实验二报告

一、 观察并回答问题

1. 关于视图

（1） sakila.mwb 模型图中共有几个 View？

一共有 7 个 View。

（2） 分析以下 3 个视图，回答以下问题：

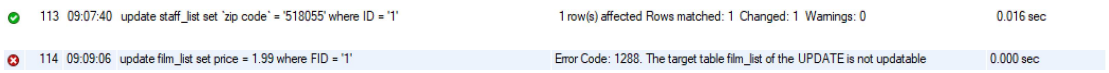
视图名	关联表	作用
actor_info	actor,film_actor,film_category,category	列出所有演员的 id，姓名以及出演的电影的信息
film_list	category,film_category,film, film_actor,actor	列出所有电影的 id，电影名，描述，所属类别，价格，片长，分级和出演的演员
sales_by_film_category	payment,rental,inventory,film, film_category,category	列出所有类别的电影的类别名和总销量

（3） 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？

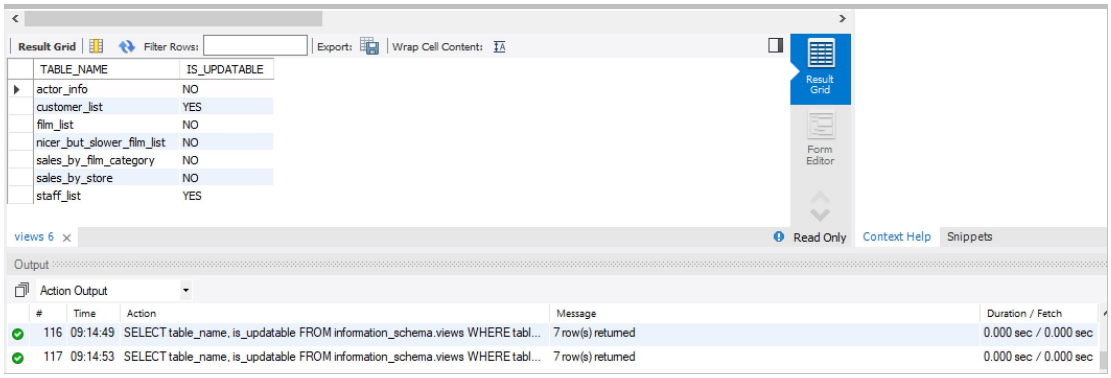
以下为分别为两句的执行结果：



视图在被设置为 updatable 时可以进行 update 操作，在没有被设置为 updatable 时不能。

（4） 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';
```



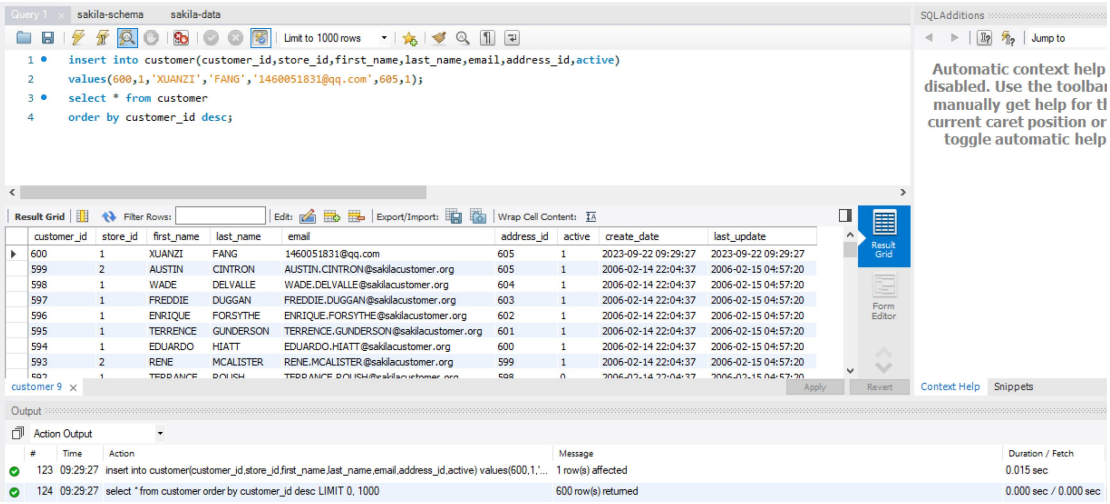
从执行结果来看，customer_list 和 staff_list 可以更新，而其他的则不能进行更新。

2. 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？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）

建在 customer 表上。该触发器实现的功能是在对 customer 表做插入操作之前设置 create_date 为 NOW()。

新增一条 customer 数据之后，通过 select 语句观察结果。

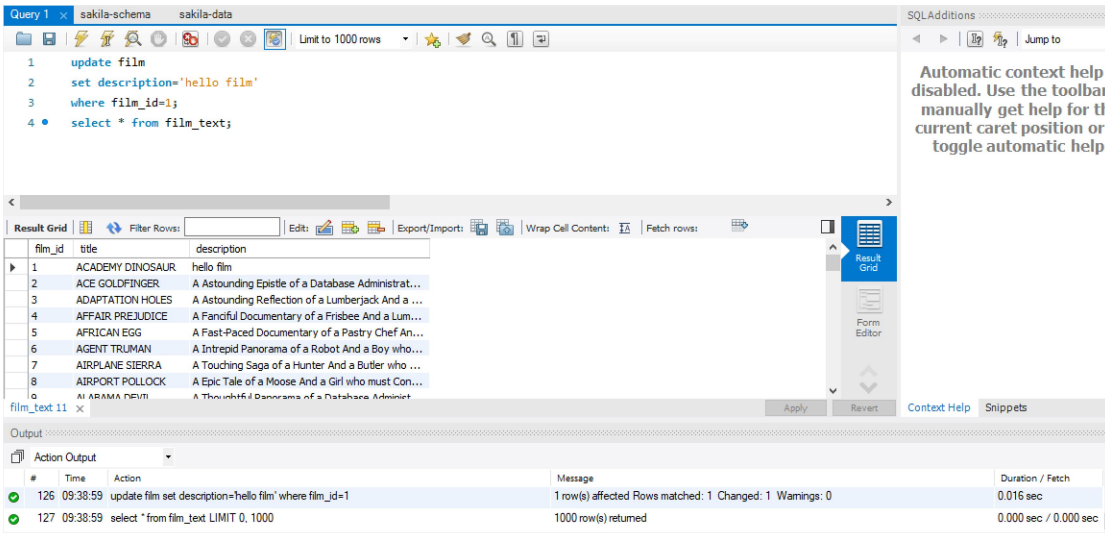


执行结果如下，相应的 create_date 结果正确。

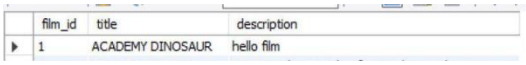
（2） 触发器 upd_film 建在哪个表上？这个触发器实现什么功能？在这个表上修改一条数据的 description 字段，验证一下触发器是否生效。（截图语句和执行结果）

建在 film 表上。该触发器实现的功能是在 film 表更新之后，对 film_text 表中的 title，description，film_id 也做相应的更新，做到与 film 表保持一致。

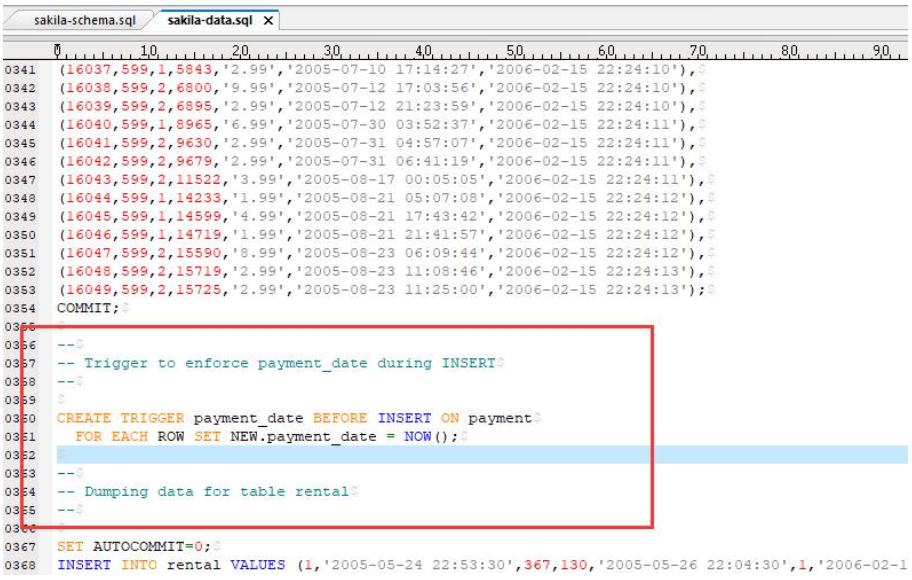
修改 id 为 1 的电影的 description 为'hello film'，再执行 select 语句查看 film_text。



执行结果如下，相应的更新正确。



（3） 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。

原因：这个触发器的功能是在每次插入 payment 表之前将 payment_date 设置为 NOW()。由于先执行 sakila-schema.sql 再执行 sakila-data.sql，如果将这个触发器移到 sakila-schema.sql 里去执行，则 sakila-data.sql 往表写入初始数据时，会使触发器生效，导致写入的所有数据的 payment_date 都变为 NOW()了。

3. 关于约束

观察 sakila-schema.sql 里面的 create table store 一段：

```
335 CREATE TABLE store (
336   store_id TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
337   manager_staff_id TINYINT UNSIGNED NOT NULL,
338   address_id SMALLINT UNSIGNED NOT NULL,
339   last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
340   PRIMARY KEY  (store_id),
341   UNIQUE KEY idx_unique_manager (manager_staff_id),
342   KEY idx_fk_address_id (address_id),
343   CONSTRAINT fk_store_staff FOREIGN KEY (manager_staff_id) REFERENCES staff (staff_id) ON DELETE RESTRICT ON UPDATE CASCADE,
344   CONSTRAINT fk_store_address FOREIGN KEY (address_id) REFERENCES address (address_id) ON DELETE RESTRICT ON UPDATE CASCADE
345 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
346
```

（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
NOT NULL 非空约束	使值不能为空
PRIMARY KEY 主键约束	使值不能为空，且不能重复，具有唯一性
UNIQUE 唯一约束	使值不能重复
FOREIGN KEY 外键约束	在两个表之间建立连接，其中一张表的一个字段被另一张表中对应的字段约束

（2） 图中第 343 行的 ON DELETE RESTRICT 和 ON UPDATE CASCADE 是什么意思？

ON DELETE RESTRICT:在表中删除记录时，先检查它是否有对应的外键，如果有则不执行删除操作。
ON UPDATE CASCADE:在表中更新记录时，先检查它是否有对应的外键，如果有则也更新包含外键的表的相应记录。

4. 关于存储过程

观察 sakila-schema.sql 里面的 rewards_report 存储过程，结合官方文档分析：

（1） 这个存储过程 rewards_report 实现了什么功能？输出参数 count_rewardees 是什么？

实现的功能是根据输入的每月最低购买量 min_monthly_purchases 和购买的最低美元金额 min_dollar_amount_purchased，提供一个有关最佳客户的报告。
输出参数 count_rewardees 是符合要求的客户数量。

（2） 图中第 483 行的 NOT DETERMINISTIC 和第 485 行的 SQL SECURITY DEFINER 分别是什么含义？

NOT DETERMINISTIC:表示该存储过程不是确定性的，给定相同的输入，不一定会返回相同的结果。
SQL SECURITY DEFINER:指定视图查询数据时的安全验证方式，DEFINER 表示按定义者拥有的权限来执行。

5. 关于函数

观察 sakila-schema.sql 里面的 get_customer_balance 函数，结合官方文档分析：

（1） 这个函数 get_customer_balance 实现了什么功能？返回值是什么？

实现的功能是在给定 CUSTOMER_ID 和日期的情况下计算当前余额。
返回值是最初租借视频所支付的费用加上之前租金的滞纳金减去之前支付的款项总额，即当前余额。

（2） 这个函数体中用到了 3 个 Mysql 内置函数，是哪几个函数？这 3 个函数的作用分别是什么？

函数	作用
IFNULL(expr_1,expr_2)	如果 expr_1 不为 NULL，则 IFNULL 函数返回 expr_1 的结果，否则返回 expr_2 的结果。
SUM([DISTINCT] expr)	返回 expr 表达式的和。如果没有返回行数，则返回 null。
TO_DAYS(date)	返回 date 和年份 0（日期 "0000-00-00"）之间的天数。

二、 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_test 用户（密码 123456）；

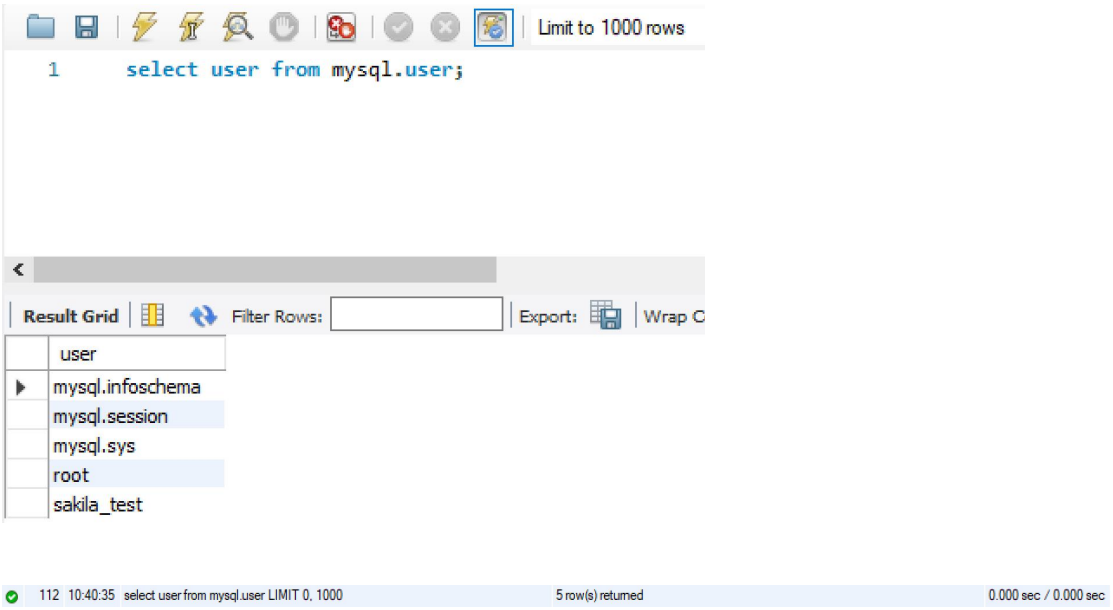
语句：

```
1 •  create user 'sakila_test'@'localhost' identified by '123456';
```

执行结果：

（2） 执行命令查看当前已有用户；

语句和执行结果如下：



（3） 执行命令把 sakila 数据库的访问权限赋予 sakila_test 用户；

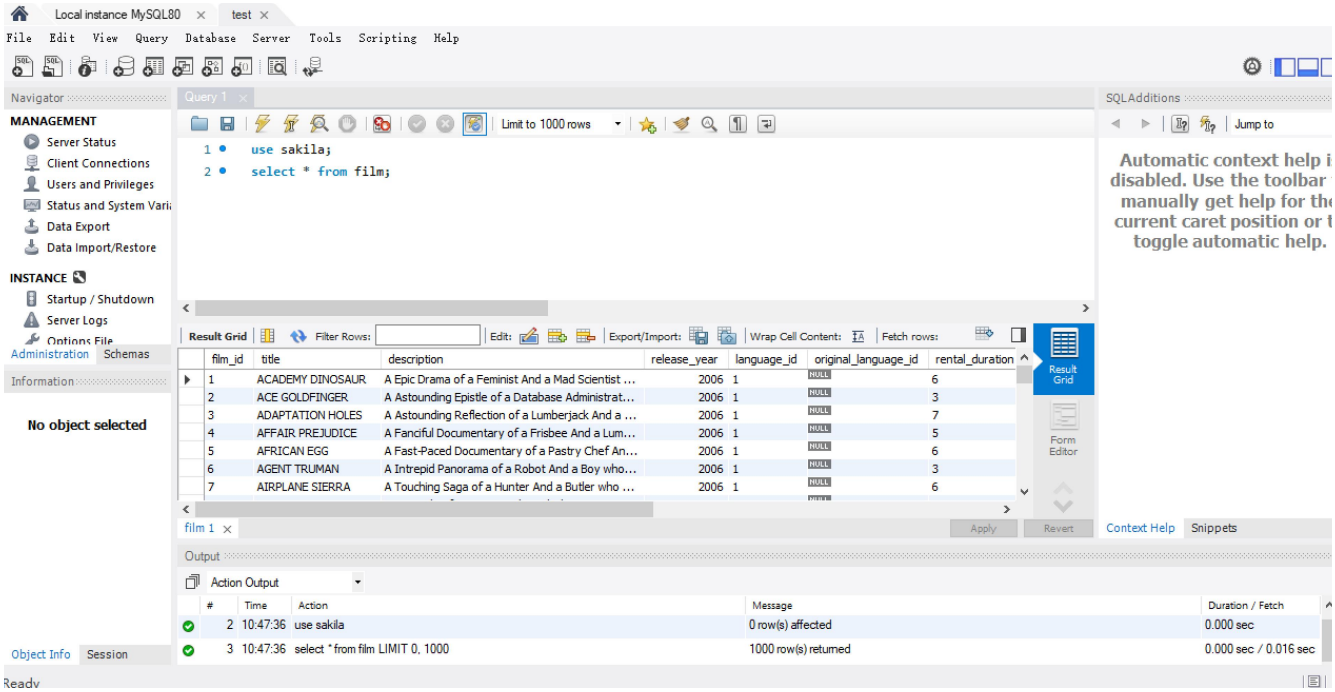
语句：

```
1    grant all privileges on sakila.* to 'sakila_test'@'localhost';
```

执行结果：

（4） 切换到 sakila_test 用户，执行 select * from film 操作。

语句和执行结果如下：



三、 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）

1. 设计 1 个视图，至少关联 2 个表；

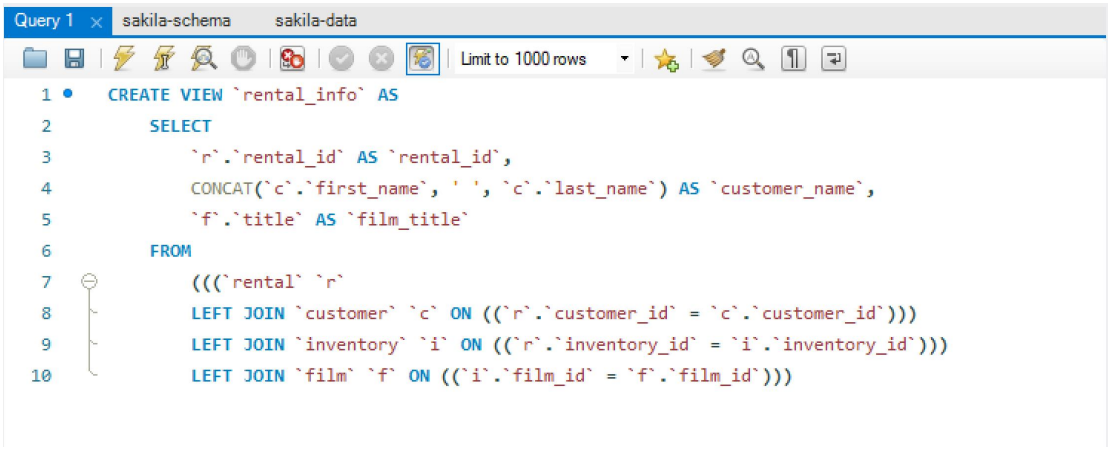
（1） 执行新建视图的语句，并截图 SQL 和执行结果：

设计视图 rental_info，功能为显示 rental_id 及其对应的顾客名字和电影名称。关联了 rental，customer，inventory 和 film 四张表。

SQL 语句为：

```
CREATE VIEW `rental_info` AS
    SELECT
        `r`.`rental_id` AS `rental_id`,
        CONCAT(`c`.`first_name`, ' ', `c`.`last_name`) AS `customer_name`,
        `f`.`title` AS `film_title`
    FROM
        (((`rental` `r`
        LEFT JOIN `customer` `c` ON ((`r`.`customer_id` = `c`.`customer_id`)))
        LEFT JOIN `inventory` `i` ON ((`r`.`inventory_id` = `i`.`inventory_id`)))
        LEFT JOIN `film` `f` ON ((`i`.`film_id` = `f`.`film_id`)))
```

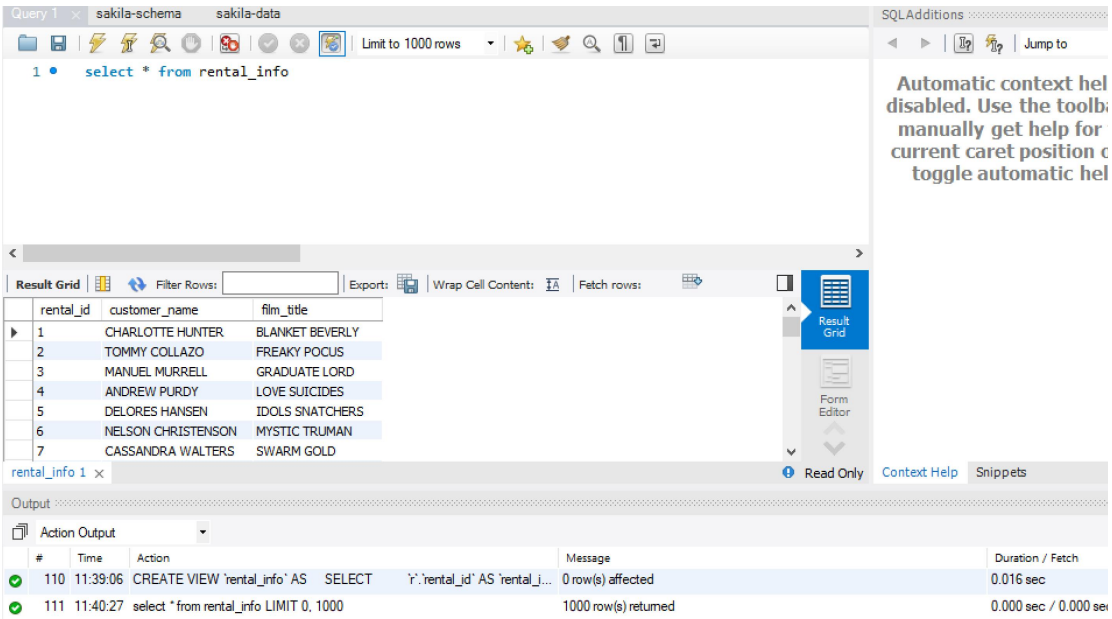
语句截图：



执行结果截图：

（2） 执行 select * from [视图名]，截图执行结果：

执行 select * from rental_info 语句，以下为执行结果：



2. 设计 1 个触发器，需要体现触发器生效。

（1） 执行新建触发器的语句，并截图 SQL 和执行结果：

设计触发器 payment_check_trigger，检查对 payment 表进行插入操作时，支付的金额 amount 是否合法。当 amount<=0 时，不合法，进行报错。

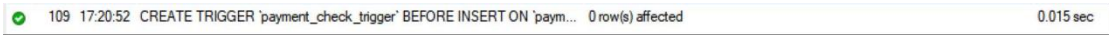
SQL 语句为：

```
DELIMITER ;;
CREATE TRIGGER `payment_check_trigger` BEFORE INSERT ON `payment`
FOR EACH ROW
BEGIN
    DECLARE message VARCHAR(20);
    IF new.amount <= 0
    THEN SET message='amount<=0,error!';
    signal sqlstate 'HY000' set message_text=message;
    END IF;
END;;
DELIMITER ;
```

语句截图如下：

```
1  DELIMITER ;;
2  CREATE TRIGGER `payment_check_trigger` BEFORE INSERT ON `payment`
3  FOR EACH ROW
4  BEGIN
5      DECLARE message VARCHAR(20);
6      IF new.amount <= 0
7      THEN SET message='amount<=0,error!';
8      signal sqlstate 'HY000' set message_text=message;
9      END IF;
10 END;;
11 DELIMITER ;
```

结果截图如下：



（2）验证触发器是否生效，截图验证过程：

验证过程如下：

首先插入一条 rental 数据，为插入 payment 提供 rental_id。再插入一个 amount 为-1 即 amount 不合法的 payment 数据。

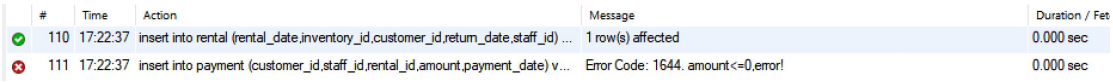
SQL 语句为：

```
insert into rental (rental_date,inventory_id,customer_id,return_date,staff_id)
values("2023-09-21 15:33:33",601,4,now(),1);
insert into payment (customer_id,staff_id,rental_id,amount,payment_date)
values(4,1,1001,-1,now());
```

语句截图如下：

```
1 • insert into rental (rental_date,inventory_id,customer_id,return_date,staff_id)
2   values("2023-09-21 15:33:33",601,4,now(),1);
3 • insert into payment (customer_id,staff_id,rental_id,amount,payment_date)
4   values(4,1,1001,-1,now());
```

结果截图如下：



执行结果报错：“amount<=0,error!”，符合预期。

接下来插入一条 rental 数据，再插入一个 amount 为 20 的 payment 数据。

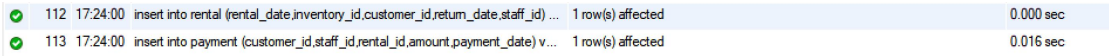
SQL 语句为：

```
insert into rental (rental_date,inventory_id,customer_id,return_date,staff_id)
values("2023-09-21 15:33:34",601,5,now(),1);
insert into payment (customer_id,staff_id,rental_id,amount,payment_date)
values(4,1,1002,20,now());
```

语句截图如下：

```
1 • insert into rental (rental_date,inventory_id,customer_id,return_date,staff_id)
2   values("2023-09-21 15:33:34",601,5,now(),1);
3 • insert into payment (customer_id,staff_id,rental_id,amount,payment_date)
4   values(4,1,1002,20,now());
```

结果截图如下：



由于数据合法，执行结果不报错，符合预期。

3. 设计 1 个存储过程，需要调用该存储过程。

（1） 执行新建存储过程的语句，并截图 SQL 和执行结果：

设计存储过程，根据输入的顾客 id 和员工 id，输出该顾客在该员工处进行租赁的次数。

SQL 语句为：

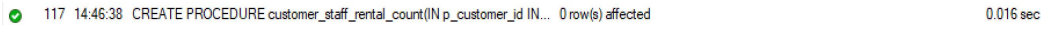
```
DELIMITER ;;
CREATE PROCEDURE customer_staff_rental_count(IN p_customer_id INT,IN p_staff_id INT,
    OUT p_rental_count INT)
READS  SQL DATA
```

```
BEGIN
    select count(rental_id) from rental
    where customer_id=p_customer_id
    and staff_id=p_staff_id
    into p_rental_count;
END ;;
DELIMITER ;
```

语句截图如下：

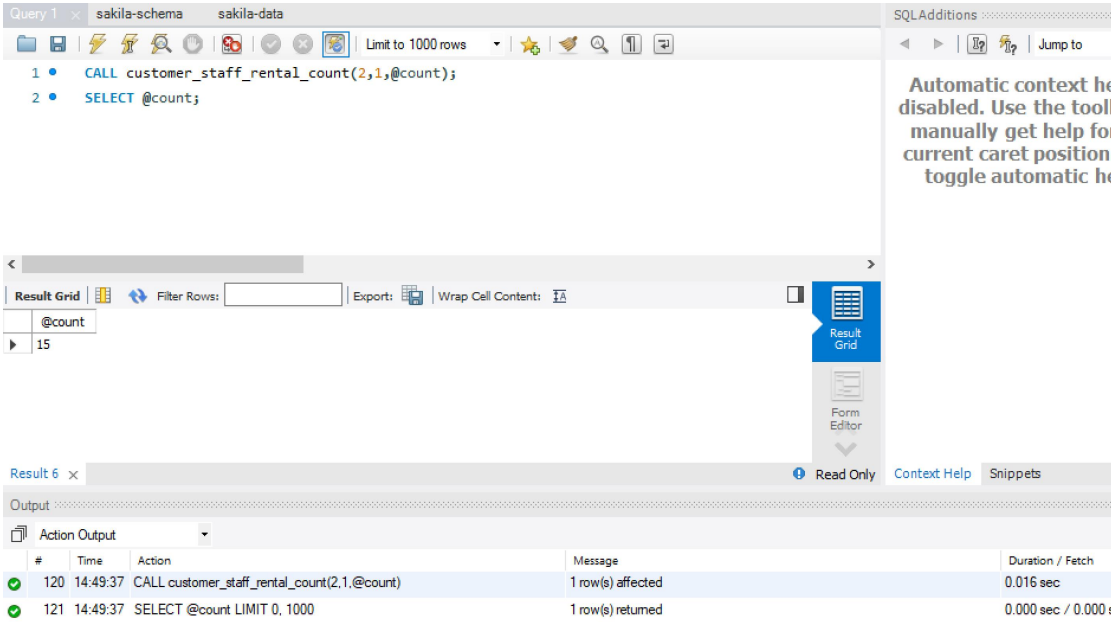
```
1   DELIMITER ;;
2   CREATE PROCEDURE customer_staff_rental_count(IN p_customer_id INT,IN p_staff_id INT,
3       OUT p_rental_count INT)
4   READS  SQL DATA
5   BEGIN
6       select count(rental_id) from rental
7       where customer_id=p_customer_id
8       and staff_id=p_staff_id
9       into p_rental_count;
10  END ;;
11  DELIMITER ;
```

结果截图如下：



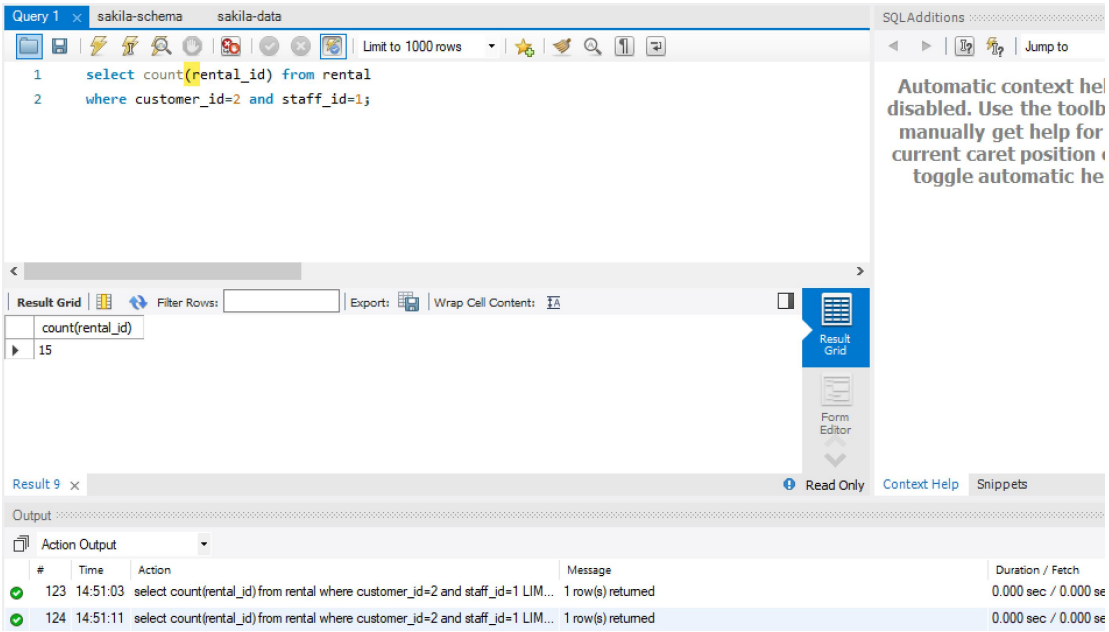
（2） 调用该存储过程，截图调用结果：

执行 CALL customer_staff_rental_count(2,1,@count)语句，对该存储过程进行调用，再执行 SELECT @count 语句获得返回值，结果如下：



结果显示，id 为 2 的顾客在 id 为 1 的员工处进行过 15 次租赁。

为了验证以上结果是否正确，执行查询 id 为 2 的顾客在 id 为 1 的员工处进行的租赁次数的语句 select count(rental_id) from rental where customer_id=2 and stuff_id=1。结果也是 15 次，说明该存储过程给出的结果是正确的。



四、 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？