**MongoDB Lab Assignments – Day1**

**> show dbs**

admin 0.000GB

config 0.000GB

local 0.000GB

**Insert Documents:**

**> use.mango\_practice**

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

WriteResult({ "nInserted" : 1 })

**> show dbs**

admin 0.000GB

config 0.000GB

local 0.000GB

mongo\_practice 0.000GB

**> show collections**

movies

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

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

WriteResult({ "nInserted" : 1 })

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

WriteResult({ "nInserted" : 1 })

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

WriteResult({ "nInserted" : 1 })

**> 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."})**

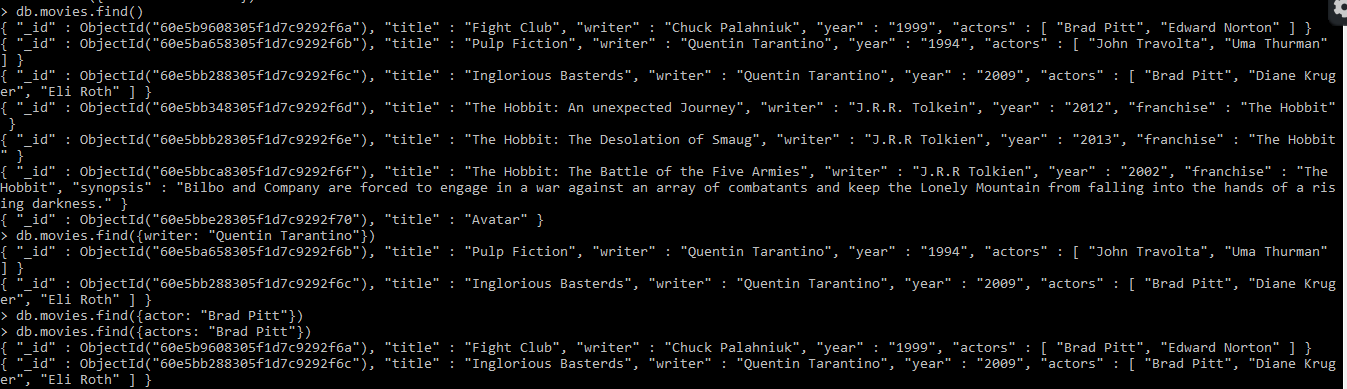
WriteResult({ "nInserted" : 1 })

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

WriteResult({ "nInserted" : 1 })

**Query/Find Documents:**

**> db.movies.find()**



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

{ "\_id" : ObjectId("60e5ba658305f1d7c9292f6b"), "title" : "Pulp Fiction", "writer" : "Quentin Tarantino", "year" : "1994", "actors" : [ "John Travolta", "Uma Thurman" ] }

{ "\_id" : ObjectId("60e5bb288305f1d7c9292f6c"), "title" : "Inglorious Basterds", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "Brad Pitt", "Diane Kruger", "Eli Roth" ] }

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

{ "\_id" : ObjectId("60e5b9608305f1d7c9292f6a"), "title" : "Fight Club", "writer" : "Chuck Palahniuk", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }

{ "\_id" : ObjectId("60e5bb288305f1d7c9292f6c"), "title" : "Inglorious Basterds", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "Brad Pitt", "Diane Kruger", "Eli Roth" ] }

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

{ "\_id" : ObjectId("60e5bb348305f1d7c9292f6d"), "title" : "The Hobbit: An unexpected Journey", "writer" : "J.R.R. Tolkein", "year" : "2012", "franchise" : "The Hobbit" }

{ "\_id" : ObjectId("60e5bbb28305f1d7c9292f6e"), "title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R Tolkien", "year" : "2013", "franchise" : "The Hobbit" }

{ "\_id" : ObjectId("60e5bbca8305f1d7c9292f6f"), "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." }

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

{ "\_id" : ObjectId("60e5b9608305f1d7c9292f6a"), "title" : "Fight Club", "writer" : "Chuck Palahniuk", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }

{ "\_id" : ObjectId("60e5ba658305f1d7c9292f6b"), "title" : "Pulp Fiction", "writer" : "Quentin Tarantino", "year" : "1994", "actors" : [ "John Travolta", "Uma Thurman" ] }

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

{ "\_id" : ObjectId("60e5b9608305f1d7c9292f6a"), "title" : "Fight Club", "writer" : "Chuck Palahniuk", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }

{ "\_id" : ObjectId("60e5ba658305f1d7c9292f6b"), "title" : "Pulp Fiction", "writer" : "Quentin Tarantino", "year" : "1994", "actors" : [ "John Travolta", "Uma Thurman" ] }

{ "\_id" : ObjectId("60e5bb348305f1d7c9292f6d"), "title" : "The Hobbit: An unexpected Journey", "writer" : "J.R.R. Tolkein", "year" : "2012", "franchise" : "The Hobbit" }

{ "\_id" : ObjectId("60e5bbb28305f1d7c9292f6e"), "title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R Tolkien", "year" : "2013", "franchise" : "The Hobbit" }

**Update Documents:**

**> db.movies.update({\_id: ObjectId("60e5bb348305f1d7c9292f6d")},{$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."}})**

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

**> db.movies.update({\_id : ObjectId("60e5bbb28305f1d7c9292f6e")}, {$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."}})**

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

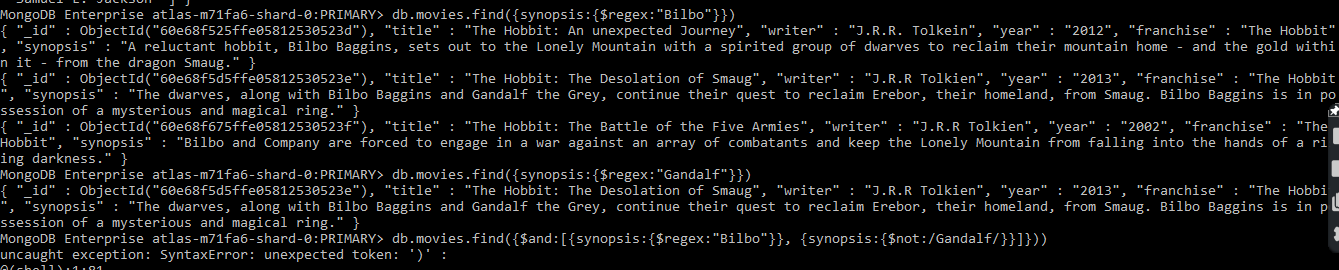
**> db.movies.update({\_id : ObjectId("60e5ba658305f1d7c9292f6b")}, {$push:{actors: "Samuel L.Jackson"}})**

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

**Text Search:**

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

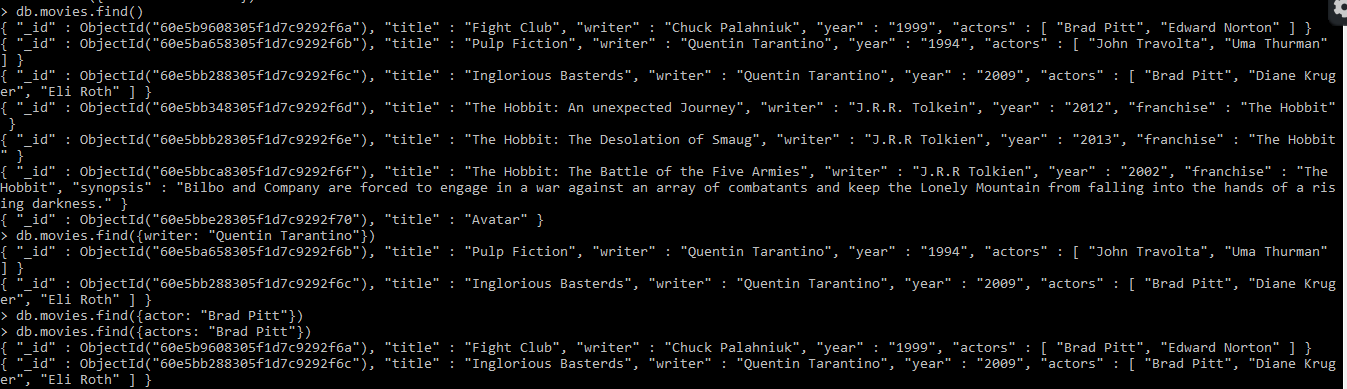
**> db.movies.find({synopsis:{$regex:"Gandalf"}})**



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

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

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



**Delete Documents:**

**> db.movies.remove({"\_id" : ObjectId("60e68f735ffe058125305240")})**

WriteResult({ "nRemoved" : 1 })

**> db.movies.remove({"\_id" : ObjectId("60e68f7f5ffe058125305241")})**

WriteResult({ "nRemoved" : 1 })

**Relationships:**

users Collection:

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

WriteResult({ "nInserted" : 1 })

**> db.users.insert({username: "ScumbagSteve", full\_name:{first:"Scumbag", last: "Steve"}})**

WriteResult({ "nInserted" : 1 })

posts collection:

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

WriteResult({ "nInserted" : 1 })

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

WriteResult({ "nInserted" : 1 })

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

WriteResult({ "nInserted" : 1 })

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

WriteResult({ "nInserted" : 1 })

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

WriteResult({ "nInserted" : 1 })

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

WriteResult({ "nInserted" : 1 })

Comments collection:

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

WriteResult({ "nInserted" : 1 })

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

WriteResult({ "nInserted" : 1 })

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

WriteResult({ "nInserted" : 1 })

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

WriteResult({ "nInserted" : 1 })

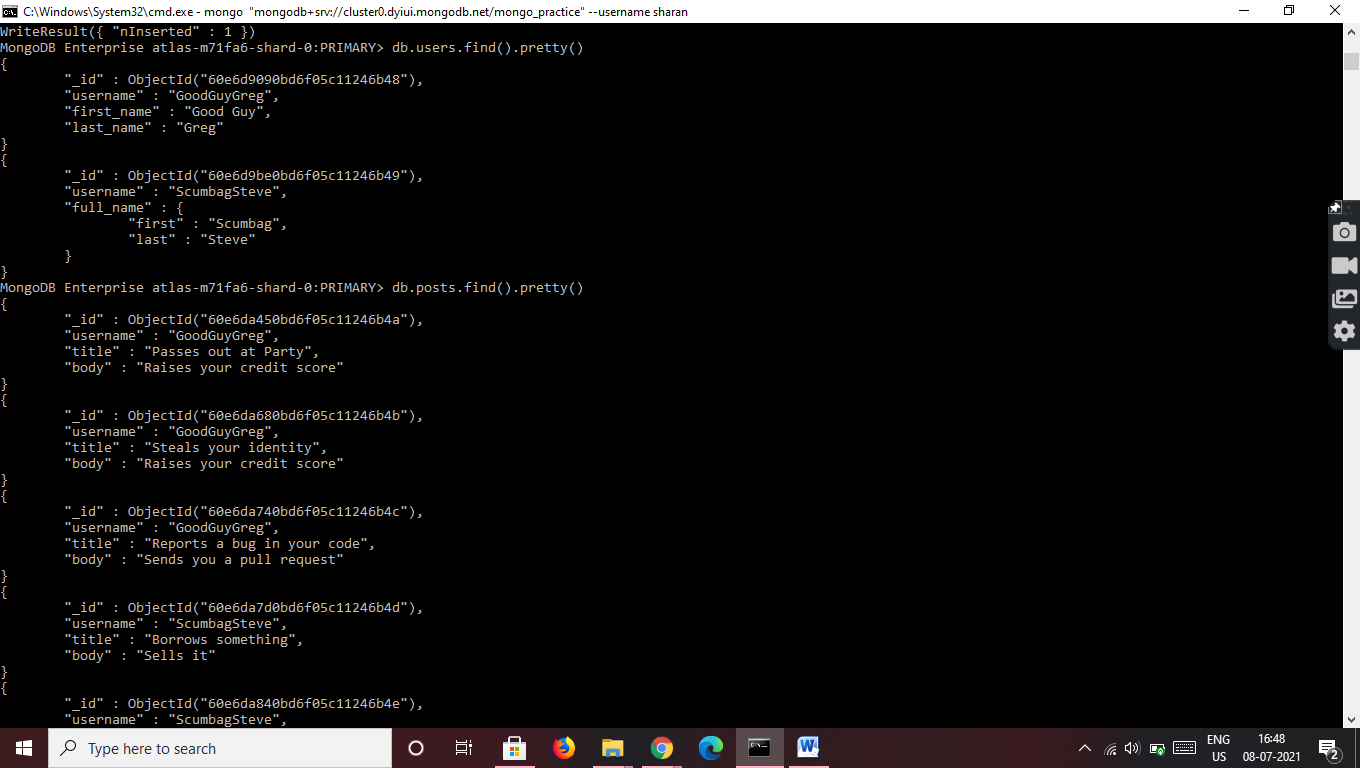
**> db.comments.insert({username:"ScumbagSteve", comment:"Denied your PR cause I found a hack", post:ObjectId("5ca0b9256435f98b5901f469")})**

WriteResult({ "nInserted" : 1 })

**Query Related Collections:**

**MongoDB Enterprise atlas-m71fa6-shard-0:PRIMARY> db.users.find().pretty()**

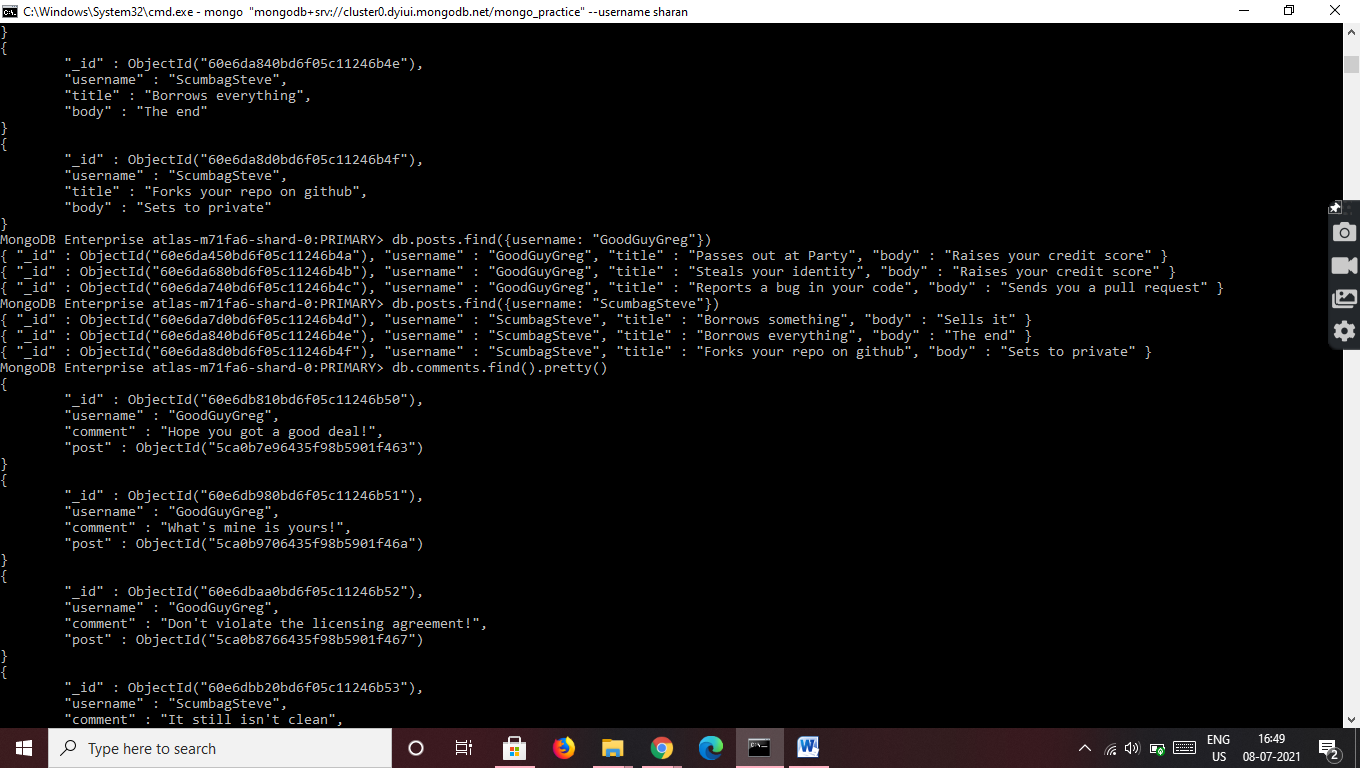
**MongoDB Enterprise atlas-m71fa6-shard-0:PRIMARY> db.posts.find().pretty()**

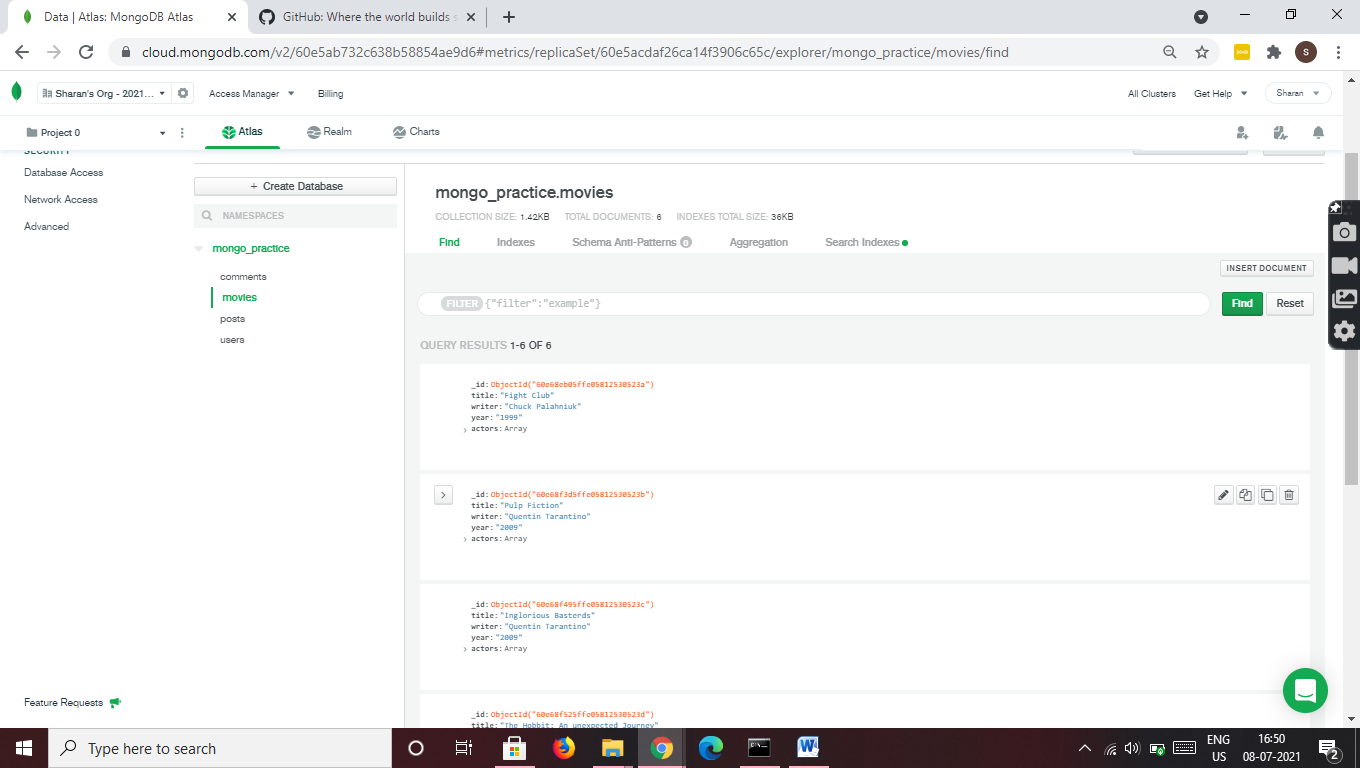


**MongoDB Enterprise atlas-m71fa6-shard-0:PRIMARY> db.posts.find({username: "GoodGuyGreg"})**

**MongoDB Enterprise atlas-m71fa6-shard-0:PRIMARY> db.posts.find({username: "ScumbagSteve"})**

**MongoDB Enterprise atlas-m71fa6-shard-0:PRIMARY> db.comments.find().pretty()**

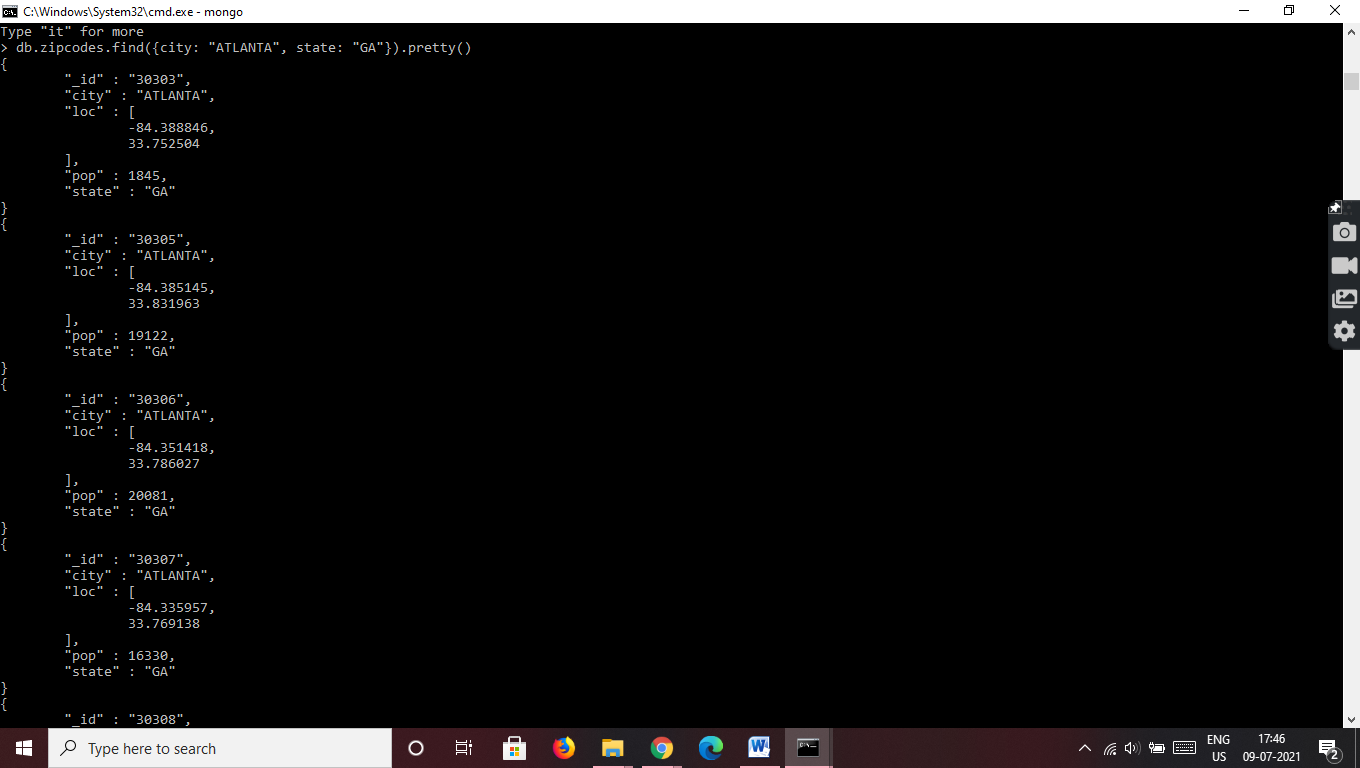
****



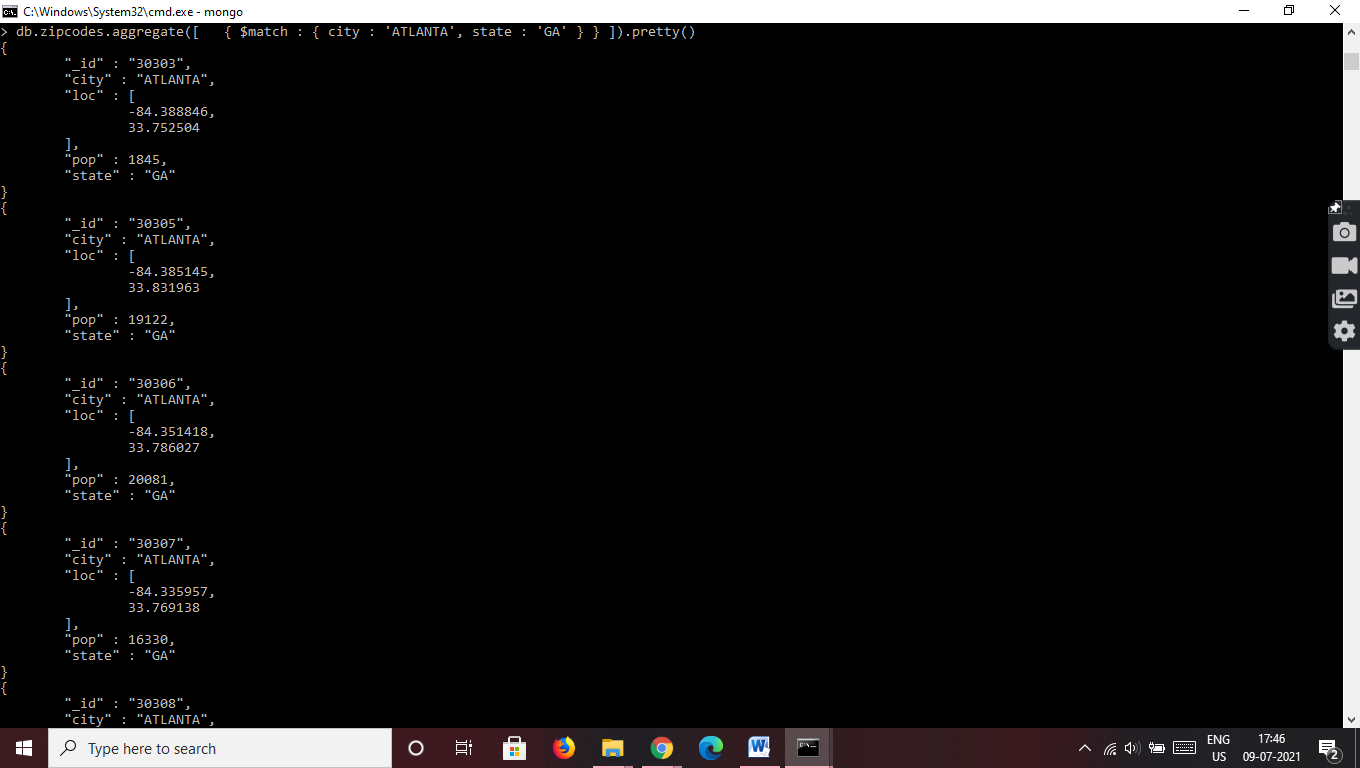
**MongoDB – Aggregation Exercise**

**Atlanta Population:**

**> db.zipcodes.find({city: "ATLANTA", state: "GA"}).pretty()**

****

**>db.zipcodes.aggregate([ { $match : { city : 'ATLANTA', state : 'GA' } } ]).pretty()**

****

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

{ "\_id" : "ATLANTA", "zipcodes" : 41 }

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

{ "\_id" : "ATLANTA", "total\_pop" : 630046 }

**Population by State:**

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

{ "\_id" : "MA", "totalPop" : 6016425 }

{ "\_id" : "TN", "totalPop" : 4876457 }

{ "\_id" : "UT", "totalPop" : 1722850 }

{ "\_id" : "DC", "totalPop" : 606900 }

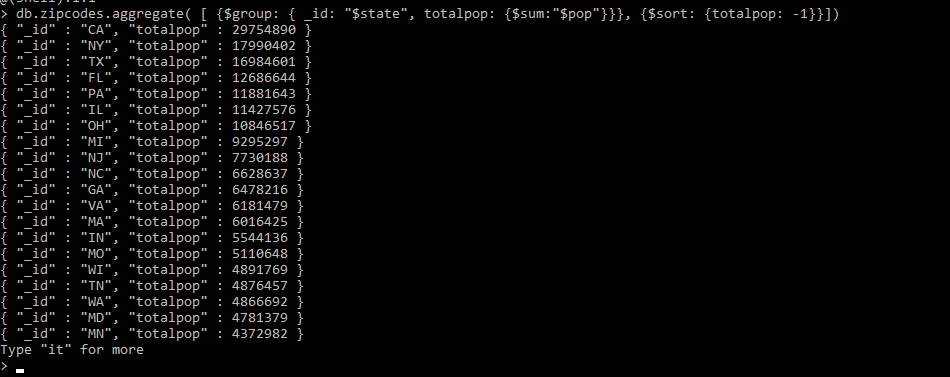
{ "\_id" : "KY", "totalPop" : 3675484 }

{ "\_id" : "IL", "totalPop" : 11427576 }

{ "\_id" : "SC", "totalPop" : 3486703 }

{ "\_id" : "OH", "totalPop" : 10846517 }

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



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

{ "\_id" : "CA", "totalpop" : 29754890 }

{ "\_id" : "NY", "totalpop" : 17990402 }

{ "\_id" : "TX", "totalpop" : 16984601 }

**Population by City:**

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



**> db.zipcodes.aggregate( [ {$group: { \_id: "$city", Totalpop: {$sum:"$pop"}}}, {$sort: {totalpop: -1}}])**

{ "\_id" : "SICILY ISLAND", "Totalpop" : 1568 }

{ "\_id" : "DARDANELLE", "Totalpop" : 8281 }

{ "\_id" : "STRAUGHN", "Totalpop" : 1300 }

{ "\_id" : "BENOIT", "Totalpop" : 1554 }

{ "\_id" : "TALBOTTON", "Totalpop" : 2324 }

{ "\_id" : "BOYD TAVERN", "Totalpop" : 2081 }

{ "\_id" : "VAN BUREN BAY", "Totalpop" : 20303 }

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

{ "\_id" : "AMARILLO", "Totalpop" : 173158 }

{ "\_id" : "TYNDALL", "Totalpop" : 1451 }

{ "\_id" : "SLAUGHTER", "Totalpop" : 4558 }

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

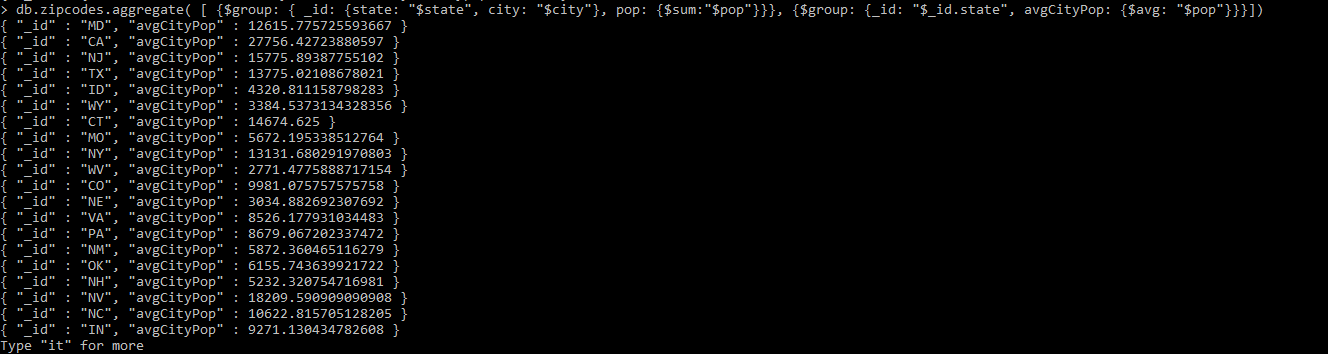
{ "\_id" : { "state" : "Texas", "city" : "BUCKFIELD" }, "citypop" : 1551 }

{ "\_id" : { "state" : "Texas", "city" : "ILWACO" }, "citypop" : 1210 }

{ "\_id" : { "state" : "Texas", "city" : "BONIFAY" }, "citypop" : 9342 }

**Bonus:**

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



**> db.zipcodes.aggregate( [ {$group: { \_id: {state: "$state", city: "$city"}, pop: {$sum:"$pop"}}}, {$group: {\_id: "$\_id.state", avgCityPop: {$avg: "$pop"}}}, {$limit: 3}])**

{ "\_id" : "WA", "avgCityPop" : 12258.670025188916 }

{ "\_id" : "SD", "avgCityPop" : 1839.6746031746031 }

{ "\_id" : "RI", "avgCityPop" : 19292.653846153848 }

**Mongo DB – Complex Queries**

**Mongo DB Exercise – With Restaurant Dataset**

**> show dbs**

admin 0.000GB

config 0.000GB

local 0.000GB

mongo\_practice 0.000GB

population 0.003GB

restaurants 0.001GB

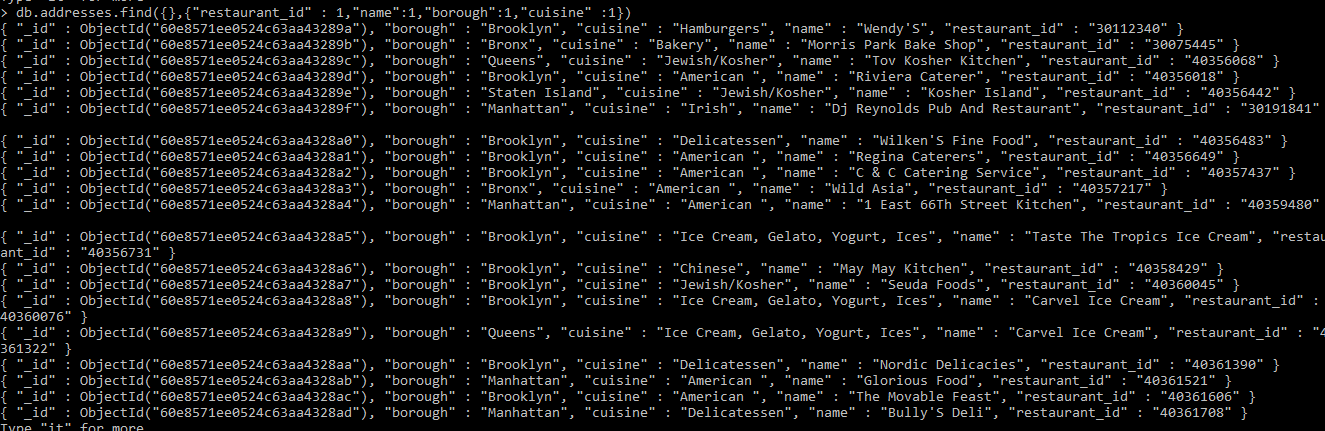
**> use restuarants**

switched to db restuarants

**> show collections**

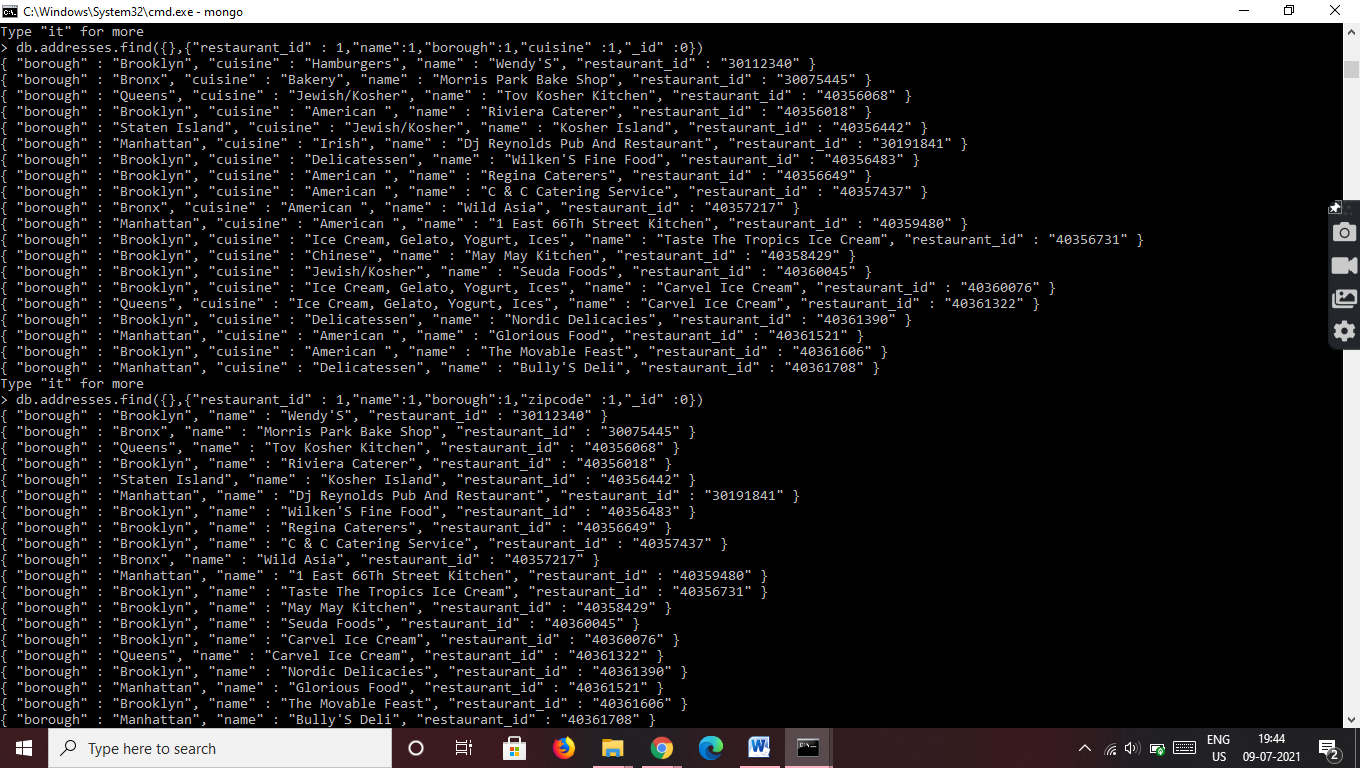
addresses

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

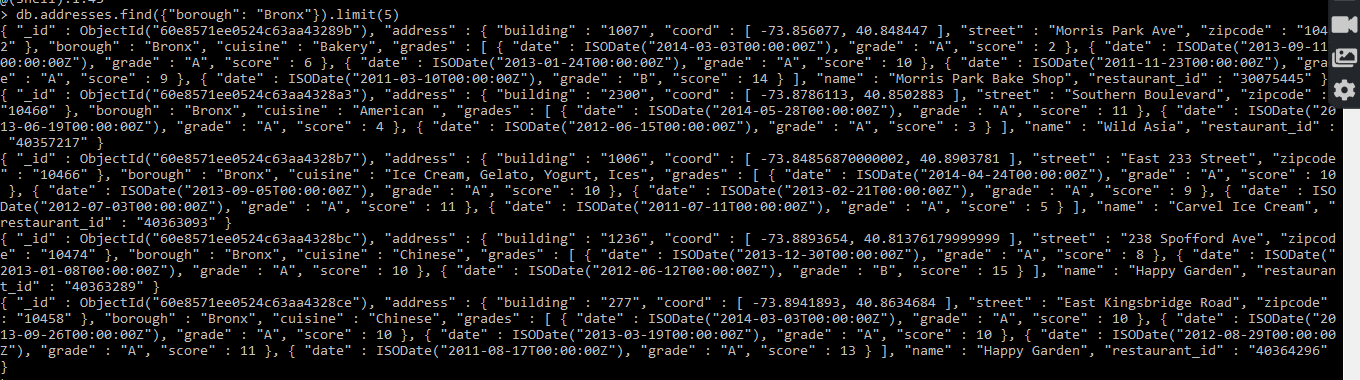
****

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

**> db.addresses.find({},{"restaurant\_id" : 1,"name":1,"borough":1,"zipcode" :1,"\_id" :0})**

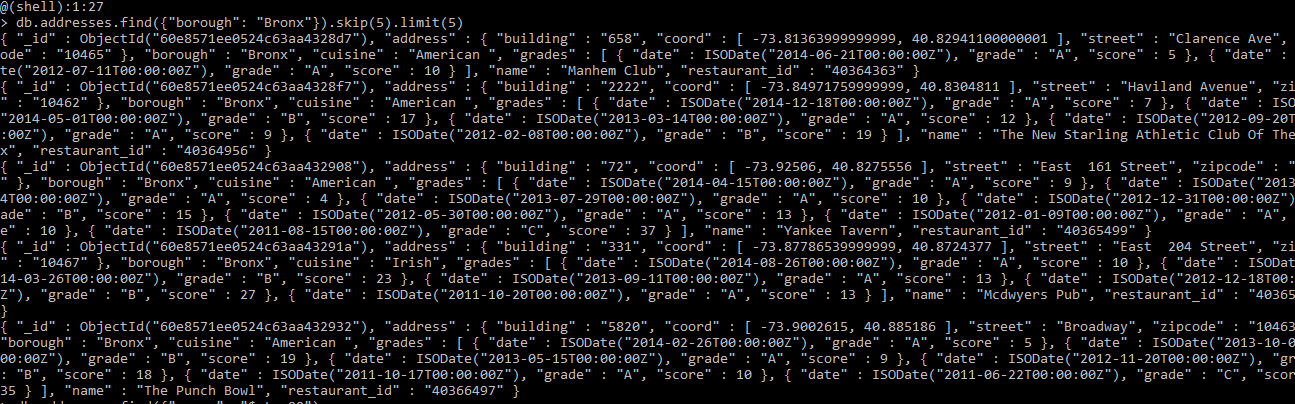
****

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

****

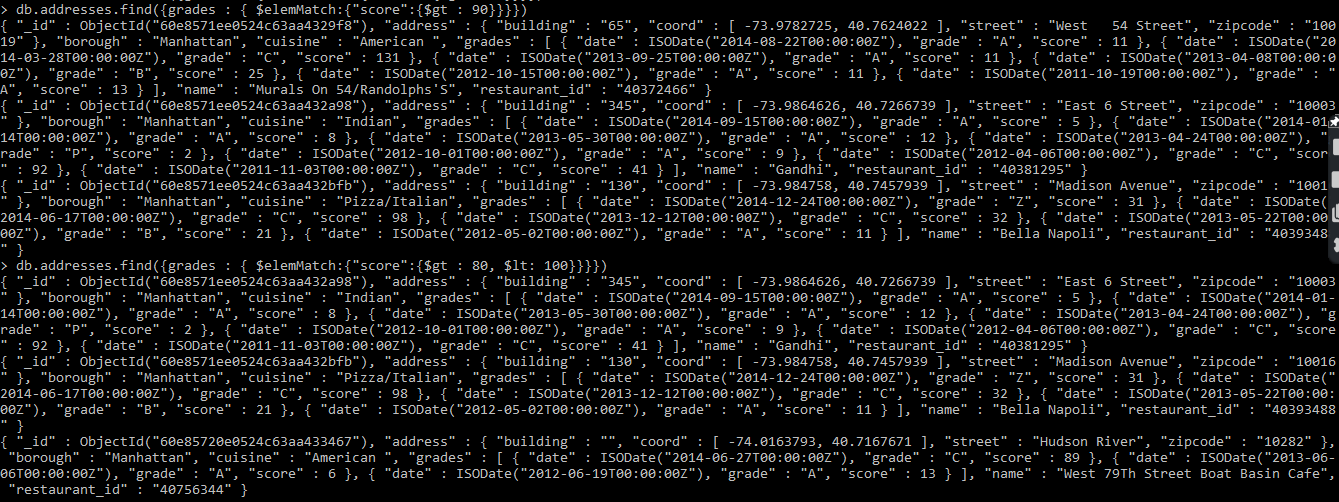
**> db.addresses.find({"borough": "Bronx"})**

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

****

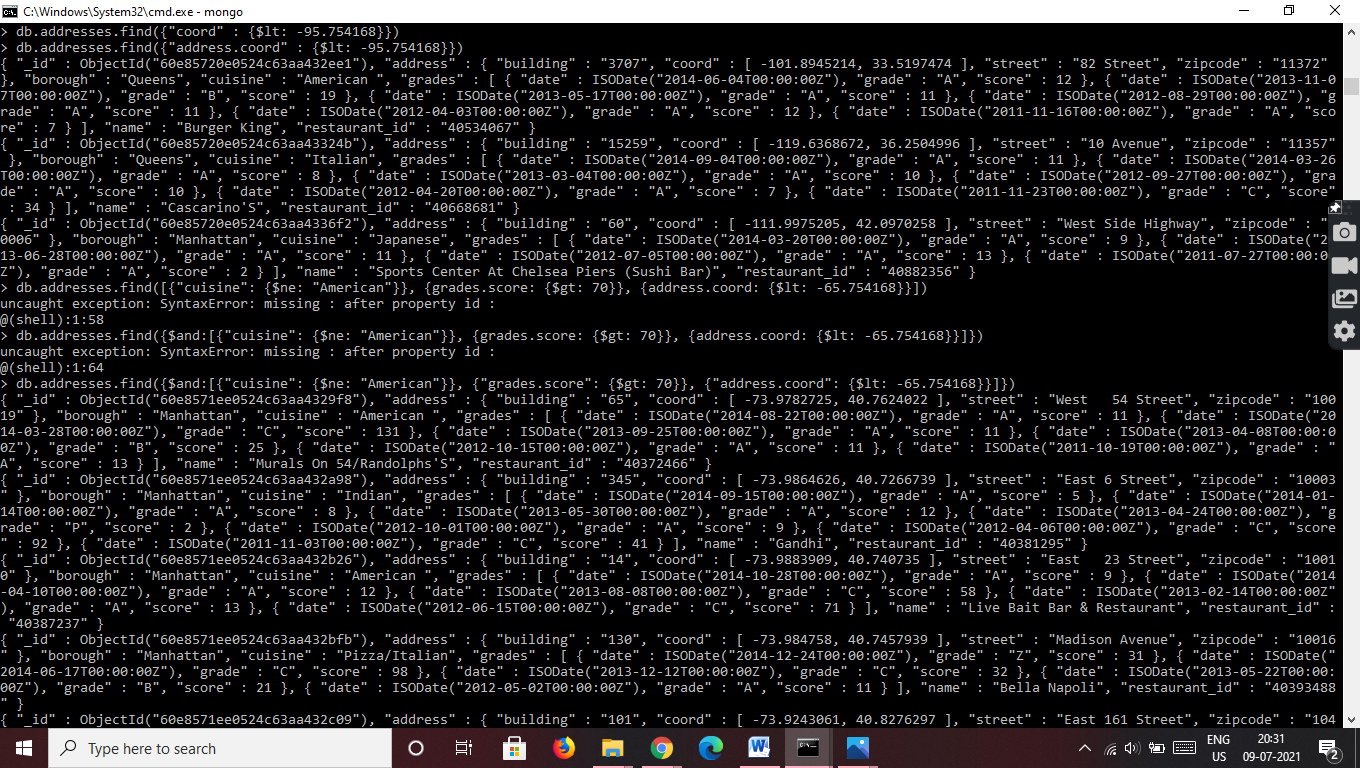
**> db.addresses.find({grades : { $elemMatch:{"score":{$gt : 90}}}})**

**> db.addresses.find({grades : { $elemMatch:{"score":{$gt : 80, $lt: 100}}}})**

****

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

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

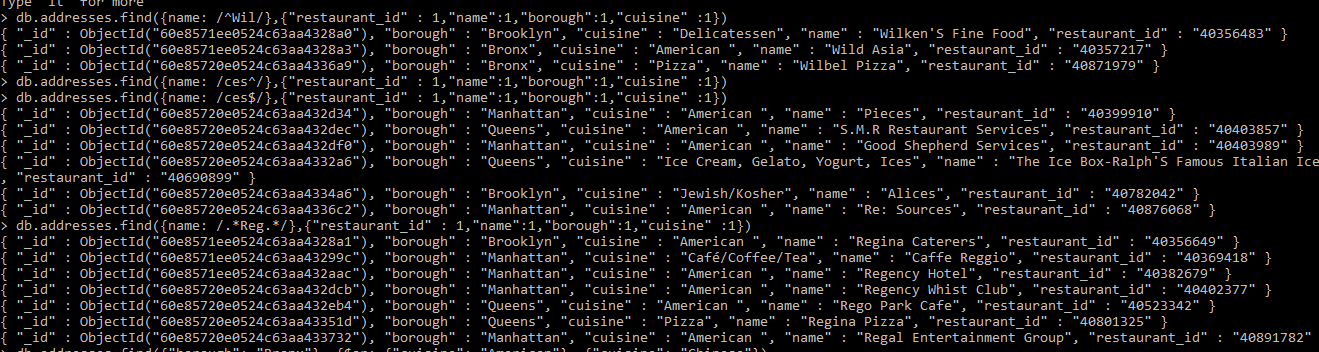
****

**> db.addresses.find( {"cuisine" : {$ne : "American "}, "grades.grade" :"A", "borough": {$ne : "Brooklyn"} } ).sort({"cuisine":-1});**

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

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

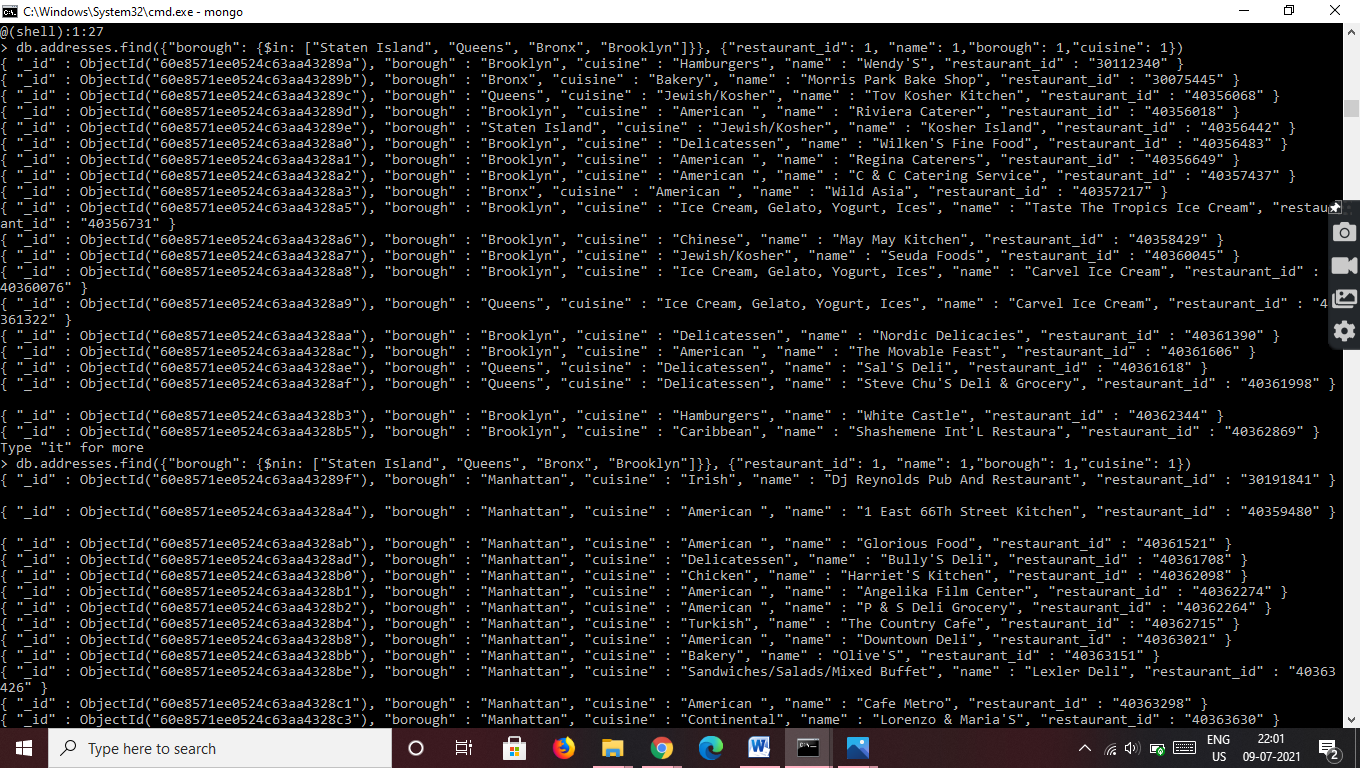
**> db.addresses.find({name: /.\*Reg.\*/},{"restaurant\_id" : 1,"name": 1, "borough": 1, "cuisine": 1})**

****

**> db.addresses.find({"borough": "Bronx", $or: [{"cuisine": "American"}, {"cuisine": "Chinese"}]})**

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

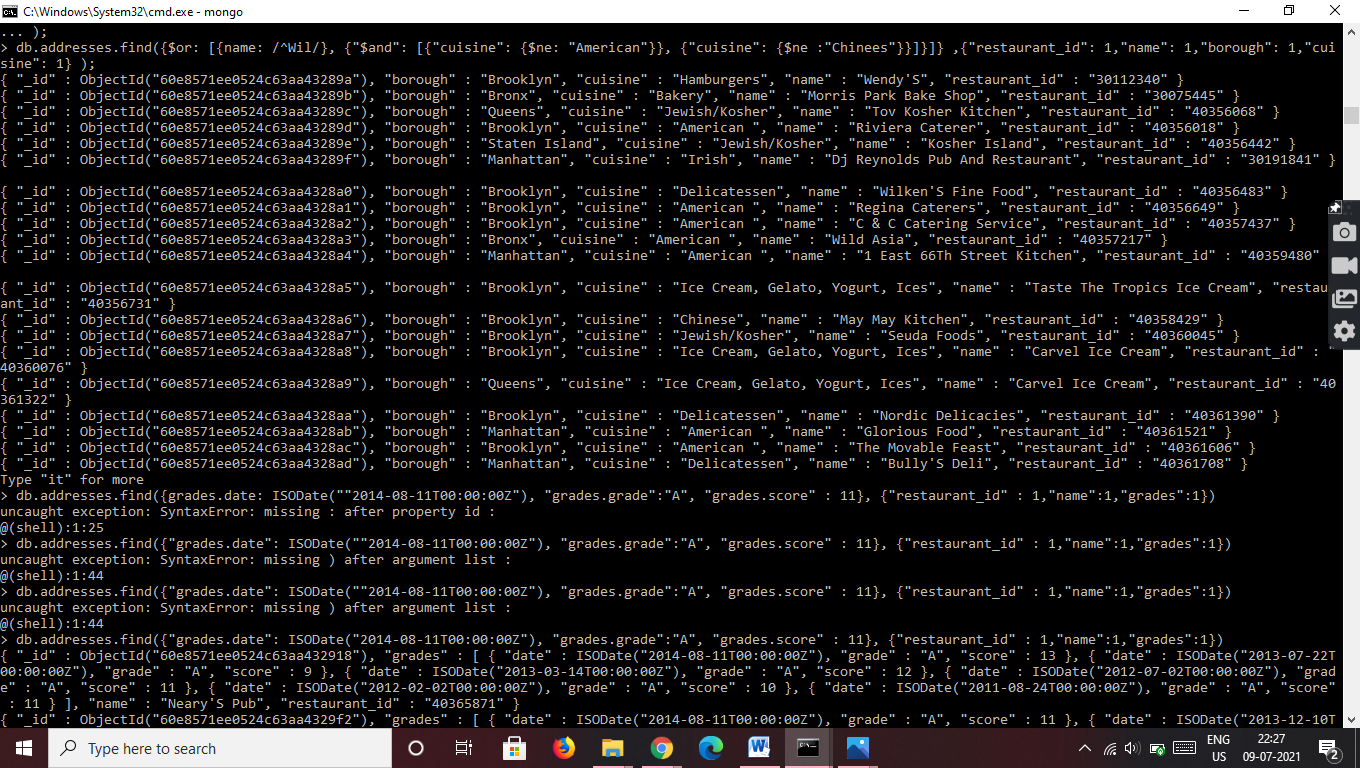
**> db.addresses.find({"borough": {$nin: ["Staten Island", "Queens", "Bronx", "Brooklyn"]}}, {"restaurant\_id": 1, "name": 1,"borough": 1,"cuisine": 1})**

****

**> db.addresses.find({"grade.score": {$not: {$gt: 10}}},{"restaurant\_id": 1,"name": 1,"borough": 1,"cuisine": 1})**

**> db.addresses.find({$or: [{name: /^Wil/}, {"$and": [{"cuisine": {$ne: "American"}}, {"cuisine": {$ne :"Chinees"}}]}]} ,{"restaurant\_id": 1,"name": 1,"borough": 1,"cuisine": 1} );**

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



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

**> db.addresses.find().sort({name:1})**

**> db.addresses.find().sort({name:-1})**

**> db.addresses.find().sort({cuisine:1,borough:-1})**

**> db.addresses.find({"address.street" : {$exists: true}})**

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

**> db.addresses.find({name: /.\*mon.\*/}, {name: 1, borough: 1, cuisine: 1, "address.coord": 1})**

**> db.addresses.find({name: /^Mad/}, {name: 1, borough: 1, cuisine: 1, "address.coord": 1})**

