRY
- last_update TIMESTAMP

Indexes

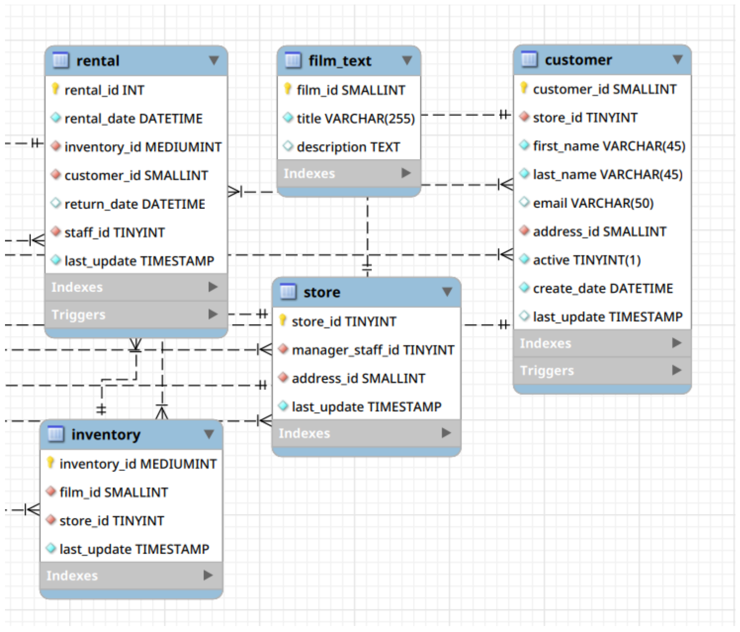
标记	意义
🔑	主键
🔹	非空的键
◇	默认为空的键
♦	来自其他表的非空外键

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？



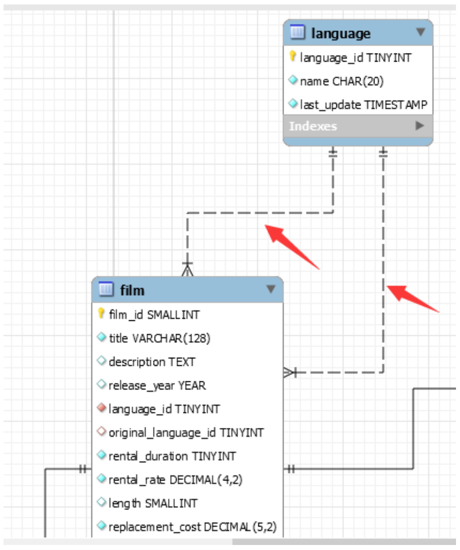
这三个表体现了影片-演员关系：film、film_actor、actor

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 rental、film_text、custormer、inventory、store 这几张表

4. film 和 language 表间的 2 条虚线表示什么意思？

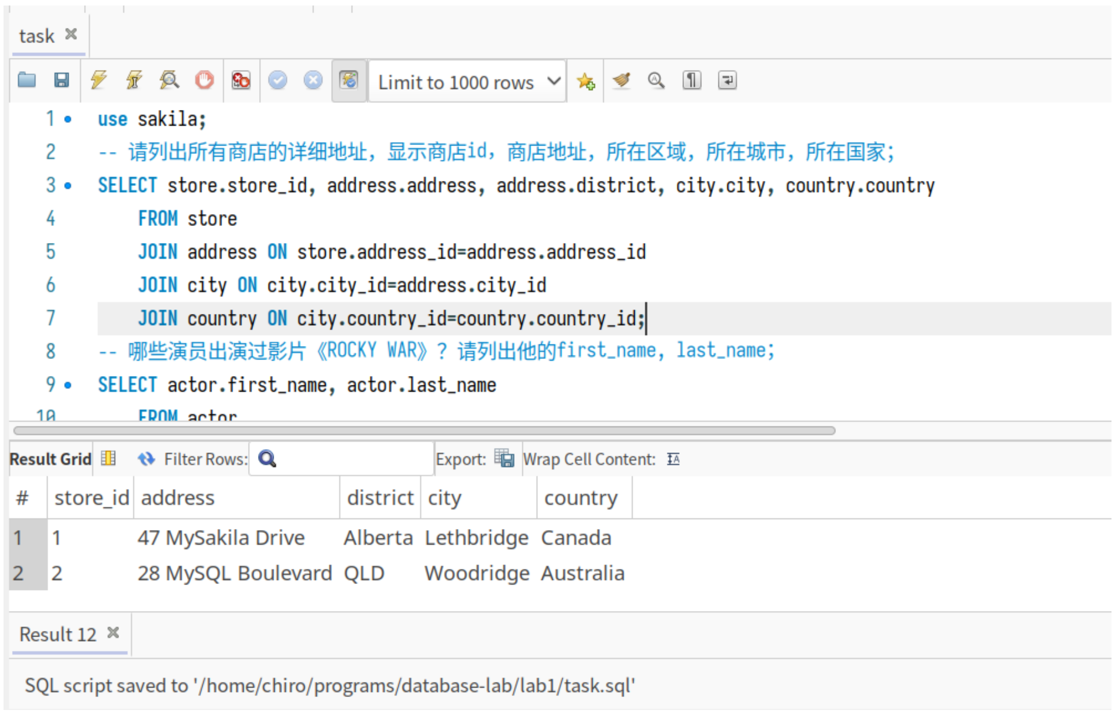


film 表中引用了两个外键：language_id、original_language_id，设置的外键都来自于表 languague 的主键 language_id

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；



```
use sakila;
-- 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.store_id, address.address, address.district, city.city, country.country
    FROM store
    JOIN address ON store.address_id=address.address_id
    JOIN city ON city.city_id=address.city_id
    JOIN country ON city.country_id=country.country_id;
-- 哪些演员出演过影片《ROCKY WAR》？请列出他的first_name, last_name;
SELECT actor.first_name, actor.last_name
    FROM actor
```

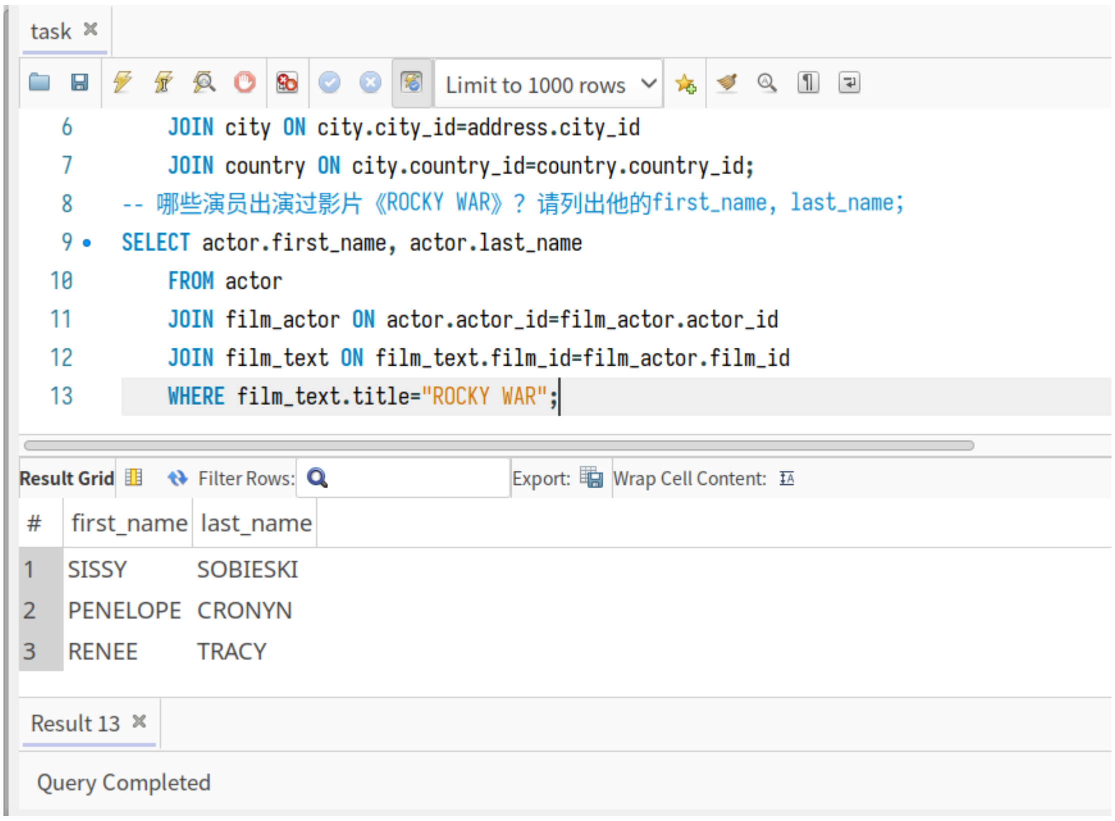
#	store_id	address	district	city	country
1	1	47 MySakila Drive	Alberta	Lethbridge	Canada
2	2	28 MySQL Boulevard	QLD	Woodridge	Australia

SQL script saved to '/home/chiro/programs/database-lab/lab1/task.sql'

```
SELECT store.store_id, address.address, address.district, city.city, country.country
FROM store
JOIN address ON store.address_id=address.address_id
JOIN city ON city.city_id=address.city_id
JOIN country ON 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 actor.first_name, actor.last_name
FROM actor
JOIN film_actor ON actor.actor_id=film_actor.actor_id
JOIN film_text ON film_text.film_id=film_actor.film_id
WHERE film_text.title="ROCKY WAR";
```



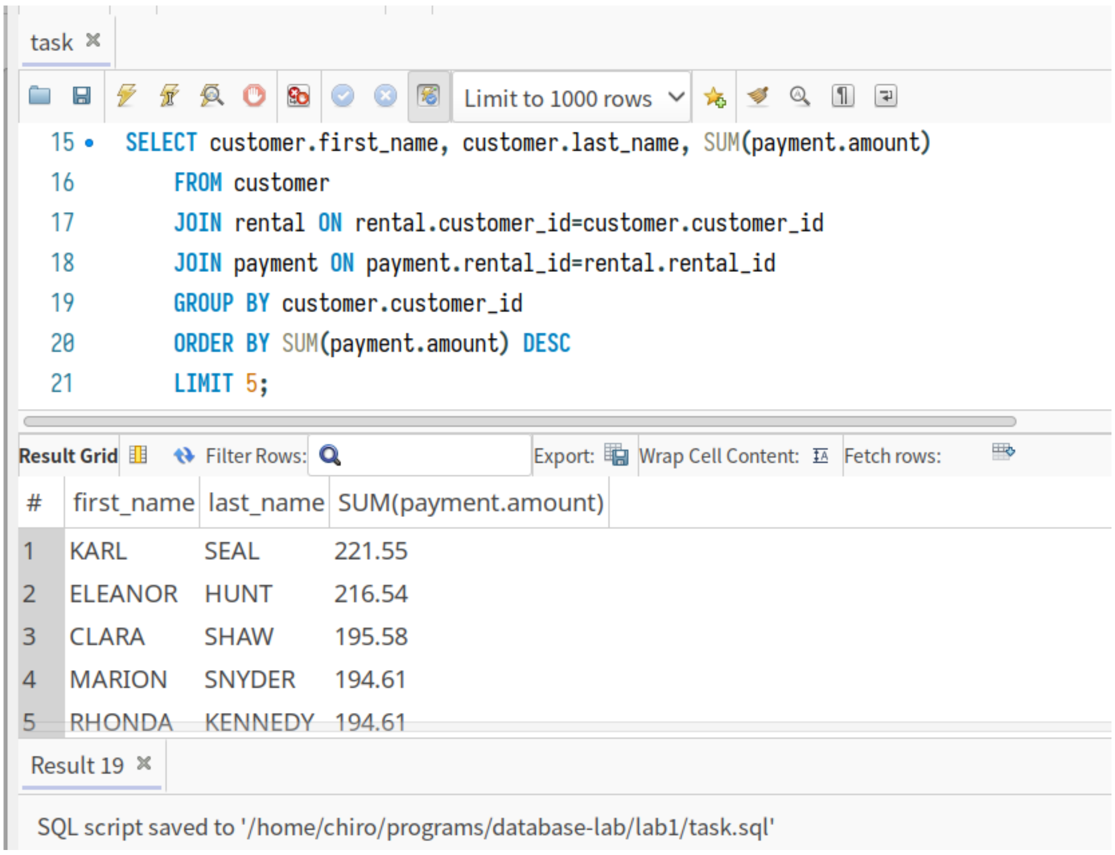
```sql
    JOIN city ON city.city_id=address.city_id
    JOIN country ON city.country_id=country.country_id;
-- 哪些演员出演过影片《ROCKY WAR》？请列出他的first_name, last_name;
SELECT actor.first_name, actor.last_name
    FROM actor
    JOIN film_actor ON actor.actor_id=film_actor.actor_id
    JOIN film_text ON film_text.film_id=film_actor.film_id
    WHERE film_text.title="ROCKY WAR";
```

```
#  first_name  last_name
1  SISSY       SOBIESKI
2  PENELOPE    CRONYN
3  RENEE       TRACY
```

Query Completed

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；

```sql
SELECT customer.first_name, customer.last_name, SUM(payment.amount)
FROM customer
JOIN rental ON rental.customer_id=customer.customer_id
JOIN payment ON payment.rental_id=rental.rental_id
GROUP BY customer.customer_id
ORDER BY SUM(payment.amount) DESC
LIMIT 5;
```


task

Limit to 1000 rows

```
SELECT customer.first_name, customer.last_name, SUM(payment.amount)
    FROM customer
    JOIN rental ON rental.customer_id=customer.customer_id
    JOIN payment ON payment.rental_id=rental.rental_id
    GROUP BY customer.customer_id
    ORDER BY SUM(payment.amount) DESC
    LIMIT 5;
```

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

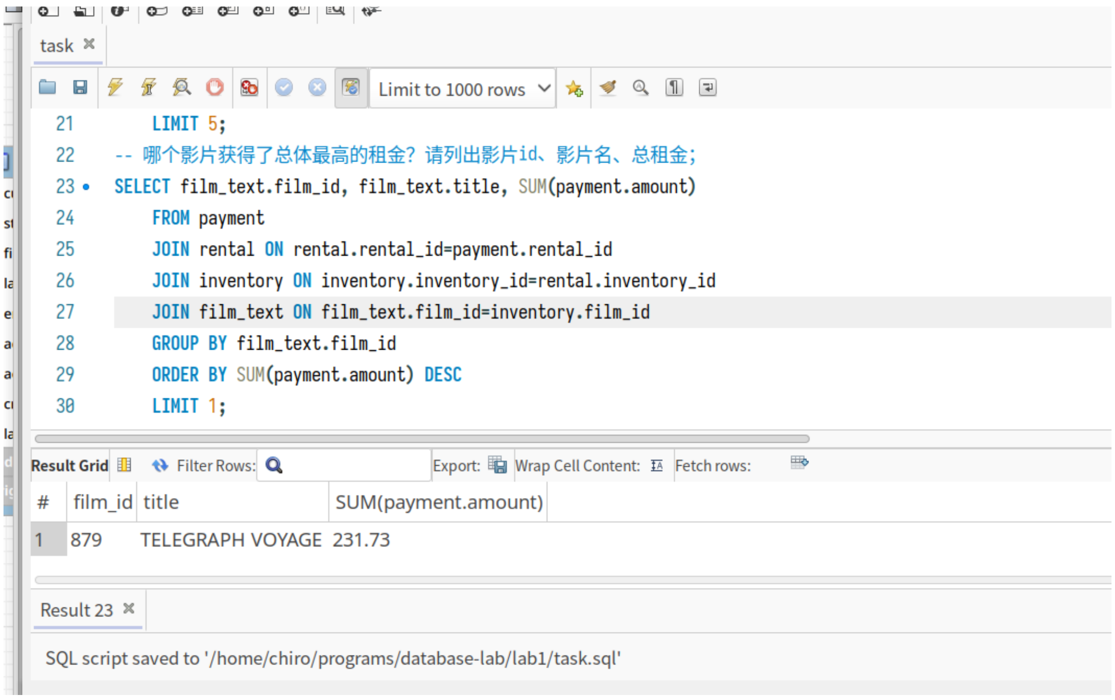
#	first_name	last_name	SUM(payment.amount)
1	KARL	SEAL	221.55
2	ELEANOR	HUNT	216.54
3	CLARA	SHAW	195.58
4	MARION	SNYDER	194.61
5	RHONDA	KENNEDY	194.61

Result 19

SQL script saved to '/home/chiro/programs/database-lab/lab1/task.sql'

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_text.film_id, film_text.title, SUM(payment.amount)
FROM payment
JOIN rental ON rental.rental_id=payment.rental_id
JOIN inventory ON inventory.inventory_id=rental.inventory_id
JOIN film_text ON film_text.film_id=inventory.film_id
GROUP BY film_text.film_id
ORDER BY SUM(payment.amount) DESC
LIMIT 1;
```



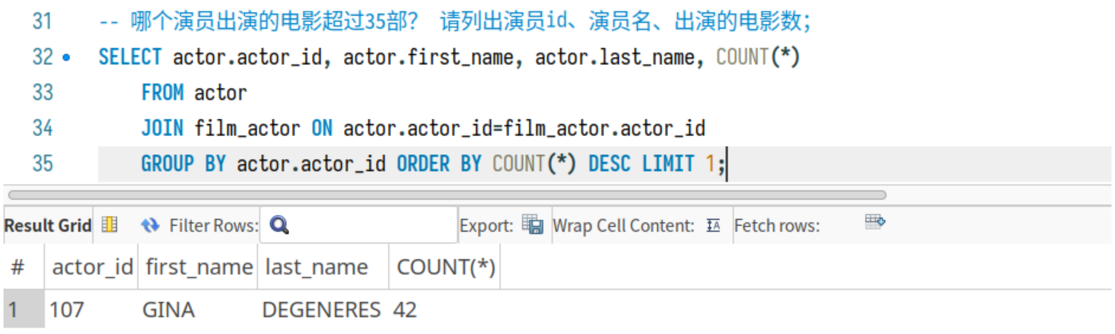
```
    LIMIT 5;
-- 哪个影片获得了总体最高的租金？请列出影片id、影片名、总租金；
SELECT film_text.film_id, film_text.title, SUM(payment.amount)
    FROM payment
    JOIN rental ON rental.rental_id=payment.rental_id
    JOIN inventory ON inventory.inventory_id=rental.inventory_id
    JOIN film_text ON film_text.film_id=inventory.film_id
    GROUP BY film_text.film_id
    ORDER BY SUM(payment.amount) DESC
    LIMIT 1;
```

```
#  film_id  title             SUM(payment.amount)
1  879      TELEGRAPH VOYAGE  231.73
```

```
SQL script saved to '/home/chiro/programs/database-lab/lab1/task.sql'
```

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, actor.first_name, actor.last_name, COUNT(*)
FROM actor
JOIN film_actor ON actor.actor_id=film_actor.actor_id
GROUP BY actor.actor_id ORDER BY COUNT(*) DESC LIMIT 1;
```

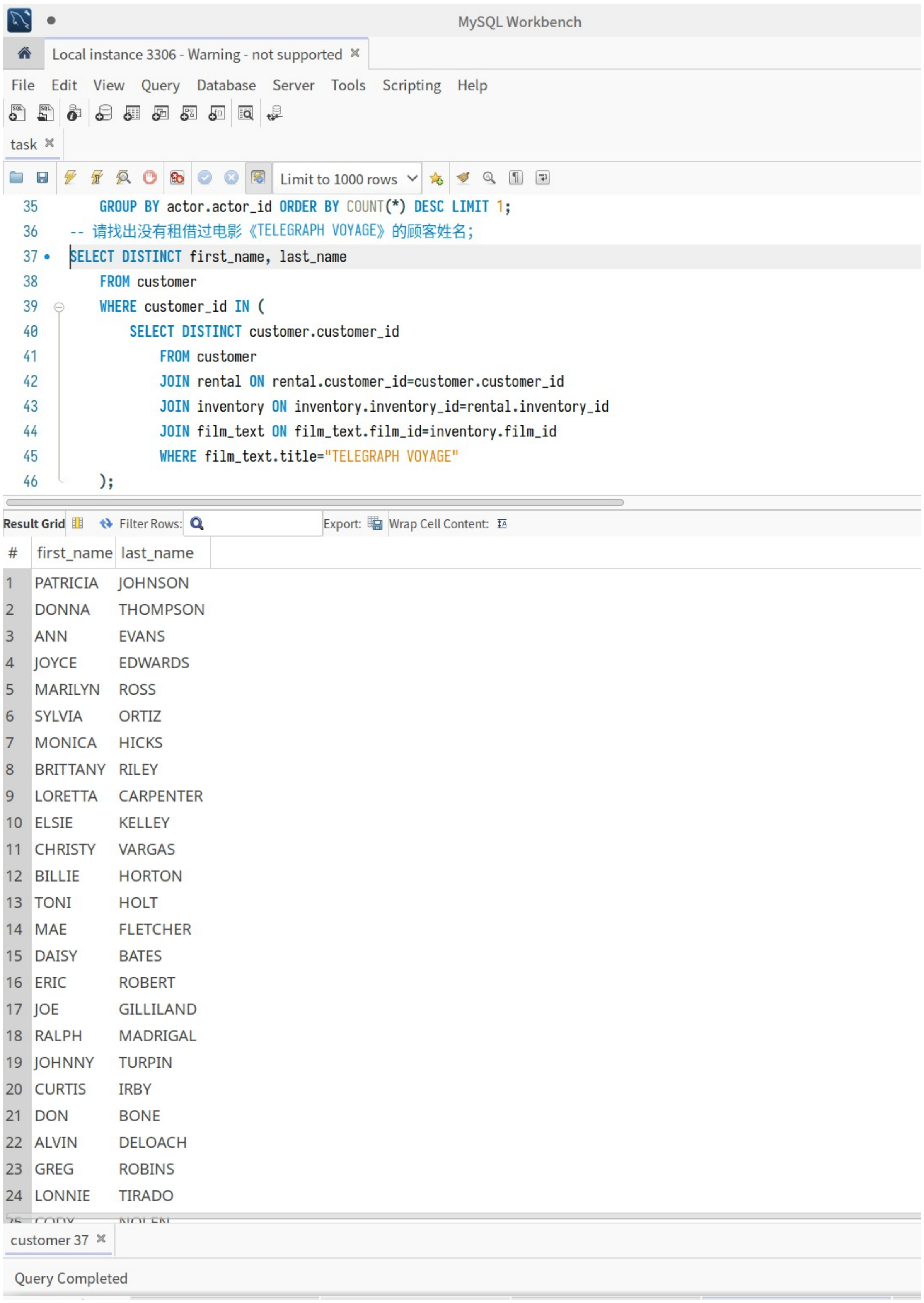


```
-- 哪个演员出演的电影超过35部？ 请列出演员id、演员名、出演的电影数；
SELECT actor.actor_id, actor.first_name, actor.last_name, COUNT(*)
    FROM actor
    JOIN film_actor ON actor.actor_id=film_actor.actor_id
    GROUP BY actor.actor_id ORDER BY COUNT(*) DESC LIMIT 1;
```

```
#  actor_id  first_name  last_name  COUNT(*)
1  107       GINA        DEGENERES  42
```

6、　　请找出没有租借过电影《TELEGRAPH VOYAGE》的顾客姓名；

```
SELECT DISTINCT first_name, last_name
FROM customer
```



```
    GROUP BY actor.actor_id ORDER BY COUNT(*) DESC LIMIT 1;
-- 请找出没有租借过电影《TELEGRAPH VOYAGE》的顾客姓名;
SELECT DISTINCT first_name, last_name
    FROM customer
    WHERE customer_id IN (
        SELECT DISTINCT customer.customer_id
            FROM customer
            JOIN rental ON rental.customer_id=customer.customer_id
            JOIN inventory ON inventory.inventory_id=rental.inventory_id
            JOIN film_text ON film_text.film_id=inventory.film_id
            WHERE film_text.title="TELEGRAPH VOYAGE"
    );
```

#	first_name	last_name
1	PATRICIA	JOHNSON
2	DONNA	THOMPSON
3	ANN	EVANS
4	JOYCE	EDWARDS
5	MARILYN	ROSS
6	SYLVIA	ORTIZ
7	MONICA	HICKS
8	BRITTANY	RILEY
9	LORETTA	CARPENTER
10	ELSIE	KELLEY
11	CHRISTY	VARGAS
12	BILLIE	HORTON
13	TONI	HOLT
14	MAE	FLETCHER
15	DAISY	BATES
16	ERIC	ROBERT
17	JOE	GILLILAND
18	RALPH	MADRIGAL
19	JOHNNY	TURPIN
20	CURTIS	IRBY
21	DON	BONE
22	ALVIN	DELOACH
23	GREG	ROBINS
24	LONNIE	TIRADO
25	CODY	NOLEN

customer 37

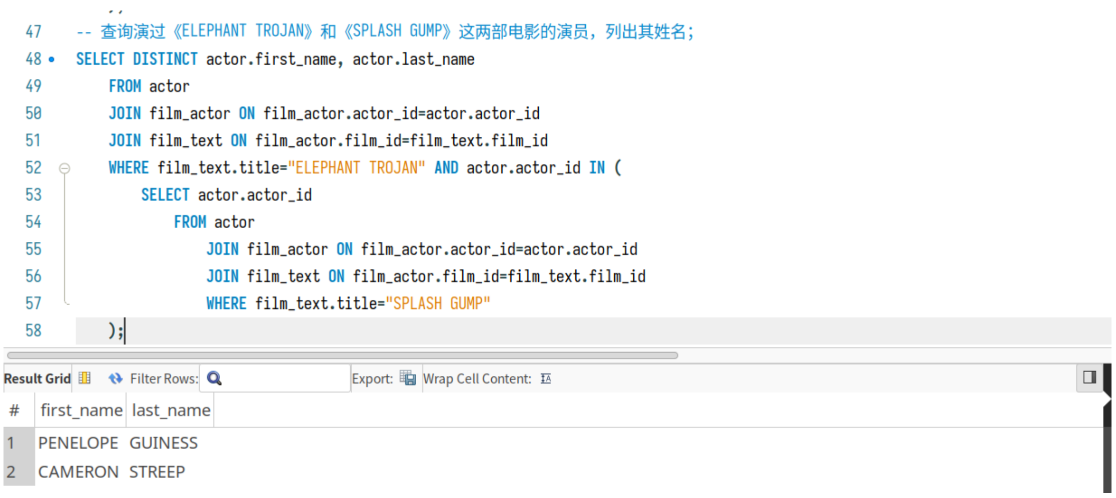
Query Completed

MySQL Workbench
Local instance 3306 - Warning - not supported
File Edit View Query Database Server Tools Scripting Help
task
Limit to 1000 rows
Result Grid
Filter Rows:
Export:
Wrap Cell Content:

```
WHERE customer_id IN (
SELECT DISTINCT customer.customer_id
FROM customer
JOIN rental ON rental.customer_id=customer.customer_id
JOIN inventory ON inventory.inventory_id=rental.inventory_id
JOIN film_text ON film_text.film_id=inventory.film_id
WHERE film_text.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 actor.first_name, actor.last_name
FROM actor
JOIN film_actor ON film_actor.actor_id=actor.actor_id
JOIN film_text ON film_actor.film_id=film_text.film_id
WHERE film_text.title="ELEPHANT TROJAN" AND actor.actor_id IN (
SELECT actor.actor_id
FROM actor
JOIN film_actor ON film_actor.actor_id=actor.actor_id
JOIN film_text ON film_actor.film_id=film_text.film_id
WHERE film_text.title="SPLASH GUMP"
);
```



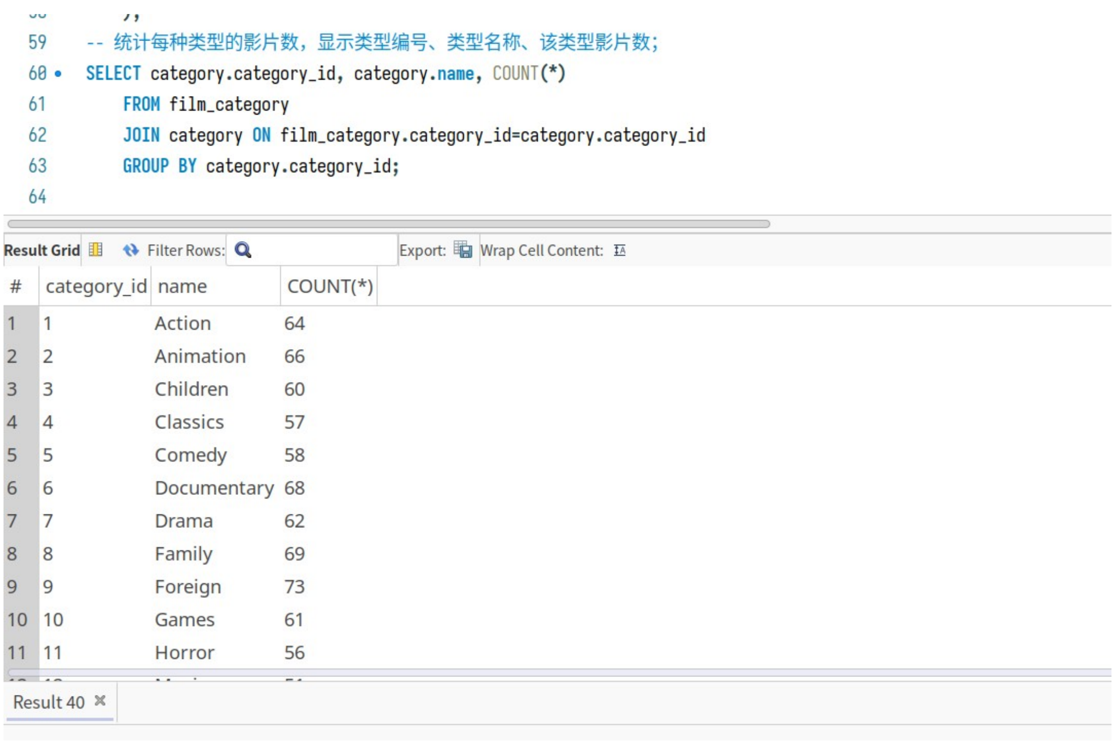
```
-- 查询演过《ELEPHANT TROJAN》和《SPLASH GUMP》这两部电影的演员，列出其姓名；
SELECT DISTINCT actor.first_name, actor.last_name
    FROM actor
    JOIN film_actor ON film_actor.actor_id=actor.actor_id
    JOIN film_text ON film_actor.film_id=film_text.film_id
    WHERE film_text.title="ELEPHANT TROJAN" AND actor.actor_id IN (
        SELECT actor.actor_id
            FROM actor
                JOIN film_actor ON film_actor.actor_id=actor.actor_id
                JOIN film_text ON film_actor.film_id=film_text.film_id
                WHERE film_text.title="SPLASH GUMP"
    );
```

#	first_name	last_name
1	PENELOPE	GUINESS
2	CAMERON	STREEP

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
```

```
FROM film_category
JOIN category ON film_category.category_id=category.category_id
GROUP BY category.category_id;
```

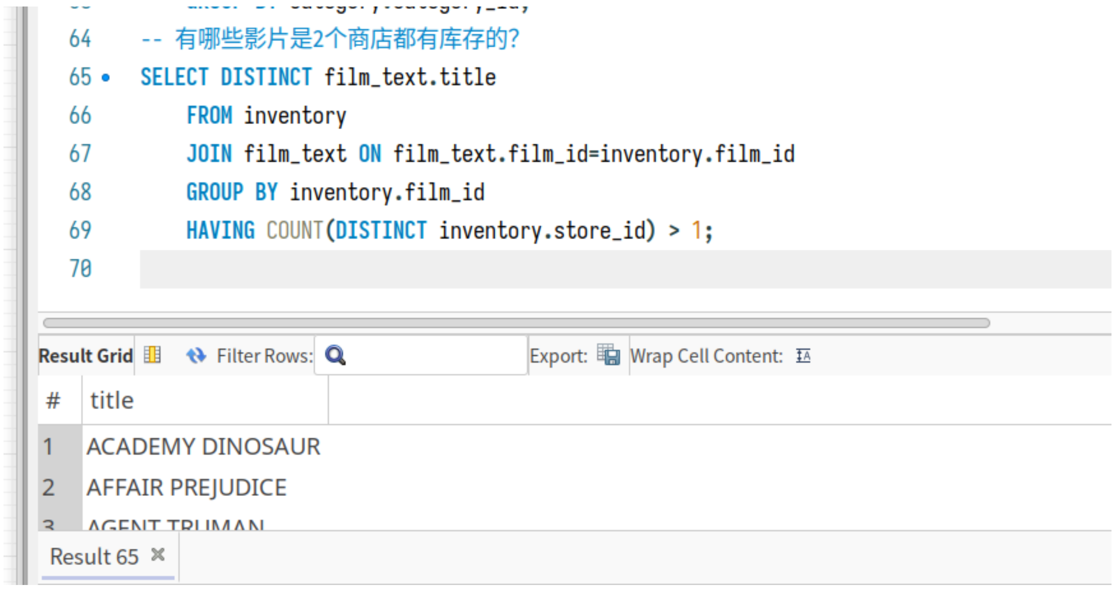


```
-- 统计每种类型的影片数，显示类型编号、类型名称、该类型影片数；
SELECT category.category_id, category.name, COUNT(*)
    FROM film_category
    JOIN category ON film_category.category_id=category.category_id
    GROUP BY category.category_id;
```

#	category_id	name	COUNT(*)
1	1	Action	64
2	2	Animation	66
3	3	Children	60
4	4	Classics	57
5	5	Comedy	58
6	6	Documentary	68
7	7	Drama	62
8	8	Family	69
9	9	Foreign	73
10	10	Games	61
11	11	Horror	56

9、 有哪些影片是 2 个商店都有库存的？

```
SELECT DISTINCT film_text.title
FROM inventory
JOIN film_text ON film_text.film_id=inventory.film_id
GROUP BY inventory.film_id
HAVING COUNT(DISTINCT inventory.store_id) > 1;
```

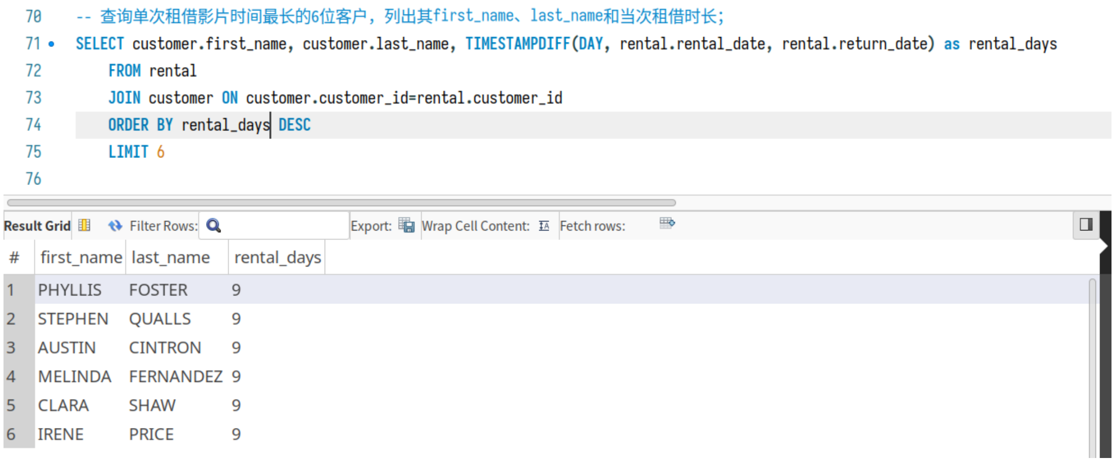


```
-- 有哪些影片是2个商店都有库存的?
SELECT DISTINCT film_text.title
    FROM inventory
    JOIN film_text ON film_text.film_id=inventory.film_id
    GROUP BY inventory.film_id
    HAVING COUNT(DISTINCT inventory.store_id) > 1;
```

#	title
1	ACADEMY DINOSAUR
2	AFFAIR PREJUDICE
3	AGENT TRUMAN

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  customer.first_name,  customer.last_name,  TIMESTAMPDIFF(DAY,  rental.rental_date,
rental.return_date) as rental_days
FROM rental
JOIN customer ON customer.customer_id=rental.customer_id
ORDER BY rental_days DESC
LIMIT 6;
```

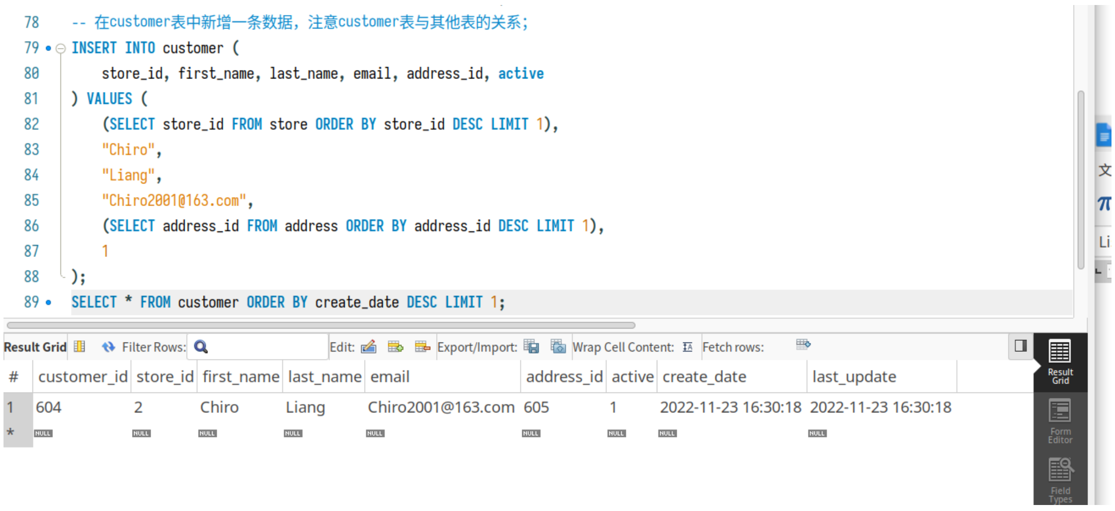


```
-- 查询单次租借影片时间最长的6位客户，列出其first_name、last_name和当次租借时长;
SELECT customer.first_name, customer.last_name, TIMESTAMPDIFF(DAY, rental.rental_date, rental.return_date) as rental_days
    FROM rental
    JOIN customer ON customer.customer_id=rental.customer_id
    ORDER BY rental_days DESC
    LIMIT 6
```

#	first_name	last_name	rental_days
1	PHYLLIS	FOSTER	9
2	STEPHEN	QUALLS	9
3	AUSTIN	CINTRON	9
4	MELINDA	FERNANDEZ	9
5	CLARA	SHAW	9
6	IRENE	PRICE	9

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

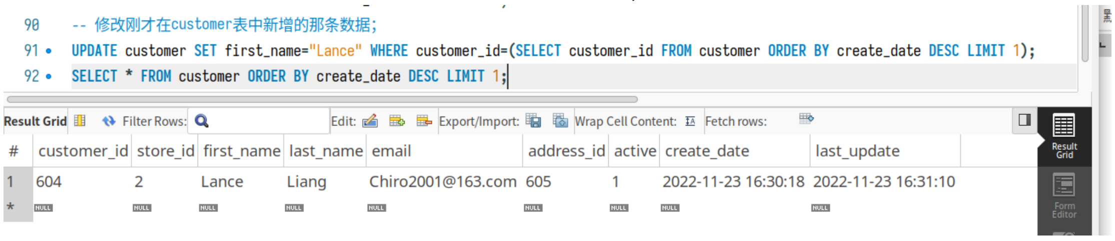
```
UPDATE customer SET first_name="Lance" WHERE customer_id=(SELECT customer_id FROM customer ORDER BY create_date DESC LIMIT 1);
SELECT * FROM customer ORDER BY create_date DESC LIMIT 1;
```



```
-- 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, email, address_id, active
) VALUES (
    (SELECT store_id FROM store ORDER BY store_id DESC LIMIT 1),
    "Chiro",
    "Liang",
    "Chiro2001@163.com",
    (SELECT address_id FROM address ORDER BY address_id DESC LIMIT 1),
    1
);
SELECT * FROM customer ORDER BY create_date DESC LIMIT 1;
```

#	customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
1	604	2	Chiro	Liang	Chiro2001@163.com	605	1	2022-11-23 16:30:18	2022-11-23 16:30:18
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

12、 修改刚才在 customer 表中新增的那条数据;



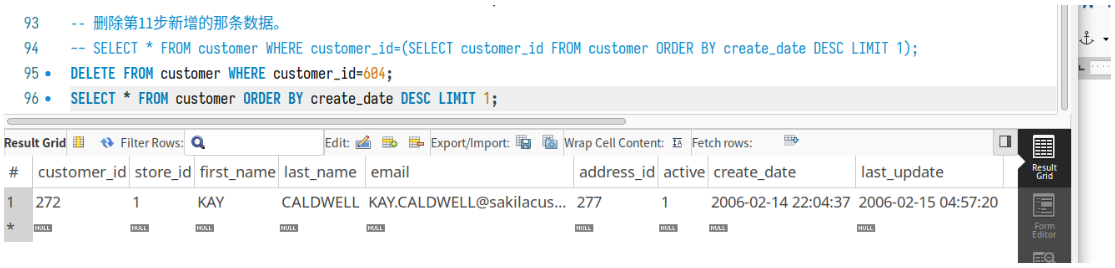
```
-- 修改刚才在customer表中新增的那条数据;
UPDATE customer SET first_name="Lance" WHERE customer_id=(SELECT customer_id FROM customer ORDER BY create_date DESC LIMIT 1);
SELECT * FROM customer ORDER BY create_date DESC LIMIT 1;
```

#	customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
1	604	2	Lance	Liang	Chiro2001@163.com	605	1	2022-11-23 16:30:18	2022-11-23 16:31:10
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
UPDATE customer SET first_name="Lance" WHERE customer_id=(SELECT customer_id FROM customer ORDER BY create_date DESC LIMIT 1);
SELECT * FROM customer ORDER BY create_date DESC LIMIT 1;
```

13、 删除第 11 步新增的那条数据。

```
DELETE FROM customer WHERE customer_id=604;
SELECT * FROM customer ORDER BY create_date DESC LIMIT 1;
```



```
-- 删除第11步新增的那条数据。
-- SELECT * FROM customer WHERE customer_id=(SELECT customer_id FROM customer ORDER BY create_date DESC LIMIT 1);
DELETE FROM customer WHERE customer_id=604;
SELECT * FROM customer ORDER BY create_date DESC LIMIT 1;
```

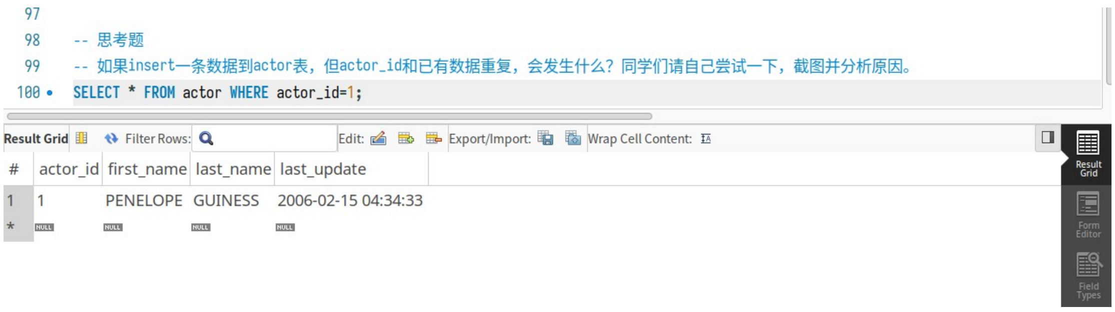
#	customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
1	272	1	KAY	CALDWELL	KAY.CALDWELL@sakilacus...	277	1	2006-02-14 22:04:37	2006-02-15 04:57:20
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

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。

```
SELECT * FROM actor WHERE actor_id=1;
INSERT INTO actor (actor_id, first_name, last_name) VALUES (1, "Chiro", "Liang");
SELECT * FROM actor WHERE actor_id=1;
```

首先查询一下原数据：



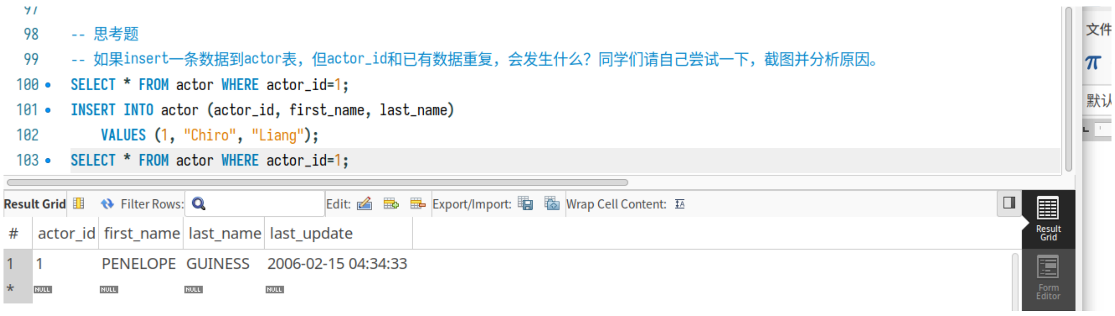
```

-- 思考题
-- 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。
SELECT * FROM actor WHERE actor_id=1;
```

#	actor_id	first_name	last_name	last_update
1	1	PENELOPE	GUINESS	2006-02-15 04:34:33
*	NULL	NULL	NULL	NULL

在执行插入语句的时候 SQL 提示语句执行有错误，Query interrupted.

执行插入后再查询：

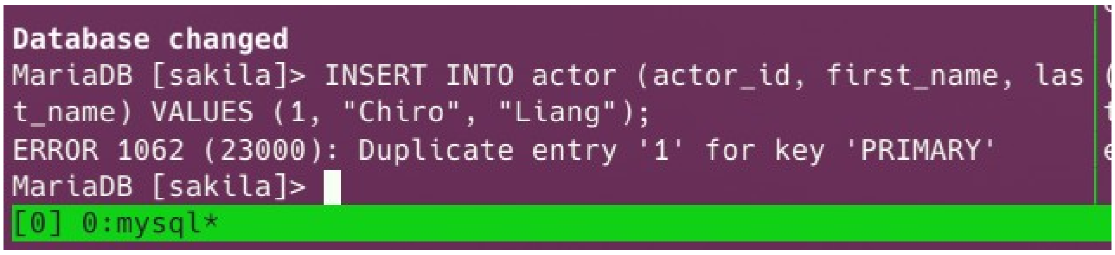


```sql
-- 思考题
-- 如果insert一条数据到actor表，但actor_id和已有数据重复，会发生什么？同学们请自己尝试一下，截图并分析原因。
SELECT * FROM actor WHERE actor_id=1;
INSERT INTO actor (actor_id, first_name, last_name)
    VALUES (1, "Chiro", "Liang");
SELECT * FROM actor WHERE actor_id=1;
```

```
#  actor_id  first_name  last_name  last_update
1  1         PENELOPE    GUINESS    2006-02-15 04:34:33
*  NULL      NULL        NULL       NULL
```

查询内容不变。

具体报错为：



```
Database changed
MariaDB [sakila]> INSERT INTO actor (actor_id, first_name, las
t_name) VALUES (1, "Chiro", "Liang");
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'
MariaDB [sakila]> 
[0] 0:mysql*
```

即主键冲突时，无法插入相同主键数据。

2) insert 语句还用了一个函数 NOW()，是做什么的呢？

NOW() 函数返回当前日期和时间。因为表中的 create_time 和 update_time 都已经配置过自动添加值，所以不使用 NOW() 函数来填充 update_time 和 create_time 也是可以的。