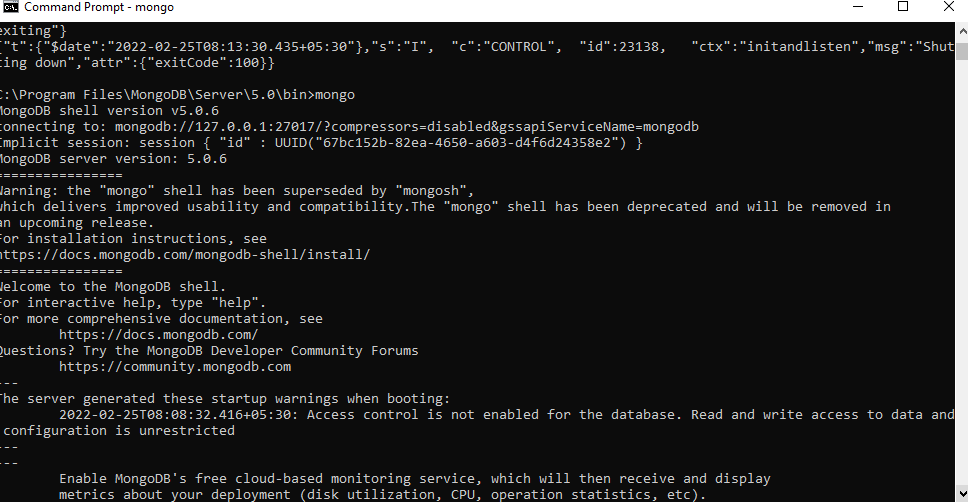
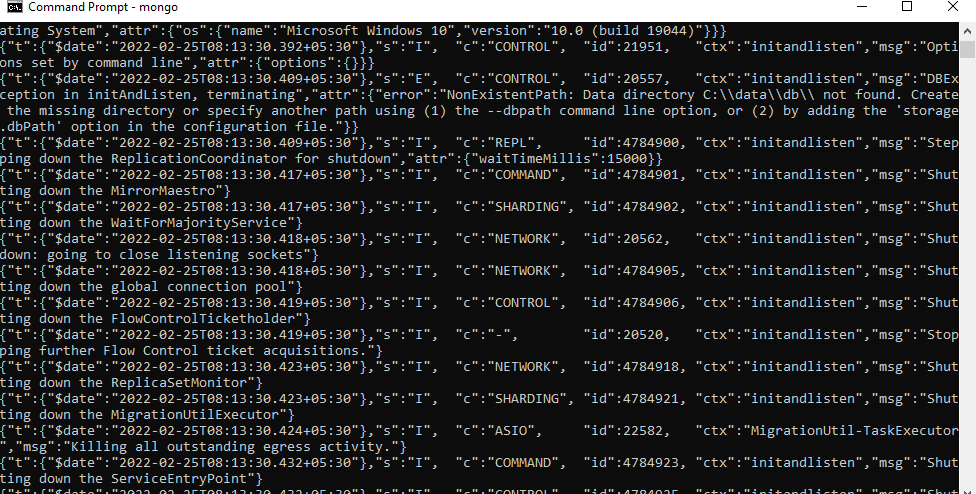
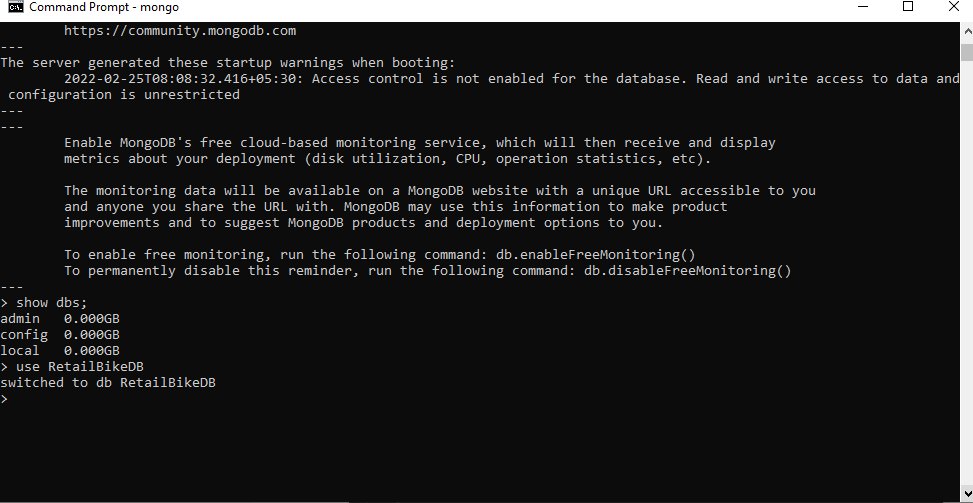
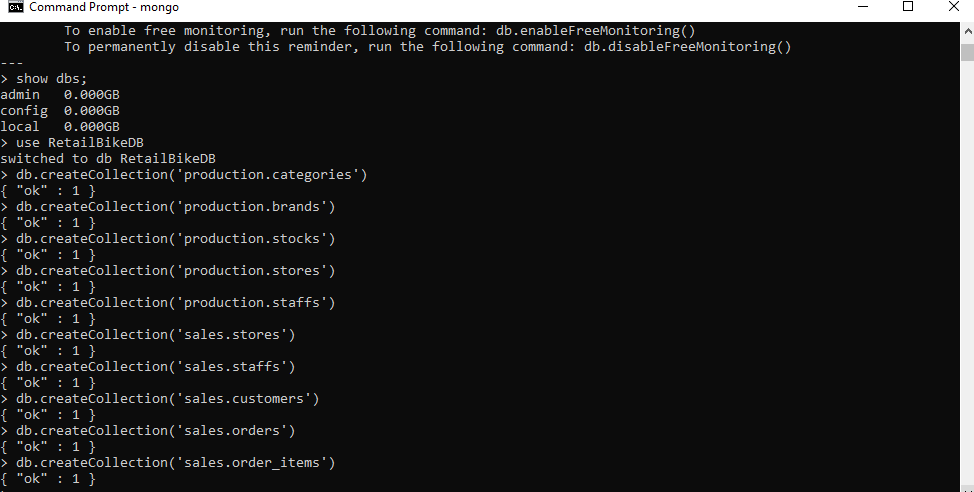
# Subject: Big Data Technology Practical\_6: Handling Unstructured

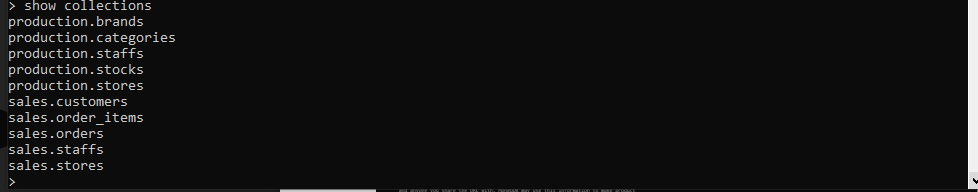
**Open the command prompt and run the cd command to change directory and run the command “mongod” to start the server and run the command “mongo” to work mongoDB.**



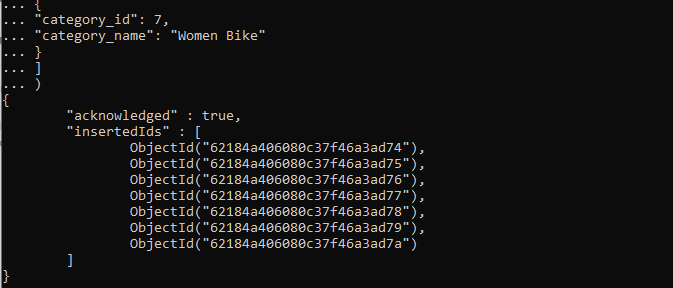
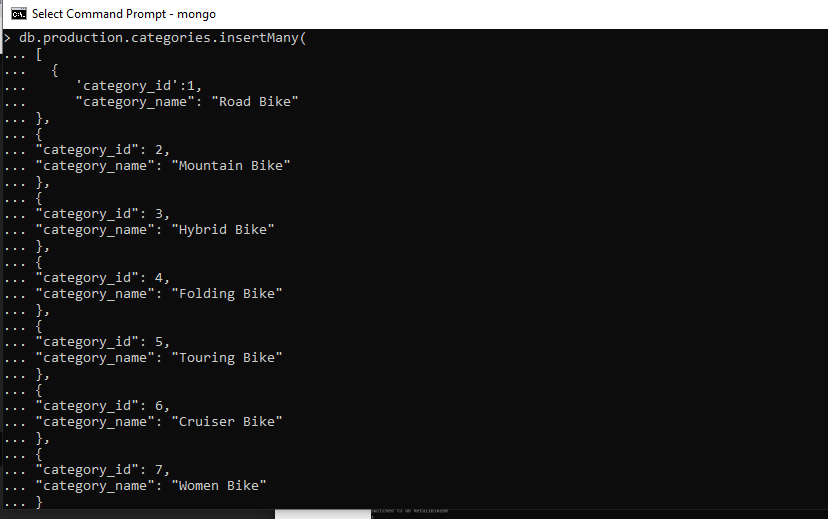
**Show list of databases**

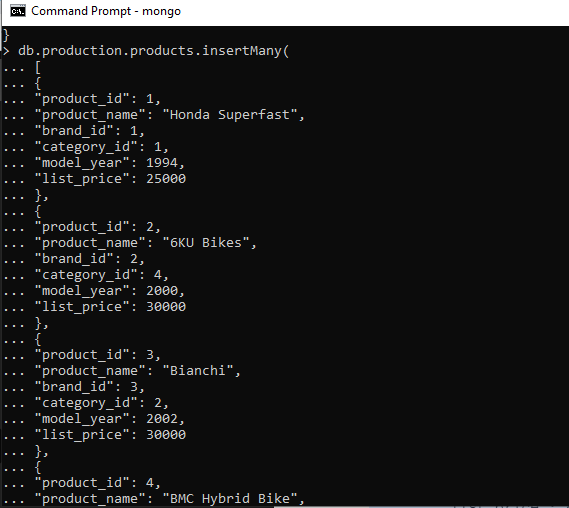
**Creation of collection can be done using db.createCollection(name)**

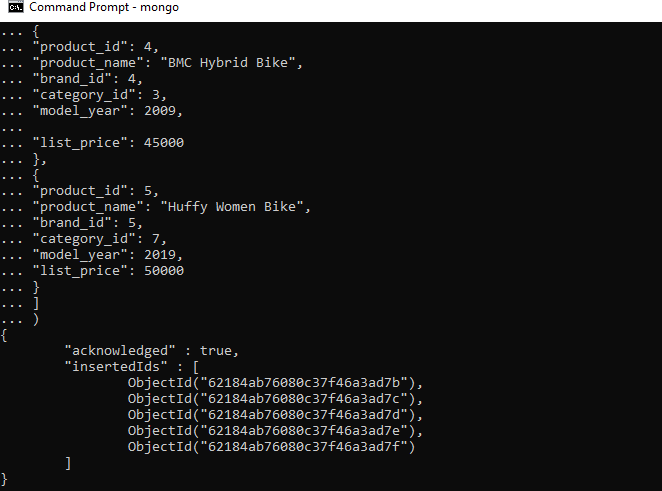
**We can show list of collection using “Show collections” commands in MongoDB.**

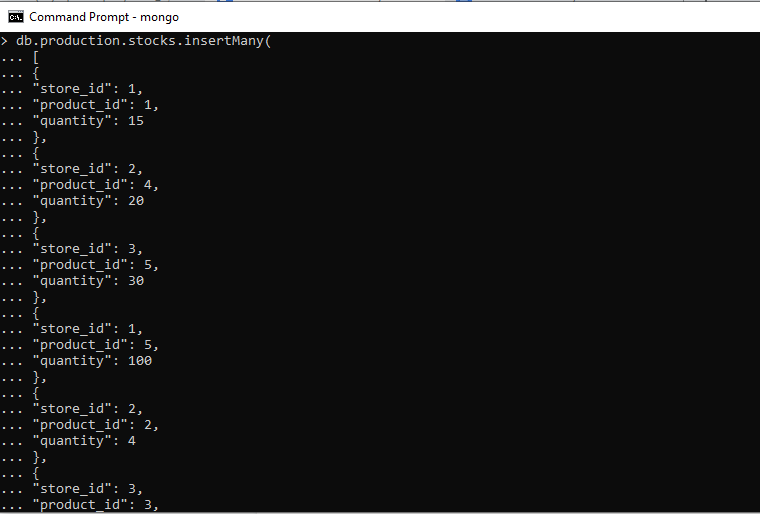


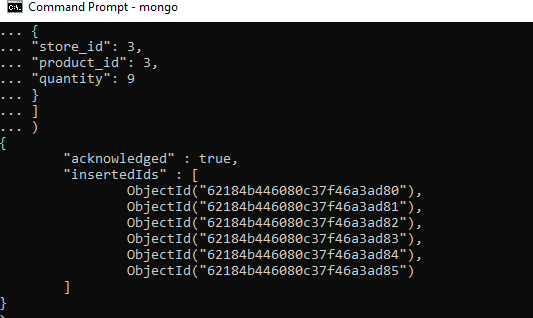
1. **production.categories**



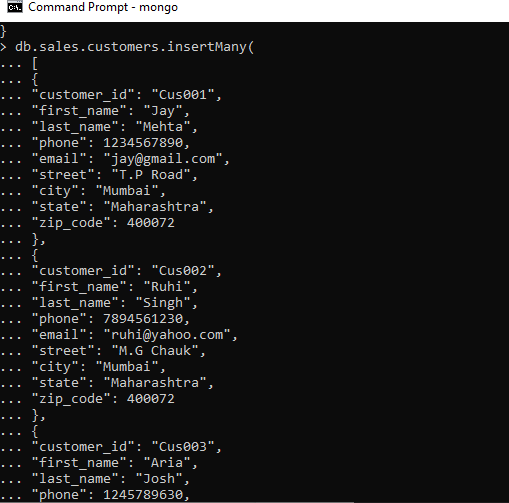
1. **production.products**

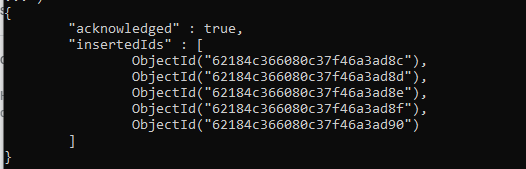
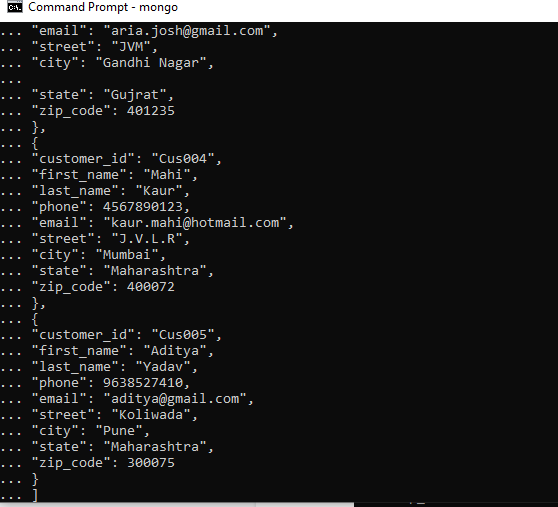


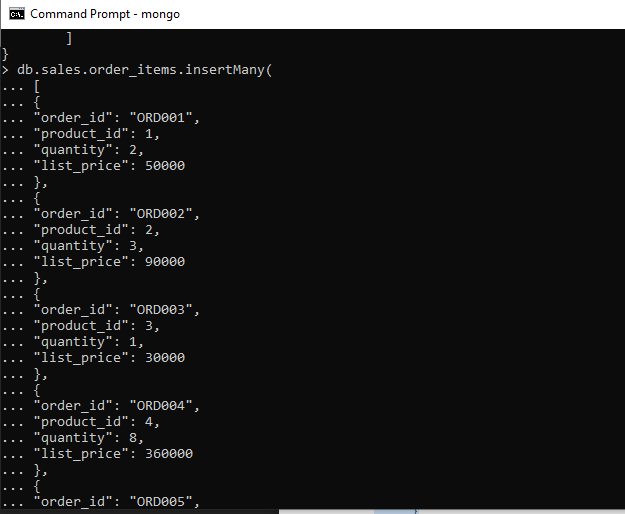
1. **production.stocks**

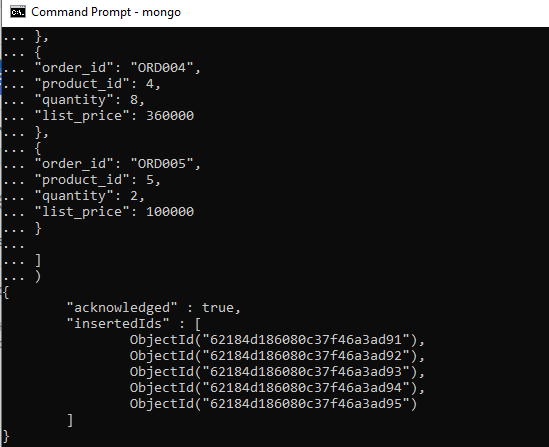


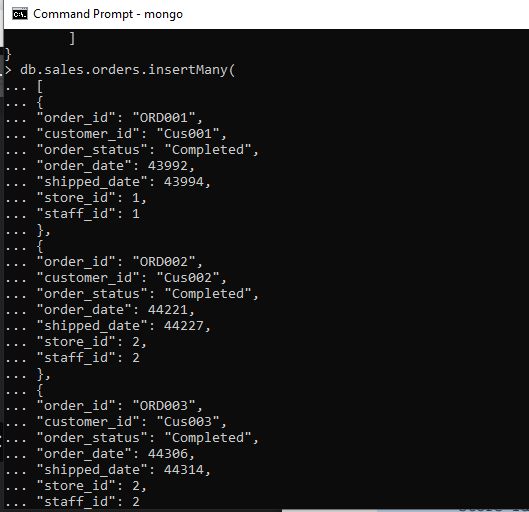
1. **sales.customers**

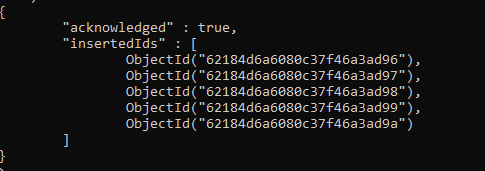
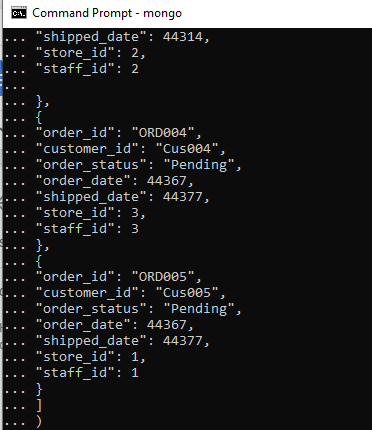




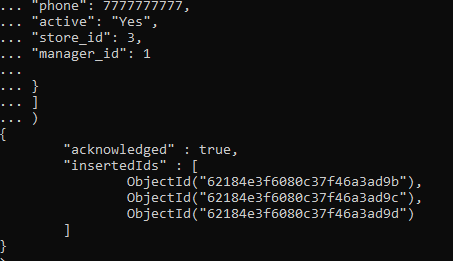
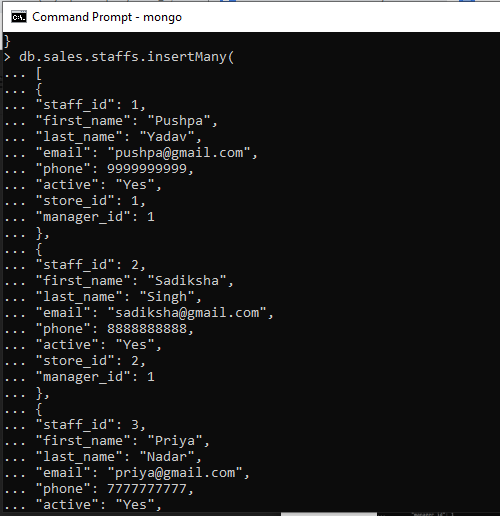
1. **sales.order\_items**



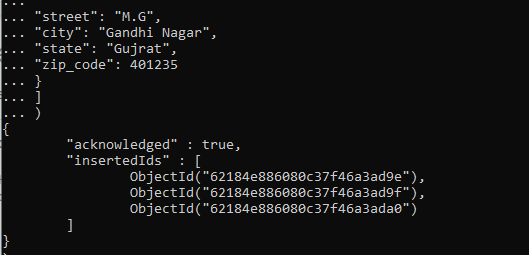
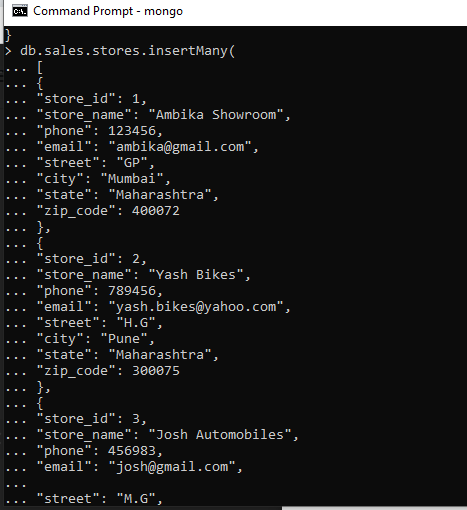
1. **Sales.orders**



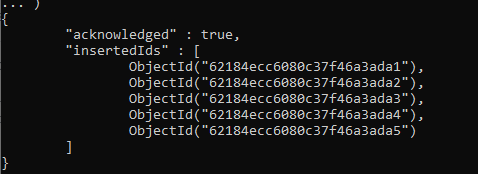
1. **sales.staffs**



1. **sales.stores**



1. **Production.brands**



If you want to get more specific with a read operation and find a desired subsection of the records, you can use the previously mentioned filtering criteria to choose what results should be returned. One of the most common ways of filtering the results is to search by value.

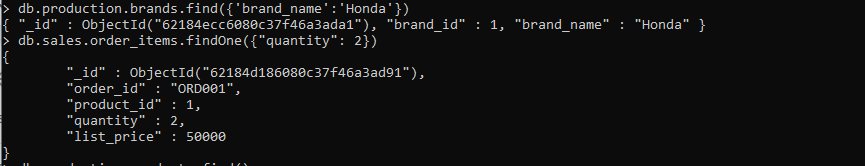
Example - db.production.brands.find({"brand\_name" : "Honda"})

findOne()

In order to get one document that satisfies the search criteria, we can simply use the findOne() method on our chosen collection. If multiple documents satisfy the query, this method returns the first document according to the natural order which reflects the order of documents on the disk. If

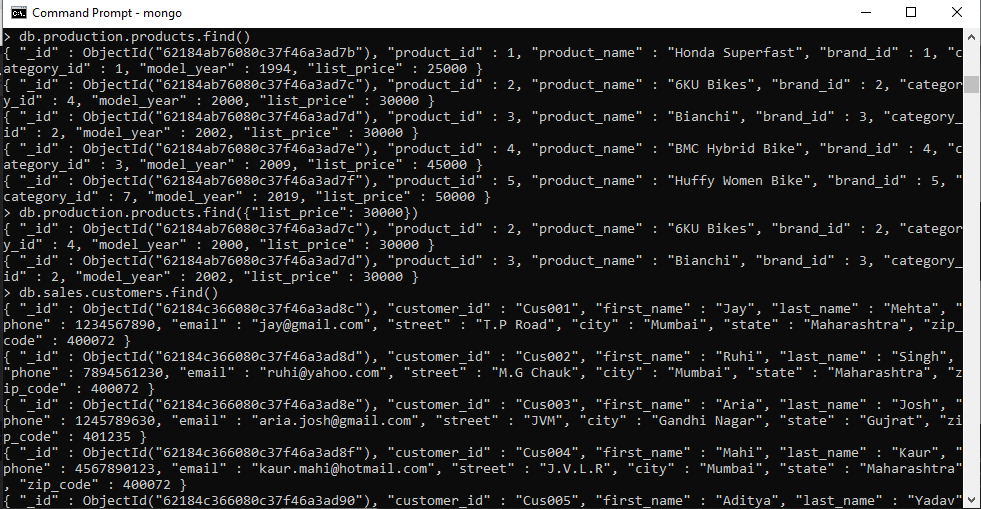
no documents satisfy the search criteria, the function returns null. The function takes the following form of syntax.

Syntax- db.{collection}.findOne({query}, {projection}) Example – db.sales.order\_items.findOne({"quantity": 2})



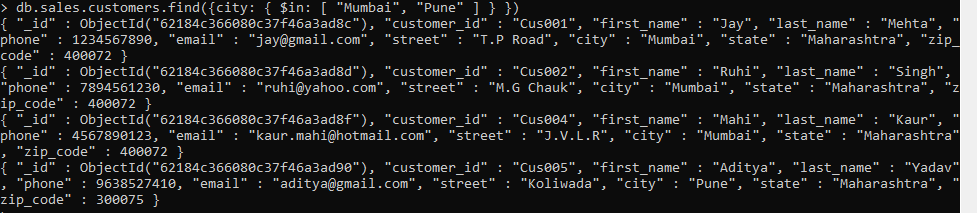
1. Specify Equality Condition

* db.production.products.find() - to show all the documents
* db.production.products.find({"list\_price": 30000}) - to show only documents which have “list\_price” as 30000



1. Specify Conditions Using Query Operators

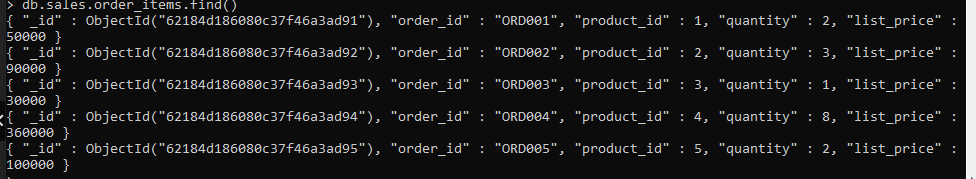
* db.sales.customers.find() - to show all the documents
* db.sales.customers.find({city: { $in: [ "Mumbai", "Pune" ] } }) - to show all the documents where "city" is either "Mumbai" or "Pune"



1. Specify AND Conditions

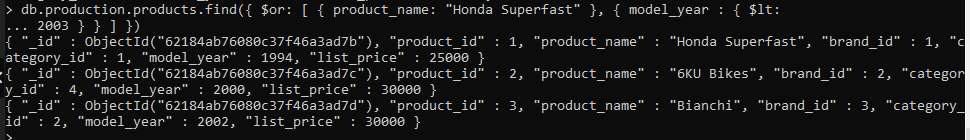
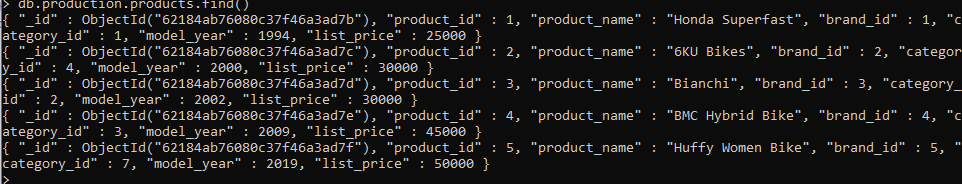
* db.sales.order\_items.find() - to show all the documents
* db.sales.order\_items.find({quantity: 2, list\_price : { $gt: 70000}}) - Here we are trying to find how

sales order item documents for which quantity is 2 and list price is greater than 70000



1. Specify OR Conditions

* db.production.products.find() - to show all the documents
* db.production.products.find({ $or: [ { product\_name: "Honda Superfast" }, { model\_year : { $lt: 2003 } } ] }) - to show all the document which is either "product name" as "Honda Superfast" or "model year" is less than year 2003



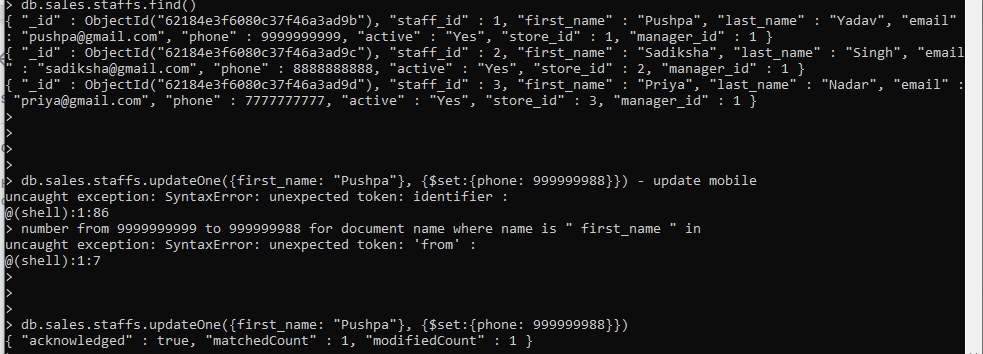
updateOne()

We can update a currently existing record and change a single document with an update operation. To do this, we use the updateOne() method on a chosen collection. To update a document, we provide the method with two arguments: an update filter and an update action.

The update filter defines which items we want to update, and the update action defines how to update those items. We first pass in the update filter. Then, we use the “$set” key and provide the fields we want to update as a value. This method will update the first record that matches the provided filter.

Example –

* db.sales.staffs.find() - to show currect document in the system
* db.sales.staffs.updateOne({first\_name: "Pushpa"}, {$set:{phone: 999999988}}) - update mobile number from 9999999999 to 999999988 for document name where name is " first\_name " in collection

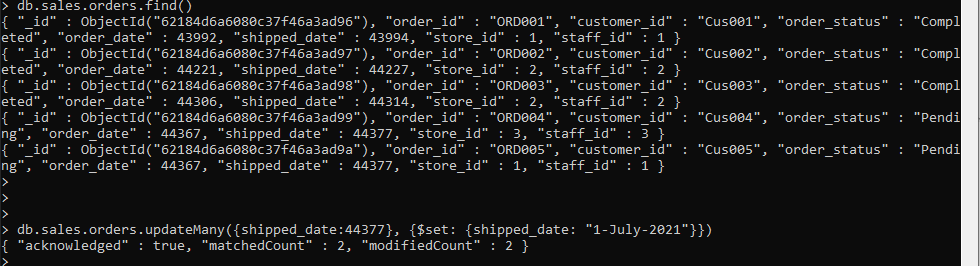


updateMany() allows us to update multiple items by passing in a list of items, just as we did when inserting multiple items. This update operation uses the same syntax for updating a single document.

Example –

db.sales.orders.find() - to show current document in the system db.sales.orders.updateMany({shipped\_date:44377}, {$set: {shipped\_date: "1-July-2021"}}) – with

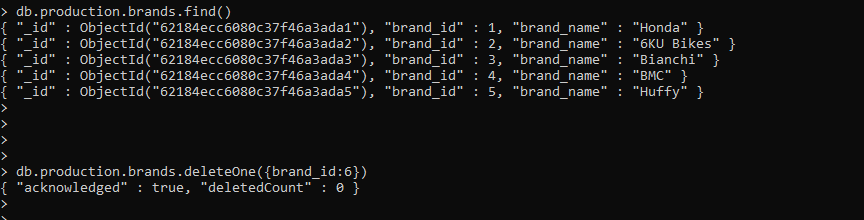
the help of this command we are updating “shipped\_date” to "1-July-2021" where shipped\_date is 44377



deleteOne()

deleteOne() is used to remove a document from a specified collection on the MongoDB server. A filter criteria is used to specify the item to delete. It deletes the first record that matches the provided filter.

Example –

* db.production.brands.find() - to show current documents in collection
* db.production.brands.deleteOne({brand\_id:6}) - delete the document where "brand\_id" is 6

deleteMany()

deleteMany() is a method used to delete multiple documents from a desired collection with a single delete operation. A list is passed into the method and the individual items are defined with filter criteria as in deleteOne().

Example –

* db.production.brands.find() - to show current documents in collection
* db.production.brands.deleteMany({brand\_id: {$gt: 5}}) - delete documents for which brand id is greater than 5

