实验二报告

## 观察并回答问题

### 关于视图

1. sakila.mwb模型图中共有几个View？

7个

1. 分析以下3个视图，回答以下问题：

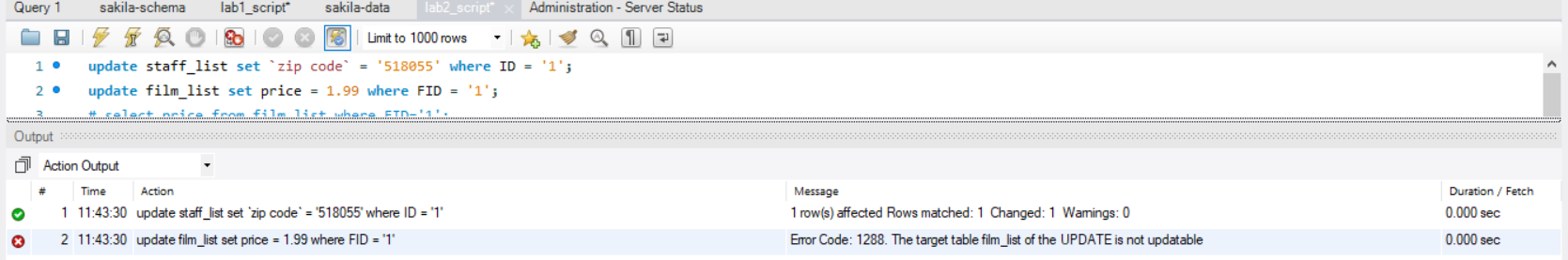
|  |  |  |
| --- | --- | --- |
| 视图名 | 关联表 | 作用 |
| actor\_info | film,film\_actor,actor | 指示演员与电影间的关系 |
| film\_list | film,category,film\_category,film\_actor | 指示电影的类别，租借率，以及电影演员 |
| sales\_by\_store | payment,rental,inventory,store, address, city, country, staff, | 指示一家商店在不同地区的销量以及负责人员 |

1. 分别执行以下2句SQL语句：

update staff\_list set `zip code` = '518055' where ID = '1';

update film\_list set price = 1.99 where FID = '1';

截图执行结果，并分析一下视图在什么情况下可以进行update操作，什么情况下不能？



staff\_list这个表的视图修改的字段只有staff这个表有，而film\_list这个视图的FID在多个表中都有，不能如此更新

1. 执行以下命令查询sakila数据库中的视图是否可更新，截图执行结果：

SELECT table\_name, is\_updatable FROM information\_schema.views

WHERE table\_schema = 'sakila';

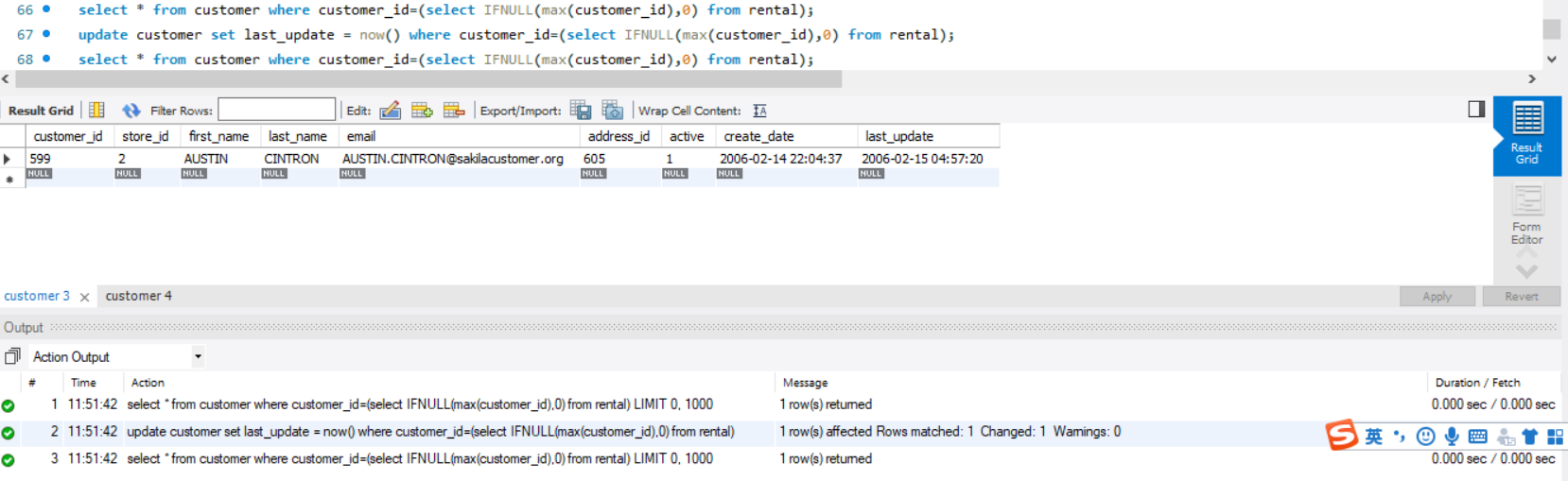


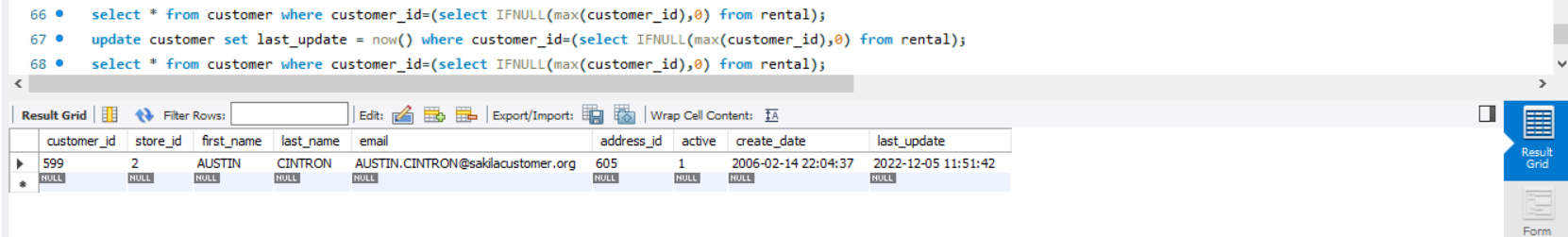
### 关于触发器

1. 触发器customer\_create\_date建在哪个表上？这个触发器实现什么功能？

建在customer表上， 用于默认设置create\_date这个属性

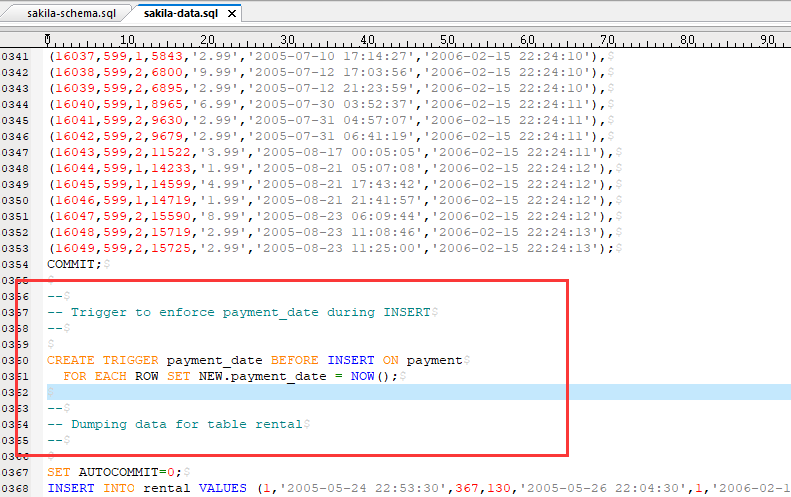
1. 在这个表上新增一条数据，验证一下触发器是否生效。（截图语句和执行结果）





1. 我们可以看到sakila-schema.sql里的语句是用于创建数据库的结构，包括表、视图、触发器等，而sakila-data.sql主要是用于往表写入数据。但sakila-data.sql里有这样一个建立触发器payment\_date的语句，这个触发器是否可以移到sakila-schema.sql里去执行？为什么？

不可以，如果移入schema中，data插入的数据中的create\_date会强制变成插入时的时间



### 关于约束

1. store表上建了哪几种约束？这些约束分别实现什么功能？（至少写3个）

|  |  |
| --- | --- |
| 约束类型 | 功能 |
| 非空 |  |
| 唯一性 | 保证商店id的唯一性 |
| 默认值 | last\_update必须要有一个默认值 |

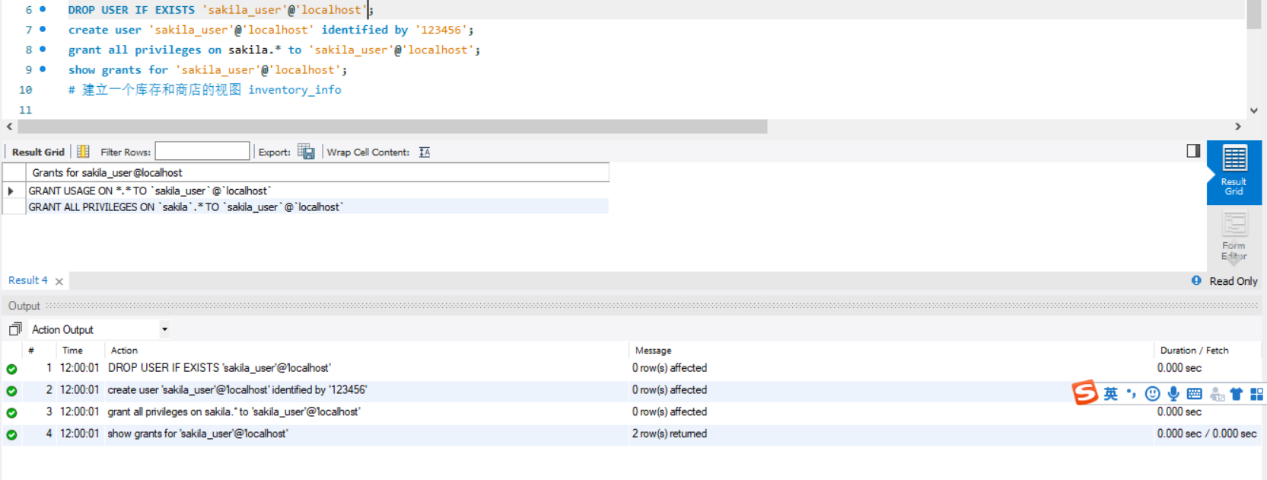
1. 图中第343行的ON DELETE RESTRICT和ON UPDATE CASCADE是什么意思？

只在delete或update时生效的约束

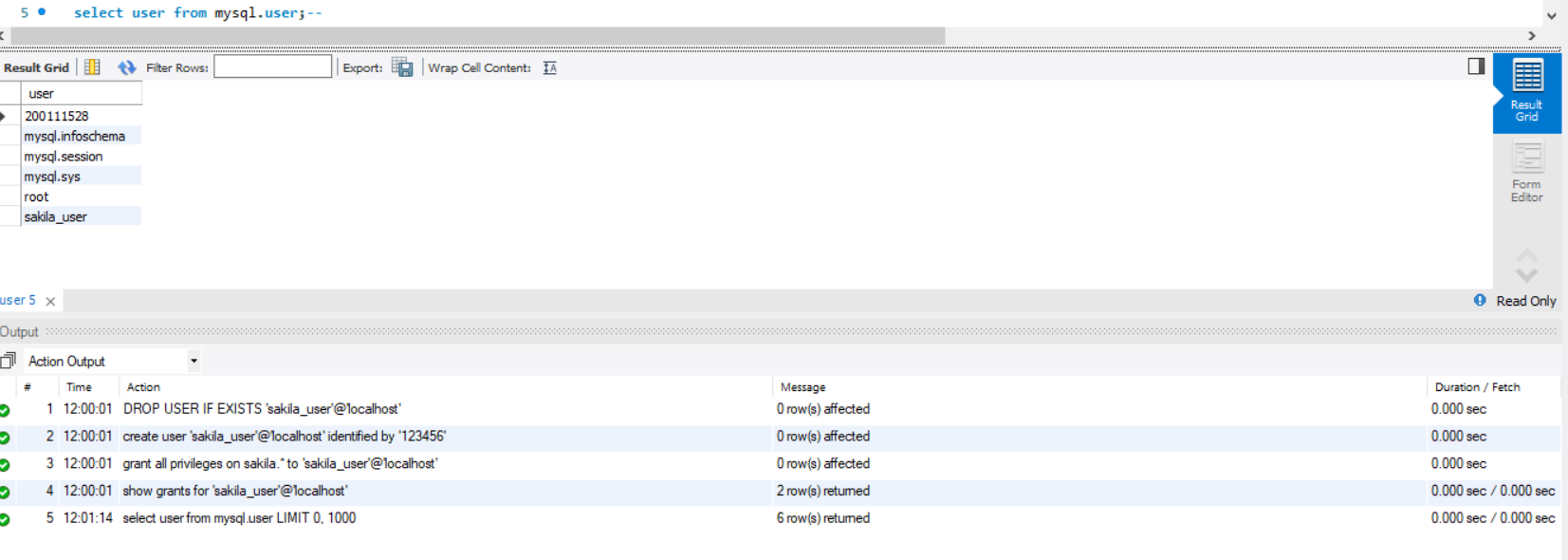
## 创建新用户并分配权限

（截图语句和执行结果）

1. 执行命令新建sakila\_user用户（密码123456）；



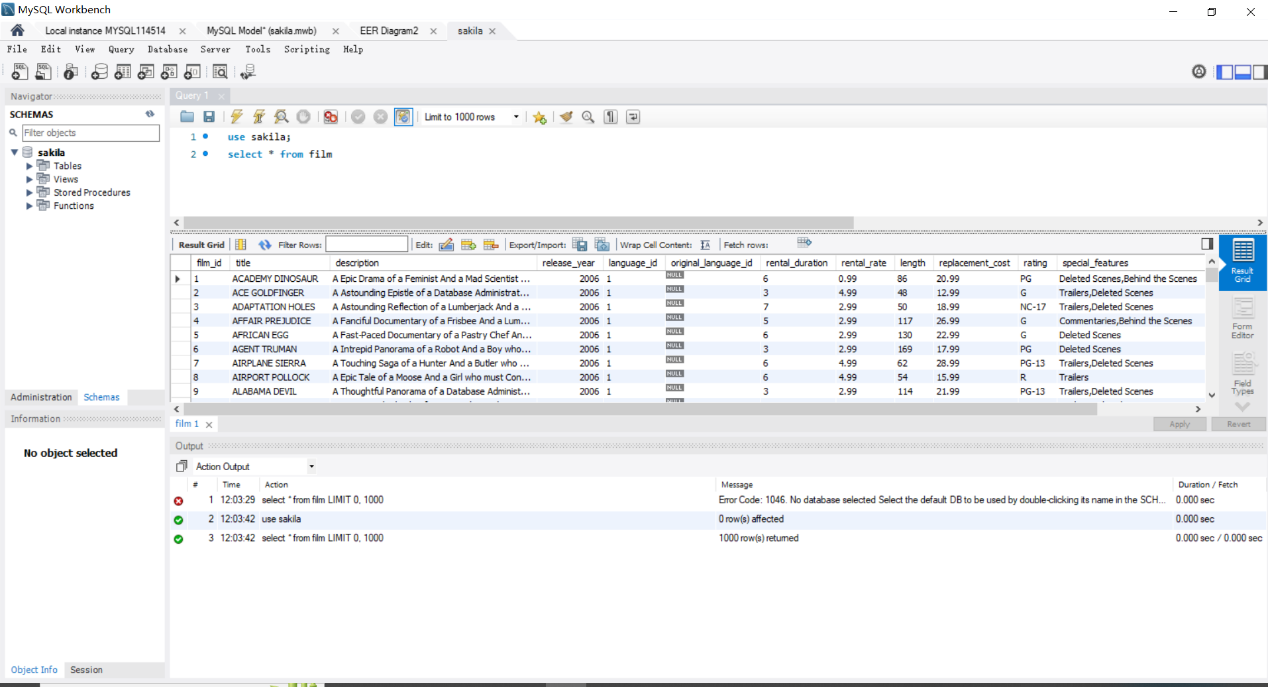
1. 执行命令查看当前已有用户；



1. 执行命令把sakila数据库的访问权限赋予sakila\_user用户；



1. 切换到sakila\_user用户，执行select \* from film操作。



## 设计并实现

根据应用场景，为Sakila数据库合理地设计并实现：

（截图语句和执行结果）

语句：

DROP VIEW IF EXISTS inventory\_info;# -- 支持多次运行 --

CREATE VIEW inventory\_info

AS

SELECT st.store\_id AS SID, inv.inventory\_id AS INVID, f.title as FilmTitle, inv.last\_update as Lastupdate

FROM store AS st JOIN inventory AS inv ON st.store\_id = inv.store\_id JOIN film AS f ON inv.film\_id = f.film\_id;

select \* from inventory\_info;

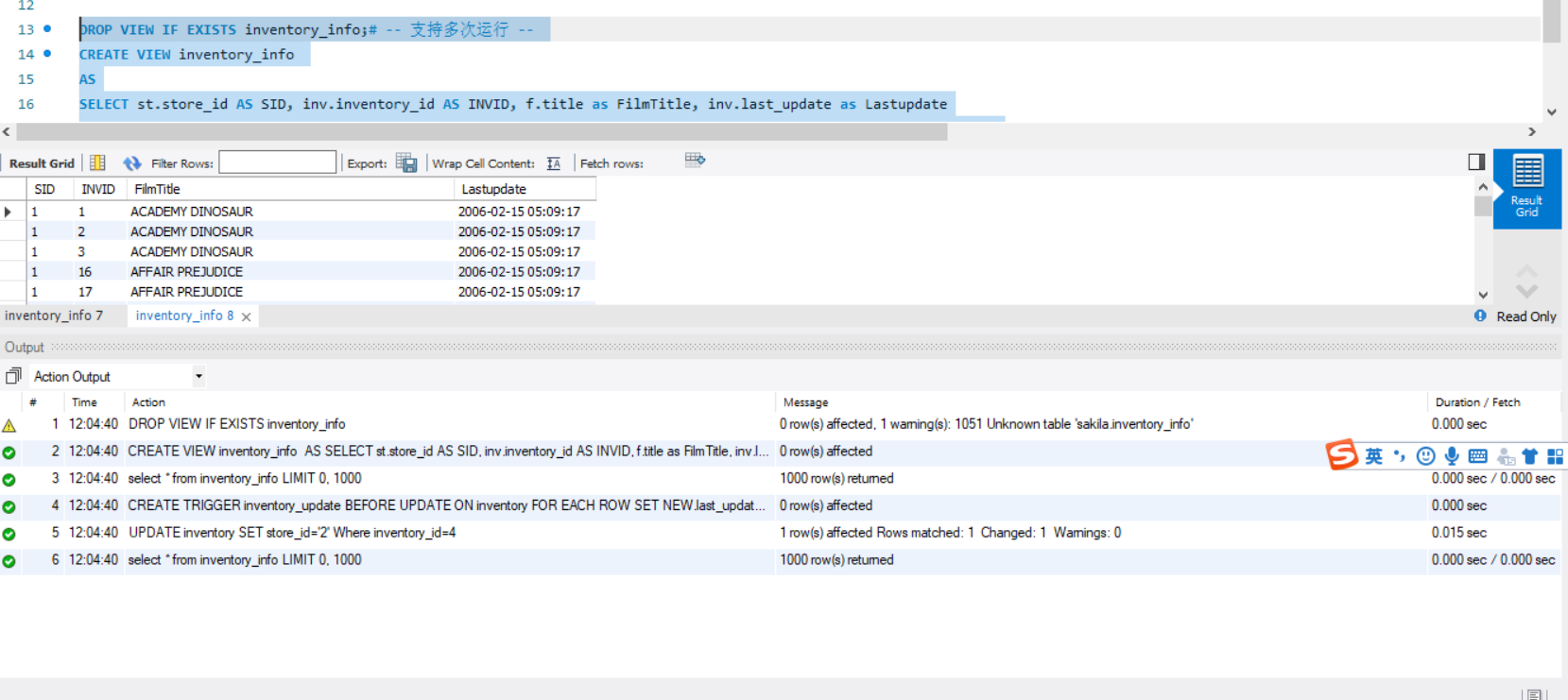
CREATE TRIGGER inventory\_update BEFORE UPDATE ON inventory

FOR EACH ROW SET NEW.last\_update = NOW();

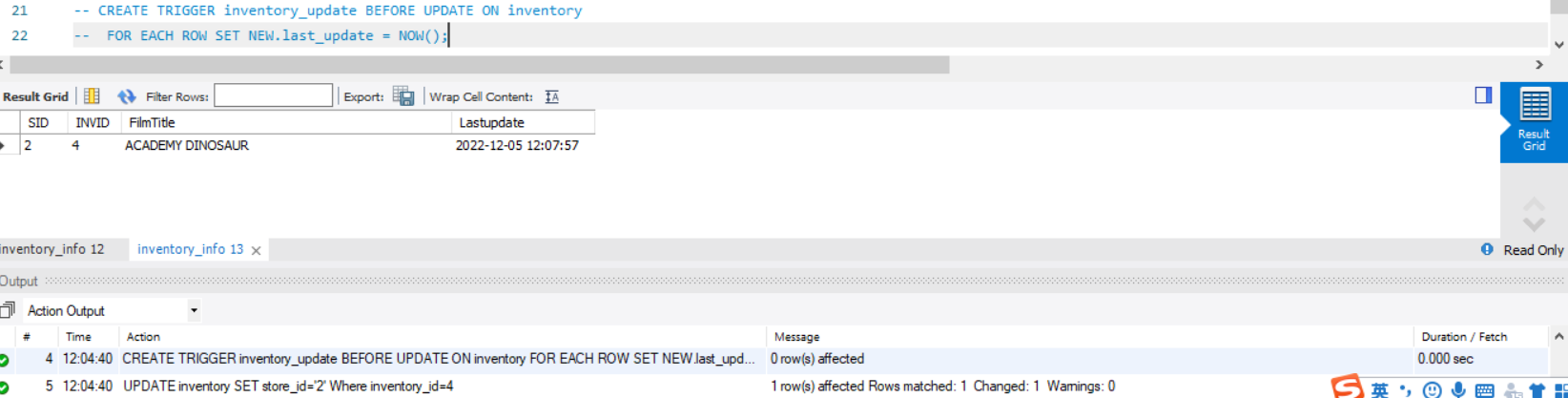
UPDATE inventory SET store\_id='2' Where inventory\_id=4;

select \* from inventory\_info;

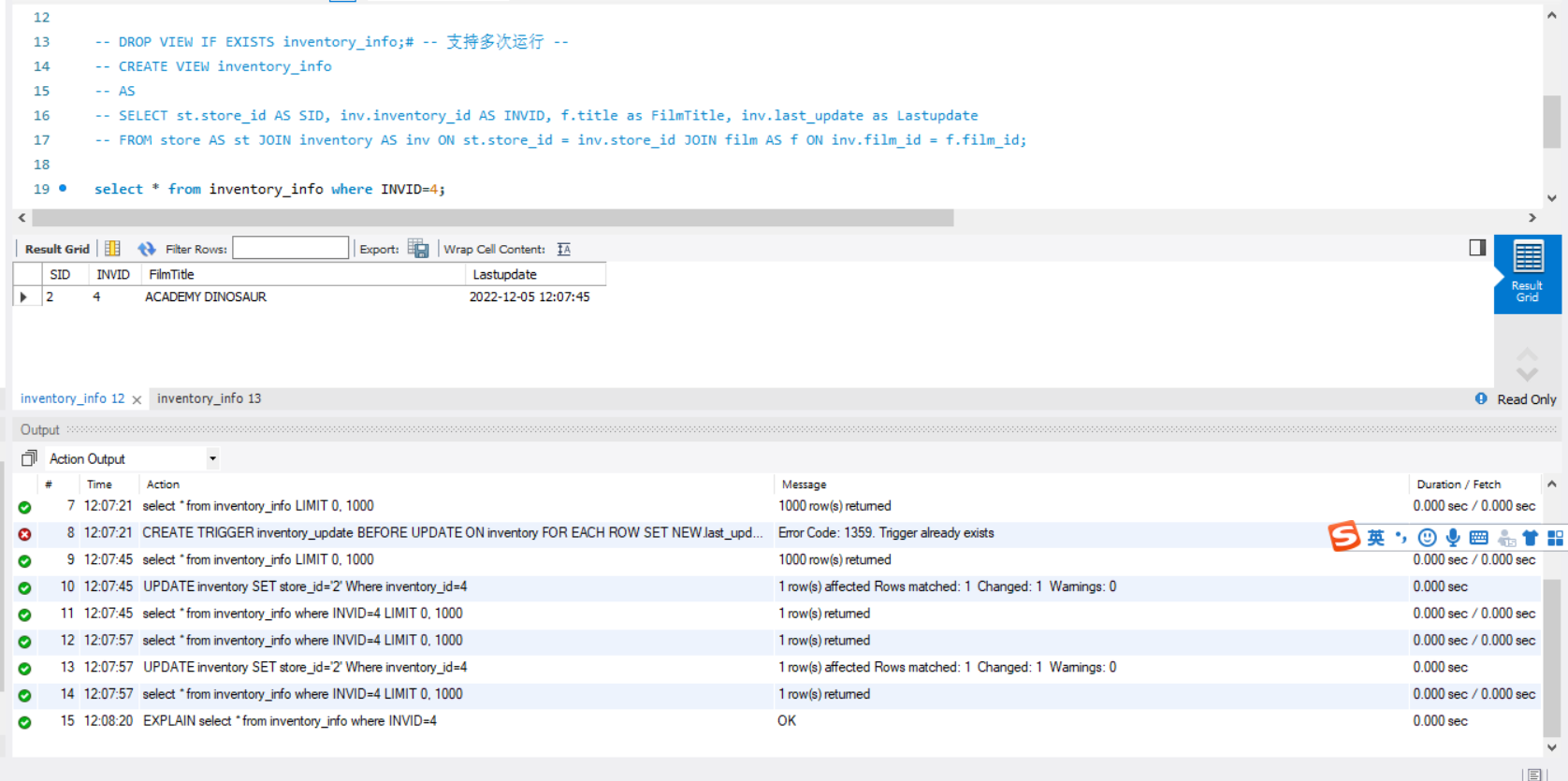
1. 设计1个视图，至少关联2个表；
2. 执行新建视图的语句，并截图SQL和执行结果：



1. 执行select \* from [视图名]，截图执行结果：
2. 设计1个触发器，需要体现触发器生效。
3. 执行新建触发器的语句，并截图SQL和执行结果：



1. 验证触发器是否生效，截图验证过程：





## 思考题

*（这部分不是必做题，供有兴趣的同学思考）*

在阿里开发规范里有一条“**【**强制**】不得使用外键与级联，一切外键概念必须在应用层解决。**”请分析一下原因。你认为外键是否没有存在的必要？

引入外键会引起维护以及添加的难度