BÀI TẬP

Một khách hàng có thể có nhiều hóa đơn, một hóa đơn chỉ thuộc một khách hàng. Một hóa đơn có thể mua nhiều sản phẩm và một sản phẩm có thể bán trên nhiều hóa đơn.

Cấu trúc dữ liệu json của các đối tượng như sau:

Customer.json


```
{
	"_id" : "AGSI14215",
	"firstName" : "Agnes",
	"lastName" : "Sims",
	"address" : {
		"city" : "Buffalo",
		"state" : "NY",
		"street" : "170 Queen Lane ",
		"zipCode" : "14215"
	},
	"registrationDate" : "1999-11-02",
	"email" : "agnes.sims@aol.com",
	"phones" : [
		{
			"number" : "799 724-303",
			"type" : "home"
		},
		{
			"number" : "33 556-775",
			"type" : "personal"
		}
	]
}
```


Order.json


```
{
	"_id" : NumberLong(2),
	"shippingAddress" : {
		"city" : "Oswego",
		"zipCode" : "13126",
		"street" : "7800 Magnolia Street ",
		"state" : "NY"
	},
	"customerID" : "ELDE13126",
	"orderDate" : "2016-04-15",
	"orderDetails" : [
		{
```

```
                "quantity" : 2,
                "color" : "blue",
                "productID" : NumberLong(16),
                "lineTotal" : 1139.981,
                "price" : 599.99,
                "discount" : 0.05000000074505806
            },
            {
                "quantity" : 1,
                "color" : "blue",
                "productID" : NumberLong(20),
                "lineTotal" : 557.9907,
                "price" : 599.99,
                "discount" : 0.07000000029802322
            }
        ]
}
```


Product.json


```
{
        "_id" : NumberLong(4),
        "price" : 469.99,
        "description" : "Brand: Surly, Category: Mountain Bikes",
        "modelYear" : 2016,
        "productName" : "Surly Ice Cream Truck Frameset - 2016"
}
```


Import các file json cho sẵn vào database có tên bikedb với tên các collection tương ứng.

Lập trình Mongodb Reactive Stream Driver, viết các phương thức với các yêu cầu sau:

1/ Tìm danh sách sản phẩm có giá cao nhất.

```
db.products.aggregate([{$group:{_id:null, maxPrice:{$max:"$price"}}},  {$lookup:{from:"products", localField:"maxPrice", foreignField:"price", as:"result"}},  {$project:{_id:0, result:1}}, {$unwind:"$result"},{$replaceRoot:{newRoot:"$result"}}]).pretty()
```

2/ Tìm danh sách sản phẩm chưa bán được lần nào.

```
db.products.aggregate([
{$lookup:{from:"orders", localField:"_id", foreignField:"orderDetails.productID", as:"listOrder"}},
{$project:{price:1,description:1, modelYear:1, productName:1, size:{$size:"$listOrder"}}},
{$match:{size:0}}])
```

3/ Thống kê số khách hàng theo từng bang.

+getNumberCustomerByState() : Map<String, Integer>

```
db.customers.aggregate({$group:{_id:"$address.state", num:{$sum:1}}})
```

4/ Tính tổng tiền của đơn hàng khi biết mã số đơn hàng.

```
db.orders.aggregate([{$match:{_id:NumberLong(2)}},
                     {$project:{total:{$sum:"$orderDetails.lineTotal"}}}] )
```

5/ Đếm số đơn hàng của từng khách hàng.

+ getOrdersByCustomers() : Map<Customer, Integer>

```
db.customers.aggregate([
{$lookup:{from:"orders", localField:"_id", foreignField:"customerID", as:"rs"}},
{$project:{n:{$size:"$rs"}}},
{$match:{n:{$gt:0}}}] ).pretty()
```

6/ Tính tổng số lượng của từng sản phẩm đã bán ra.

+ getTotalProduct(): Map<Product, Integer>

```
db.orders.aggregate({$project:{_id:0, orderDetails:1}},
{$unwind:"$orderDetails"},
{$replaceRoot:{newRoot:"$orderDetails"}},
{$group:{_id:"$productID", sum:{$sum:"$quantity"}}},
{$lookup:{from:"products", localField:"_id", foreignField:"_id",as:"rs"}},
{$unwind:"$rs"}).pretty()
```

7/ Dùng text search để tìm kiếm sản phẩm theo tên sản phẩm và mô tả của sản phẩm.

```
db.products.createIndex({productName:"text", description:"text"})
db.products.find({$text:{$search:"Go! Ice"}})
```

8/ Tính tổng tiền của tất cả các hóa đơn trong một ngày nào đó.

```
db.orders.aggregate({$match:{orderDate:{$regex:/2016-04-15/}}},
{$project:{total:{$sum:"$orderDetails.lineTotal"}}},
{$group:{_id:null, sum:{$sum:"$total"}}},
{$project:{_id:0}}).pretty()
```

9/ Thêm dữ liệu vào từng collection.

```
db.products.insertOne(…)
db.customers. insertOne(…)
db.orders. insertOne(…)
```

10/ Cập nhật giá của sản phẩm khi biết mã sản phẩm.

```
db.products.updateOne({_id:1},{$set:{price:100}} )
```

11/ Xóa tất cả các khách hàng chưa mua hàng.

```
db.customers.aggregate([
{$lookup:{from:"orders", localField:"_id", foreignField:"customerID", as:"rs"}},
{$project:{num:{$size:"$rs"}}},
{$match:{num:0}},
{$unset:"num"}])
```

12/ Tìm sản phẩm khi biết mã sản phẩm

```
db.products.find({_id:1})
```

Entity Class

```
public class Address {

        private String city;
        private String state;
        private String street;
        private String zipCode;

        public Address() {
        }

}
```

```
public class Phone {

        private String number;
        private String type;

        public Phone() {
        }
}
```

```
public class OrderDetail {

        private int quantity;
        private String color;
        private Product product;
        private double lineTotal;
        private double price;
        private double discount;

        public OrderDetail() {
        }
```

```
public class Product {

        private Long _id;
        private double price;
        private String description;
        private int modelYear;
        private String productName;

        public Product() {
        }

}
```

```
        public OrderDetail(int quantity, String
        color, Product product, double price,
        double discount) {
                this.quantity = quantity;
                this.color = color;
                this.product = product;
                this.price = price;
                this.discount = discount;
                this.lineTotal = quantity * price *
        (1 - discount);
        }
}
```

```
public class Customer {

        private String _id;
        private String firstName;
        private String lastName;
        private Address address;
        private LocalDate registrationDate;
        private String email;
        private List<Phone> phones;

        public Customer() {
        }
}
```

```
public class Order {

        private Long _id;
        private Address shippingAddress;
        private Customer customer;
        private LocalDate orderDate;
        private List<OrderDetail> orderDetails;

        public Order() {
                this(0l,new Address());
        }

        /**
         * @param _id
         * @param shippingAddress
         */
        public Order(Long _id, Address shippingAddress) {
                this._id = _id;
                this.shippingAddress = shippingAddress;
                orderDate = LocalDate.now();
                orderDetails = new ArrayList<OrderDetail>();
        }


        /**
         * @return the orderDetails
         */
        public List<OrderDetail> getOrderDetails() {
                return orderDetails;
        }

        /**
         * Add one OrderDetail
```

```
         * @param quantity
         * @param color
         * @param product
         * @param price
         * @param discount
         */
        public void addOrderDetail(int quantity, String color, Product product, double price, double
discount) {
                OrderDetail orderDetail = new OrderDetail(quantity, color, product, price, discount);
                this.getOrderDetails().add(orderDetail);
        }
}
```

Full Source Code: http://www.mediafire.com/file/zdrvx2l5kqiij87/Week04_Sol.rar