**MongoDB LAB Assignments –Day 1**

**INSERT DOCUMENTS-**

1.

db.movies.insert({title:"Fight Club", writer: "Chuck Palahniuk", year: "1999", actors:["Brad Pitt", "Edward Norton"]})

2.

db.movies.insert({title:"Pulp Fiction", writer:"Quentin Tarantino", year:"2009", actors:["John Travolta", "Uma Thurman"]})

3.

db.movies.insert({title:"Inglorious Basterds", writer:"Quentin Tarantino", year:"2009", actors:["Brad Pitt", "Diane Kruger", "Eli Roth"]})

4.

db.movies.insert({title:"The Hobbit: An unexpected Journey", writer:"J.R.R. Tolkein", year:"2012",franchise:"The Hobbit"})

5.

db.movies.insert({title:"The Hobbit: The Desolation of Smaug", writer:"J.R.R Tolkien", year:"2013", franchise:"The Hobbit"})

6.

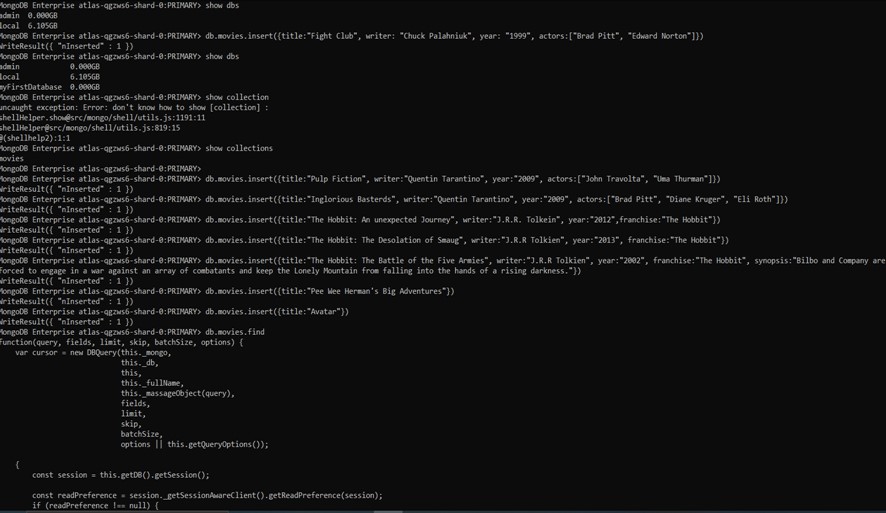
db.movies.insert({title:"The Hobbit: The Battle of the Five Armies", writer:"J.R.R Tolkien", year:"2002", franchise:"The Hobbit", synopsis:"Bilbo and Company are forced to engage in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of a rising darkness."})

7.

db.movies.insert({title:"Pee Wee Herman's Big Adventures"})

8.

db.movies.insert({title:"Avatar"})



**Query / Find Documents-**

1.

db.movies.find()

2.

db.movies.find({writer:"Quentin Tarantino"})

3.

db.movies.find({actors:"Brad Pitt"})

4.

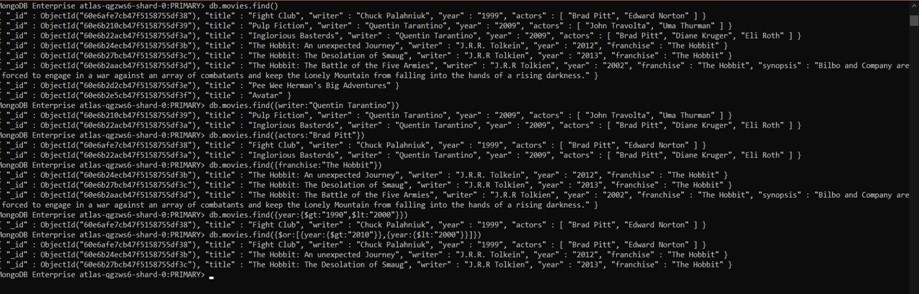
db.movies.find({franchise:"The Hobbit"})

5.

db.movies.find({year:{$gt:"1990", $lt:"2000"}})

6.

db.movies.find({$or:[{year:{$gt:"2010"}},{year: {$lt:"2000"}}]})



**Update Documents -**

1.

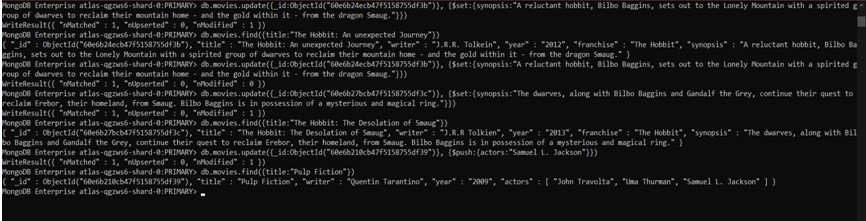
db.movies.update({\_id:ObjectId("5c9f98e5e5c2dfe9b3729bfe")}, {$set:{synopsis:"A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug."}})

2.

db.movies.update({\_id:ObjectId("5c9fa42ae5c2dfe9b3729c03")}, {$set:{synopsis:"The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."}})

3.

db.movies.update({\_id:ObjectId("5c9f983ce5c2dfe9b3729bfc")}, {$push:{actors:"Samuel L. Jackson"}})

****

**TEXT SEARCH-**

1.

db.movies.find({synopsis:{$regex:"Bilbo"}})

2.

db.movies.find({synopsis:{$regex:"Gandalf"}})

3.

db.movies.find({$and:[{synopsis:{$regex:"Bilbo"}}, {synopsis:{$not:/Gandalf/}}]})

4.

db.movies.find({$or:[{synopsis:{$regex:"dwarves"}}, {synopsis:{$regex:"hobbit"}}]})

5.

db.movies.find({$and:[{synopsis:{$regex:"gold"}}, {synopsis:{$regex:"dragon"}}]})

****

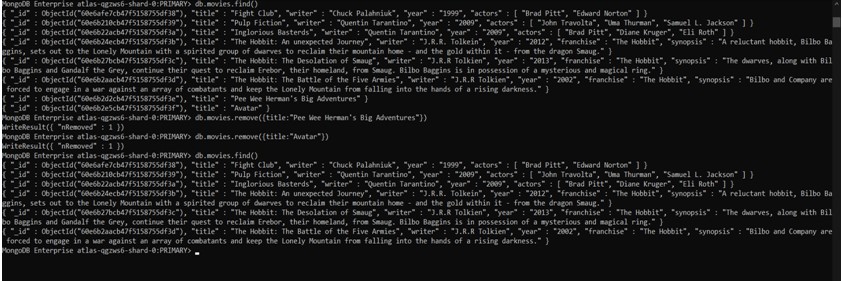
**DELETE DOCUMENTS -**

1.

db.movies.remove({title:"Pee Wee Herman's Big Adventures"})

2.

db.movies.remove({title:"Avatar"})

****

**RELATIONSHIPS -**

1.

db.users.insert({\_id:1,username:"GoodGuyGreg", first\_name:"Good Guy", last\_name:"Greg"})

2.

db.users.insert({\_id:2, username:"ScumbagSteve", fullname:{first: "Scumbag", last:"Steve"}})

3.

db.posts.insert({username:"GoodGuyGreg", title:"Passes out at Party", body:"Raises your credit score"})

4.

db.posts.insert({ username:"GoodGuyGreg", title:"Steals your identity", body:"Raises your credit score"})

5.

db.posts.insert({username:"GoodGuyGreg", title:"Reports a bug in your code", body:"Sends you a pull request"})

6.

db.posts.insert({ username:"ScumbagSteve", title:"Borrows something", body:"Sells it"})

7.

db.posts.insert({ username:"ScumbagSteve", title:"Borrows everything", body:"The end"})

8.

db.posts.insert({username:"ScumbagSteve", title:"Forks your repo on github", body:"Sets to private"})

Insert the following documents into a **comments** collection-

1.

db.comments.insert({ username:"GoodGuyGreg", comment:"Hope you got a good deal!", post:ObjectId("5ca0b7e96435f98b5901f463")})

2.

db.comments.insert({username:"GoodGuyGreg", comment:"What's mine is yours!", post:ObjectId("5ca0b9706435f98b5901f46a")})

3.

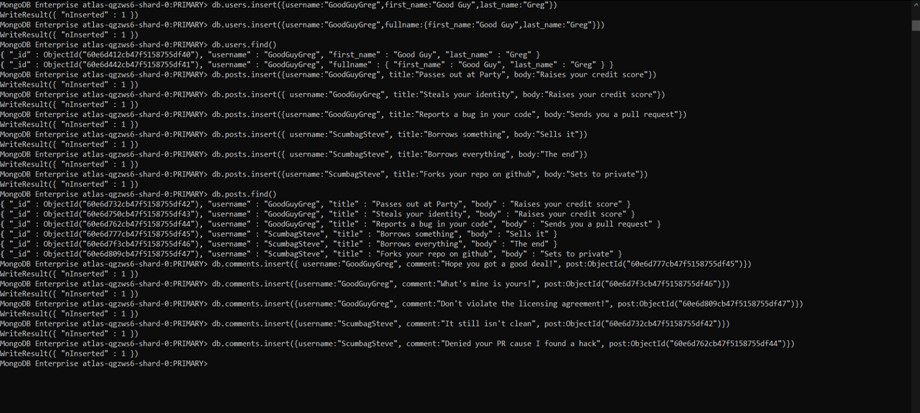
db.comments.insert({username:"GoodGuyGreg", comment:"Don't violate the licensing agreement!", post:ObjectId("5ca0b8766435f98b5901f467")})

4.

db.comments.insert({username:"ScumbagSteve", comment:"It still isn't clean", post:ObjectId("5ca0b8546435f98b5901f466")})

5.

db.comments.insert({username:"ScumbagSteve", comment:"Denied your PR cause I found a hack", post:ObjectId("5ca0b9256435f98b5901f469")})



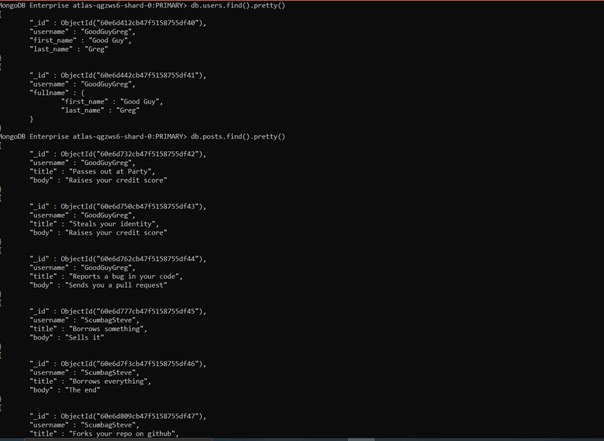
**Querying related collections-**

1.

db.users.find().pretty()

2.

db.posts.find().pretty()



3.

db.posts.find({username:"GoodGuyGreg"})

4.

db.posts.find({username:"ScumbagSteve"})

5.

db.comments.find().pretty()

6.

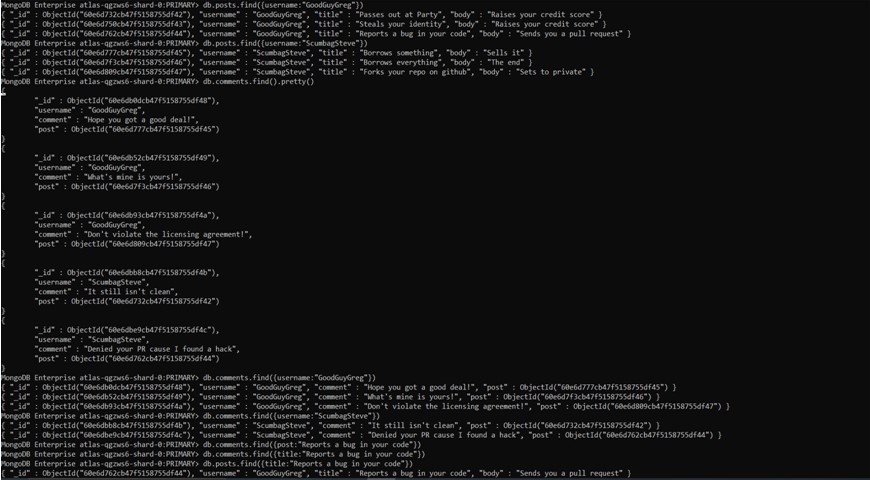
db.comments.find({username:"GoodGuyGreg"})

7.

db.comments.find({username:"ScumbagSteve"})

8.

db.posts.find({title:"Reports a bug in your code"})



**MongoDB Aggregate Assignments- DAY-2**

**ATLANTA POPULATION -**

1.

db.zipcodes.find({city:"ATLANTA"})

2.

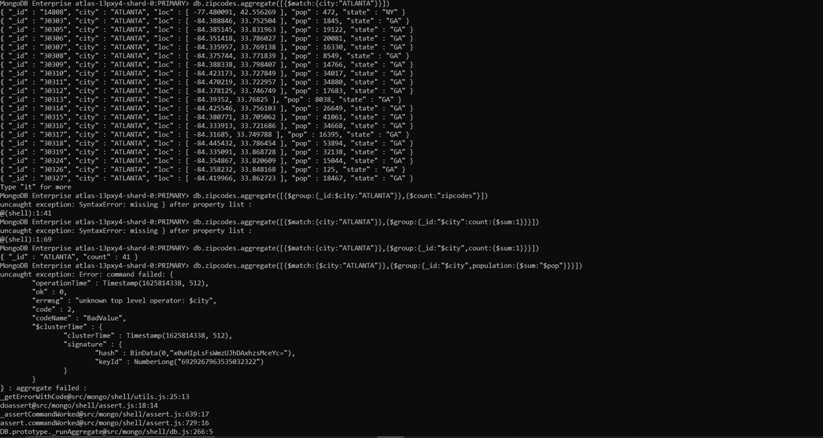
db.zipcodes.aggregate([{$match:{city:"ATLANTA"}}])

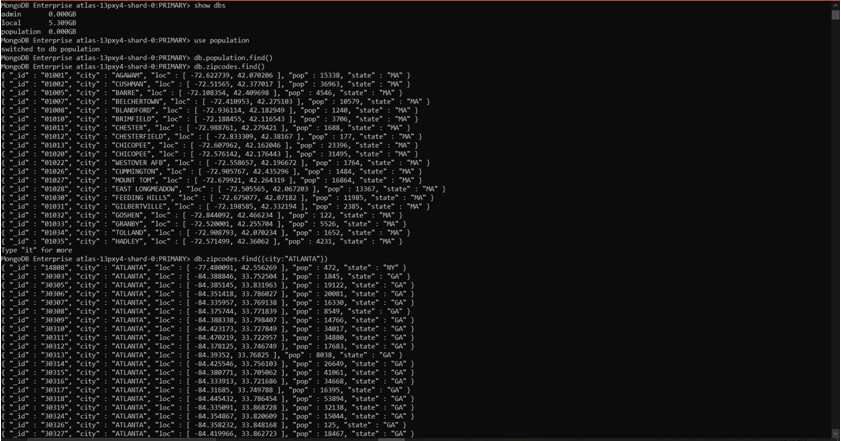
3.

db.zipcodes.aggregate([{$match:{city:"ATLANTA"}},{$group:{\_id:"$city",count:{$sum:1}}}])

4.

db.zipcodes.aggregate([{$match:{city:"ATLANTA"}},{$group:{\_id:"$city",population:{$sum:"$pop"}}}])





aggregate 4.2.jpg

**POPULATIONS BY STATE-**

1.

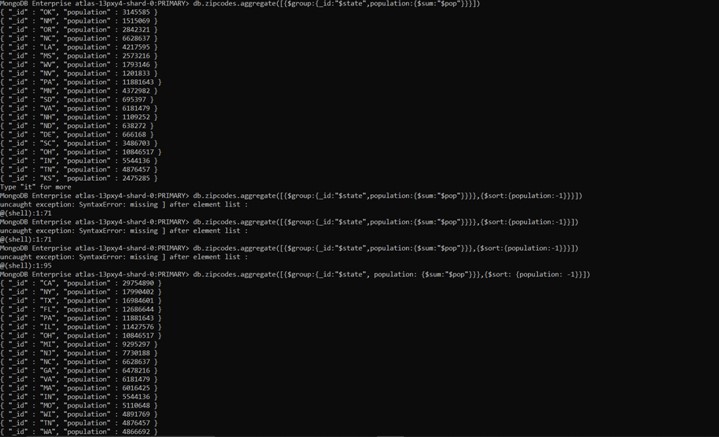
db.zipcodes.aggregate([{$group:{\_id:”$state”,population:{$sum:”$pop”}}}])

2.

db.zipcodes.aggregate([{$group:{\_id:"$state", population: {$sum:"$pop"}}},{$sort: {population: -1}}])

3.

db.zipcodes.aggregate([{$group:{\_id:"$state", population: {$sum:"$pop"}}},{$sort: {population: -1}}, {$limit:3}])





**POPULATIONS BY CITY-**

1.

db.zipcodes.aggregate([{$group:{\_id:{city:"$city",state:"$state"},populationIs:{$sum:"$pop"}}}])

2.

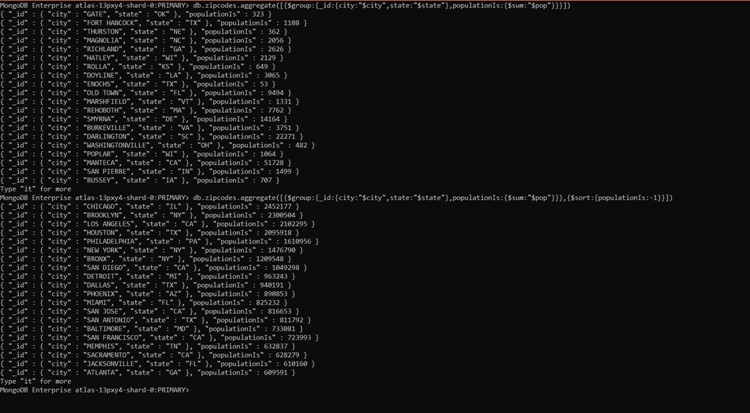
db.zipcodes.aggregate([{$group:{\_id:{city:"$city",state:"$state"},populationIs:{$sum:"$pop"}}},{$sort:{populationIs:-1}}])

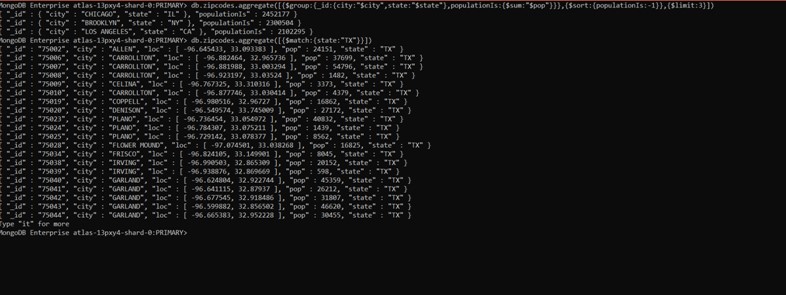
3.

db.zipcodes.aggregate([{$group:{\_id:{city:"$city",state:"$state"},populationIs:{$sum:"$pop"}}},{$sort:{populationIs:-1}},{$limit:3}])

4.

db.zipcodes.aggregate([{$match:{state:"TX"}}])





**BONUS-**

1.

db.zipcodes.aggregate([{$group: {\_id: {state: "$state"}, avgpopulationIs: {$avg: "$pop"}}}])

2.

db.zipcodes.aggregate([{$group: {\_id: {state: "$state"}, avgpopulationIs: {$avg: "$pop"}}},{$sort:{avgpopulationIs:-1}},{$limit:3}])

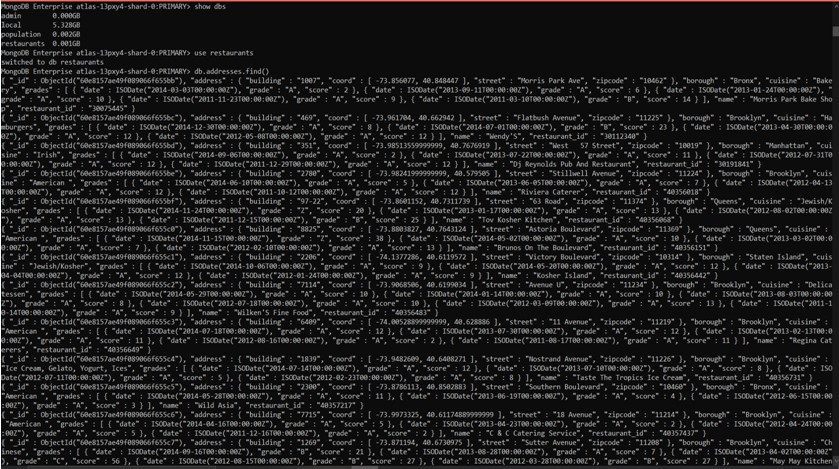


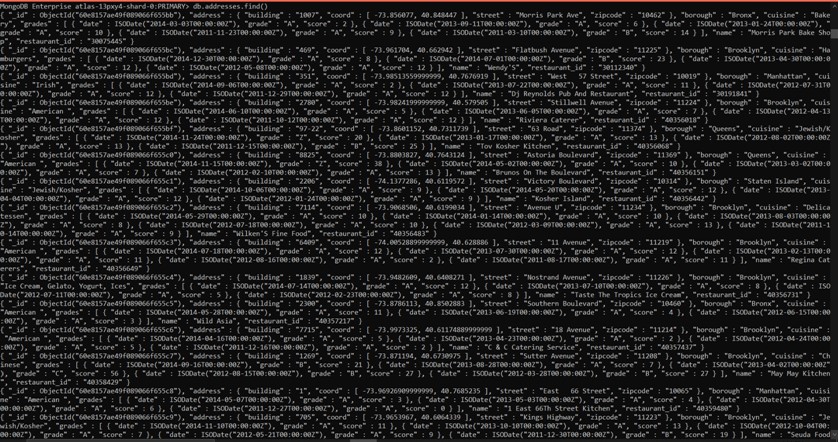
**MongoDB Complex Queries – DAY -3**

**Exercise Questions -**

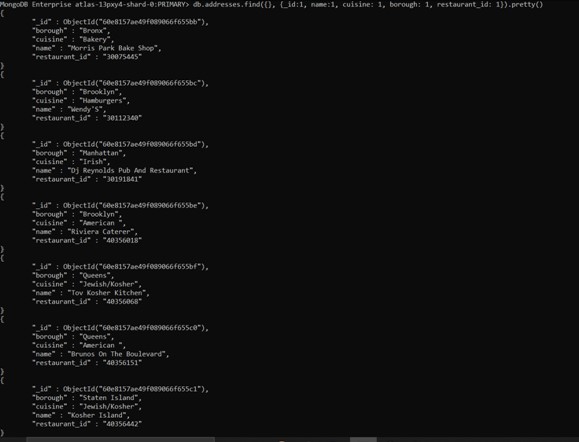
1.

Show collections



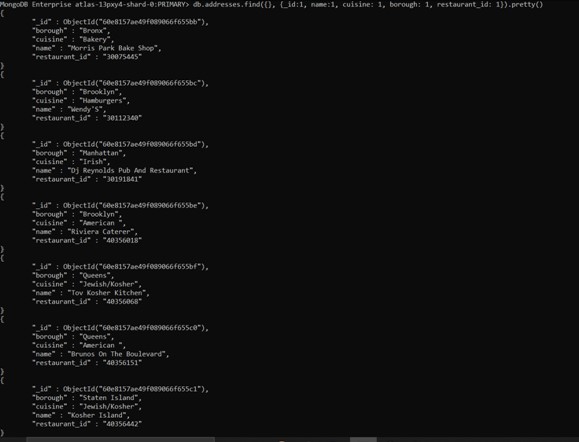


db.addresses.find()



2.

db.addresses.find({}, {\_id:1, name:1, cuisine: 1, borough: 1, restaurant\_id: 1}).pretty()



3.

db.addresses.find({}, {\_id:0, name:1, cuisine: 1, borough: 1, restaurant\_id: 1})



4.

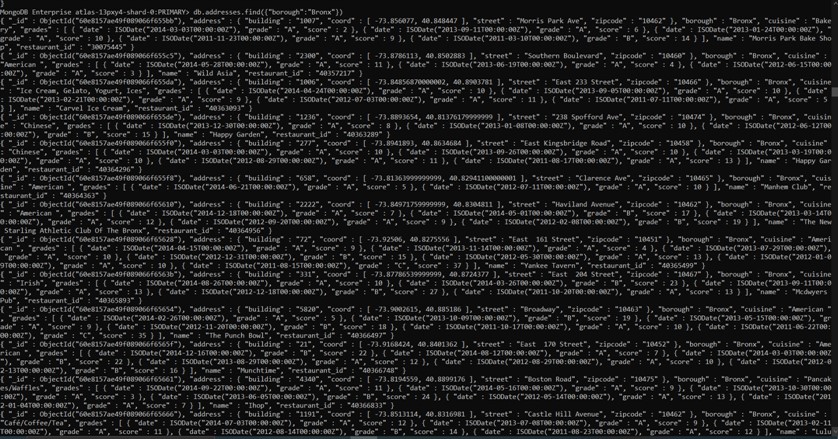
db.addresses.find({}, {\_id:0, name:1, cuisine: 1, borough: 1, restaurant\_id: 1, "address.zip code":1})

5.

db.addresses.find({"borough":"Bronx"}).limit(5)

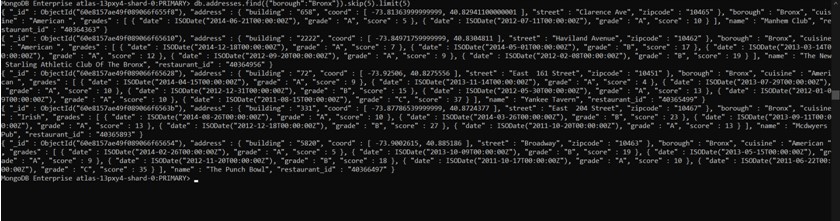
6.

db.addresses.find({"borough":"Bronx"})



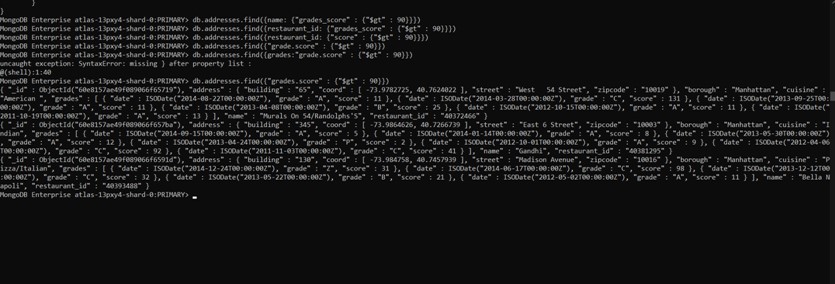
7.

db.addresses.find({"borough":"Bronx"}).skip(5).limit(5)



8.

db.addresses.find({"grades.score" : {"$gt" : 90}})



9.

db.addresses.find({$and:[{"grades.score" : {"$gt" : 80}},{"grades.score" : {"$lt" : 100}}]})

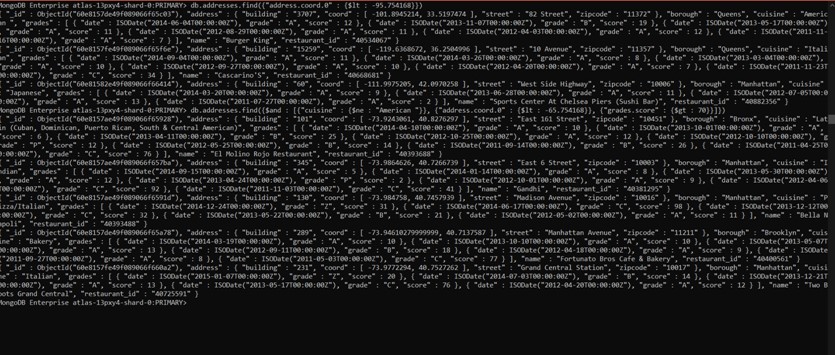


10.

db.addresses.find({"address.coord.0" : {$lt : -95.754168}})

11.

db.addresses.find({$and : [{"cuisine" : {$ne : "American "}}, {"address.coord.1" : {$lt : -65.754168}}, {"grades.score" : {$gt : 70}}]})



12.

db.addresses.find({$and : [{"cuisine" : {$ne : "American "}}, {"address.coord" : {$lt : -65.754168}}, {"grades.score" : {$gt : 70}}]})

13.

db.addresses.find({$and : [{"cuisine" : {$ne : "American "}}, {"grades.grade" : "A"}, {"borough" : {$ne : "Brooklyn "}}]}).sort({cuisine : -1})



14.

db.addresses.find({"name" : { $regex: /^Wil.\*/}}, {\_id:0, restaurant\_id:1, name:1, borough:1, cuisine:1})

15.

db.addresses.find({"name" : { $regex: /.\*ces$/}}, {\_id:0, restaurant\_id:1, name:1, borough:1, cuisine:1})

16.

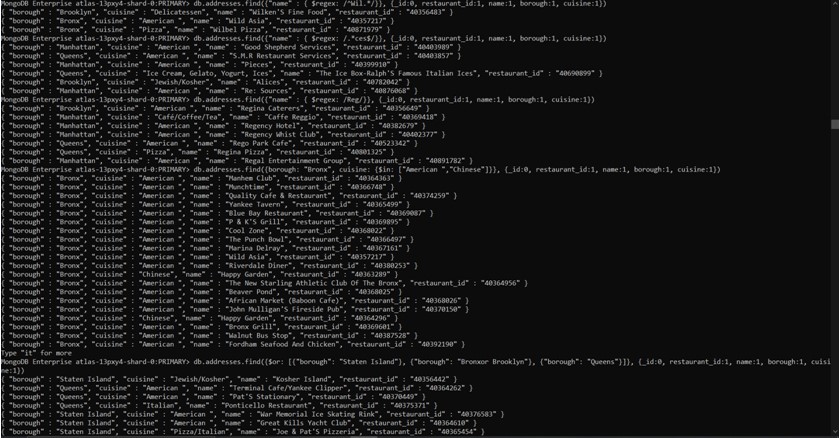
db.addresses.find({"name" : { $regex: /Reg/}}, {\_id:0, restaurant\_id:1, name:1, borough:1, cuisine:1})

17.

db.addresses.find({borough: "Bronx", cuisine: {$in: ["American ","Chinese"]}}, {\_id:0, restaurant\_id:1, name:1, borough:1, cuisine:1})

18.

db.addresses.find({$or: [{"borough": "Staten Island"}, {"borough": "Bronxor Brooklyn"}, {"borough": "Queens"}]}, {\_id:0, restaurant\_id:1, name:1, borough:1, cuisine:1})



19.

db.addresses.find( {borough: {$nin: ["Staten Island","Queens","Bronx","Brooklyn"]}} , {\_id:0, restaurant\_id:1, name:1, borough:1, cuisine:1})

20.

db.addresses.find({"grades.score": {$lt: 10}}, {\_id:0, restaurant\_id:1, name:1, borough:1, cuisine:1})

21.

db.addresses.find({$nor: [{cuisine: {$in: ["American ","Chinese"]}},{name: /^Wil.\*/}]},{\_id:0, restaurant\_id:1, name:1, borough:1, cuisine:1})



22.

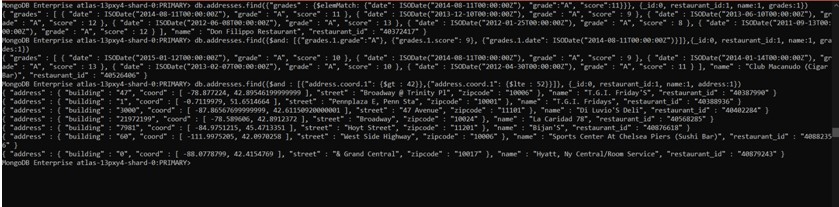
db.addresses.find({"grades" : {$elemMatch: {"date": ISODate("2014-08-11T00:00:00Z"), "grade":"A", "score":11}}}, {\_id:0, restaurant\_id:1, name:1, grades:1})

23.

db.addresses.find({$and: [{"grades.1.grade":"A"}, {"grades.1.score": 9}, {"grades.1.date": ISODate("2014-08-11T00:00:00Z")}]},{\_id:0, restaurant\_id:1, name:1, grades:1})

24.

db.addresses.find({$and : [{"address.coord.1": {$gt : 42}},{"address.coord.1": {$lte : 52}}]}, {\_id:0, restaurant\_id:1, name:1, address:1})

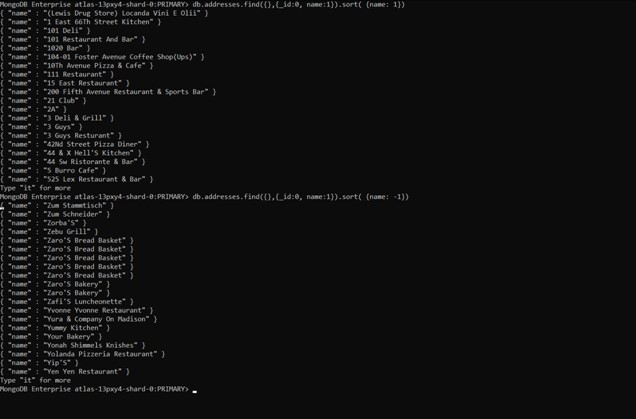


25.

db.addresses.find({},{\_id:0, name:1}).sort( {name: 1})

26.

db.addresses.find({},{\_id:0, name:1}).sort( {name: -1})

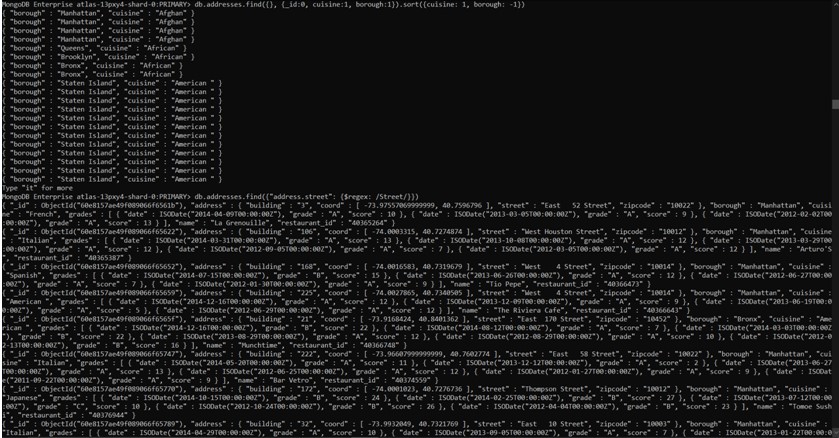


27.

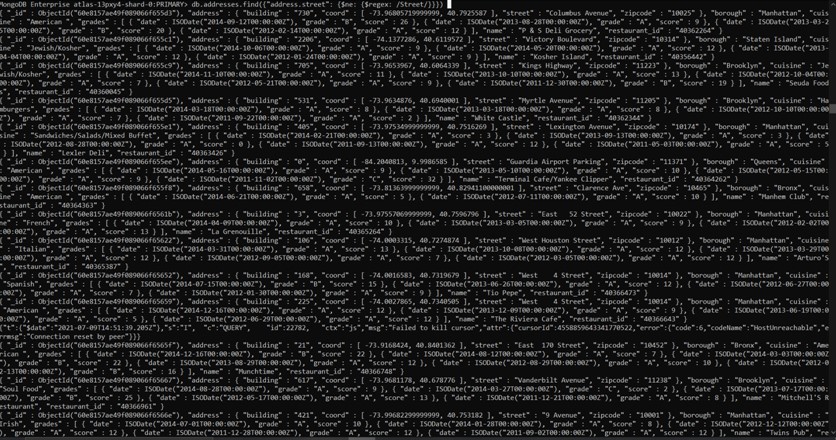
db.addresses.find({}, {\_id:0, cuisine:1, borough:1}).sort({cuisine: 1, borough: -1})

28.

db.addresses.find({"address.street": {$regex: /Street/}})



db.addresses.find({"address.street": {$ne: {$regex: /Street/}}})



29.

db.addresses.find({"address.coord": {$type: "double"}}, {\_id:0, address:1})

30.

db.addresses.find({"grades": {$elemMatch: {"score": {$mod: [7,0]}}}},{\_id:0, restaurant\_id:1, name:1, grades:1})



31.

db.addresses.find({name: {$regex: /mon/}},{\_id:0, name:1, borough:1, "address.coord":1, cuisine:1})

32.

db.addresses.find({name: {$regex: /^Mad.\*/}},{\_id:0, name:1, borough:1, "address.coord":1, cuisine:1})



-------------------------------------------------------------------------------------------------------------------------------