**ASSIGNMENT-1:**

**MongoDB Practice**

MongoDB Exercise in mongo shell, Mongo Atlas

Connected the MongoAtlas to the local database

Created a database in Atlas cluster (mongo\_practice)

**Create database**

Connect to a running mongo instance, use a database named mongo\_practice.

Use mongo\_practice

**Insert Documents**

Insert the following documents into a movies collection.

Created a collection in the the db mongo\_practice at Mongo Atlas

title : Fight Club

writer : Chuck Palahniuk

year : 1999

actors : [

Brad Pitt

Edward Norton

]

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

title : Pulp Fiction

writer : Quentin Tarantino

year : 1994

actors : [

John Travolta

Uma Thurman

]

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

title : Inglorious Basterds

writer : Quentin Tarantino

year : 2009

actors : [

Brad Pitt

Diane Kruger

Eli Roth

]

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

title : The Hobbit: An Unexpected Journey

writer : J.R.R. Tolkein

year : 2012

franchise : The Hobbit

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

title : The Hobbit: The Desolation of Smaug

writer : J.R.R. Tolkein

year : 2013

franchise : The Hobbit

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

title : The Hobbit: The Battle of the Five Armies

writer : J.R.R. Tolkein

year : 2012

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.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."})**

title : Pee Wee Herman's Big Adventure

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

title : Avatar

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

**Query / Find Documents**

query the movies collection to

1. get all documents

**db.movies.find()**

1. get all documents with writer set to "Quentin Tarantino"

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

1. get all documents where actors include "Brad Pitt"

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

1. get all documents with franchise set to "The Hobbit"

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

1. get all movies released in the 90s

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

1. get all movies released before the year 2000 or after 2010

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

**Update Documents**

1. add a synopsis to "The Hobbit: An Unexpected Journey" : "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."

**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."}})**

1. add a synopsis to "The Hobbit: The Desolation of Smaug" : "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."

**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."}})**

1. add an actor named "Samuel L. Jackson" to the movie "Pulp Fiction"

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

**Text Search**

1. find all movies that have a synopsis that contains the word "Bilbo"

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

1. find all movies that have a synopsis that contains the word "Gandalf"

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

1. find all movies that have a synopsis that contains the word "Bilbo" and not the word "Gandalf"

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

1. find all movies that have a synopsis that contains the word "dwarves" or "hobbit"

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

1. find all movies that have a synopsis that contains the word "gold" and "dragon"

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

**Delete Documents**

1. delete the movie "Pee Wee Herman's Big Adventure"

**db.movies.remove({\_id:ObjectId("5c9f992ae5c2dfe9b3729c00")})**

1. delete the movie "Avatar"

**db.movies.remove({\_id:ObjectId("5c9f9936e5c2dfe9b3729c01")})**

**Relationships**

**Insert the following documents into a users collection**

username : GoodGuyGreg

first\_name : "Good Guy"

last\_name : "Greg"

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

username : ScumbagSteve

full\_name :

first : "Scumbag"

last : "Steve"

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

**Insert the following documents into a posts collection**

username : GoodGuyGreg

title : Passes out at party

body : Wakes up early and cleans house

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

username : GoodGuyGreg

title : Steals your identity

body : Raises your credit score

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

username : GoodGuyGreg

title : Reports a bug in your code

body : Sends you a Pull Request

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

username : ScumbagSteve

title : Borrows something

body : Sells it

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

username : ScumbagSteve

title : Borrows everything

body : The end

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

username : ScumbagSteve

title : Forks your repo on github

body : Sets to private

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

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

username : GoodGuyGreg

comment : Hope you got a good deal!

post : [post\_obj\_id]

where [post\_obj\_id] is the ObjectId of the posts document: "Borrows something"

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

username : GoodGuyGreg

comment : What's mine is yours!

post : [post\_obj\_id]

where [post\_obj\_id] is the ObjectId of the posts document: "Borrows everything"

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

username : GoodGuyGreg

comment : Don't violate the licensing agreement!

post : [post\_obj\_id]

where [post\_obj\_id] is the ObjectId of the posts document: "Forks your repo on github

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

username : ScumbagSteve

comment : It still isn't clean

post : [post\_obj\_id]

where [post\_obj\_id] is the ObjectId of the posts document: "Passes out at party"

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

username : ScumbagSteve

comment : Denied your PR cause I found a hack

post : [post\_obj\_id]

where [post\_obj\_id] is the ObjectId of the posts document: "Reports a bug in your code"

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

**Querying related collections**

1. find all users

**db.users.find().pretty()**

1. find all posts

**db.posts.find().pretty()**

1. find all posts that was authored by "GoodGuyGreg"

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

1. find all posts that was authored by "ScumbagSteve"

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

1. find all comments

**db.comments.find().pretty()**

1. find all comments that was authored by "GoodGuyGreg"

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

1. find all comments that was authored by "ScumbagSteve"

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

1. find all comments belonging to the post "Reports a bug in your code"

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

**ASSIGNMENT-2: Aggregation excercises**

Installed mongodatabase tool for windows , opened cmd in that, further tracked the location of JSON files, and implemented the code:

mongoimport –db population –collection zipcodes –file (location of Json file)

Opened new cmd and connected to mongo server by command line mongo

Then checked for dbs by show dbs

Used population dbs by use population

Further bellows are the queries implemented:

**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" }}}])

**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}]

**Population 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",},populationIs: { $sum:"$pop"}} } ,{$sort:{populationIs: -1}},{$limit: 3}])
4. db.zipcodes.aggregate([{$match: {state:"TX"}},{$sort:{pop: -1}},{$limit:3}])

**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} ])

**ASSIGNMENT – 3: COMPLEX QUERIES**

**With restaurants data set:**

Unzipped the restaurants.zip file

Ran MongoDB server

Use MongoDb additional tools and imported the restaurants.json files into the database through the command:

**Mongoimport –-db restaurants –-collection addresses –-file C:\Users\Lenovo\Downloads\restaurants.json 🡪 command line**

Ran MongoShell

Ran command show dbs

Switched to restaurants db by use restaurants command line

Ran **db.addresses.find()**

**Exercise Questions:**

1. **db.addresses.find()**

**db.stats()- to get the complete current db status**

**db.addressess.find().pretty()**

1. **db.addresses.find({},{restaurant\_id:1,name:1,borough:1,cuisine:1})**
2. **db.addresses.find({},{restaurant\_id:1,name:1,borough:1,cuisine:1,\_id:0})**
3. **db.addresses.find({},{restaurant\_id:1,name:1,borough:1,”address.zipcode”:1,\_id:0})** -🡪replace those double quotes as per the need as it results in uncaught exception while execution
4. **db.addresses.find({borough:”Bronx”}).limit(5)** -🡪replace those double quotes as per the need as it results in uncaught exception while execution
5. **db.addresses.find({borough:”Bronx”})** -🡪replace those double quotes as per the need as it results in uncaught exception while execution
6. **db.addresses.find({borough:”Bronx”}).skip(5).limit(5)** -🡪replace those double quotes as per the need as it results in uncaught exception while execution
7. **db.addresses.find({“grades.score”:{$gt:90}})** -🡪replace those double quotes as per the need as it results in uncaught exception while execution
8. **db.addresses.find({$and:[{"grades.score":{$gt:80}},{"grades.score":{$lt:100}}]})**
9. **db.addresses.find({"address.coord.0":{$lt:-95.754168}})**
10. **db.addresses.find({$and:[{cuisine:{$not:{$regex:"American"}}},{"grades.score":{$gt:70}},{"address.coord.0":{$lt:-65.754168}}]})**
11. **db.addresses.find({$and:[{cuisine:{$not:{$regex:"American"}}},{"grades.score":{$gt:70}},{"address.coord.1":{$lt:-65.754168}}]})**
12. **db.addresses.find({$and:[{cuisine:{$not:{$regex:"American"}}},{"grades.grade":"A"},{borough:{$not:{$regex:"Brooklyn"}}}]}).sort({cuisine:-1})**
13. **db.addresses.find({name:/^Wil/},{restaurant\_id:1,name:1,borough:1,cuisine:1})**
14. **db.addresses.find({name:/ces$/},{restaurant\_id:1,name:1,borough:1,cuisine:1})**
15. **db.addresses.find({name:/.\*Reg.\*/},{restaurant\_id:1,name:1,borough:1,cuisine:1})**
16. **db.addresses.find({borough:"Bronx",$or:[{cuisine:"American"},{cuisine:"Chinese"}]})**
17. **db.addresses.find({$or:[{borough:"Staten Island"},{borough:"Queens"},{borough:"Bronx"},{borough:"Brooklyn"}]},{restaurant\_id:1,name:1,borough:1,cuisine:1})**
18. **db.addresses.find({$and:[{borough:{$not:/Bronx/}},{borough:{$not:/Brooklyn/}},{borough:{$not:/Staten Island/}},{borough:{$not:/Queens/}}]},{restaurant\_id:1,name:1,borough:1,cuisine:1})**
19. **db.addresses.find({"grades.score":{$lte:10}},{restaurant\_id:1,name:1,borough:1,cuisine:1})**
20. **db.addresses.find({$or:[{name:/^Wil/},{$and:[{cuisine:{$not:/American/}},{cuisine:{$not:/Chinese/}}]}]},{restaurant\_id:1,name:1,borough:1,cuisine:1})**
21. **db.addresses.find({"grades.grade":"A","grades.score":11,"grades.date":ISODate('2014-08-11T00:00:00Z')}, {restaurant\_id:1,name:1,grades:1})**
22. **db.addresses.find({"grades.1.grade":"A","grades.1.score":9,"grades.1.date":ISODate('2014-08-11T00:00:00Z')}, {restaurant\_id:1,name:1,grades:1})**
23. **db.addresses.find({$and:[{"address.coord.1":{$gt:42}},{"address.coord.1":{$lte:52}}]},{restaurant\_id:1,name:1,"address":1})**
24. **db.addresses.find().sort({name:1})**
25. **db.addresses.find().sort({name:-1})**
26. **db.addresses.find().sort({cuisine:1,borough:-1})**
27. **db.addresses.find({$or:[{“addresses.street”:{$exists:true}},{“addresses.street”:{$exists:false}}]})**
28. **db.addresses.find({“address.coord”:{$type:1}})**
29. **db.addresses.find({“grades.score”:{$mod:[7,0]}},{restaurant\_id:1,name:1,grades:1});**
30. **db.addresses.find({name:/.\*mon.\*/}, {name:1,borough:1,cuisine:1,"address.coord":1})**
31. **db.addresses.find({name:/^Mad/}, {name:1,borough:1,cuisine:1,"address.coord":1})**

|  |
| --- |
|  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |