MONGO Database (No SQL)

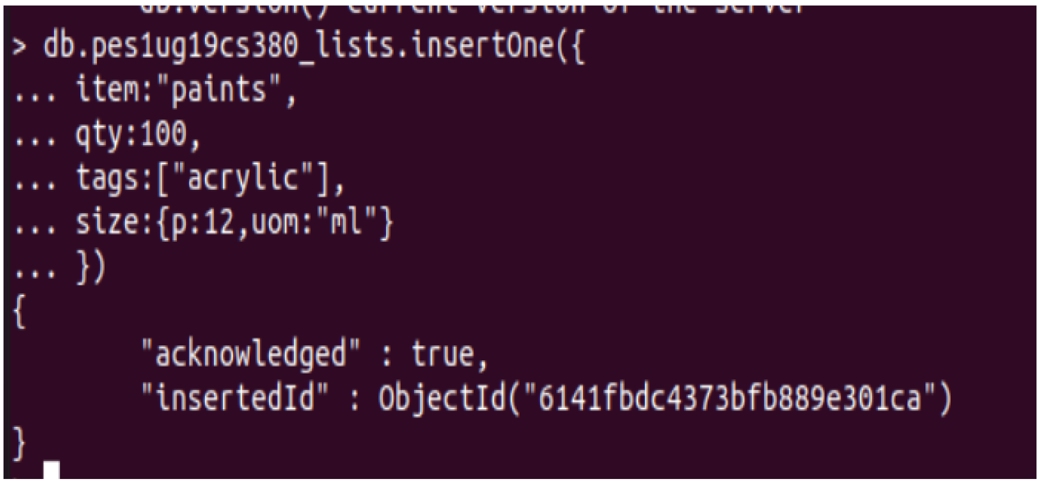
Week #1

Problem Statement	Use MongoDB and create a sample database and perform the following operations on it 1. Create documents with varying fields. 2. Use the insert and save commands. 3. Drop an existing database and a collection and use remove() to delete documents. 4. Update an existing document in a collection. (update()) 5. Use find() and findOne() for querying the database.

1.Create Operations

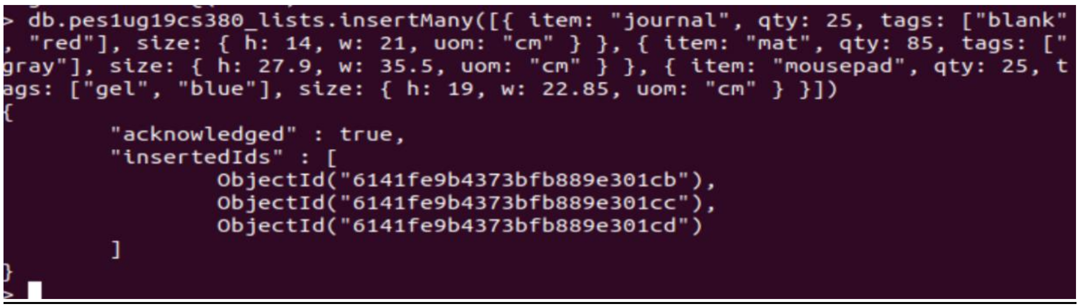
Create or insert operations add new documents to a collection. If the collection does not currently exist, insert operations will create the collection

Insert one :



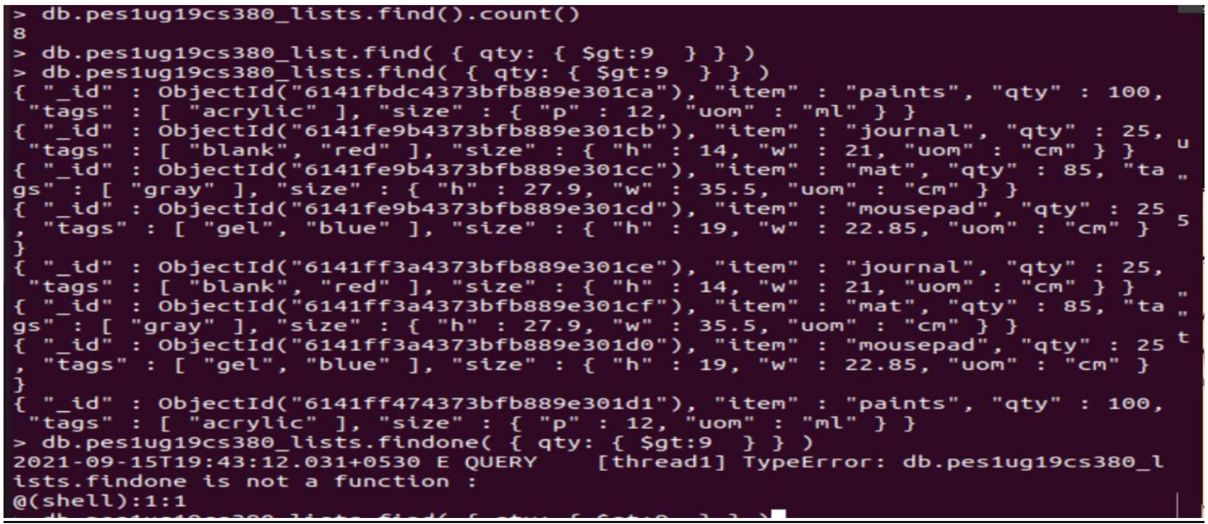
```
> db.pes1ug19cs380_lists.insertOne({
... item:"paints",
... qty:100,
... tags:["acrylic"],
... size:{p:12,uom:"ml"}
... })
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6141fbdc4373bfb889e301ca")
}
```

Insert Many:



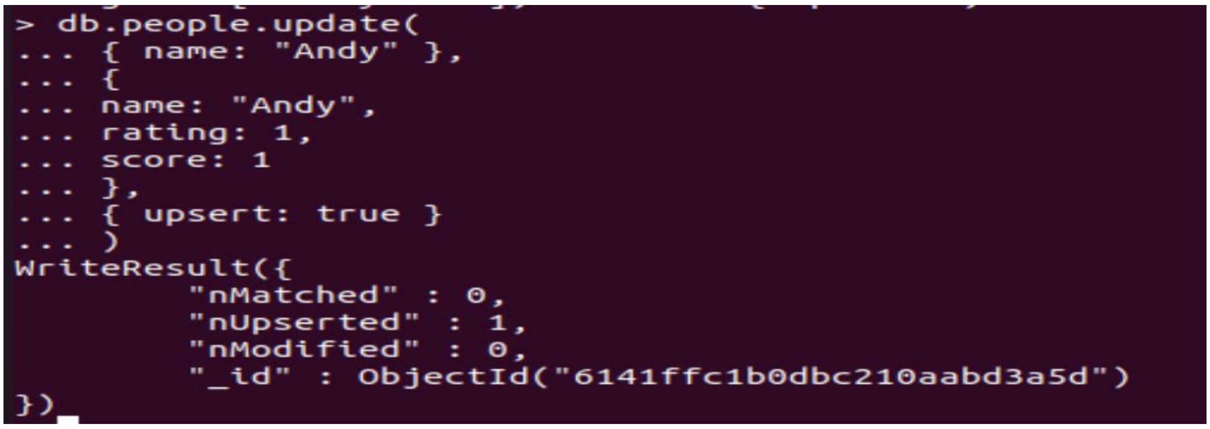
```
> db.pes1ug19cs380_lists.insertMany([{ item: "journal", qty: 25, tags: ["blank"
, "red"], size: { h: 14, w: 21, uom: "cm" } }, { item: "mat", qty: 85, tags: ["
gray"], size: { h: 27.9, w: 35.5, uom: "cm" } }, { item: "mousepad", qty: 25, t
ags: ["gel", "blue"], size: { h: 19, w: 22.85, uom: "cm" } }])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("6141fe9b4373bfb889e301cb"),
                ObjectId("6141fe9b4373bfb889e301cc"),
                ObjectId("6141fe9b4373bfb889e301cd")
        ]
}
>
```

2. Read Operations



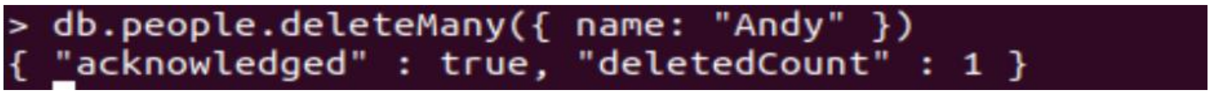
```
> db.pes1ug19cs380_lists.find().count()
8
> db.pes1ug19cs380_list.find( { qty: { $gt:9  } } )
> db.pes1ug19cs380_lists.find( { qty: { $gt:9  } } )
{ "_id" : ObjectId("6141fbdc4373bfb889e301ca"), "item" : "paints", "qty" : 100,
 "tags" : [ "acrylic" ], "size" : { "p" : 12, "uom" : "ml" } }
{ "_id" : ObjectId("6141fe9b4373bfb889e301cb"), "item" : "journal", "qty" : 25,
 "tags" : [ "blank", "red" ], "size" : { "h" : 14, "w" : 21, "uom" : "cm" } }
{ "_id" : ObjectId("6141fe9b4373bfb889e301cc"), "item" : "mat", "qty" : 85, "ta
gs" : [ "gray" ], "size" : { "h" : 27.9, "w" : 35.5, "uom" : "cm" } }
{ "_id" : ObjectId("6141fe9b4373bfb889e301cd"), "item" : "mousepad", "qty" : 25
, "tags" : [ "gel", "blue" ], "size" : { "h" : 19, "w" : 22.85, "uom" : "cm" }
}
{ "_id" : ObjectId("6141ff3a4373bfb889e301ce"), "item" : "journal", "qty" : 25,
 "tags" : [ "blank", "red" ], "size" : { "h" : 14, "w" : 21, "uom" : "cm" } }
{ "_id" : ObjectId("6141ff3a4373bfb889e301cf"), "item" : "mat", "qty" : 85, "ta
gs" : [ "gray" ], "size" : { "h" : 27.9, "w" : 35.5, "uom" : "cm" } }
{ "_id" : ObjectId("6141ff3a4373bfb889e301d0"), "item" : "mousepad", "qty" : 25
, "tags" : [ "gel", "blue" ], "size" : { "h" : 19, "w" : 22.85, "uom" : "cm" }
}
{ "_id" : ObjectId("6141ff474373bfb889e301d1"), "item" : "paints", "qty" : 100,
 "tags" : [ "acrylic" ], "size" : { "p" : 12, "uom" : "ml" } }
> db.pes1ug19cs380_lists.findone( { qty: { $gt:9  } } )
2021-09-15T19:43:12.031+0530 E QUERY    [thread1] TypeError: db.pes1ug19cs380_l
ists.findone is not a function :
@(shell):1:1
```

3.Update Operations



```
> db.people.update(
... { name: "Andy" },
... {
... name: "Andy",
... rating: 1,
... score: 1
... },
... { upsert: true }
... )
WriteResult({
        "nMatched" : 0,
        "nUpserted" : 1,
        "nModified" : 0,
        "_id" : ObjectId("6141ffc1b0dbc210aabd3a5d")
})
```

4.Delete Operations

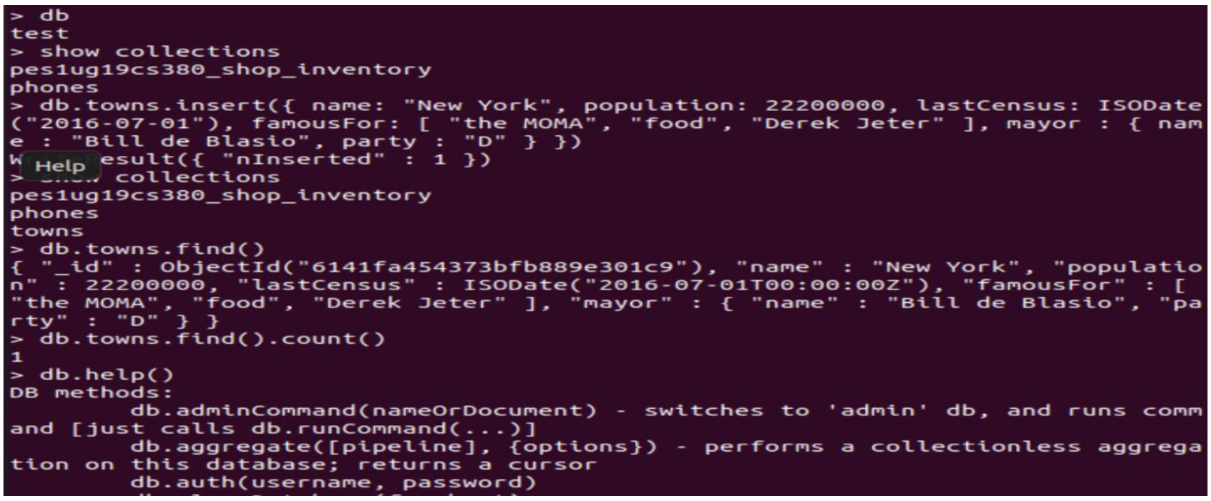


```
> db.people.deleteMany({ name: "Andy" })
{ "acknowledged" : true, "deletedCount" : 1 }
```

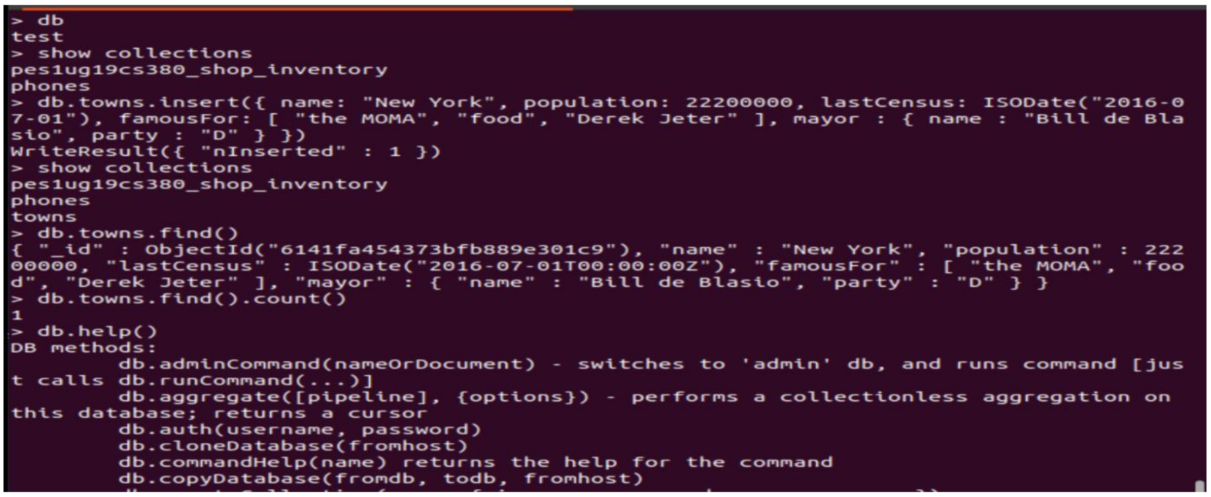
Mongo DB Demo:

Making 4 collections[towns, PES1ug19cs380_studentlist phones, pes1ug19cs380_shop_inventory], and applying the prior mentioned methods to execute queries and perform CRUD operations create, read, update, and delete on these collections

/*Create*/

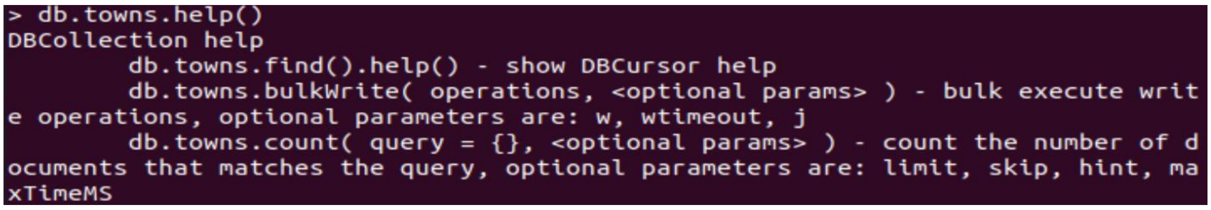


```
> db
test
> show collections
pes1ug19cs380_shop_inventory
phones
> db.towns.insert({ name: "New York", population: 22200000, lastCensus: ISODate
("2016-07-01"), famousFor: [ "the MOMA", "food", "Derek Jeter" ], mayor : { nam
e : "Bill de Blasio", party : "D" } })
W      esult({ "nInserted" : 1 })
>      collections
pes1ug19cs380_shop_inventory
phones
towns
> db.towns.find()
{ "_id" : ObjectId("6141fa454373bfb889e301c9"), "name" : "New York", "populatio
n" : 22200000, "lastCensus" : ISODate("2016-07-01T00:00:00Z"), "famousFor" : [
"the MOMA", "food", "Derek Jeter" ], "mayor" : { "name" : "Bill de Blasio", "pa
rty" : "D" } }
> db.towns.find().count()
1
> db.help()
DB methods:
        db.adminCommand(nameOrDocument) - switches to 'admin' db, and runs comm
and [just calls db.runCommand(...)]
        db.aggregate([pipeline], {options}) - performs a collectionless aggrega
tion on this database; returns a cursor
        db.auth(username, password)
```



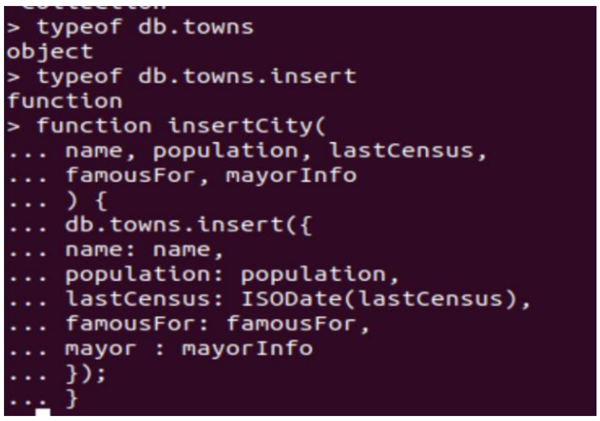
```
> db
test
> show collections
pes1ug19cs380_shop_inventory
phones
> db.towns.insert({ name: "New York", population: 22200000, lastCensus: ISODate("2016-0
7-01"), famousFor: [ "the MOMA", "food", "Derek Jeter" ], mayor : { name : "Bill de Bla
sio", party : "D" } })
WriteResult({ "nInserted" : 1 })
> show collections
pes1ug19cs380_shop_inventory
phones
towns
> db.towns.find()
{ "_id" : ObjectId("6141fa454373bfb889e301c9"), "name" : "New York", "population" : 222
00000, "lastCensus" : ISODate("2016-07-01T00:00:00Z"), "famousFor" : [ "the MOMA", "foo
d", "Derek Jeter" ], "mayor" : { "name" : "Bill de Blasio", "party" : "D" } }
> db.towns.find().count()
1
> db.help()
DB methods:
        db.adminCommand(nameOrDocument) - switches to 'admin' db, and runs command [jus
t calls db.runCommand(...)]
        db.aggregate([pipeline], {options}) - performs a collectionless aggregation on
this database; returns a cursor
        db.auth(username, password)
        db.cloneDatabase(fromhost)
        db.commandHelp(name) returns the help for the command
        db.copyDatabase(fromdb, todb, fromhost)
```

/*help*/



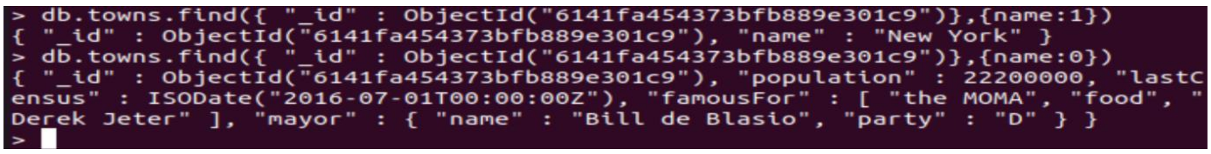
```
> db.towns.help()
DBCollection help
        db.towns.find().help() - show DBCursor help
        db.towns.bulkWrite( operations, <optional params> ) - bulk execute writ
e operations, optional parameters are: w, wtimeout, j
        db.towns.count( query = {}, <optional params> ) - count the number of d
ocuments that matches the query, optional parameters are: limit, skip, hint, ma
xTimeMS
```

/*insert*/



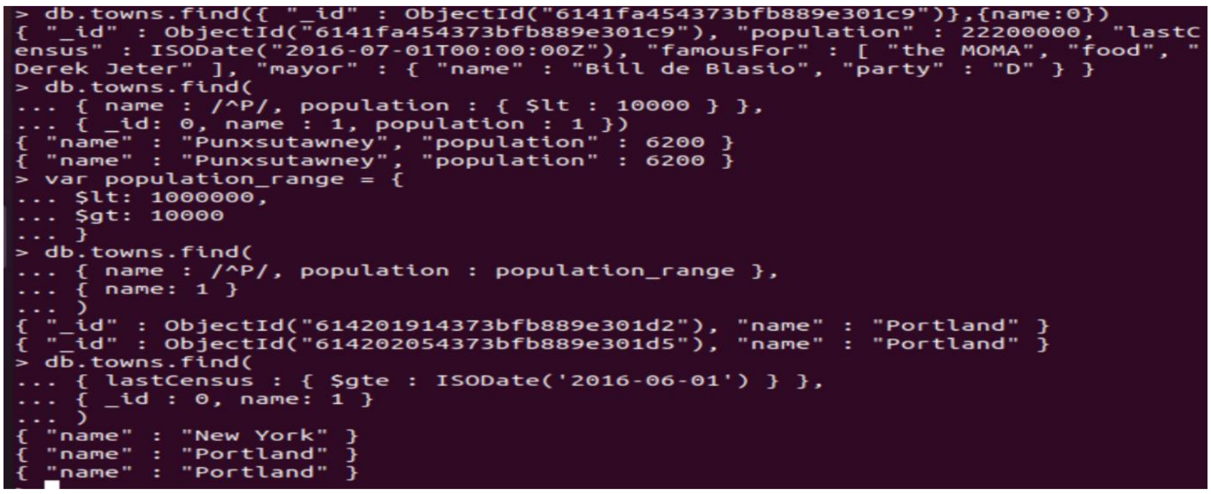
```
> typeof db.towns
object
> typeof db.towns.insert
function
> function insertCity(
... name, population, lastCensus,
... famousFor, mayorInfo
... ) {
... db.towns.insert({
... name: name,
... population: population,
... lastCensus: ISODate(lastCensus),
... famousFor: famousFor,
... mayor : mayorInfo
... });
... }
```

/*Read*/



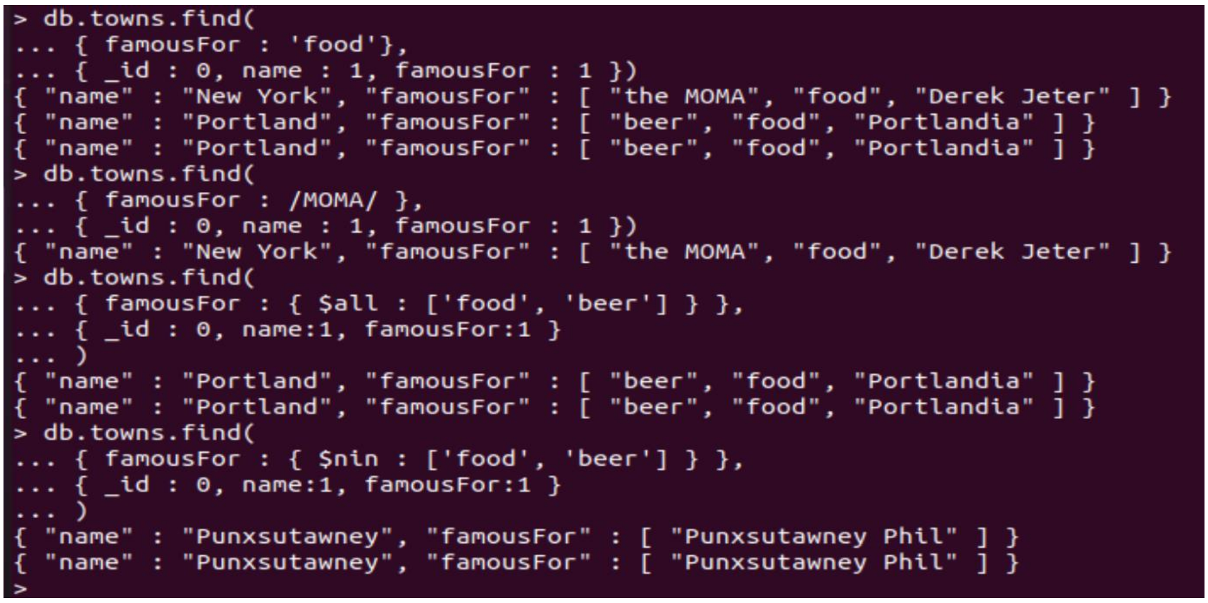
```
> db.towns.find({ "_id" : ObjectId("6141fa454373bfb889e301c9")},{name:1})
{ "_id" : ObjectId("6141fa454373bfb889e301c9"), "name" : "New York" }
> db.towns.find({ "_id" : ObjectId("6141fa454373bfb889e301c9")},{name:0})
{ "_id" : ObjectId("6141fa454373bfb889e301c9"), "population" : 22200000, "lastC
ensus" : ISODate("2016-07-01T00:00:00Z"), "famousFor" : [ "the MOMA", "food", "
Derek Jeter" ], "mayor" : { "name" : "Bill de Blasio", "party" : "D" } }
>
```

/* find all towns that begin with the letter P and have a population less than 10,000 */



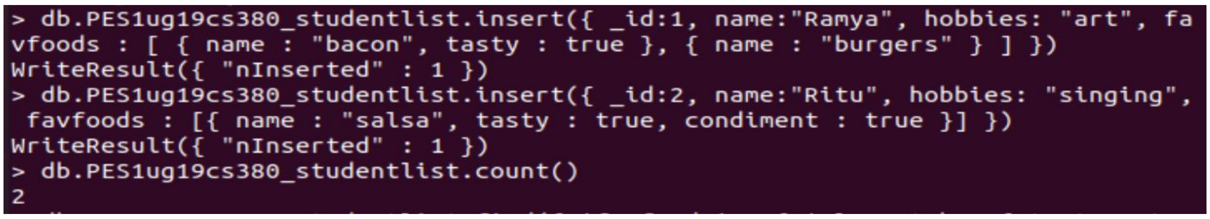
```
> db.towns.find({ "_id" : ObjectId("6141fa454373bfb889e301c9")},{name:0})
{ "_id" : ObjectId("6141fa454373bfb889e301c9"), "population" : 22200000, "lastC
ensus" : ISODate("2016-07-01T00:00:00Z"), "famousFor" : [ "the MOMA", "food", "
Derek Jeter" ], "mayor" : { "name" : "Bill de Blasio", "party" : "D" } }
> db.towns.find(
... { name : /^P/, population : { $lt : 10000 } },
... { _id: 0, name : 1, population : 1 })
{ "name" : "Punxsutawney", "population" : 6200 }
{ "name" : "Punxsutawney", "population" : 6200 }
> var population_range = {
... $lt: 1000000,
... $gt: 10000
... }
> db.towns.find(
... { name : /^P/, population : population_range },
... { name: 1 }
... )
{ "_id" : ObjectId("614201914373bfb889e301d2"), "name" : "Portland" }
{ "_id" : ObjectId("614202054373bfb889e301d5"), "name" : "Portland" }
> db.towns.find(
... { lastCensus : { $gte : ISODate('2016-06-01') } },
... { _id : 0, name: 1 }
... )
{ "name" : "New York" }
{ "name" : "Portland" }
{ "name" : "Portland" }
```

/* Querying nested data */



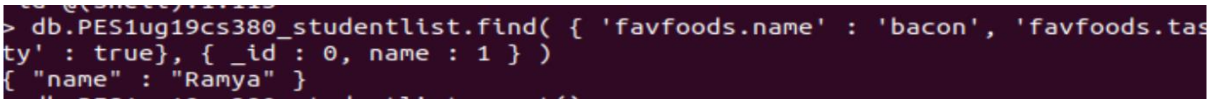
```
> db.towns.find(
... { famousFor : 'food'},
... { _id : 0, name : 1, famousFor : 1 })
{ "name" : "New York", "famousFor" : [ "the MOMA", "food", "Derek Jeter" ] }
{ "name" : "Portland", "famousFor" : [ "beer", "food", "Portlandia" ] }
{ "name" : "Portland", "famousFor" : [ "beer", "food", "Portlandia" ] }
> db.towns.find(
... { famousFor : /MOMA/ },
... { _id : 0, name : 1, famousFor : 1 })
{ "name" : "New York", "famousFor" : [ "the MOMA", "food", "Derek Jeter" ] }
> db.towns.find(
... { famousFor : { $all : ['food', 'beer'] } },
... { _id : 0, name:1, famousFor:1 }
... )
{ "name" : "Portland", "famousFor" : [ "beer", "food", "Portlandia" ] }
{ "name" : "Portland", "famousFor" : [ "beer", "food", "Portlandia" ] }
> db.towns.find(
... { famousFor : { $nin : ['food', 'beer'] } },
... { _id : 0, name:1, famousFor:1 }
... )
{ "name" : "Punxsutawney", "famousFor" : [ "Punxsutawney Phil" ] }
{ "name" : "Punxsutawney", "famousFor" : [ "Punxsutawney Phil" ] }
>
```

/* New collection – PES1ug19cs380_studentlist*/

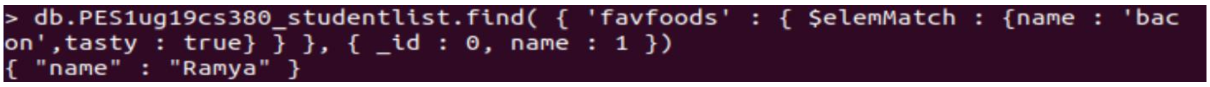


```
> db.PES1ug19cs380_studentlist.insert({ _id:1, name:"Ramya", hobbies: "art", fa
vfoods : [ { name : "bacon", tasty : true }, { name : "burgers" } ] })
WriteResult({ "nInserted" : 1 })
> db.PES1ug19cs380_studentlist.insert({ _id:2, name:"Ritu", hobbies: "singing",
 favfoods : [{ name : "salsa", tasty : true, condiment : true }] })
WriteResult({ "nInserted" : 1 })
> db.PES1ug19cs380_studentlist.count()
2
```

/*Read*/

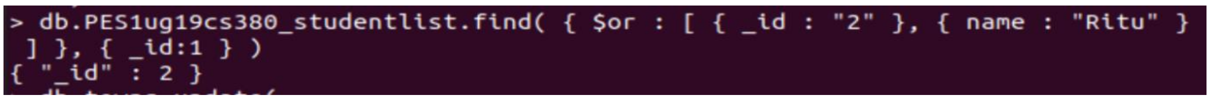


```
> db.PES1ug19cs380_studentlist.find( { 'favfoods.name' : 'bacon', 'favfoods.tas
ty' : true}, { _id : 0, name : 1 } )
{ "name" : "Ramya" }
```



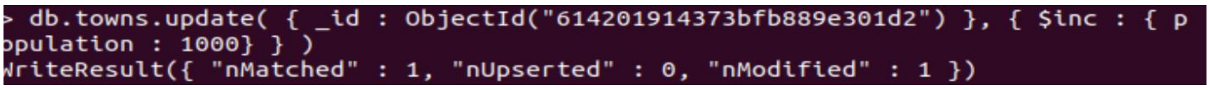
```
> db.PES1ug19cs380_studentlist.find( { 'favfoods' : { $elemMatch : {name : 'bac
on',tasty : true} } }, { _id : 0, name : 1 })
{ "name" : "Ramya" }
```

/*OR*/



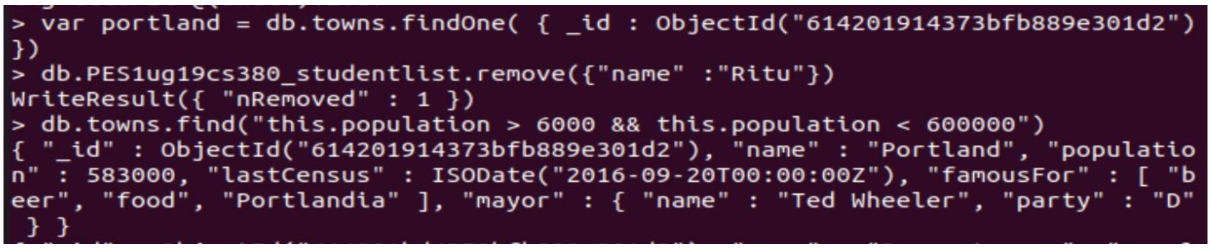
```
> db.PES1ug19cs380_studentlist.find( { $or : [ { _id : "2" }, { name : "Ritu" }
 ] }, { _id:1 } )
{ "_id" : 2 }
```

/*Update*/



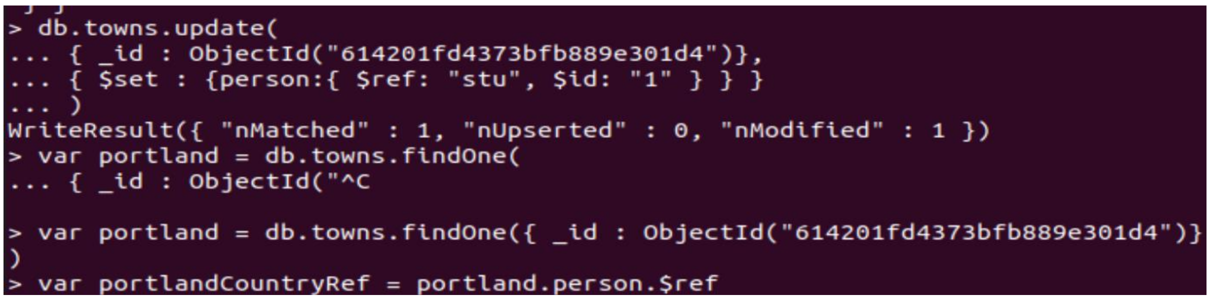
```
> db.towns.update( { _id : ObjectId("614201914373bfb889e301d2") }, { $inc : { p
opulation : 1000} } )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

/* Increment */



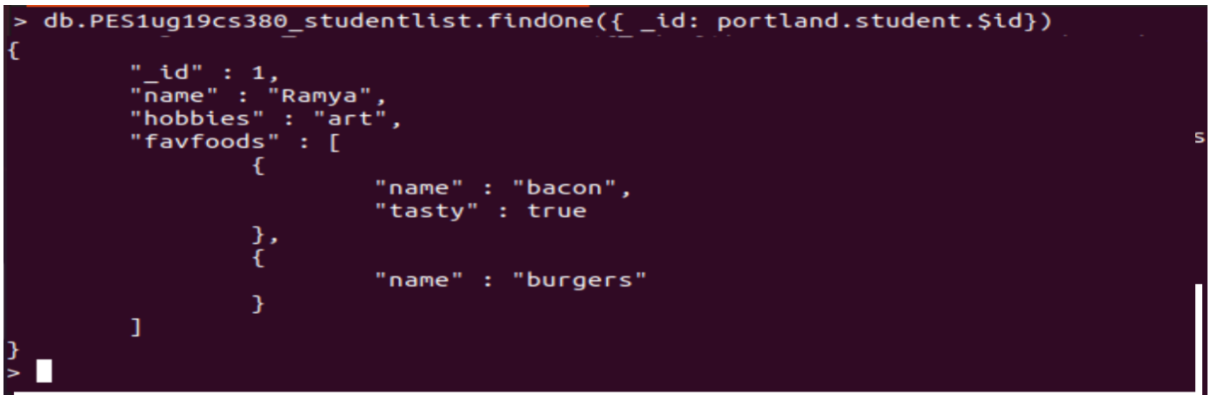
```
> var portland = db.towns.findOne( { _id : ObjectId("614201914373bfb889e301d2")
})
> db.PES1ug19cs380_studentlist.remove({"name" :"Ritu"})
WriteResult({ "nRemoved" : 1 })
> db.towns.find("this.population > 6000 && this.population < 600000")
{ "_id" : ObjectId("614201914373bfb889e301d2"), "name" : "Portland", "populatio
n" : 583000, "lastCensus" : ISODate("2016-09-20T00:00:00Z"), "famousFor" : [ "b
eer", "food", "Portlandia" ], "mayor" : { "name" : "Ted Wheeler", "party" : "D"
 } }
```

/* References */

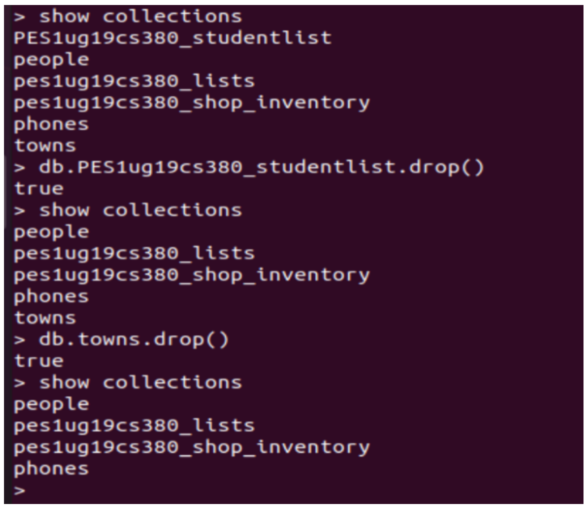


```
> db.towns.update(
... { _id : ObjectId("614201fd4373bfb889e301d4")},
... { $set : {person:{ $ref: "stu", $id: "1" } } }
... )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> var portland = db.towns.findOne(
... { _id : ObjectId("^C

> var portland = db.towns.findOne({ _id : ObjectId("614201fd4373bfb889e301d4")}
)
> var portlandCountryRef = portland.person.$ref
```

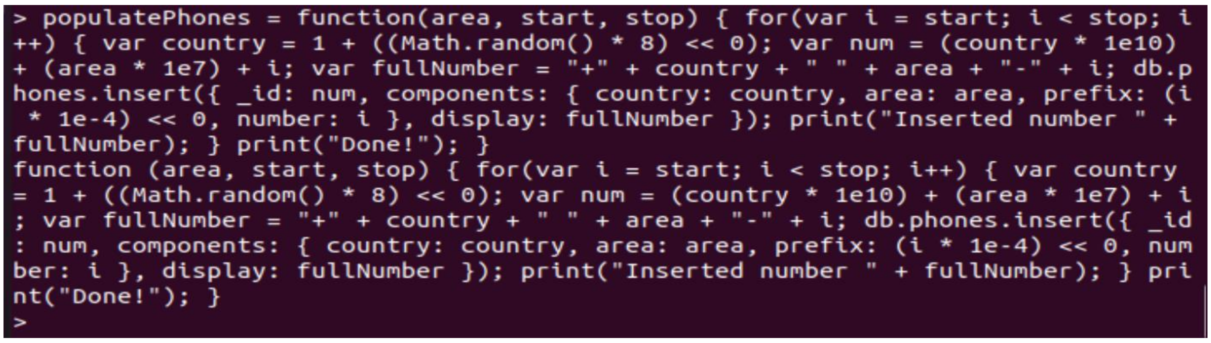


```
> db.PES1ug19cs380_studentlist.findOne({ _id: portland.student.$id})
{
        "_id" : 1,
        "name" : "Ramya",
        "hobbies" : "art",
        "favfoods" : [
                {
                        "name" : "bacon",
                        "tasty" : true
                },
                {
                        "name" : "burgers"
                }
        ]
}
>
```

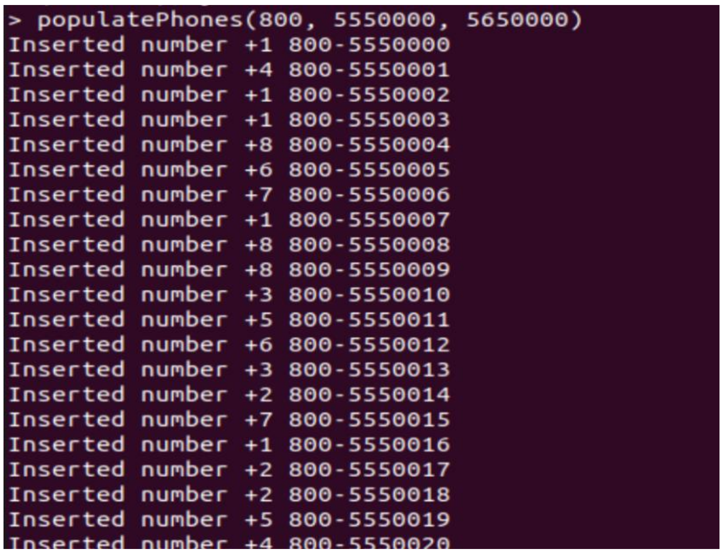


```
> show collections
PES1ug19cs380_studentlist
people
pes1ug19cs380_lists
pes1ug19cs380_shop_inventory
phones
towns
> db.PES1ug19cs380_studentlist.drop()
true
> show collections
people
pes1ug19cs380_lists
pes1ug19cs380_shop_inventory
phones
towns
> db.towns.drop()
true
> show collections
people
pes1ug19cs380_lists
pes1ug19cs380_shop_inventory
phones
>
```

/*New collections: Phones*/

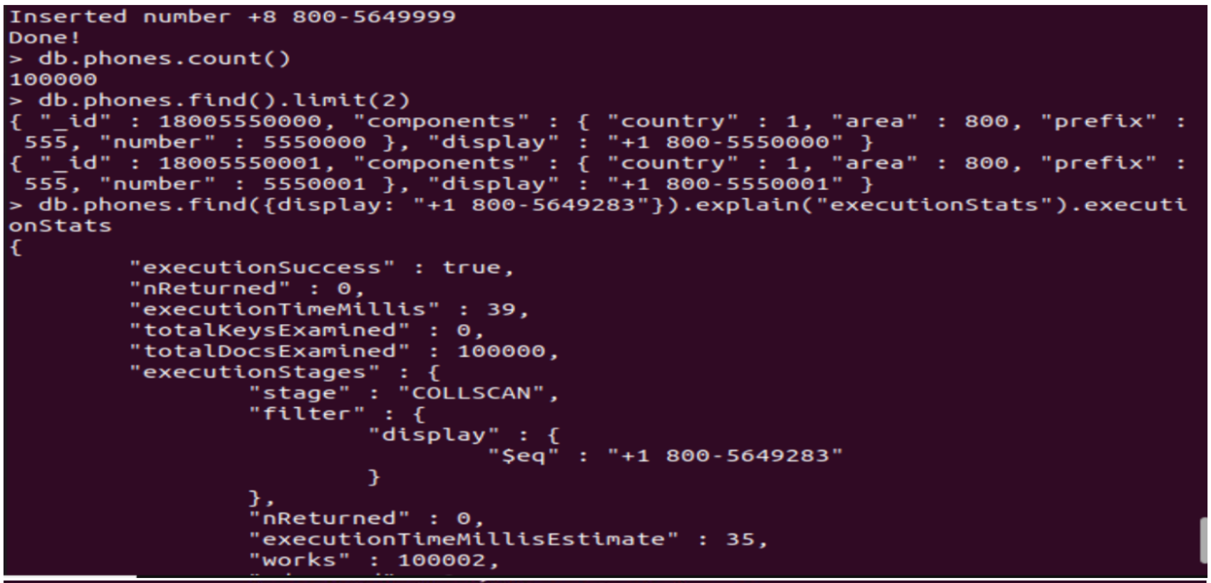


```
> populatePhones = function(area, start, stop) { for(var i = start; i < stop; i
++) { var country = 1 + ((Math.random() * 8) << 0); var num = (country * 1e10)
+ (area * 1e7) + i; var fullNumber = "+" + country + " " + area + "-" + i; db.p
hones.insert({ _id: num, components: { country: country, area: area, prefix: (i
 * 1e-4) << 0, number: i }, display: fullNumber }); print("Inserted number " +
fullNumber); } print("Done!"); }
function (area, start, stop) { for(var i = start; i < stop; i++) { var country
= 1 + ((Math.random() * 8) << 0); var num = (country * 1e10) + (area * 1e7) + i
; var fullNumber = "+" + country + " " + area + "-" + i; db.phones.insert({ _id
: num, components: { country: country, area: area, prefix: (i * 1e-4) << 0, num
ber: i }, display: fullNumber }); print("Inserted number " + fullNumber); } pri
nt("Done!"); }
>
```

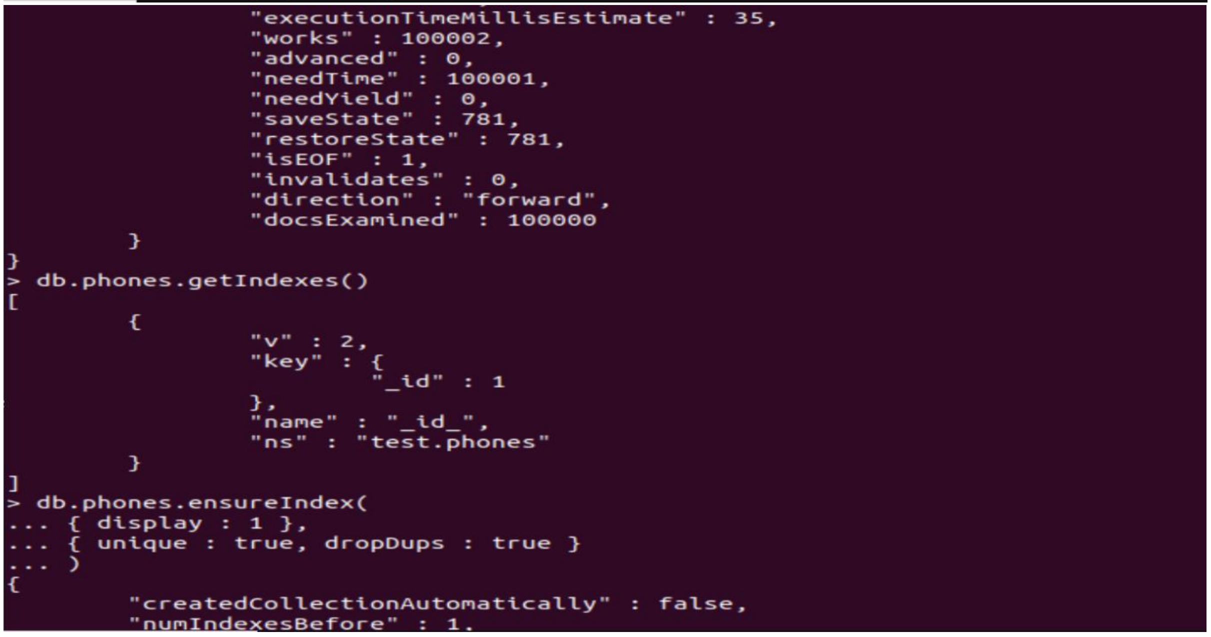


```
> populatePhones(800, 5550000, 5650000)
Inserted number +1 800-5550000
Inserted number +4 800-5550001
Inserted number +1 800-5550002
Inserted number +1 800-5550003
Inserted number +8 800-5550004
Inserted number +6 800-5550005
Inserted number +7 800-5550006
Inserted number +1 800-5550007
Inserted number +8 800-5550008
Inserted number +8 800-5550009
Inserted number +3 800-5550010
Inserted number +5 800-5550011
Inserted number +6 800-5550012
Inserted number +3 800-5550013
Inserted number +2 800-5550014
Inserted number +7 800-5550015
Inserted number +1 800-5550016
Inserted number +2 800-5550017
Inserted number +2 800-5550018
Inserted number +5 800-5550019
Inserted number +4 800-5550020
```

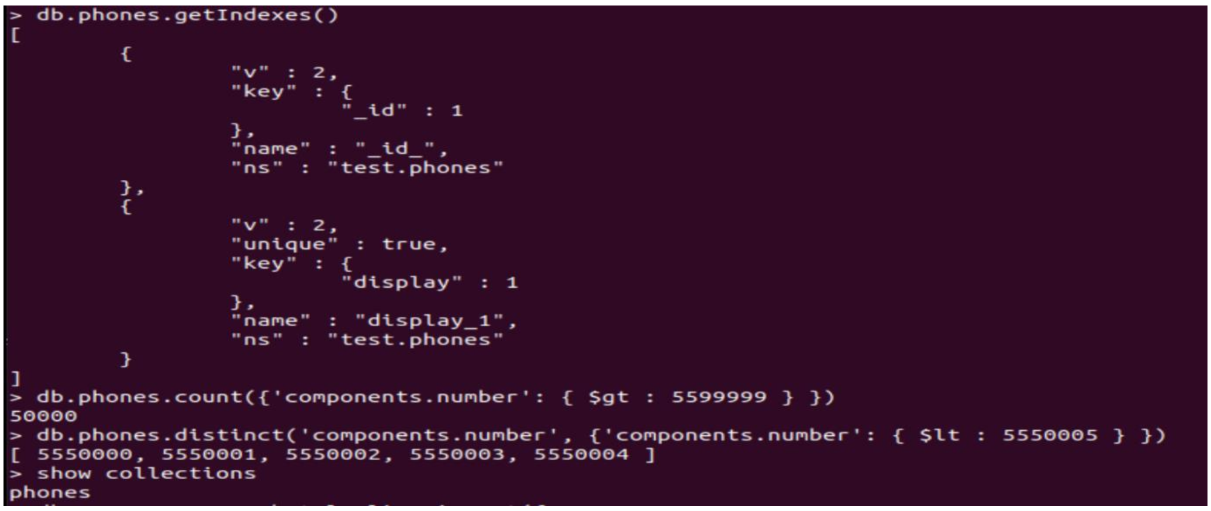
/* create index */



```
Inserted number +8 800-5649999
Done!
> db.phones.count()
100000
> db.phones.find().limit(2)
{ "_id" : 18005550000, "components" : { "country" : 1, "area" : 800, "prefix" :
 555, "number" : 5550000 }, "display" : "+1 800-5550000" }
{ "_id" : 18005550001, "components" : { "country" : 1, "area" : 800, "prefix" :
 555, "number" : 5550001 }, "display" : "+1 800-5550001" }
> db.phones.find({display: "+1 800-5649283"}).explain("executionStats").executi
onStats
{
        "executionSuccess" : true,
        "nReturned" : 0,
        "executionTimeMillis" : 39,
        "totalKeysExamined" : 0,
        "totalDocsExamined" : 100000,
        "executionStages" : {
                "stage" : "COLLSCAN",
                "filter" : {
                        "display" : {
                                "$eq" : "+1 800-5649283"
                        }
                },
                "nReturned" : 0,
                "executionTimeMillisEstimate" : 35,
                "works" : 100002,
```

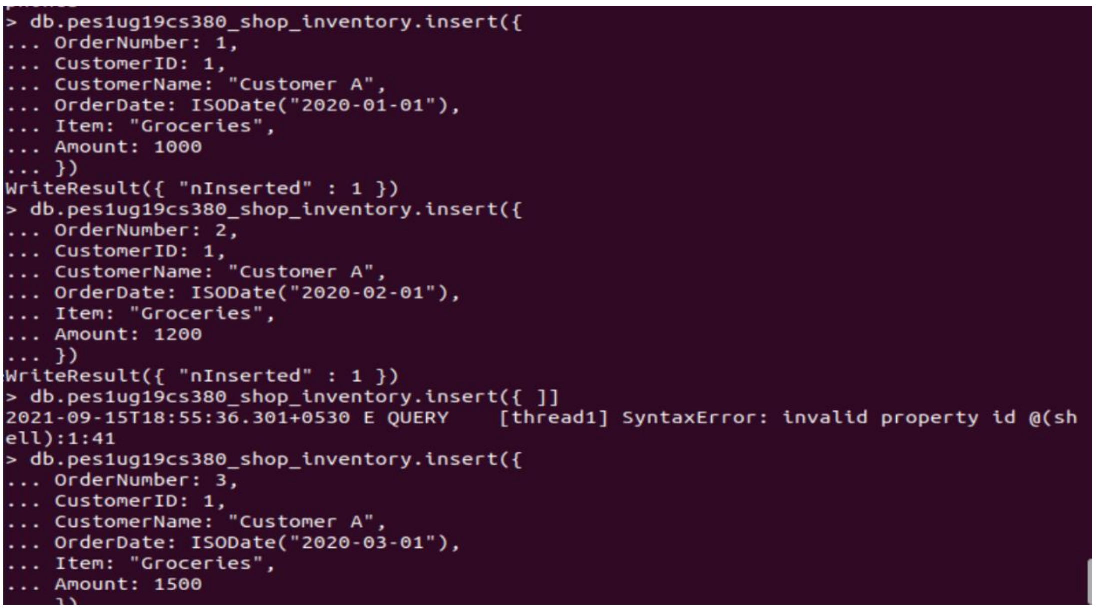


```
                "executionTimeMillisEstimate" : 35,
                "works" : 100002,
                "advanced" : 0,
                "needTime" : 100001,
                "needYield" : 0,
                "saveState" : 781,
                "restoreState" : 781,
                "isEOF" : 1,
                "invalidates" : 0,
                "direction" : "forward",
                "docsExamined" : 100000
        }
}
> db.phones.getIndexes()
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "test.phones"
        }
]
> db.phones.ensureIndex(
... { display : 1 },
... { unique : true, dropDups : true }
... )
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
```

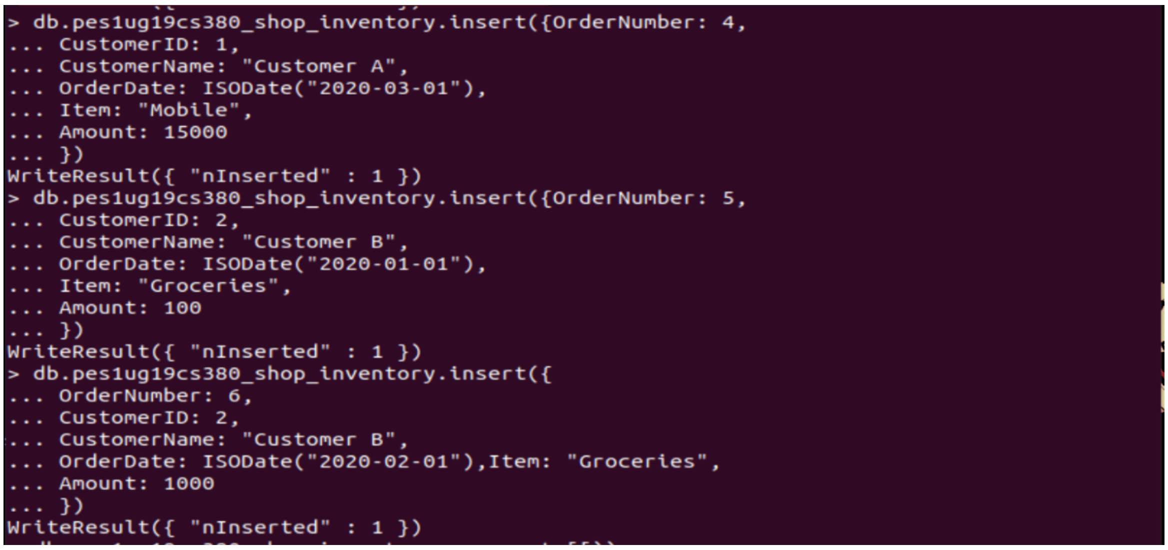


```
> db.phones.getIndexes()
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "test.phones"
        },
        {
                "v" : 2,
                "unique" : true,
                "key" : {
                        "display" : 1
                },
                "name" : "display_1",
                "ns" : "test.phones"
        }
]
> db.phones.count({'components.number': { $gt : 5599999 } })
50000
> db.phones.distinct('components.number', {'components.number': { $lt : 5550005 } })
[ 5550000, 5550001, 5550002, 5550003, 5550004 ]
> show collections
phones
```

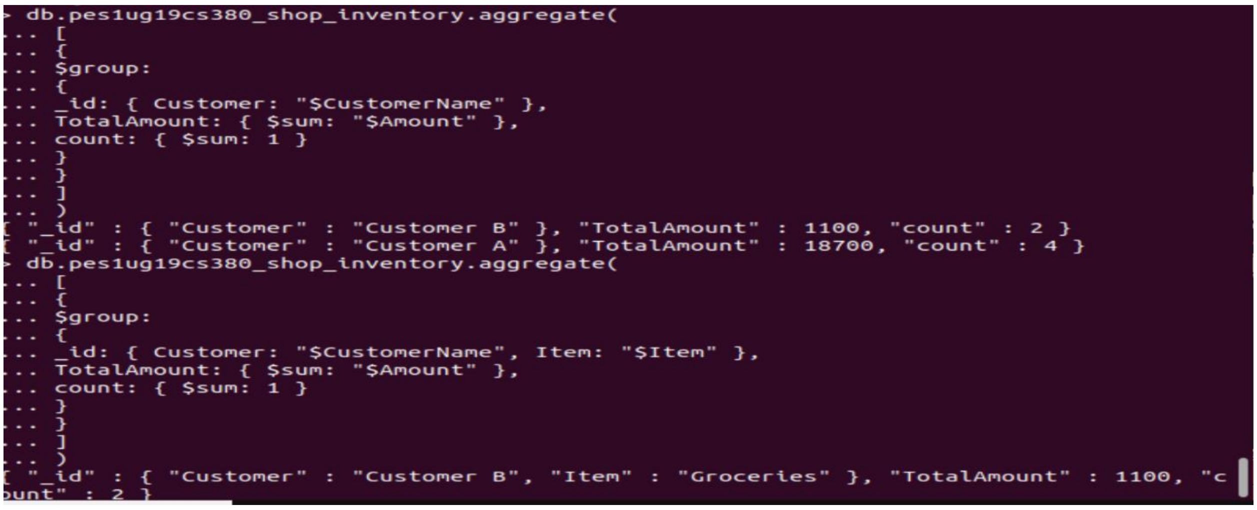
/* Aggregation */



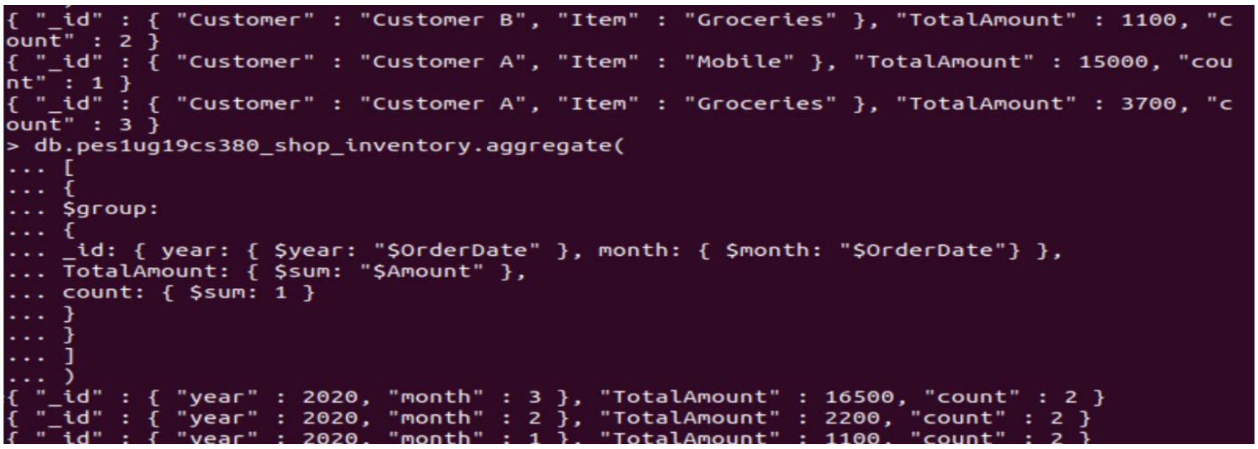
```
> db.pes1ug19cs380_shop_inventory.insert({
... OrderNumber: 1,
... CustomerID: 1,
... CustomerName: "Customer A",
... OrderDate: ISODate("2020-01-01"),
... Item: "Groceries",
... Amount: 1000
... })
WriteResult({ "nInserted" : 1 })
> db.pes1ug19cs380_shop_inventory.insert({
... OrderNumber: 2,
... CustomerID: 1,
... CustomerName: "Customer A",
... OrderDate: ISODate("2020-02-01"),
... Item: "Groceries",
... Amount: 1200
... })
WriteResult({ "nInserted" : 1 })
> db.pes1ug19cs380_shop_inventory.insert({ ]]
2021-09-15T18:55:36.301+0530 E QUERY    [thread1] SyntaxError: invalid property id @(sh
ell):1:41
> db.pes1ug19cs380_shop_inventory.insert({
... OrderNumber: 3,
... CustomerID: 1,
... CustomerName: "Customer A",
... OrderDate: ISODate("2020-03-01"),
... Item: "Groceries",
... Amount: 1500
```



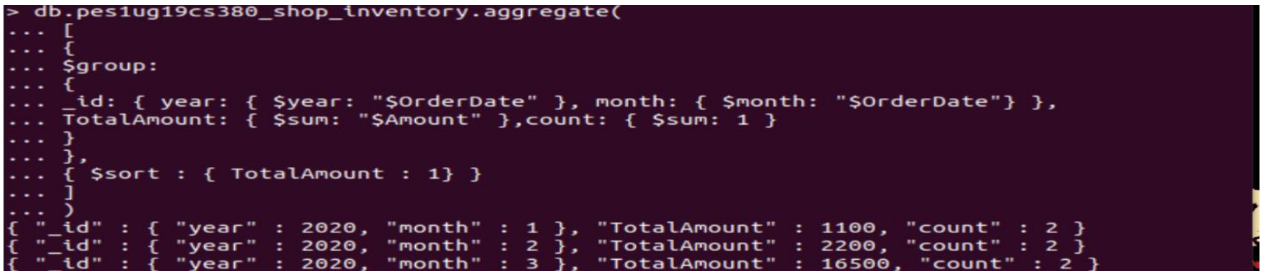
```
> db.pes1ug19cs380_shop_inventory.insert({OrderNumber: 4,
... CustomerID: 1,
... CustomerName: "Customer A",
... OrderDate: ISODate("2020-03-01"),
... Item: "Mobile",
... Amount: 15000
... })
WriteResult({ "nInserted" : 1 })
> db.pes1ug19cs380_shop_inventory.insert({OrderNumber: 5,
... CustomerID: 2,
... CustomerName: "Customer B",
... OrderDate: ISODate("2020-01-01"),
... Item: "Groceries",
... Amount: 100
... })
WriteResult({ "nInserted" : 1 })
> db.pes1ug19cs380_shop_inventory.insert({
... OrderNumber: 6,
... CustomerID: 2,
... CustomerName: "Customer B",
... OrderDate: ISODate("2020-02-01"),Item: "Groceries",
... Amount: 1000
... })
WriteResult({ "nInserted" : 1 })
```



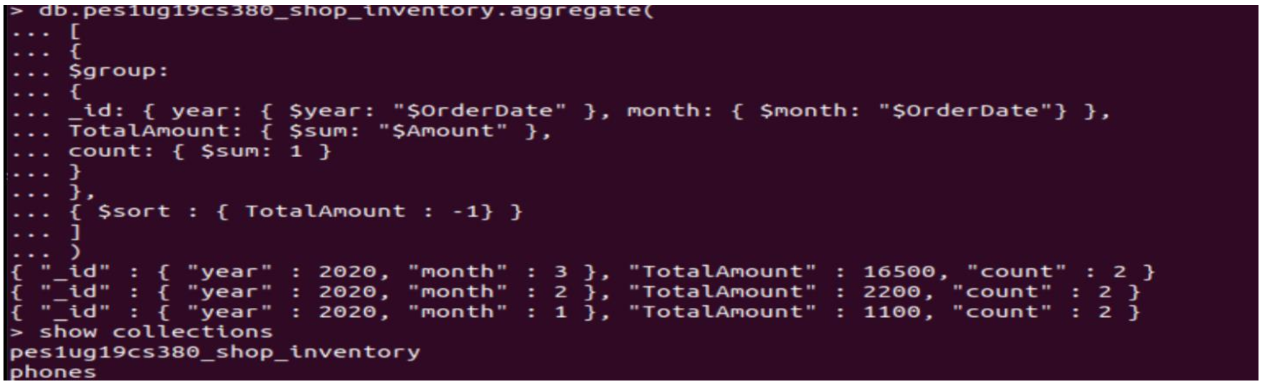
```
> db.pes1ug19cs380_shop_inventory.aggregate(
... [
... {
... $group:
... {
... _id: { Customer: "$CustomerName" },
... TotalAmount: { $sum: "$Amount" },
... count: { $sum: 1 }
... }
... }
... ]
... )
{ "_id" : { "Customer" : "Customer B" }, "TotalAmount" : 1100, "count" : 2 }
{ "_id" : { "Customer" : "Customer A" }, "TotalAmount" : 18700, "count" : 4 }
> db.pes1ug19cs380_shop_inventory.aggregate(
... [
... {
... $group:
... {
... _id: { Customer: "$CustomerName", Item: "$Item" },
... TotalAmount: { $sum: "$Amount" },
... count: { $sum: 1 }
... }
... }
... ]
... )
{ "_id" : { "Customer" : "Customer B", "Item" : "Groceries" }, "TotalAmount" : 1100, "c
ount" : 2 }
```



```
{ "_id" : { "Customer" : "Customer B", "Item" : "Groceries" }, "TotalAmount" : 1100, "c
ount" : 2 }
{ "_id" : { "Customer" : "Customer A", "Item" : "Mobile" }, "TotalAmount" : 15000, "cou
nt" : 1 }
{ "_id" : { "Customer" : "Customer A", "Item" : "Groceries" }, "TotalAmount" : 3700, "c
ount" : 3 }
> db.pes1ug19cs380_shop_inventory.aggregate(
... [
... {
... $group:
... {
... _id: { year: { $year: "$OrderDate" }, month: { $month: "$OrderDate"} },
... TotalAmount: { $sum: "$Amount" },
... count: { $sum: 1 }
... }
... }
... ]
... )
{ "_id" : { "year" : 2020, "month" : 3 }, "TotalAmount" : 16500, "count" : 2 }
{ "_id" : { "year" : 2020, "month" : 2 }, "TotalAmount" : 2200, "count" : 2 }
{ "_id" : { "year" : 2020, "month" : 1 }, "TotalAmount" : 1100, "count" : 2 }
```



```
> db.pes1ug19cs380_shop_inventory.aggregate(
... [
... {
... $group:
... {
... _id: { year: { $year: "$OrderDate" }, month: { $month: "$OrderDate"} },
... TotalAmount: { $sum: "$Amount" },count: { $sum: 1 }
... }
... },
... { $sort : { TotalAmount : 1} }
... ]
... )
{ "_id" : { "year" : 2020, "month" : 1 }, "TotalAmount" : 1100, "count" : 2 }
{ "_id" : { "year" : 2020, "month" : 2 }, "TotalAmount" : 2200, "count" : 2 }
{ "_id" : { "year" : 2020, "month" : 3 }, "TotalAmount" : 16500, "count" : 2 }
```



```
> db.pes1ug19cs380_shop_inventory.aggregate(
... [
... {
... $group:
... {
... _id: { year: { $year: "$OrderDate" }, month: { $month: "$OrderDate"} },
... TotalAmount: { $sum: "$Amount" },
... count: { $sum: 1 }
... }
... },
... { $sort : { TotalAmount : -1} }
... ]
... )
{ "_id" : { "year" : 2020, "month" : 3 }, "TotalAmount" : 16500, "count" : 2 }
{ "_id" : { "year" : 2020, "month" : 2 }, "TotalAmount" : 2200, "count" : 2 }
{ "_id" : { "year" : 2020, "month" : 1 }, "TotalAmount" : 1100, "count" : 2 }
> show collections
pes1ug19cs380_shop_inventory
phones
```