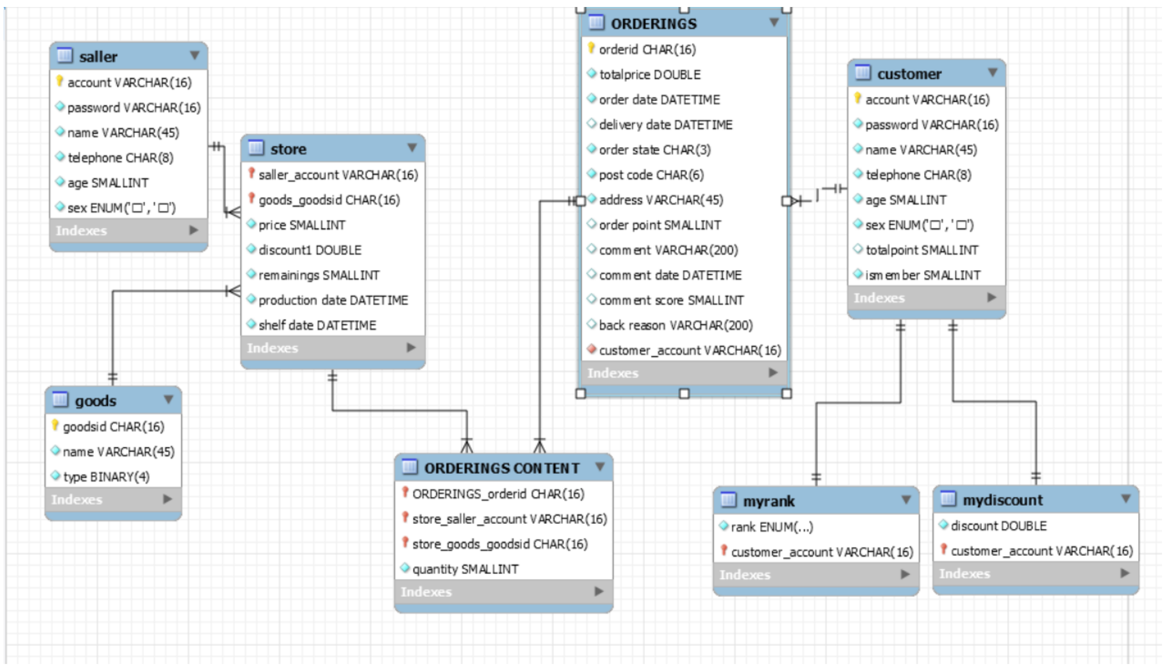
数据库技术第四次上机作业

潘亦晟

515021910384

1. 在数据库的设计过程中，为了保证 3NF，你做了哪些调整？3NF 在本数据库中带来的好处是什么（如果有）？坏处是什么（如果有）？



ER 图：

customer: 顾客信息表; myrank : 顾客用户等级表；
mydiscount:顾客额外折扣表；saller: 供应商信息表；
goods：商品信息表； store：供应商供应商品；
ORDERINGS：订单信息表； ODRERINGS CONTENT：订单内容

①由于一件商品可以由不同的供应商供应，且设置不同的价格，折扣，库存，一个供应商同时也可以供应多件不同的商品，对供应商和商品之间建立 m:n 关系 store。
②由于一件商品可以属于多种类别，一种商品类别的商品可以包括多个商品，将商品类型设置为四位二进制（eg.0001 代表衣物，1001 代表奢侈品和衣服），大大减少了新建表的工作量
③由于一种商品可以被多个客户多次购买，建立 ORDERINGS 并构建 orderid 加以区分不同的订单；由于一个订单中可以包括多种不同的商品，对订单和商品储存之间构建 m:n 关系 ORDEINGS CONTENT。

建立数据库流程：
①插入顾客信息（插入后自动更新等级和额外折扣）

②插入供应商信息（saller）
③插入商品信息（goods）
④插入供应商-商品储存信息（store）
⑤插入订单信息（其中 state，totalprice 为 null）
⑥插入订单内容（插入后自动更新商品库存，订单金额，并设置订单状态为未处理）
⑦更新订单状态（更新后自动更新顾客积分，库存）

边界条件：
a.基本信息格式，域值
b.对于订单，未处理和已备货状态无法设置发货日期（设为 null），仅有已完成状态才能更新评价内容，评价内容，评价分数，订单积分（且需要更新顾客积分），仅有已退货状态可以更新退货理由（且需要对顾客积分和库存进行回滚）

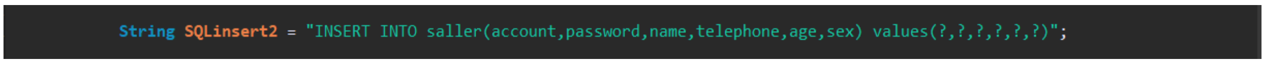
调整：为保证 3NF，主要做出的调整是将顾客的用户等级和额外折扣两个属性进行了单独建表来消除用户积分，用户等级，额外折扣三者之间的依赖性

好处：

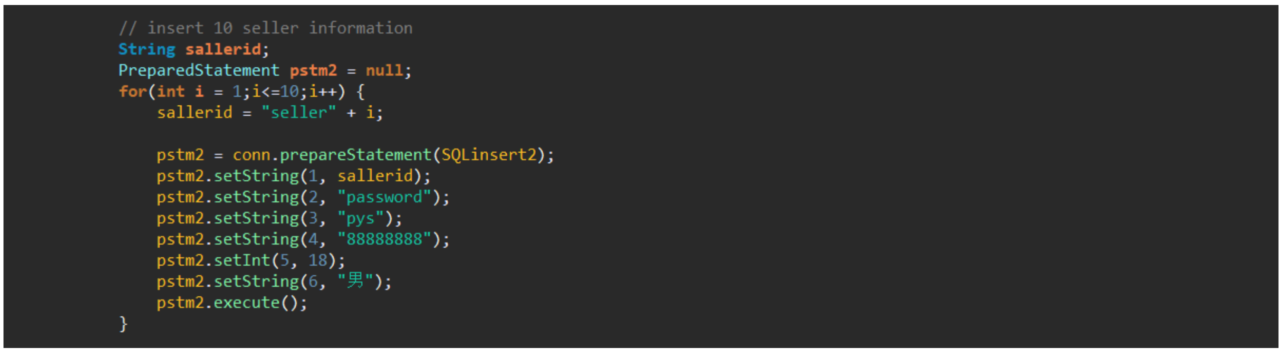
坏处：增加了代码的复杂性。对于维护用户等级和额外折扣这两个表不对不新建多个触发器来维护，同时也降低了性能。

2. 内容 1（写出下列三步操作对应的 sql 语句）

(1) 随机插入 10 名供应商的信息
思想：由于后续操作与供应商的其他信息无关，为了方便快速插入，对于密码，姓名，电话，年龄，性别都固定相同的信息插入；对于账号信息，为了方便检查，都以“seller+num”的格式进行插入。JDBC 代码实现如下：



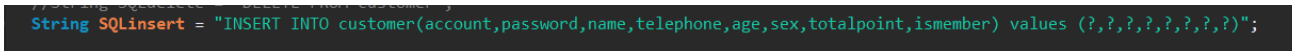
```java
String SQLinsert2 = "INSERT INTO saller(account,password,name,telephone,age,sex) values(?,?,?,?,?,?)";
```



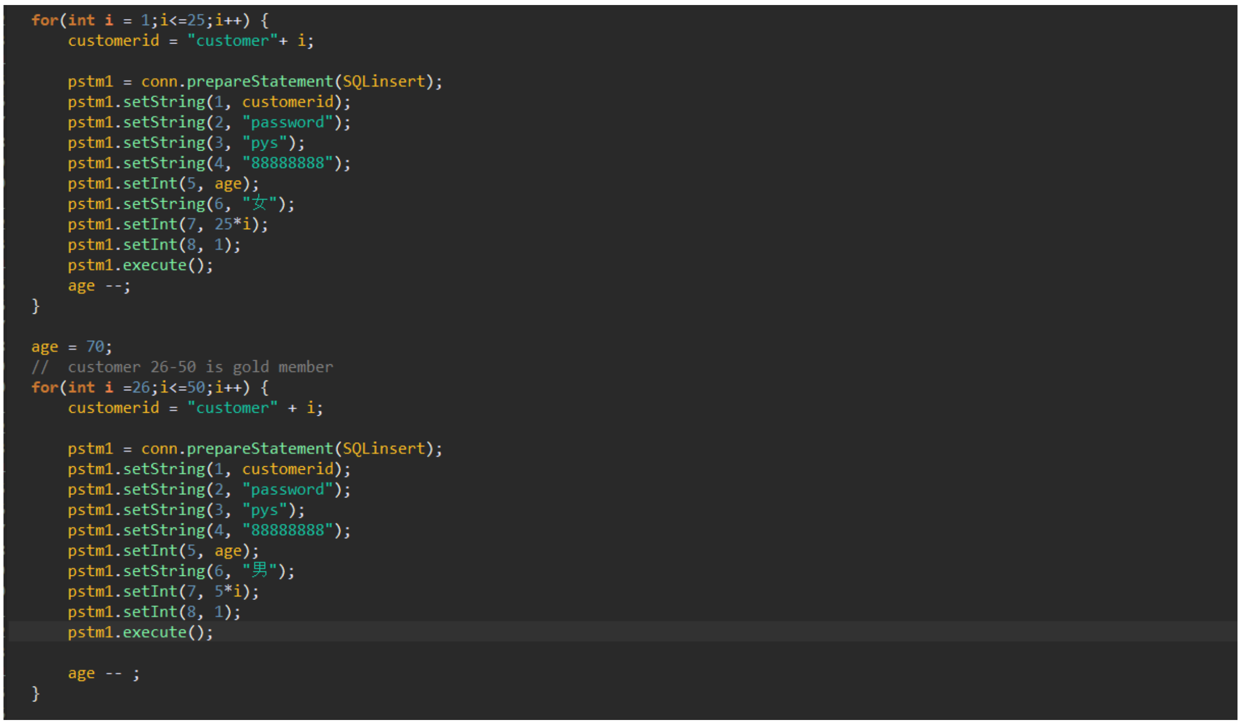
```java
// insert 10 seller information
String sallerid;
PreparedStatement pstm2 = null;
for(int i = 1;i<=10;i++) {
    sallerid = "seller" + i;

    pstm2 = conn.prepareStatement(SQLinsert2);
    pstm2.setString(1, sallerid);
    pstm2.setString(2, "password");
    pstm2.setString(3, "pys");
    pstm2.setString(4, "88888888");
    pstm2.setInt(5, 18);
    pstm2.setString(6, "男");
    pstm2.execute();
}
```

(2) 随机插入 100 名客户的信息
思想：对于密码，姓名，电话，性别等无关信息，为了方便插入，都固定相同的信息插入；对于账号信息，为了方便检查，都以“customer+num”的格式进行插入；为了尽可

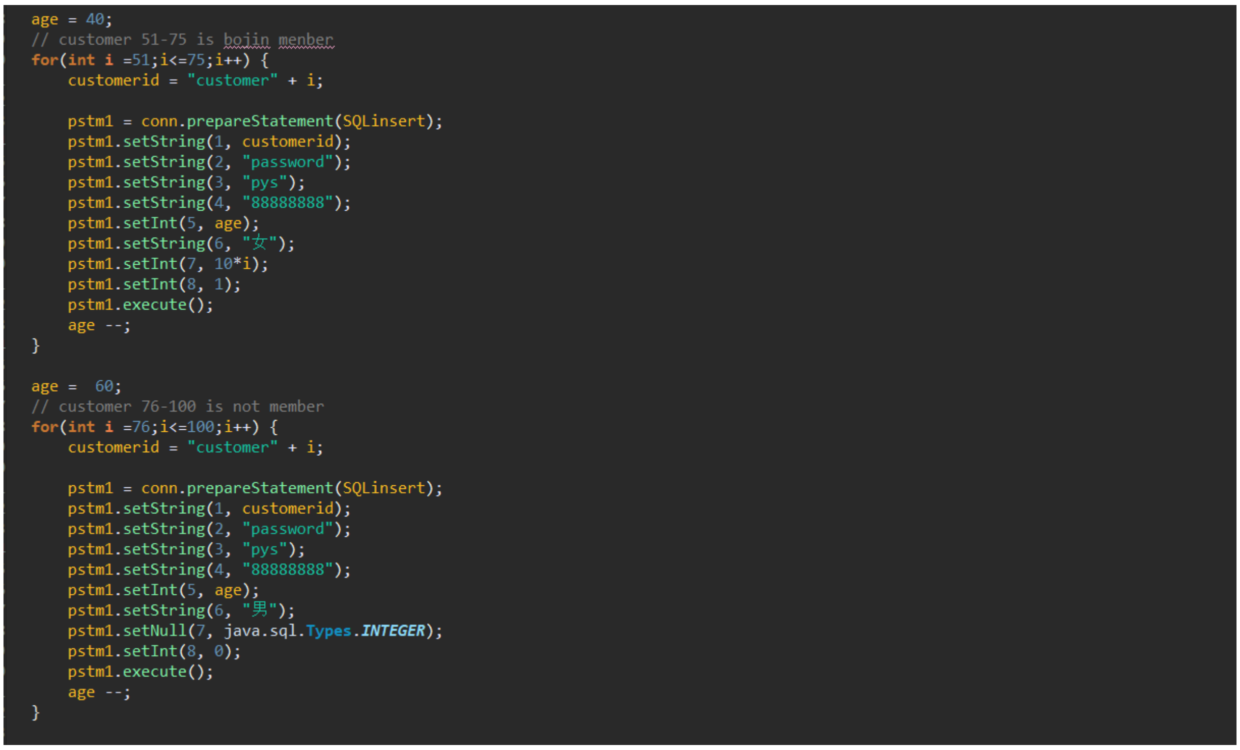
能验证所键数据库的鲁棒性，将 100 客户均分为 4 份，分配给 4 种不同的用户等级，并给每个用户分配与其用户等级相对应的不同的积分（非会员的积分为 null）。
JDBC 代码实现如下：



```
String SQLinsert = "INSERT INTO customer(account,password,name,telephone,age,sex,totalpoint,ismember) values (?,?,?,?,?,?,?,?)";
```



```
for(int i = 1;i<=25;i++) {
    customerid = "customer"+ i;

    pstm1 = conn.prepareStatement(SQLinsert);
    pstm1.setString(1, customerid);
    pstm1.setString(2, "password");
    pstm1.setString(3, "pys");
    pstm1.setString(4, "88888888");
    pstm1.setInt(5, age);
    pstm1.setString(6, "女");
    pstm1.setInt(7, 25*i);
    pstm1.setInt(8, 1);
    pstm1.execute();
    age --;
}

age = 70;
//  customer 26-50 is gold member
for(int i =26;i<=50;i++) {
    customerid = "customer" + i;

    pstm1 = conn.prepareStatement(SQLinsert);
    pstm1.setString(1, customerid);
    pstm1.setString(2, "password");
    pstm1.setString(3, "pys");
    pstm1.setString(4, "88888888");
    pstm1.setInt(5, age);
    pstm1.setString(6, "男");
    pstm1.setInt(7, 5*i);
    pstm1.setInt(8, 1);
    pstm1.execute();

    age -- ;
}
```

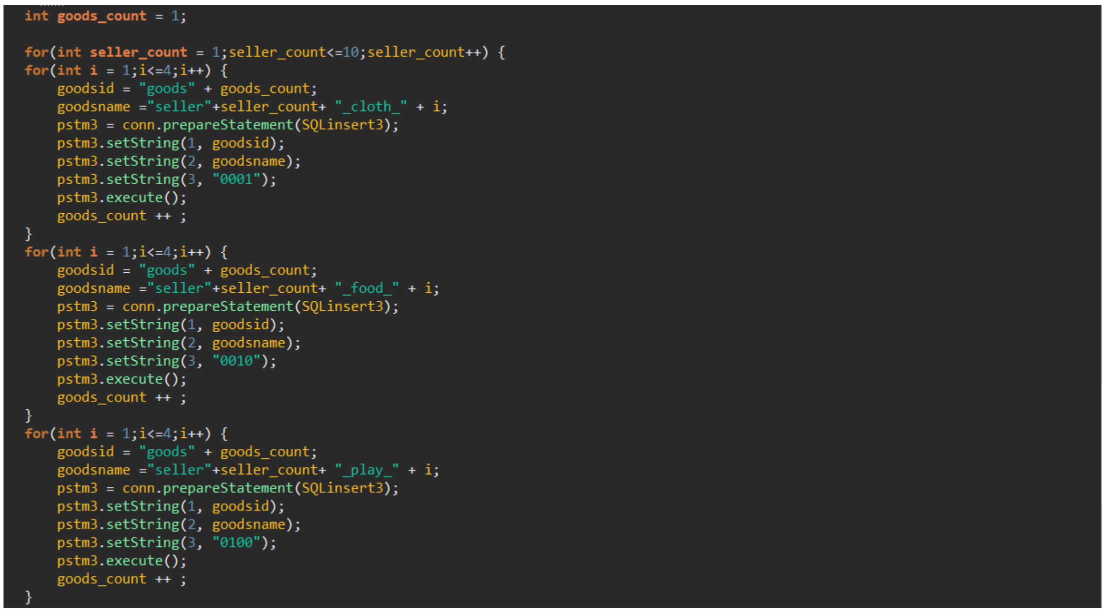


```
age = 40;
// customer 51-75 is bojin menber
for(int i =51;i<=75;i++) {
    customerid = "customer" + i;

    pstm1 = conn.prepareStatement(SQLinsert);
    pstm1.setString(1, customerid);
    pstm1.setString(2, "password");
    pstm1.setString(3, "pys");
    pstm1.setString(4, "88888888");
    pstm1.setInt(5, age);
    pstm1.setString(6, "女");
    pstm1.setInt(7, 10*i);
    pstm1.setInt(8, 1);
    pstm1.execute();
    age --;
}

age =  60;
// customer 76-100 is not member
for(int i =76;i<=100;i++) {
    customerid = "customer" + i;

    pstm1 = conn.prepareStatement(SQLinsert);
    pstm1.setString(1, customerid);
    pstm1.setString(2, "password");
    pstm1.setString(3, "pys");
    pstm1.setString(4, "88888888");
    pstm1.setInt(5, age);
    pstm1.setString(6, "男");
    pstm1.setNull(7, java.sql.Types.INTEGER);
    pstm1.setInt(8, 0);
    pstm1.execute();
    age --;
}
```

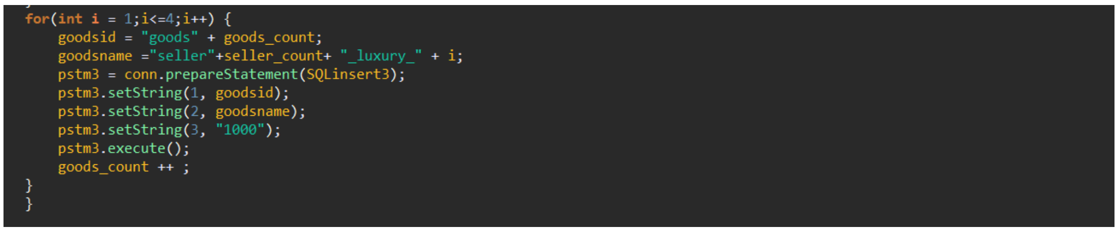
(3)　供应商发布四类商品的情况

思想：在问题 1 中已经说明，供应商发布商品需要先插入商品信息，再建立供应商-商品关系。为了简化问题，将生产日期，保质期等信息固定不变插入；同时将商品价格和折扣设置为 50-300 和 0.5-1 的随机数。每个供应商每种商品都发布 4 种商品，每种商品的库存为 10-30 中的随机数。

insert goods

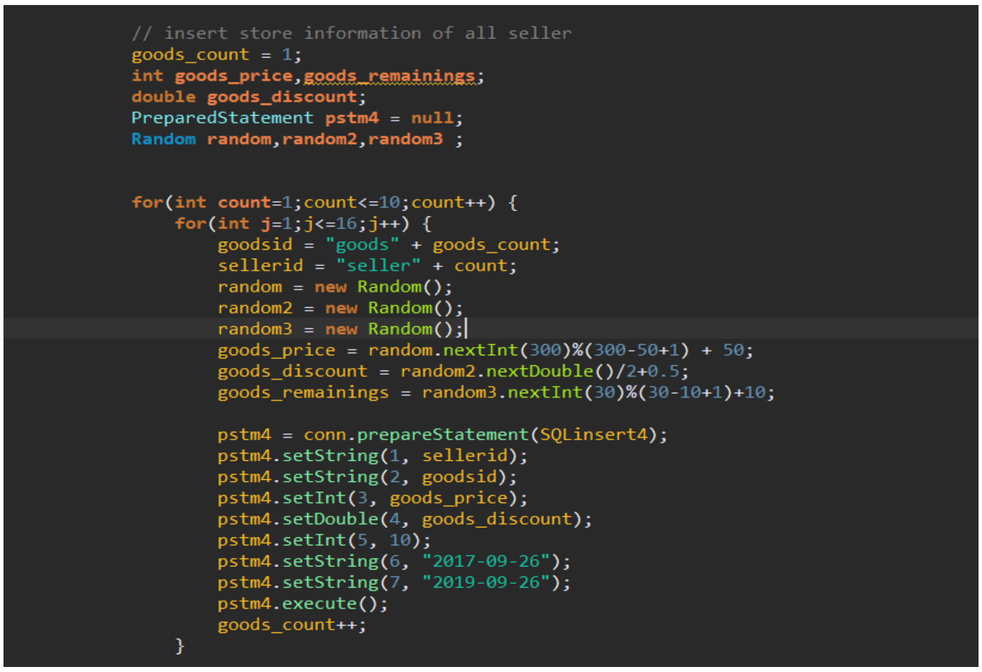


```java
int goods_count = 1;

for(int seller_count = 1;seller_count<=10;seller_count++) {
for(int i = 1;i<=4;i++) {
    goodsid = "goods" + goods_count;
    goodsname ="seller"+seller_count+ "_cloth_" + i;
    pstm3 = conn.prepareStatement(SQLinsert3);
    pstm3.setString(1, goodsid);
    pstm3.setString(2, goodsname);
    pstm3.setString(3, "0001");
    pstm3.execute();
    goods_count ++ ;
}
for(int i = 1;i<=4;i++) {
    goodsid = "goods" + goods_count;
    goodsname ="seller"+seller_count+ "_food_" + i;
    pstm3 = conn.prepareStatement(SQLinsert3);
    pstm3.setString(1, goodsid);
    pstm3.setString(2, goodsname);
    pstm3.setString(3, "0010");
    pstm3.execute();
    goods_count ++ ;
}
for(int i = 1;i<=4;i++) {
    goodsid = "goods" + goods_count;
    goodsname ="seller"+seller_count+ "_play_" + i;
    pstm3 = conn.prepareStatement(SQLinsert3);
    pstm3.setString(1, goodsid);
    pstm3.setString(2, goodsname);
    pstm3.setString(3, "0100");
    pstm3.execute();
    goods_count ++ ;
}
```



```java
for(int i = 1;i<=4;i++) {
    goodsid = "goods" + goods_count;
    goodsname ="seller"+seller_count+ "_luxury_" + i;
    pstm3 = conn.prepareStatement(SQLinsert3);
    pstm3.setString(1, goodsid);
    pstm3.setString(2, goodsname);
    pstm3.setString(3, "1000");
    pstm3.execute();
    goods_count ++ ;
}
}
```

insert store：

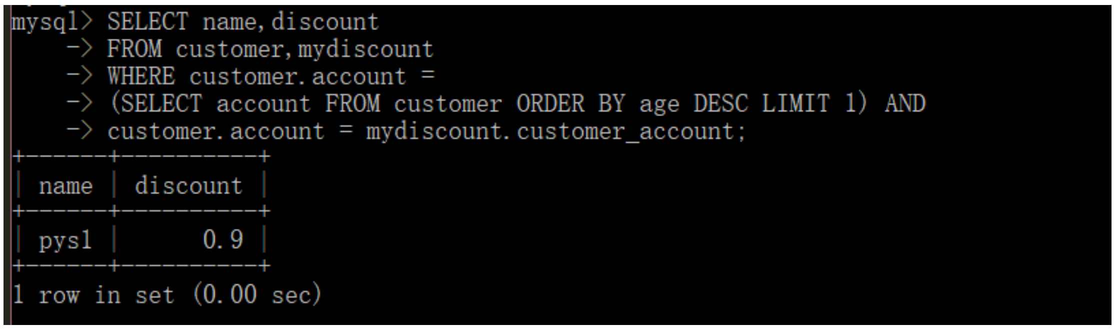


```java
        // insert store information of all seller
        goods_count = 1;
        int goods_price,goods_remainings;
        double goods_discount;
        PreparedStatement pstm4 = null;
        Random random,random2,random3 ;


        for(int count=1;count<=10;count++) {
            for(int j=1;j<=16;j++) {
                goodsid = "goods" + goods_count;
                sellerid = "seller" + count;
                random = new Random();
                random2 = new Random();
                random3 = new Random();
                goods_price = random.nextInt(300)%(300-50+1) + 50;
                goods_discount = random2.nextDouble()/2+0.5;
                goods_remainings = random3.nextInt(30)%(30-10+1)+10;

                pstm4 = conn.prepareStatement(SQLinsert4);
                pstm4.setString(1, sellerid);
                pstm4.setString(2, goodsid);
                pstm4.setInt(3, goods_price);
                pstm4.setDouble(4, goods_discount);
                pstm4.setInt(5, 10);
                pstm4.setString(6, "2017-09-26");
                pstm4.setString(7, "2019-09-26");
                pstm4.execute();
                goods_count++;
            }
```

3. 内容 2

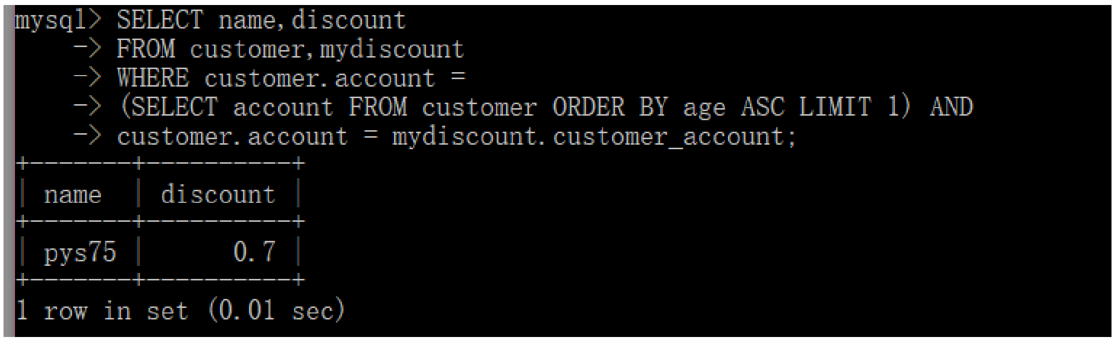
（1）给出以下操作对应的 sql 语句和查询结果：“查询所有客户中年龄最大和最小的用户的名字（记为 customer1 和 customer2）和享受的额外折扣率”。

对于 customer1：



```
mysql> SELECT name,discount
    -> FROM customer,mydiscount
    -> WHERE customer.account =
    -> (SELECT account FROM customer ORDER BY age DESC LIMIT 1) AND
    -> customer.account = mydiscount.customer_account;
+-------+----------+
| name  | discount |
+-------+----------+
| pys1  |      0.9 |
+-------+----------+
1 row in set (0.00 sec)
```

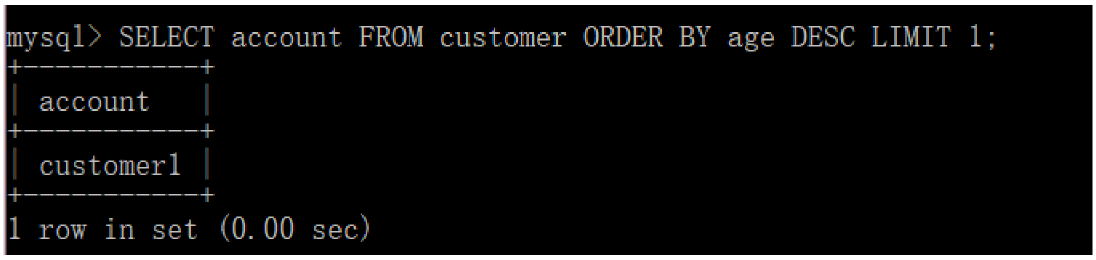
对于 customer2：



```
mysql> SELECT name,discount
    -> FROM customer,mydiscount
    -> WHERE customer.account =
    -> (SELECT account FROM customer ORDER BY age ASC LIMIT 1) AND
    -> customer.account = mydiscount.customer_account;
+--------+----------+
| name   | discount |
+--------+----------+
| pys75  |      0.7 |
+--------+----------+
1 row in set (0.01 sec)
```

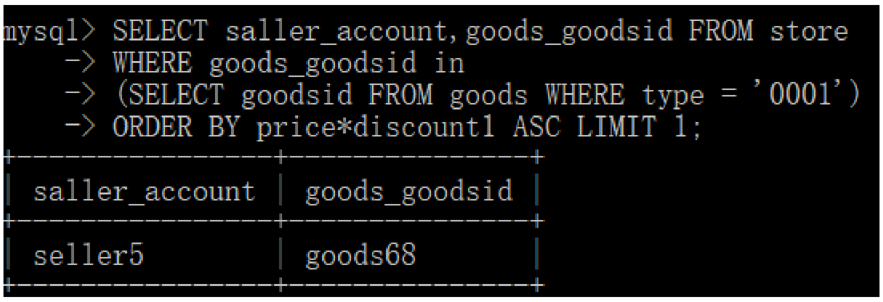
（2）给出 customer1 下单对应的 sql 语句 （若分步进行，依次给出 sql 语句和中间结果）

①查询 customer 1 的 accout

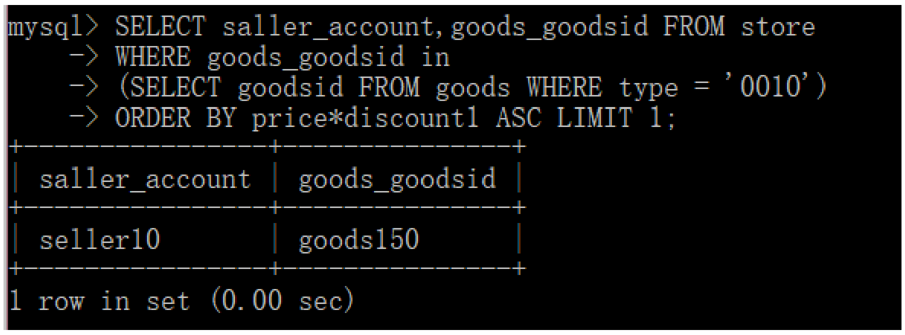


```
mysql> SELECT account FROM customer ORDER BY age DESC LIMIT 1;
+-----------+
| account   |
+-----------+
| customer1 |
+-----------+
1 row in set (0.00 sec)
```

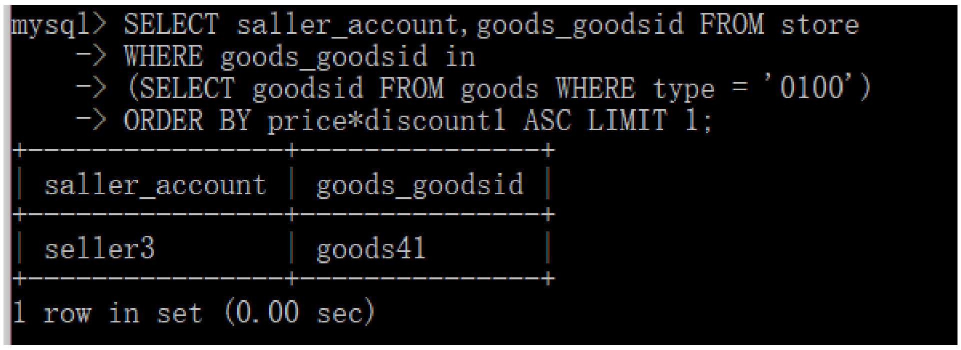
②查询各类商品中最便宜的商品编号



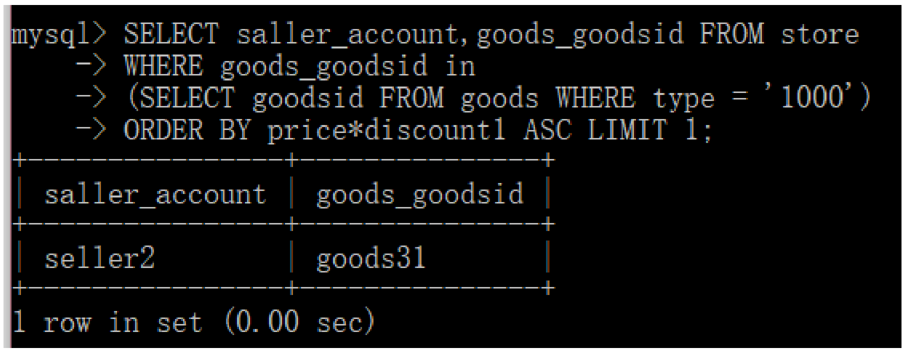
```
mysql> SELECT saller_account,goods_goodsid FROM store
    -> WHERE goods_goodsid in
    -> (SELECT goodsid FROM goods WHERE type = '0001')
    -> ORDER BY price*discount1 ASC LIMIT 1;
+----------------+---------------+
| saller_account | goods_goodsid |
+----------------+---------------+
| seller5        | goods68       |
+----------------+---------------+
```



```
mysql> SELECT saller_account,goods_goodsid FROM store
    -> WHERE goods_goodsid in
    -> (SELECT goodsid FROM goods WHERE type = '0010')
    -> ORDER BY price*discount1 ASC LIMIT 1;
+----------------+---------------+
| saller_account | goods_goodsid |
+----------------+---------------+
| seller10       | goods150      |
+----------------+---------------+
1 row in set (0.00 sec)
```

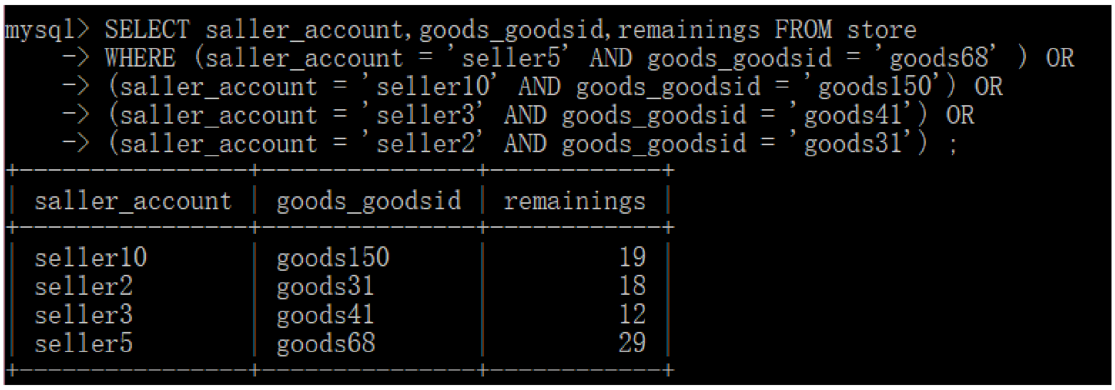


```
mysql> SELECT saller_account,goods_goodsid FROM store
    -> WHERE goods_goodsid in
    -> (SELECT goodsid FROM goods WHERE type = '0100')
    -> ORDER BY price*discount1 ASC LIMIT 1;
+----------------+---------------+
| saller_account | goods_goodsid |
+----------------+---------------+
| seller3        | goods41       |
+----------------+---------------+
1 row in set (0.00 sec)
```



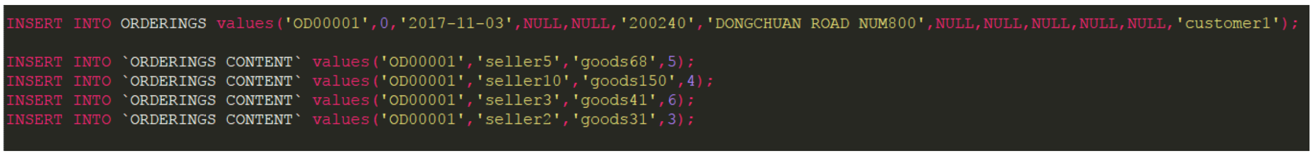
```
mysql> SELECT saller_account,goods_goodsid FROM store
    -> WHERE goods_goodsid in
    -> (SELECT goodsid FROM goods WHERE type = '1000')
    -> ORDER BY price*discount1 ASC LIMIT 1;
+----------------+---------------+
| saller_account | goods_goodsid |
+----------------+---------------+
| seller2        | goods31       |
+----------------+---------------+
1 row in set (0.00 sec)
```

③查询这四种商品的原本的库存



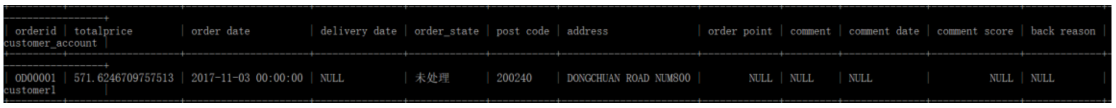
```
mysql> SELECT saller_account,goods_goodsid,remainings FROM store
    -> WHERE (saller_account = 'seller5' AND goods_goodsid = 'goods68' ) OR
    -> (saller_account = 'seller10' AND goods_goodsid = 'goods150') OR
    -> (saller_account = 'seller3' AND goods_goodsid = 'goods41') OR
    -> (saller_account = 'seller2' AND goods_goodsid = 'goods31') ;
+----------------+---------------+------------+
| saller_account | goods_goodsid | remainings |
+----------------+---------------+------------+
| seller10       | goods150      |         19 |
| seller2        | goods31       |         18 |
| seller3        | goods41       |         12 |
| seller5        | goods68       |         29 |
+----------------+---------------+------------+
```

④顾客 1 下单

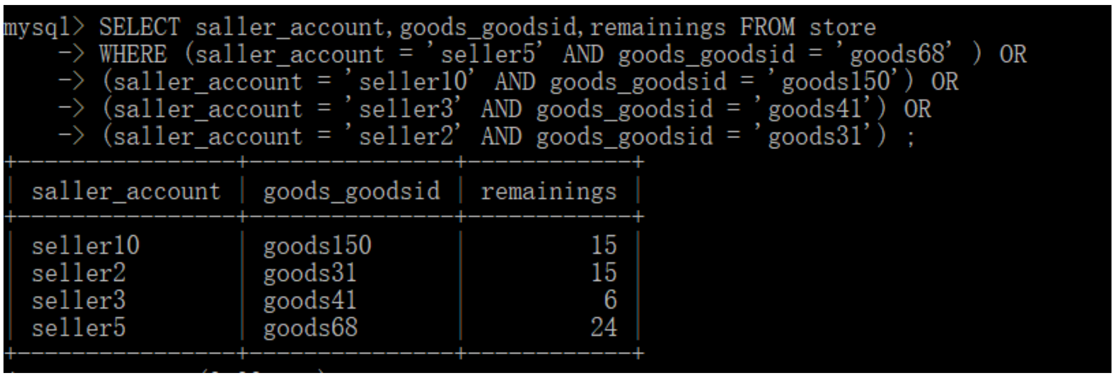


```
INSERT INTO ORDERINGS values('OD00001',0,'2017-11-03',NULL,NULL,'200240','DONGCHUAN ROAD NUM800',NULL,NULL,NULL,NULL,NULL,'customer1');

INSERT INTO `ORDERINGS CONTENT` values('OD00001','seller5','goods68',5);
INSERT INTO `ORDERINGS CONTENT` values('OD00001','seller10','goods150',4);
INSERT INTO `ORDERINGS CONTENT` values('OD00001','seller3','goods41',6);
INSERT INTO `ORDERINGS CONTENT` values('OD00001','seller2','goods31',3);
```

⑤查询下单后的订单信息和库存



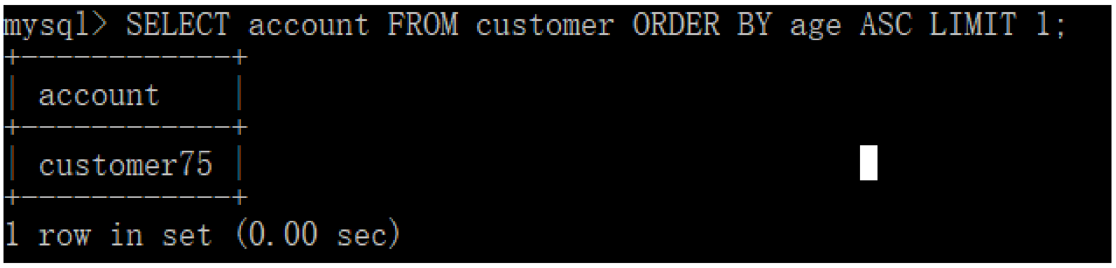
```
| orderid | totalprice        | order date          | delivery date | order_state | post code | address                | order point | comment | comment date | comment score | back reason |
customer_account |
| OD00001 | 571.6246709757513 | 2017-11-03 00:00:00 | NULL          | 未处理      | 200240    | DONGCHUAN ROAD NUM800 |        NULL | NULL    | NULL         |          NULL | NULL        |
customer1        |
```



```
mysql> SELECT saller_account,goods_goodsid,remainings FROM store
    -> WHERE (saller_account = 'seller5' AND goods_goodsid = 'goods68' ) OR
    -> (saller_account = 'seller10' AND goods_goodsid = 'goods150') OR
    -> (saller_account = 'seller3' AND goods_goodsid = 'goods41') OR
    -> (saller_account = 'seller2' AND goods_goodsid = 'goods31') ;
+----------------+---------------+------------+
| saller_account | goods_goodsid | remainings |
+----------------+---------------+------------+
| seller10       | goods150      |         15 |
| seller2        | goods31       |         15 |
| seller3        | goods41       |          6 |
| seller5        | goods68       |         24 |
+----------------+---------------+------------+
```

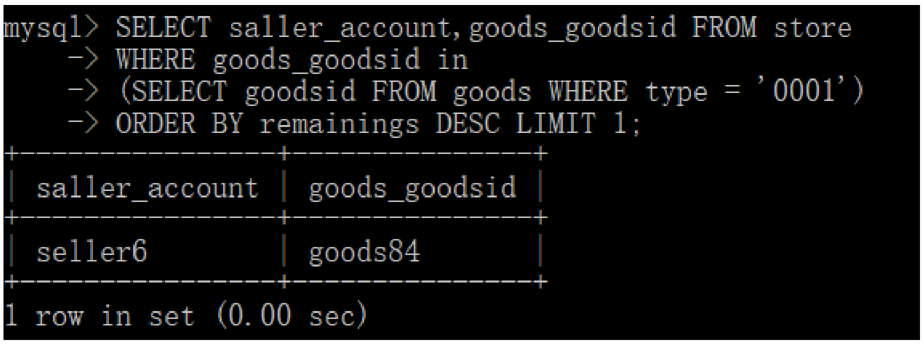
（3）给出 customer2 下单对应的 sql 语句（若分步进行，依次给出 sql 语句和中间结果）

①查询顾客 2 的 account

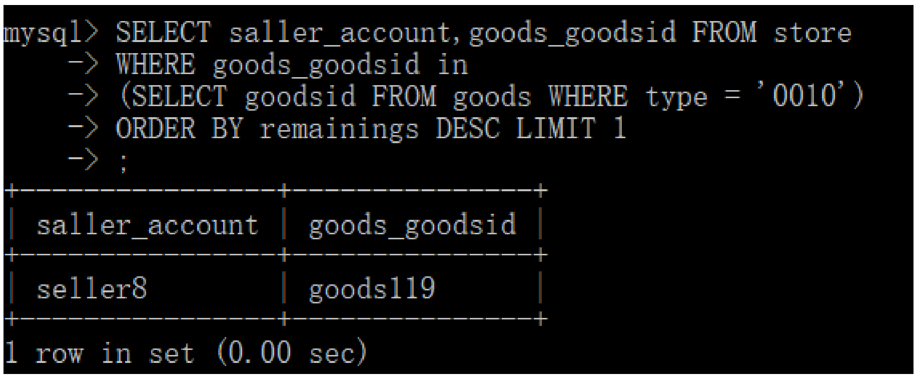


```
mysql> SELECT account FROM customer ORDER BY age ASC LIMIT 1;
+------------+
| account    |
+------------+
| customer75 |
+------------+
1 row in set (0.00 sec)
```

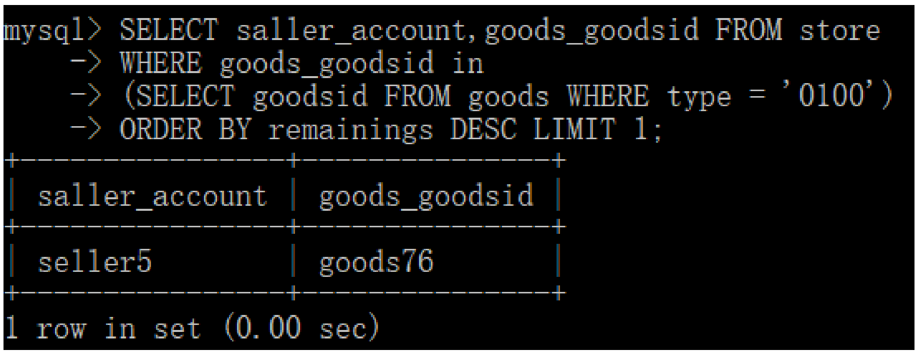
②查询每种类别中库存最多的商品



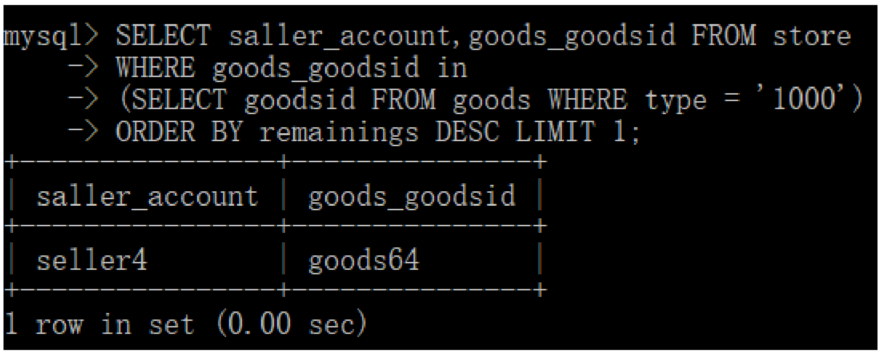
```
mysql> SELECT saller_account,goods_goodsid FROM store
    -> WHERE goods_goodsid in
    -> (SELECT goodsid FROM goods WHERE type = '0001')
    -> ORDER BY remainings DESC LIMIT 1;
+----------------+---------------+
| saller_account | goods_goodsid |
+----------------+---------------+
| seller6        | goods84       |
+----------------+---------------+
1 row in set (0.00 sec)
```



```
mysql> SELECT saller_account,goods_goodsid FROM store
    -> WHERE goods_goodsid in
    -> (SELECT goodsid FROM goods WHERE type = '0010')
    -> ORDER BY remainings DESC LIMIT 1
    -> ;
+----------------+---------------+
| saller_account | goods_goodsid |
+----------------+---------------+
| seller8        | goods119      |
+----------------+---------------+
1 row in set (0.00 sec)
```

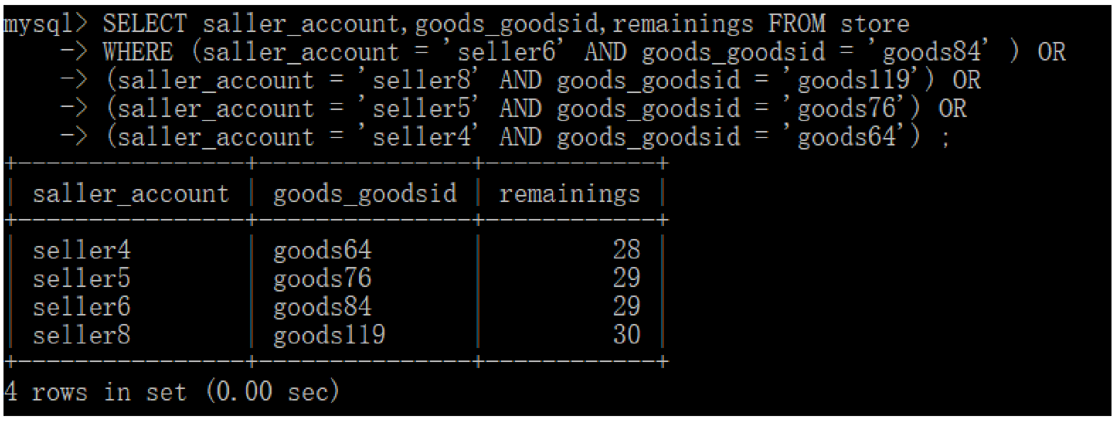


```
mysql> SELECT saller_account,goods_goodsid FROM store
    -> WHERE goods_goodsid in
    -> (SELECT goodsid FROM goods WHERE type = '0100')
    -> ORDER BY remainings DESC LIMIT 1;
+-----------------+-----------------+
| saller_account  | goods_goodsid  |
+-----------------+-----------------+
| seller5         | goods76         |
+-----------------+-----------------+
1 row in set (0.00 sec)
```



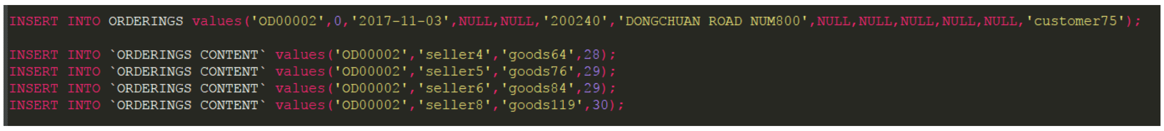
```
mysql> SELECT saller_account,goods_goodsid FROM store
    -> WHERE goods_goodsid in
    -> (SELECT goodsid FROM goods WHERE type = '1000')
    -> ORDER BY remainings DESC LIMIT 1;
+-----------------+-----------------+
| saller_account  | goods_goodsid  |
+-----------------+-----------------+
| seller4         | goods64         |
+-----------------+-----------------+
1 row in set (0.00 sec)
```

③查询四种要购买的商品的库存



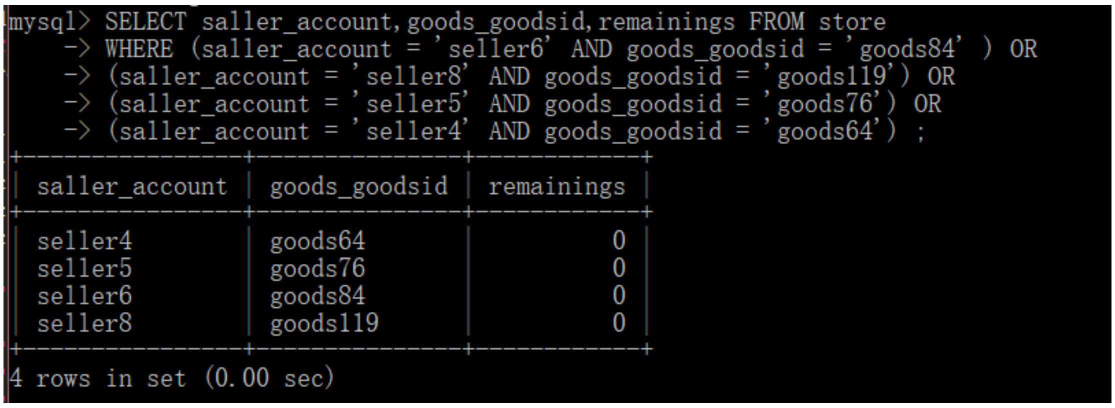
```
mysql> SELECT saller_account,goods_goodsid,remainings FROM store
    -> WHERE (saller_account = 'seller6' AND goods_goodsid = 'goods84' ) OR
    -> (saller_account = 'seller8' AND goods_goodsid = 'goods119') OR
    -> (saller_account = 'seller5' AND goods_goodsid = 'goods76') OR
    -> (saller_account = 'seller4' AND goods_goodsid = 'goods64') ;
+----------------+---------------+------------+
| saller_account | goods_goodsid | remainings |
+----------------+---------------+------------+
| seller4        | goods64       |         28 |
| seller5        | goods76       |         29 |
| seller6        | goods84       |         29 |
| seller8        | goods119      |         30 |
+----------------+---------------+------------+
4 rows in set (0.00 sec)
```

④顾客 2 下单

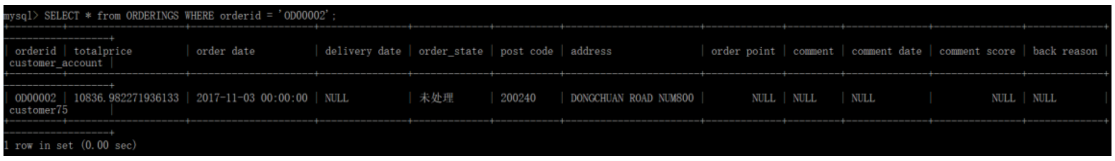


```
INSERT INTO ORDERINGS values('OD00002',0,'2017-11-03',NULL,NULL,'200240','DONGCHUAN ROAD NUM800',NULL,NULL,NULL,NULL,NULL,'customer75');

INSERT INTO `ORDERINGS CONTENT` values('OD00002','seller4','goods64',28);
INSERT INTO `ORDERINGS CONTENT` values('OD00002','seller5','goods76',29);
INSERT INTO `ORDERINGS CONTENT` values('OD00002','seller6','goods84',29);
INSERT INTO `ORDERINGS CONTENT` values('OD00002','seller8','goods119',30);
```

⑤下单后查询各个状态



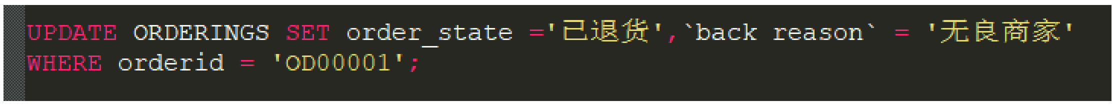
```
mysql> SELECT saller_account,goods_goodsid,remainings FROM store
    -> WHERE (saller_account = 'seller6' AND goods_goodsid = 'goods84' ) OR
    -> (saller_account = 'seller8' AND goods_goodsid = 'goods119') OR
    -> (saller_account = 'seller5' AND goods_goodsid = 'goods76') OR
    -> (saller_account = 'seller4' AND goods_goodsid = 'goods64') ;
+----------------+---------------+------------+
| saller_account | goods_goodsid | remainings |
+----------------+---------------+------------+
| seller4        | goods64       |          0 |
| seller5        | goods76       |          0 |
| seller6        | goods84       |          0 |
| seller8        | goods119      |          0 |
+----------------+---------------+------------+
4 rows in set (0.00 sec)
```



```
mysql> SELECT * from ORDERINGS WHERE orderid = '0D00002';
+---------+--------------------+---------------------+---------------+-------------+-----------+-------------------------+-------------+---------+--------------+---------------+-------------+
-------------------+
| orderid | totalprice         | order date          | delivery date | order_state | post code | address                 | order point | comment | comment date | comment score | back reason |
 customer_account |
+---------+--------------------+---------------------+---------------+-------------+-----------+-------------------------+-------------+---------+--------------+---------------+-------------+
-------------------+
| 0D00002 | 10836.982271936133 | 2017-11-03 00:00:00 | NULL          | 未处理      | 200240    | DONGCHUAN ROAD NUM800 |        NULL | NULL    | NULL         |          NULL | NULL        |
 customer75       |
+---------+--------------------+---------------------+---------------+-------------+-----------+-------------------------+-------------+---------+--------------+---------------+-------------+
-------------------+
1 row in set (0.00 sec)
```

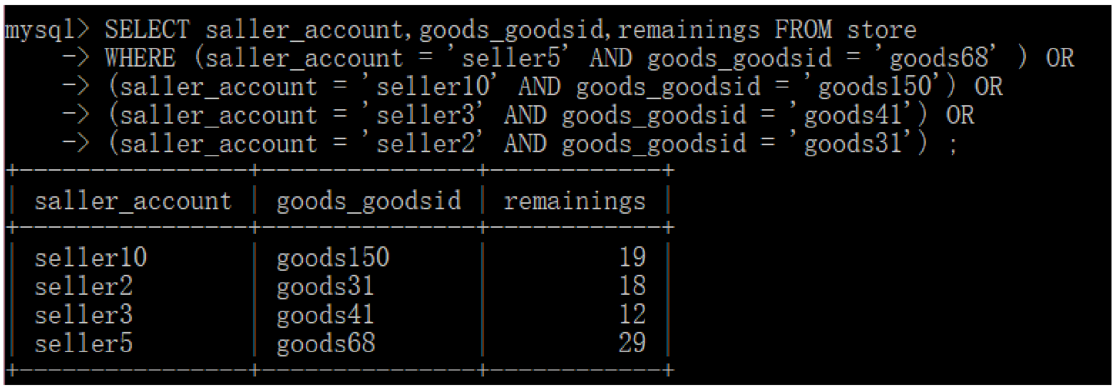
（4）给出 customer1 退货对应的 sql 语句（若分步进行，依次给出 sql 语句和中间结果）

①更新订单状态为已退货



```
UPDATE ORDERINGS SET order_state ='已退货',`back reason` = '无良商家'
WHERE orderid = 'OD00001';
```

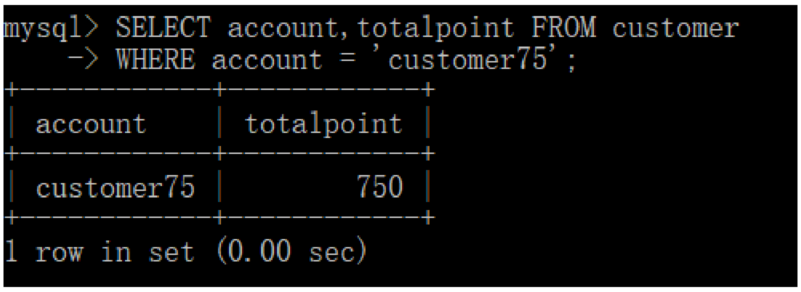
②查询退货后 sotre 中的库存



```
mysql> SELECT saller_account,goods_goodsid,remainings FROM store
    -> WHERE (saller_account = 'seller5' AND goods_goodsid = 'goods68' ) OR
    -> (saller_account = 'seller10' AND goods_goodsid = 'goods150') OR
    -> (saller_account = 'seller3' AND goods_goodsid = 'goods41') OR
    -> (saller_account = 'seller2' AND goods_goodsid = 'goods31') ;
+----------------+---------------+------------+
| saller_account | goods_goodsid | remainings |
+----------------+---------------+------------+
| seller10       | goods150      |         19 |
| seller2        | goods31       |         18 |
| seller3        | goods41       |         12 |
| seller5        | goods68       |         29 |
+----------------+---------------+------------+
```

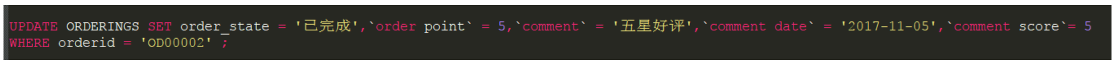
（5）给出 customer2 接受订单对应的 sql 语句（若分步进行，依次给出 sql 语句和中间结果）

①查询接受订单前顾客积分



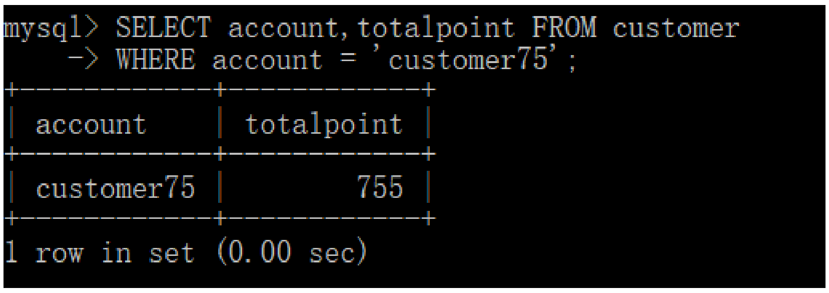
```
mysql> SELECT account,totalpoint FROM customer
    -> WHERE account = 'customer75';
+------------+------------+
| account    | totalpoint |
+------------+------------+
| customer75 |        750 |
+------------+------------+
1 row in set (0.00 sec)
```

②更新订单状态为已完成



```
UPDATE ORDERINGS SET order_state = '已完成',`order point` = 5,`comment` = '五星好评',`comment date` = '2017-11-05',`comment score`= 5
WHERE orderid = 'OD00002' ;
```

③查询接受订单后顾客积分



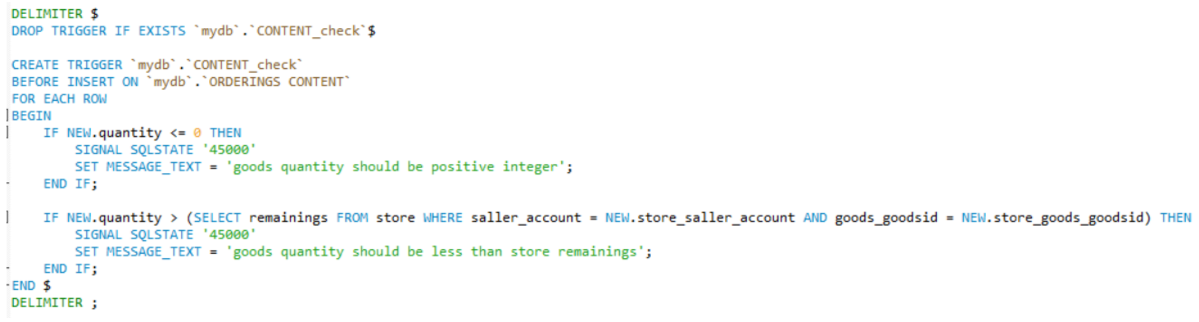
```
mysql> SELECT account,totalpoint FROM customer
    -> WHERE account = 'customer75';
+------------+------------+
| account    | totalpoint |
+------------+------------+
| customer75 |        755 |
+------------+------------+
1 row in set (0.00 sec)
```

4. Bonus

对每种完整性约束，给出对应的 trigger 代码截图

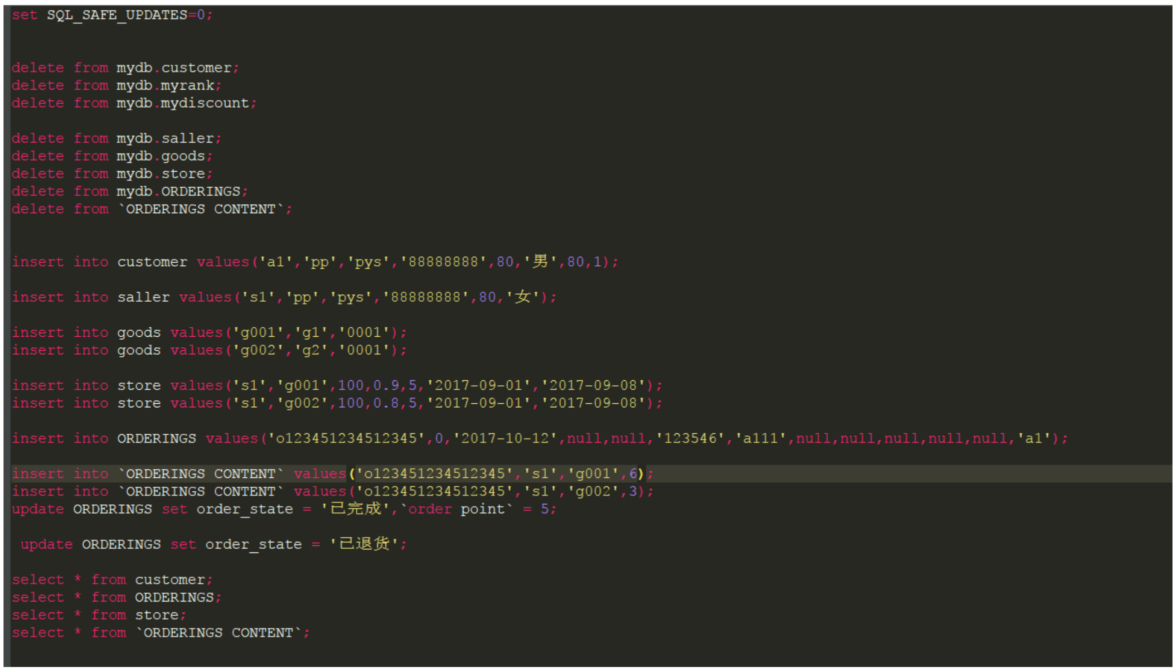
（1）　购买数量和库存量的约束

思想：在对订单内容 ORDERINGS CONTENT 插入时需要保证购买该商品的数量不能超过库存量，需要在 store 表中对库存量 remainings 进行查询，然后进行比较。



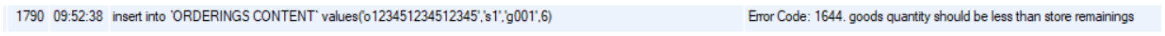
```
DELIMITER $
DROP TRIGGER IF EXISTS `mydb`.`CONTENT_check`$

CREATE TRIGGER `mydb`.`CONTENT_check`
BEFORE INSERT ON `mydb`.`ORDERINGS CONTENT`
FOR EACH ROW
BEGIN
    IF NEW.quantity <= 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'goods quantity should be positive integer';
    END IF;

    IF NEW.quantity > (SELECT remainings FROM store WHERE saller_account = NEW.store_saller_account AND goods_goodsid = NEW.store_goods_goodsid) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'goods quantity should be less than store remainings';
    END IF;
END $
DELIMITER ;
```

验证：



```
set SQL_SAFE_UPDATES=0;


delete from mydb.customer;
delete from mydb.myrank;
delete from mydb.mydiscount;

delete from mydb.saller;
delete from mydb.goods;
delete from mydb.store;
delete from mydb.ORDERINGS;
delete from `ORDERINGS CONTENT`;


insert into customer values('a1','pp','pys','88888888',80,'男',80,1);

insert into saller values('s1','pp','pys','88888888',80,'女');

insert into goods values('g001','g1','0001');
insert into goods values('g002','g2','0001');

insert into store values('s1','g001',100,0.9,5,'2017-09-01','2017-09-08');
insert into store values('s1','g002',100,0.8,5,'2017-09-01','2017-09-08');

insert into ORDERINGS values('o123451234512345',0,'2017-10-12',null,null,'123546','a111',null,null,null,null,null,'a1');

insert into `ORDERINGS CONTENT` values('o123451234512345','s1','g001',6);
insert into `ORDERINGS CONTENT` values('o123451234512345','s1','g002',3);
update ORDERINGS set order_state = '已完成',`order point` = 5;

 update ORDERINGS set order_state = '已退货';

select * from customer;
select * from ORDERINGS;
select * from store;
select * from `ORDERINGS CONTENT`;
```

结果符合预期，如下：



```
1790  09:52:38  insert into 'ORDERINGS CONTENT' values('o123451234512345','s1','g001',6)    Error Code: 1644. goods quantity should be less than store remainings
```

（2）　库存量减少机制

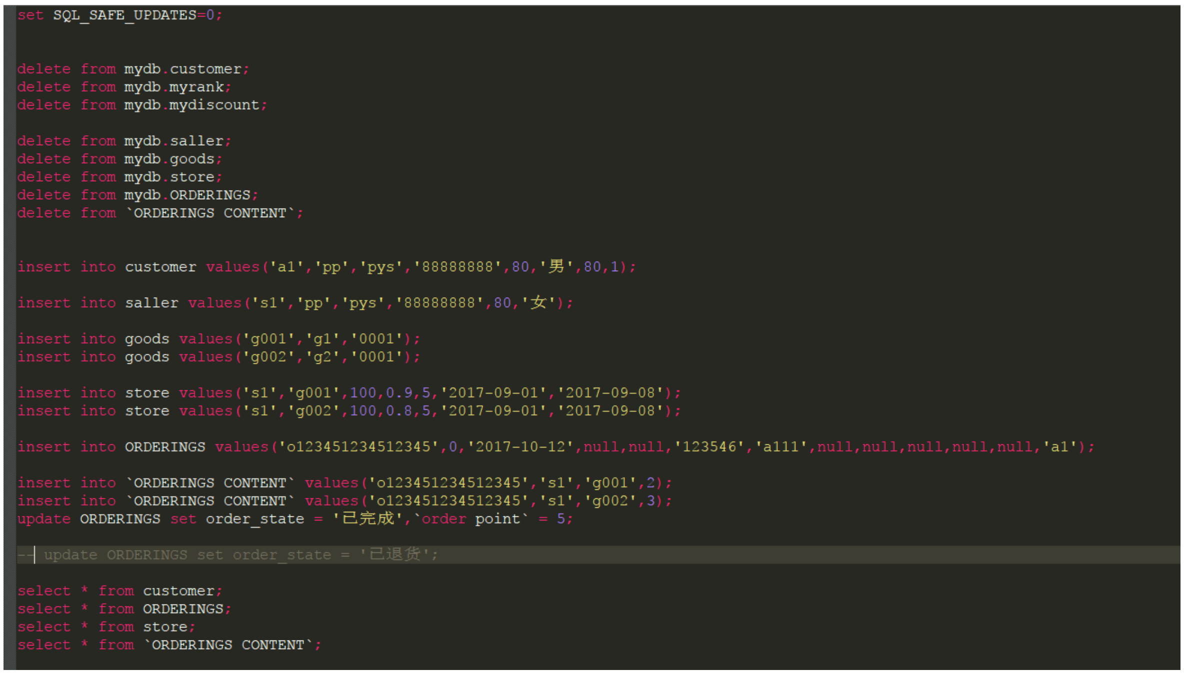
　　思想：当插入订单内容后，需要首先将更新订单表中的订单金额，订单状态（设为未处理），计算订单内容的时候需要查询订单内容汇总各供应发布各个商品的价格，折扣，同时查询顾客的额外折扣，进行累加计算；然后需要更新供应商-商品表 store 中各个商品的库存量。

DELIMITER $
DROP TRIGGER IF EXISTS `mydb`.`auto_updte_order_store`$

CREATE TRIGGER `mydb`.`auto_updte_order_store`
BEFORE INSERT ON `mydb`.`ORDERINGS CONTENT`
FOR EACH ROW

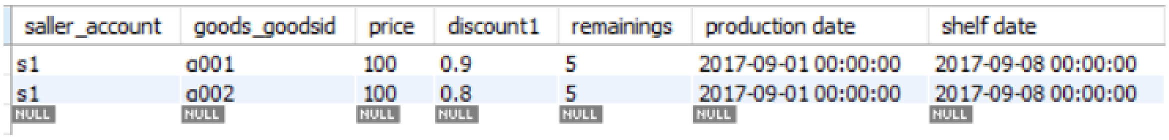
```
BEGIN
        DECLARE x varchar(16);

        UPDATE ORDERINGS SET `order_state` = '未处理' WHERE `orderid` = NEW.`ORDERINGS_orderid`;

        SET x = (SELECT DISTINCT customer_account FROM ORDERINGS WHERE orderid = NEW.`ORDERINGS_orderid`);

        IF (((SELECT ismember FROM customer WHERE `account` = x) = 1) AND ((SELECT totalpoint FROM customer WHERE `account` = x) BETWEEN 0 AND 100) ) THEN
          UPDATE ORDERINGS SET totalprice = totalprice + NEW.quantity*(SELECT discount1*price FROM store WHERE goods_goodsid = NEW.store_goods_goodsid AND saller_account = NEW.store_saller_account)*0.9
          WHERE orderid = NEW.`ORDERINGS_orderid`;

          UPDATE store SET remainings = remainings - NEW.quantity WHERE store.goods_goodsid = NEW.store_goods_goodsid AND store.saller_account = NEW.store_saller_account;
        END IF;

          IF (((SELECT ismember FROM customer WHERE `account` = x) = 1) AND ((SELECT totalpoint FROM customer WHERE `account` = x) BETWEEN 101 AND 500) ) THEN
          UPDATE ORDERINGS SET totalprice = totalprice + NEW.quantity*(SELECT discount1*price FROM store WHERE goods_goodsid = NEW.store_goods_goodsid AND saller_account = NEW.store_saller_account)*0.8
          WHERE orderid = NEW.`ORDERINGS_orderid`;

        UPDATE store SET remainings = remainings - NEW.quantity WHERE store.goods_goodsid = NEW.store_goods_goodsid AND store.saller_account = NEW.store_saller_account;
        END IF;

        IF (((SELECT ismember FROM customer WHERE `account` = x) = 1) AND ((SELECT totalpoint FROM customer WHERE `account` = x) >500) ) THEN
          UPDATE ORDERINGS SET totalprice = totalprice + NEW.quantity*(SELECT discount1*price FROM store WHERE goods_goodsid = NEW.store_goods_goodsid AND saller_account = NEW.store_saller_account)*0.7
          WHERE orderid = NEW.`ORDERINGS_orderid`;

        UPDATE store SET remainings = remainings - NEW.quantity WHERE store.goods_goodsid = NEW.store_goods_goodsid AND store.saller_account = NEW.store_saller_account;
        END IF;

        IF (((SELECT ismember FROM customer WHERE `account` = x) = 0) AND ((SELECT totalpoint FROM customer WHERE `account` = x) IS NULL) ) THEN
        UPDATE ORDERINGS SET totalprice = totalprice + NEW.quantity*(SELECT discount1*price
FROM store WHERE goods_goodsid = NEW.store_goods_goodsid AND saller_account =
NEW.store_saller_account)
        WHERE orderid = NEW.`ORDERINGS_orderid`;

    UPDATE store SET remainings = remainings - NEW.quantity WHERE store.goods_goodsid =
NEW.store_goods_goodsid AND store.saller_account = NEW.store_saller_account;
    END IF;
END $
DELIMITER ;
```

验证：

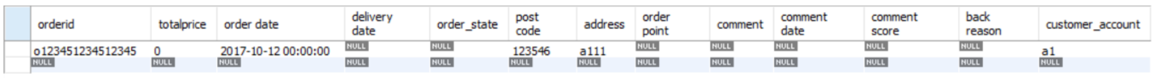


```
set SQL_SAFE_UPDATES=0;


delete from mydb.customer;
delete from mydb.myrank;
delete from mydb.mydiscount;

delete from mydb.saller;
delete from mydb.goods;
delete from mydb.store;
delete from mydb.ORDERINGS;
delete from `ORDERINGS CONTENT`;


insert into customer values('a1','pp','pys','88888888',80,'男',80,1);

insert into saller values('s1','pp','pys','88888888',80,'女');

insert into goods values('g001','g1','0001');
insert into goods values('g002','g2','0001');

insert into store values('s1','g001',100,0.9,5,'2017-09-01','2017-09-08');
insert into store values('s1','g002',100,0.8,5,'2017-09-01','2017-09-08');

insert into ORDERINGS values('o123451234512345',0,'2017-10-12',null,null,'123546','a111',null,null,null,null,null,'a1');

insert into `ORDERINGS CONTENT` values('o123451234512345','s1','g001',2);
insert into `ORDERINGS CONTENT` values('o123451234512345','s1','g002',3);
update ORDERINGS set order_state = '已完成',`order point` = 5;

-- update ORDERINGS set order_state = '已退货';

select * from customer;
select * from ORDERINGS;
select * from store;
select * from `ORDERINGS CONTENT`;
```

初始时

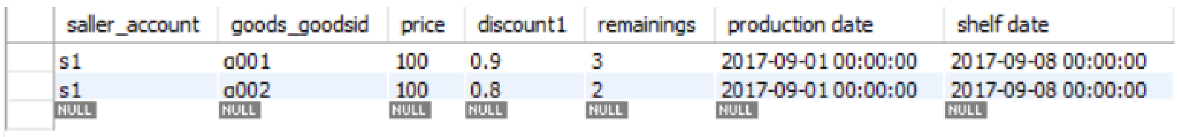


saller_account	goods_goodsid	price	discount1	remainings	production date	shelf date
s1	g001	100	0.9	5	2017-09-01 00:00:00	2017-09-08 00:00:00
s1	g002	100	0.8	5	2017-09-01 00:00:00	2017-09-08 00:00:00
NULL	NULL	NULL	NULL	NULL	NULL	NULL

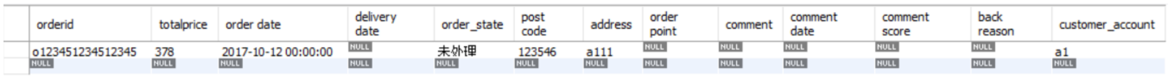


orderid	totalprice	order date	delivery date	order_state	post code	address	order point	comment	comment date	comment score	back reason	customer_account
o123451234512345	0	2017-10-12 00:00:00	NULL	NULL	123546	a111	NULL	NULL	NULL	NULL	NULL	a1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

插入订单内容后



saller_account	goods_goodsid	price	discount1	remainings	production date	shelf date
s1	g001	100	0.9	3	2017-09-01 00:00:00	2017-09-08 00:00:00
s1	g002	100	0.8	2	2017-09-01 00:00:00	2017-09-08 00:00:00
NULL	NULL	NULL	NULL	NULL	NULL	NULL

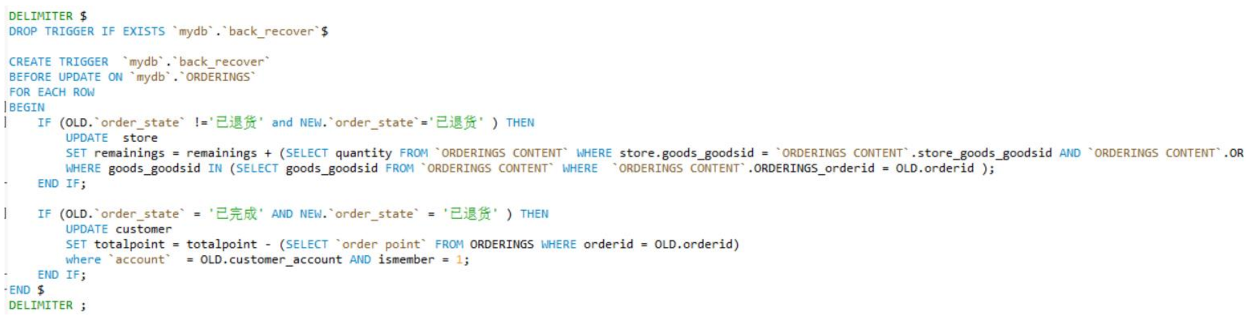


orderid	totalprice	order date	delivery date	order_state	post code	address	order point	comment	comment date	comment score	back reason	customer_account
o123451234512345	378	2017-10-12 00:00:00	NULL	未处理	123546	a111	NULL	NULL	NULL	NULL	NULL	a1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

结果符合预期

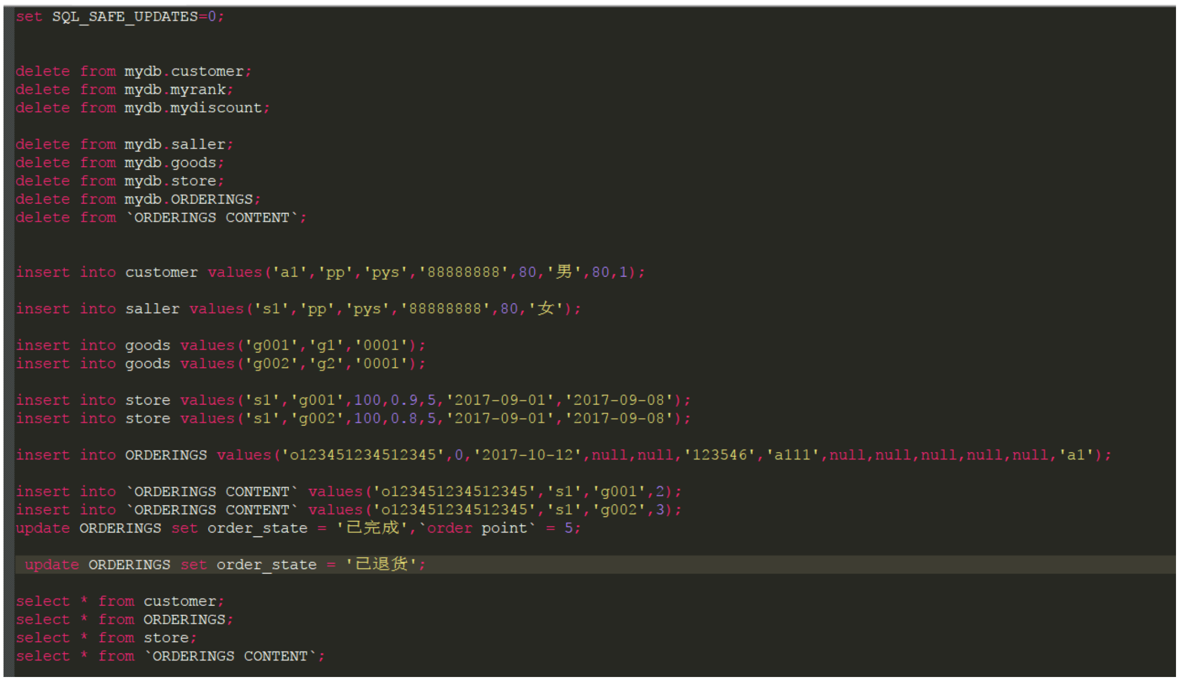
（3）　库存量在订单失效后恢复机制

思想：当订单状态更新为已退货后，需要将供应商-商品关系 store 中的库存量回滚，如果更新前的订单状态为已完成，还需要将顾客增加的积分回滚。



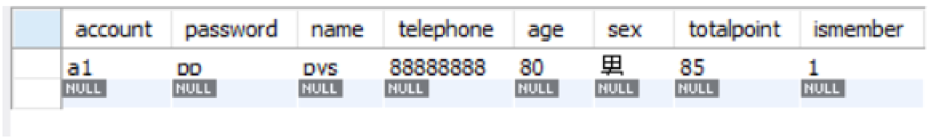
```
DELIMITER $
DROP TRIGGER IF EXISTS `mydb`.`back_recover`$

CREATE TRIGGER  `mydb`.`back_recover`
BEFORE UPDATE ON `mydb`.`ORDERINGS`
FOR EACH ROW
BEGIN
    IF (OLD.`order_state` !='已退货' and NEW.`order_state`='已退货' ) THEN
        UPDATE  store
        SET remainings = remainings + (SELECT quantity FROM `ORDERINGS CONTENT` WHERE store.goods_goodsid = `ORDERINGS CONTENT`.store_goods_goodsid AND `ORDERINGS CONTENT`.OR
        WHERE goods_goodsid IN (SELECT goods_goodsid FROM `ORDERINGS CONTENT` WHERE  `ORDERINGS CONTENT`.ORDERINGS_orderid = OLD.orderid );
    END IF;

    IF (OLD.`order_state` = '已完成' AND NEW.`order_state` = '已退货' ) THEN
        UPDATE customer
        SET totalpoint = totalpoint - (SELECT `order point` FROM ORDERINGS WHERE orderid = OLD.orderid)
        where `account`  = OLD.customer_account AND ismember = 1;
    END IF;
END $
DELIMITER ;
```

验证：

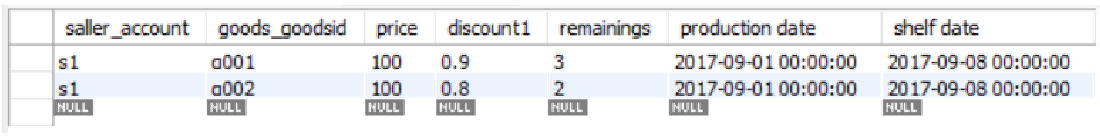


```
set SQL_SAFE_UPDATES=0;


delete from mydb.customer;
delete from mydb.myrank;
delete from mydb.mydiscount;

delete from mydb.saller;
delete from mydb.goods;
delete from mydb.store;
delete from mydb.ORDERINGS;
delete from `ORDERINGS CONTENT`;


insert into customer values('a1','pp','pys','88888888',80,'男',80,1);

insert into saller values('s1','pp','pys','88888888',80,'女');

insert into goods values('g001','g1','0001');
insert into goods values('g002','g2','0001');

insert into store values('s1','g001',100,0.9,5,'2017-09-01','2017-09-08');
insert into store values('s1','g002',100,0.8,5,'2017-09-01','2017-09-08');

insert into ORDERINGS values('o123451234512345',0,'2017-10-12',null,null,'123546','a111',null,null,null,null,null,'a1');

insert into `ORDERINGS CONTENT` values('o123451234512345','s1','g001',2);
insert into `ORDERINGS CONTENT` values('o123451234512345','s1','g002',3);
update ORDERINGS set order_state = '已完成',`order point` = 5;

 update ORDERINGS set order_state = '已退货';

select * from customer;
select * from ORDERINGS;
select * from store;
select * from `ORDERINGS CONTENT`;
```

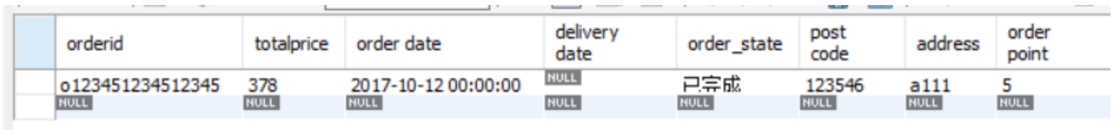
订单为已完成时



account	password	name	telephone	age	sex	totalpoint	ismember
a1	pp	pys	88888888	80	男	85	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

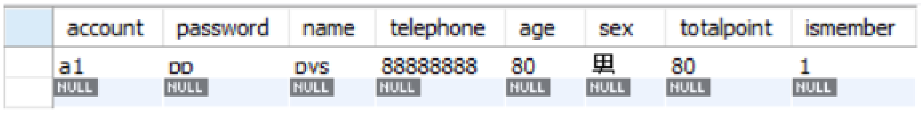


saller_account	goods_goodsid	price	discount1	remainings	production date	shelf date
s1	g001	100	0.9	3	2017-09-01 00:00:00	2017-09-08 00:00:00
s1	g002	100	0.8	2	2017-09-01 00:00:00	2017-09-08 00:00:00
NULL	NULL	NULL	NULL	NULL	NULL	NULL

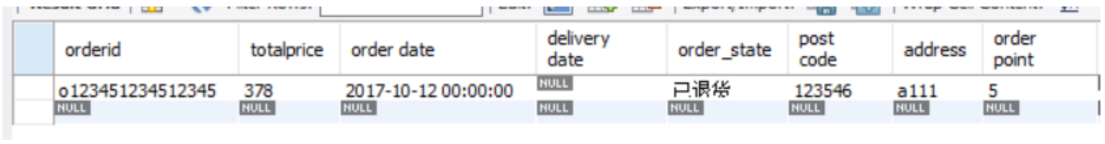


orderid	totalprice	order date	delivery date	order_state	post code	address	order point
o123451234512345	378	2017-10-12 00:00:00	NULL	已完成	123546	a111	5
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

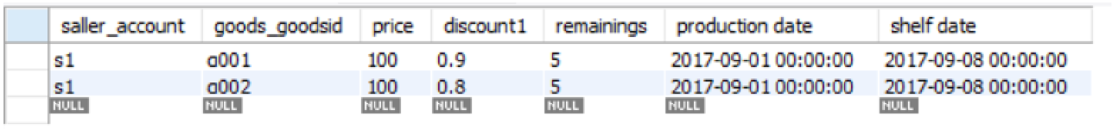
订单为已退货时



	account	password	name	telephone	age	sex	totalpoint	ismember
	a1	pp	pvs	88888888	80	男	80	1
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



	orderid	totalprice	order date	delivery date	order_state	post code	address	order point
	o123451234512345	378	2017-10-12 00:00:00	NULL	已退货	123546	a111	5
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

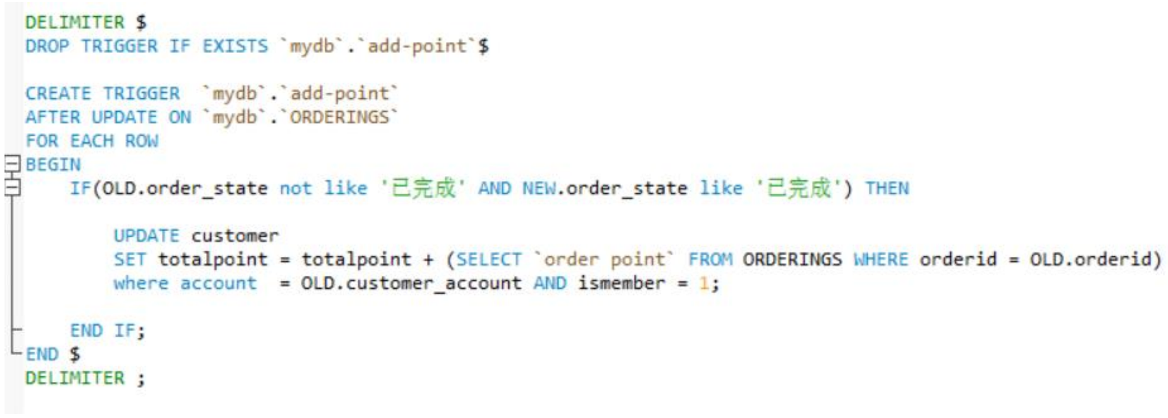


	saller_account	goods_goodsid	price	discount1	remainings	production date	shelf date
	s1	g001	100	0.9	5	2017-09-01 00:00:00	2017-09-08 00:00:00
	s1	g002	100	0.8	5	2017-09-01 00:00:00	2017-09-08 00:00:00
	NULL	NULL	NULL	NULL	NULL	NULL	NULL

结果符合预期

（4） 订单评价后的积分增加机制

思想：当订单状态更新为已完成时，需要对顾客积分进行更新

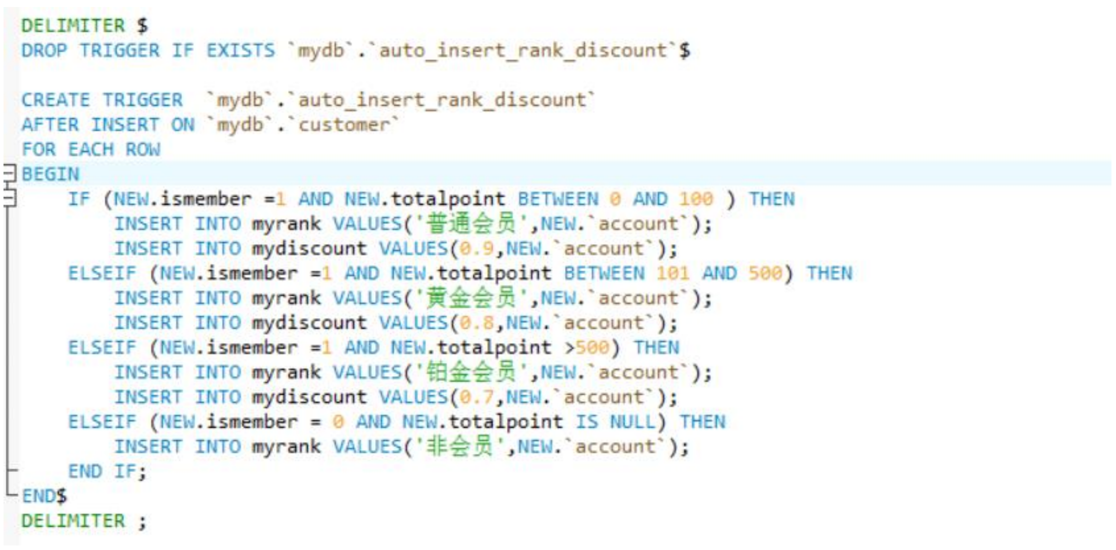


```
DELIMITER $
DROP TRIGGER IF EXISTS `mydb`.`add-point`$

CREATE TRIGGER  `mydb`.`add-point`
AFTER UPDATE ON `mydb`.`ORDERINGS`
FOR EACH ROW
BEGIN
    IF(OLD.order_state not like '已完成' AND NEW.order_state like '已完成') THEN

        UPDATE customer
        SET totalpoint = totalpoint + (SELECT `order point` FROM ORDERINGS WHERE orderid = OLD.orderid)
        where account  = OLD.customer_account AND ismember = 1;

    END IF;
END $
DELIMITER ;
```

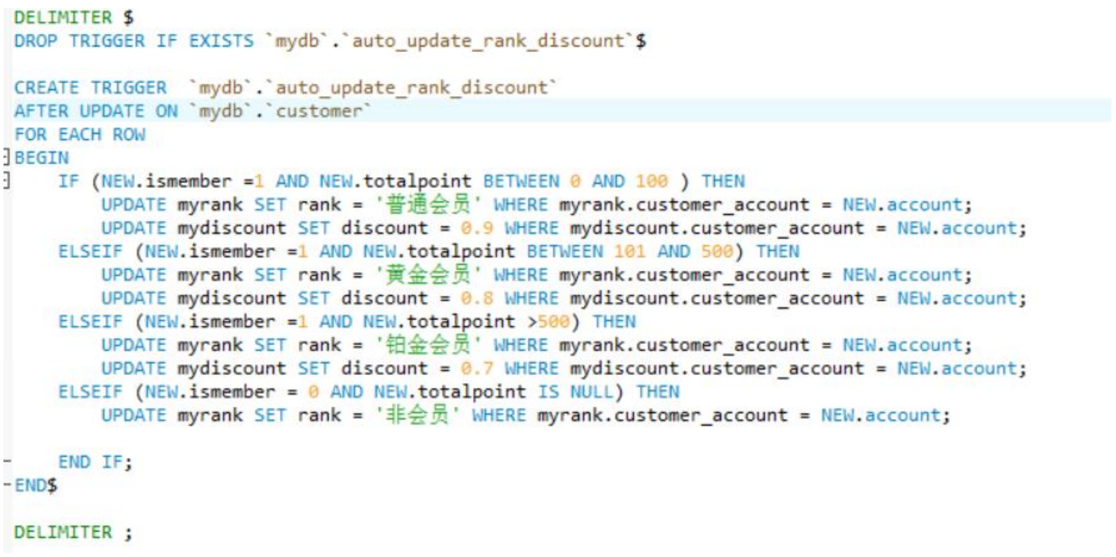
验证已经在（3）中完成；

（5） 会员积分与等级、折扣率的对应机制

思想：在插入顾客信息和更新顾客积分时，都需要相应的插入和更新 myrank 和 mydiscount 表，保证信息的一致性。



```
DELIMITER $
DROP TRIGGER IF EXISTS `mydb`.`auto_insert_rank_discount`$

CREATE TRIGGER  `mydb`.`auto_insert_rank_discount`
AFTER INSERT ON `mydb`.`customer`
FOR EACH ROW
BEGIN
    IF (NEW.ismember =1 AND NEW.totalpoint BETWEEN 0 AND 100 ) THEN
        INSERT INTO myrank VALUES('普通会员',NEW.`account`);
        INSERT INTO mydiscount VALUES(0.9,NEW.`account`);
    ELSEIF (NEW.ismember =1 AND NEW.totalpoint BETWEEN 101 AND 500) THEN
        INSERT INTO myrank VALUES('黄金会员',NEW.`account`);
        INSERT INTO mydiscount VALUES(0.8,NEW.`account`);
    ELSEIF (NEW.ismember =1 AND NEW.totalpoint >500) THEN
        INSERT INTO myrank VALUES('铂金会员',NEW.`account`);
        INSERT INTO mydiscount VALUES(0.7,NEW.`account`);
    ELSEIF (NEW.ismember = 0 AND NEW.totalpoint IS NULL) THEN
        INSERT INTO myrank VALUES('非会员',NEW.`account`);
    END IF;
END$
DELIMITER ;
```



```
DELIMITER $
DROP TRIGGER IF EXISTS `mydb`.`auto_update_rank_discount`$

CREATE TRIGGER  `mydb`.`auto_update_rank_discount`
AFTER UPDATE ON `mydb`.`customer`
FOR EACH ROW
BEGIN
    IF (NEW.ismember =1 AND NEW.totalpoint BETWEEN 0 AND 100 ) THEN
        UPDATE myrank SET rank = '普通会员' WHERE myrank.customer_account = NEW.account;
        UPDATE mydiscount SET discount = 0.9 WHERE mydiscount.customer_account = NEW.account;
    ELSEIF (NEW.ismember =1 AND NEW.totalpoint BETWEEN 101 AND 500) THEN
        UPDATE myrank SET rank = '黄金会员' WHERE myrank.customer_account = NEW.account;
        UPDATE mydiscount SET discount = 0.8 WHERE mydiscount.customer_account = NEW.account;
    ELSEIF (NEW.ismember =1 AND NEW.totalpoint >500) THEN
        UPDATE myrank SET rank = '铂金会员' WHERE myrank.customer_account = NEW.account;
        UPDATE mydiscount SET discount = 0.7 WHERE mydiscount.customer_account = NEW.account;
    ELSEIF (NEW.ismember = 0 AND NEW.totalpoint IS NULL) THEN
        UPDATE myrank SET rank = '非会员' WHERE myrank.customer_account = NEW.account;

    END IF;
END$

DELIMITER ;
```

5. Survey（不算分、选做）

关于上机课的这种形式（上机作业的次数、难度和时间安排等方面），大家是否有收获，欢迎同学们对不足的地方给出批评和改进的建议，感谢大家的配合！

优点：上机次数适中，对于第一，三，四次上机，难度适中，与所学内容紧密相关，学到了不少东西。
缺点：第二次上机与课程严重脱离；每次上机时间 DDL 可以延后，但是 demo 可以取消；上机尽量给出基本教程。问题叙述尽量明确，减少歧义。