**MongoDB assessment 1**

**Inserting Documents into Movie collection**

**Connecting to database**

mongo "mongodb+srv://cluster0.srt2w.mongodb.net/myFirstDatabase" --username user01 //for connecting cluster

use mongo\_practice //create db

db.createCollection('movies') // create collection

db.movies.insertOne({title: "Fight Club", writer: "Chuck Palahniuko", year: 1999, actor: ["Brad Pitt","Edward Norton" ]})

db.movies.insertMany([{title: "Pulp Fiction", writer: "Quentin Tarantino", year:1994, actor: ["john Travolta","Uma Thurman" ]},

{title : "Inglorious Basterds", writer: "Quentin Tarantino", year: 2009, actor: ["Brad Pitt", "Diane Kruger", "Eli Roth" ]},

{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"},

{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."},

{title: "Pee Wee Herman's Big Adventure"},

{title: "Avatar"}])

**Find Documents:**

db.movies.find().pretty() //// all documents

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

db.movies.find({actor: "Brad Pitt"}).pretty() //actor: Brad Pitt

db.movies.find( { $and:[ {year: {$gt:1989}}, {year: {$lt: 2000}} ] },{\_id:0, title:1}).pretty() //movies in 90s

db.movies.find( {year: {$gt:2010}} ,{\_id:0, title:1}).pretty() // movies after 2010

db.movies.find({ $or: [{ year: { $lt: 2000 } }, { year: { $gt: 2010 } }] },{\_id:0, title:1}).pretty() //movies before 2000 or after 2010

**Update Documents:**

db.movies.update({title: "The Hobbit: An Unexpected Journey"},{$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"}})

db.movies.update({title: "The Hobbit: The Desolation of Smaug"},{$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"}})

db.movies.update({title: "Pulp Fiction"},{$push: {actor: "Samuel L. Jackson"}})

Search

db.movies.find({"synopsis": /Bilbo/},{\_id:0, title:1}).pretty()

db.movies.find({"synopsis": /Gandalf/},{\_id:0, title:1}).pretty()

db.movies.find({"synopsis": /Bilbo -Gandalf/},{\_id:0, title:1}).pretty()

Text search

db.movies.createIndex({synopsis:"text"})

db.movies.find({$text: {$search: "Bilbo"}}, {\_id:0, title:1})

db.movies.find({$text: {$search: "Gandalf"}}, {\_id:0, title:1})

db.movies.find({$text: {$search: "Bilbo -Gandalf"}}, {\_id:0, title:1})

db.movies.find({$text: {$search: "dwarves hobbit"}}, {\_id:0, title:1})

db.movies.find({$text: {$search: " (gold.\*dragon|dragon.\*gold)"}}, {\_id:0, title:1})

Deleting Documents:

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

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

Relationships:

db.users.insertMany([{username:"GoodGuyGrey", first\_name:"Good Guy", last\_name:"Grey"},{username: "ScumbagSteve", fullname:{first: "Scumbag", last:"steve"}}])

db.posts.insertMany([{username: "GoodGuyGreg", title: "Passes out at party", body: "Wakes up early and cleans house"},{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"},{username:"ScumbagSteve",title:"Borrows something", body:"Sells it"},{username:"ScumbagSteve",title:"Borrows everthing",body:"The end"},{username: "ScumbagSteve",title:"Forks your repo on github",body:"Sets to private" }])

db.comments.insertMany([{username:"GoodGuyGreg",comment:"Hope you got a good deal!",post:ObjectId("60e05d6cd82b74bc01165229")},{username: "GoodGuyGrey", comment :"What's mini is yours", post:ObjectId("60e05d6cd82b74bc0116522a")},{username:"GoodGuyGrey",comment:"Don't violate the licensing agreement!", post:ObjectId("60e05d6cd82b74bc0116522b")},{username:"ScumbagSteve",comment: "it still isn't clean", post:ObjectId("60e05d6cd82b74bc01165226")},{username:"ScumbagSteve",comment:"Denied your PR cause I found a hack", post:ObjectId("60e05d6cd82b74bc01165228")}])

db.users.find({},{\_id:0,username:1}).pretty()

db.posts.find({},{\_id:0}).pretty()

db.posts.find({ username: 'GoodGuyGreg' },{\_id:0}).pretty()

db.posts.find({ username: 'ScumbagSteve' },{\_id:0}).pretty()

db.comments.find().pretty()

db.comments.find({ username: 'GoodGuyGreg' },{\_id:0}).pretty()

db.post.find().comments

db.comments.find({post: db.posts.findOne({title: "Reports a bug in your code"}).\_id})

Assignment 2

mongo "mongodb+srv://cluster0.srt2w.mongodb.net/myFirstDatabase" --username user01

Use mongodb compass

Atlanta Population

1.db.zipcodes.find({city:"ATLANTA", state: "GA"})

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

3.db.zipcodes.aggregate([{$match: {city:"ATLANTA",state:"GA"}},{$group:{\_id:{zipcodes:"$\_id"}}},{$count:"Total number of zip codes"}]).pretty()

4.db.zipcodes.aggregate([{$match: {city:"ATLANTA",state:"GA"}},{$group:{\_id:"Null","Totalpopulation in atlanta":{$sum:"$pop"}}}])

Population By State

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

Population by city

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

Bonus:

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

Assignment 3

mongo "mongodb+srv://cluster0.srt2w.mongodb.net/myFirstDatabase" --username user01

|  |  |
| --- | --- |
|  | db.addresses.find() |
|  | db.addresses.find({},{restaurant\_id:1,name:1,borough:1,cuisine:1}).pretty() |
|  | db.addresses.find({},{\_id:0,restaurant\_id:1,name:1,borough:1,cuisine:1}).pretty() |
|  | db.addresses.find({},{\_id:0,restaurant\_id:1,name:1,borough:1,address:{zipcode:1}}).pretty() |
|  | db.addresses.aggregate([{$match:{borough:"Bronx"}},{$limit:5}]).pretty() |
|  | db.addresses.aggregate([{$match:{borough:"Bronx"}}]).pretty() |
|  | db.addresses.aggregate([{$match:{borough:"Bronx"}},{$skip:5},{$limit:5}]).pretty() |
|  | db.addresses.find({grades : { $elemMatch:{"score":{$gt : 80}}}}) |
|  | 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(  {$and:  [  {"cuisine" : {$ne : "American "}},  {"grades.score" :{$gt: 70}},  {"address.coord" : {$lt : -65.754168}}]  }  ) |
|  | db.addresses.find( {$and:[  {"cuisine" : {$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" ,  $or : [  { "cuisine" : "American " },  { "cuisine" : "Chinese" }  ]  }  ) |
|  | db.addresses.find({},  {\_id:0,restaurant\_id:1,name:1,borough:1,cuisine:1},  {$or : [  {borough:"Satenislan"},  {borough:"Queens"},  {Borough:"Bronxor Brooklyn"}]  }) |
|  | db.addresses.find(  {"borough" :{$nin :["Staten Island","Queens","Bronx","Brooklyn"]}},  {  "restaurant\_id" : 1,  "name":1,"borough":1,  "cuisine" :1  }  ) |
|  | db.addresses.find(  {"grades.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(  { "grades.1.date": ISODate("2014-08-11T00:00:00Z"),  "grades.1.grade":"A" ,  "grades.1.score" : 9  },  {"restaurant\_id" : 1,"name":1,"grades":1}  ) |
|  | db.addresses.find(  {  "address.coord.1": {$gt : 42, $lte : 52}  },  {"restaurant\_id" : 1,"name":1,"address":1,"coord":1}  ) |
|  | db.addresses.find().sort({"name":1}) |
|  | db.addresses.find().sort({"name":-1}) |
|  | db.addresses.find().sort({"name":1},{"borough":-1}) |
|  | db.addresses.find(  {"address.street" :  { $exists : true }  }  ) |
|  | db.addresses.find(  {"address.coord" :  {$type : 1}  }  ) |
|  | db.addresses.find(  {"grades.score" :  {$mod : [7,0]}  },  {"restaurant\_id" : 1,"name":1,"grades":1}  ) |
|  | db.addresses.find(  { name :  { $regex : "mon.\*", $options: "i" }  },  {  "name":1,  "borough":1,  "address.coord":1,  "cuisine" :1  }  ) |
|  | db.addresses.find(  { name :  { $regex : /^Mad/i, }  },  {  "name":1,  "borough":1,  "address.coord":1,  "cuisine" :1  }  ) |