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？

7 个。

（2） 分析以下 3 个视图，回答以下问题：

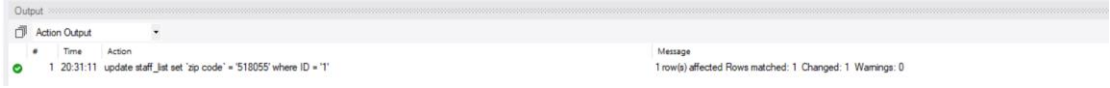
视图名	关联表	作用
actor_info	film,film_actor,film_category,category	聚合演员的出演电影信息
film_list	film,film_actor,film_category,category， actor	聚合电影的各个信息
sales_by_film_category	payment,inventory,rental,film,film_category,category	聚合每个类型电影的销售额

（3） 分别执行以下 2 句 SQL 语句：

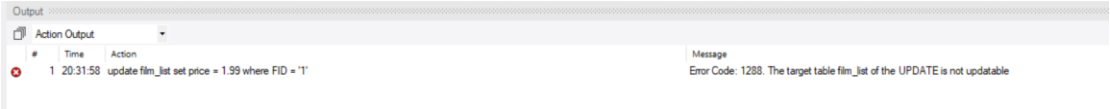
```
update staff_list set `zip code` = '518055' where ID = '1';
update film_list set price = 1.99 where FID = '1';
```

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？

第一句：



第二句：



视图在拥有复杂结构，例如含有 distinct 关键字、含有 group by,order by 之类的聚合分组操作、拥有子查询等时是不可以进行 update 操作的；视图的 select 语句在选择列表中没有聚合函数、没有派生列、FROM 子句至少引用一个表时是可以进行 update 操作的。

（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';
```

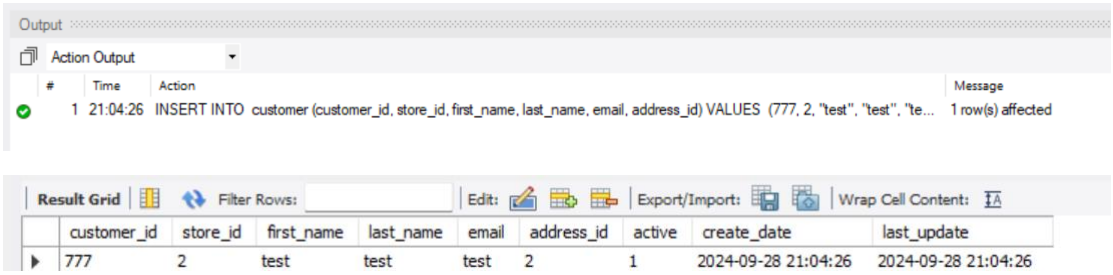
2. 关于触发器

（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）

customer。

每当有新的数据要插入到 customer 表时，在实际插入之前，触发器会自动将 create_date 字段的值设置为当前的日期和时间。

```
INSERT INTO
    customer (customer_id, store_id, first_name, last_name, email, address_id)
VALUES
    (777, 2, "test", "test", "test", "2")
```



（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）

film。

在 film 更新之后更新 film_text 表。

```sql
UPDATE
    film
SET
    description = "text"
WHERE
    film_id = 1
```

```
Output
Action Output
#  Time      Action                                                    Message
1  21:16:42  UPDATE film SET description = "text" WHERE film_id = 1    1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
```

Result Grid | Filter Rows: | Edit: | Export/Import:

film_id	title	description
1	ACADEMY DINOSAUR	text

（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？

sakila-schema.sql | sakila-data.sql

```
0341  (16037,599,1,5843,'2.99','2005-07-10 17:14:27','2006-02-15 22:24:10'),$
0342  (16038,599,2,6800,'9.99','2005-07-12 17:03:56','2006-02-15 22:24:10'),$
0343  (16039,599,2,6895,'2.99','2005-07-12 21:23:59','2006-02-15 22:24:10'),$
0344  (16040,599,1,8965,'6.99','2005-07-30 03:52:37','2006-02-15 22:24:11'),$
0345  (16041,599,2,9630,'2.99','2005-07-31 04:57:07','2006-02-15 22:24:11'),$
0346  (16042,599,2,9679,'2.99','2005-07-31 06:41:19','2006-02-15 22:24:11'),$
0347  (16043,599,2,11522,'3.99','2005-08-17 00:05:05','2006-02-15 22:24:11'),$
0348  (16044,599,1,14233,'1.99','2005-08-21 05:07:08','2006-02-15 22:24:12'),$
0349  (16045,599,1,14599,'4.99','2005-08-21 17:43:42','2006-02-15 22:24:12'),$
0350  (16046,599,1,14719,'1.99','2005-08-21 21:41:57','2006-02-15 22:24:12'),$
0351  (16047,599,2,15590,'8.99','2005-08-23 06:09:44','2006-02-15 22:24:12'),$
0352  (16048,599,2,15719,'2.99','2005-08-23 11:08:46','2006-02-15 22:24:13'),$
0353  (16049,599,2,15725,'2.99','2005-08-23 11:25:00','2006-02-15 22:24:13');$
0354  COMMIT;$
0355  $
0356  --$
0357  -- Trigger to enforce payment_date during INSERT$
0358  --$
0359  $
0360  CREATE TRIGGER payment_date BEFORE INSERT ON payment$
0361    FOR EACH ROW SET NEW.payment_date = NOW();$
0362  $
0363  --$
0364  -- Dumping data for table rental$
0365  --$
0366  $
0367  SET AUTOCOMMIT=0;$
0368  INSERT INTO rental VALUES (1,'2005-05-24 22:53:30',367,130,'2005-05-26 22:04:30',1,'2006-02-1
```

可以将创建触发器的语句移动到 sakila-schema.sql 中。

该触发器用于强制在 payment 表插入数据时自动更新 payment_date，属于数据库结构的一部分，可以放在创建表结构的 sakila-schema.sql 文件中，而不是数据文件中。

3. 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

主键约束	确保每个店铺有唯一的标识符，不允许重复值
非空约束	确保这些字段必须有值，防止出现空值
无符号约束	限制列只能存储非负数

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

ON DELETE RESTRICT: 表示当尝试删除 staff 表中被引用的行时,会阻止删除操作。如果 store 表中有记录引用了 staff 表的某个 staff_id,那么就不允许删除 staff 表中相应的行。

ON UPDATE CASCADE: 表示当 staff 表中被引用的 staff_id 更新时,store 表中相应的 manager_staff_id 也会自动更新。

4. 关于存储过程

（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？

Rewards_report 功能为：找出在上个月中，购买次数和总金额都超过指定阈值的客户，并返回这些客户的详细信息以及符合条件的客户总数。

count_rewardees 是符合奖励条件的客户数量。

（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 表示该存储过程是非确定性的，给定相同的输入参数，该存储过程可能在不同的调用中返回不同的结果。

SQL SECURITY DEFINER 是关于存储过程的安全上下文的声明，表示存储过程将以创建该存储过程的用户的权限来执行，而不是调用该存储过程的用户的权限。

5. 关于函数

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

get_customer_balance 计算指定客户在给定日期的余额，返回的是余额的值。

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

函数	作用
IFNULL	接受两个参数，若第一个参数不是 NULL，则返回第一个参数的值，否则返回第二个参数的值
SUM	用于计算一组值的总和
IF	用作条件判断，接受三个参数：一个条件表达式，如果条件为真时的返回值，以及条件为假时的返回值。

二、 创建新用户并分配权限

（截图语句和执行结果）

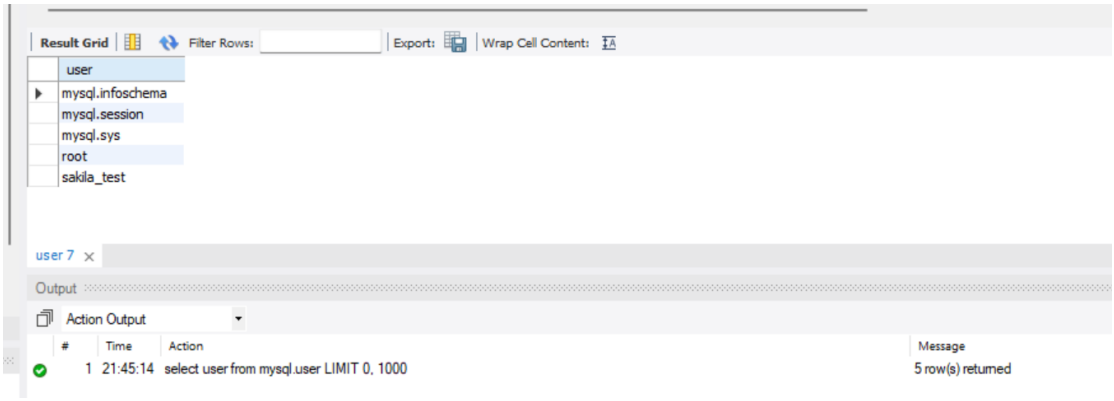
（1） 执行命令新建 sakila_test 用户（密码 123456）；

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

```
#  Time      Action                                                    Message
1  21:44:28  create user 'sakila_test'@'localhost' identified by '123456'  0 row(s) affected
```

（2） 执行命令查看当前已有用户；

```
1  select user from mysql.user
```



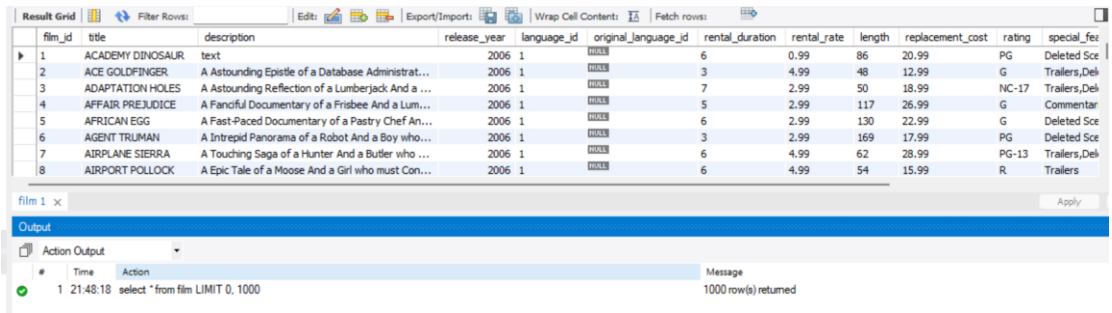
（3） 执行命令把 sakila 数据库的访问权限赋予 sakila_test 用户；

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

```
#  Time      Action                                                    Message
1  21:46:36  grant all privileges on sakila.* to 'sakila_test'@'localhost'  0 row(s) affected
```

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

```
1  select * from film
```



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：

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY INVOKER
VIEW `customer_rentals` AS
    SELECT
        customer_id,
        CONCAT(first_name, ' ', last_name) AS customer_name,
        rental_date,
        return_date,
        film_id,
        title
    FROM
        customer
        JOIN rental USING (customer_id)
        JOIN inventory USING (inventory_id)
        JOIN film USING (film_id)
    ORDER BY
        customer_id
```

```
Output
Action Output
#  Time      Action                                                                                                  Message
1  10:15:13  CREATE  ALGORITHM = UNDEFINED  DEFINER = `root`@`localhost`  SQL SECURITY INVOKER VIEW `customer_r...  0 row(s) affected
```

（2） 执行 select * from [视图名]，截图执行结果：

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

customer_id	customer_name	rental_date	return_date	film_id	title
1	MARY SMITH	2005-07-28 19:20:07	2005-07-29 22:54:07	982	WOMEN DORADO
1	MARY SMITH	2005-06-21 06:24:45	2005-06-28 03:28:45	997	YOUTH KICK
1	MARY SMITH	2005-08-21 23:33:57	2005-08-23 01:30:57	924	UNFORGIVEN ZOOLANDER
1	MARY SMITH	2005-05-28 10:35:23	2005-06-03 06:32:23	875	TALENTED HOMICIDE
1	MARY SMITH	2005-07-28 16:18:23	2005-07-30 17:56:23	929	USUAL UNTOUCHABLES
1	MARY SMITH	2005-07-08 07:33:56	2005-07-12 13:25:56	764	SATURDAY LAMBS
1	MARY SMITH	2005-06-18 13:33:59	2005-06-19 17:40:59	766	SAVANNAH TOWN
1	MARY SMITH	2005-07-09 13:24:07	2005-07-14 14:01:07	814	SNATCH SLIPPER
1	MARY SMITH	2005-08-01 08:51:04	2005-08-10 12:12:04	3	ADAPTATION HOLES
1	MARY SMITH	2005-07-28 09:04:45	2005-07-30 12:37:45	243	DOORS PRESIDENT
1	MARY SMITH	2005-07-29 03:58:49	2005-08-01 05:16:49	22	AMISTAD MIDSUMMER
1	MARY SMITH	2005-06-16 15:18:57	2005-06-17 21:05:57	159	CLOSER BANG
1	MARY SMITH	2005-06-15 18:02:53	2005-06-19 15:54:53	228	DETECTIVE VISION
1	MARY SMITH	2005-07-09 16:38:01	2005-07-13 18:02:01	174	CONFIDENTIAL INTERVIEW
1	MARY SMITH	2005-06-18 08:41:48	2005-06-22 03:36:48	44	ATTACKS HATE

customer_rentals 12 ×

```
Output
Action Output
#  Time      Action                                  Message
1  10:15:49  SELECT * FROM customer_rentals LIMIT 0, 1000  1000 row(s) returned
```

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：

```
DELIMITER //

CREATE TRIGGER after_rental_insert
AFTER INSERT ON rental
FOR EACH ROW
BEGIN
    UPDATE inventory
    SET last_update = NOW()
    WHERE inventory_id = NEW.inventory_id;
END//

DELIMITER ;
```

Output

Action Output

#	Time	Action	Message
1	10:51:28	CREATE TRIGGER after_rental_insert AFTER INSERT ON rental FOR EACH ROW BEGIN UPDATE inventory SET last...	0 row(s) affected

（2） 验证触发器是否生效，截图验证过程：

更新前：

Result Grid | Filter Rows: | Edit:

inventory_id	film_id	store_id	last_update
1	1	1	2006-02-15 05:09:17
NULL	NULL	NULL	NULL

更新：

```
INSERT INTO rental (rental_date, inventory_id, customer_id, staff_id)
VALUES (NOW(), 1, 1, 1);
```

Output

Action Output

#	Time	Action
1	10:49:23	INSERT INTO rental (rental_date, inventory_id, customer_id, staff_id) VALUES (NOW(), 1, 1, 1)

更新后：

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

inventory_id	film_id	store_id	last_update
1	1	1	2024-09-29 10:50:20
NULL	NULL	NULL	NULL

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：

```sql
DELIMITER //

CREATE PROCEDURE GetCustomerLoyalty(IN p_customer_id INT)
BEGIN
    DECLARE v_rental_count INT;
    DECLARE v_favorite_category VARCHAR(25);
    DECLARE v_total_amount DECIMAL(5,2);
    DECLARE v_loyalty_points INT;

    SELECT COUNT(*) INTO v_rental_count
    FROM rental
    WHERE customer_id = p_customer_id;

    SELECT c.name INTO v_favorite_category
    FROM rental r
    JOIN inventory i USING(inventory_id)
    JOIN film f USING (film_id)
    JOIN film_category fc USING (film_id)
    JOIN category c USING (category_id)
    WHERE r.customer_id = p_customer_id
    GROUP BY c.category_id
    ORDER BY COUNT(*) DESC
    LIMIT 1;

    SELECT COALESCE(SUM(p.amount), 0) INTO v_total_amount
    FROM payment p
    WHERE p.customer_id = p_customer_id;

    SET v_loyalty_points = v_rental_count * 10 + FLOOR(v_total_amount);

    SELECT
        v_rental_count AS rental_count,
        v_favorite_category AS favorite_category,
        v_total_amount AS total_amount,
        v_loyalty_points AS loyalty_points;
```

Output

Action Output

#	Time	Action	Message
1	10:57:50	CREATE PROCEDURE GetCustomerLoyalty(IN p_customer_id INT) BEGIN DECLARE v_rental_count INT; DECLARE v...	0 row(s) affected

（2） 调用该存储过程，截图调用结果：

```sql
CALL GetCustomerLoyalty(1)
```

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

rental_count	favorite_category	total_amount	loyalty_points
34	Classics	118.68	458

Result 1

Output

Action Output

#	Time	Action	Message
1	10:58:23	CALL GetCustomerLoyalty(1)	1 row(s) returned

四、 附加题

1. 为什么在 AFTER 触发器中只能对 NEW 取值，不能对 NEW 进行赋值？

AFTER 触发器是在数据库操作完成之后触发的。此时，数据已经被写入表中。

允许在 AFTER 触发器中修改 NEW 会导致数据不一致，因为主操作已经完成,修改 NEW 不会影响到已经提交的数据。

2. 请结合具体场景举例说明如何利用视图实现权限控制。

如下：

先根据不同的需求创建不同的视图：

```sql
-- 普通员工创建视图，只能看到电影基本信息
CREATE VIEW employee_film_info AS
SELECT film_id, title, description, release_year, rating
FROM film;

-- 为库存管理员创建视图，可以看到库存信息
CREATE VIEW inventory_manager_view AS
SELECT i.inventory_id, f.film_id, f.title, i.store_id
FROM inventory i
JOIN film f ON i.film_id = f.film_id;
```

再将视图权限授予给特定的用户：

```sql
GRANT SELECT ON employee_film_info TO 'all_employees';
GRANT SELECT ON inventory_manager_view TO 'inventory_managers';
```

这种方式实现了以下权限控制：

普通员工只能看到电影的基本信息，库存管理员可以查看库存状况。