**MongoDB Lab Assignments -Day 1**

**use mongo\_practice**

**switched to db mongo\_practice**

**> db.createCollection("movies")**

**{ "ok" : 1 }**

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

**db.movies.insert({title:"Pulp Fiction",writer:"Quentin Tarantino",year:"1994",actors:["John Travolta","Uma Thurman"]})**

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

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

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

**db.movies.insert({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 failing into the hands of a rising darkness"})**

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

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

1. get all documents

**db.movies.find().pretty()**

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:{$gt:"1900",$lt:"2000"}})**

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

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

**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("61765dacf9d7dfa73783b893")},{$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 withi 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("61765eccf9d7dfa73783b895")},{$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({title:"Pulp Fiction"},{$push:{actors:"Samuel L.Jackson"}})**

Text Search

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

**db.movies.find({synopsis:/Bilbo/g}).pretty()**

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

**db.movies.find({synopsis:/Gandalf/g}).pretty()**

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

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

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

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

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

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

**Delete Documents**

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

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

1. delete the movie "Avatar”

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

**Relationships**

Insert the following documents into a users collection

username : GoodGuyGreg

first\_name : "Good Guy"

last\_name : "Greg"

username : ScumbagSteve

full\_name : first : "Scumbag" last : "Steve"

**db.users.insertMany([{username:"GoodGuyGrey",first\_name:"Good Guy",last\_name:"Grey"},{username:"ScumbagSteve",full\_name:{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

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 everything

body : The end

username : ScumbagSteve

title : Forks your repo on github

body : Sets to private

**db.posts.insertMany([{username:"GoodGuyGrey",title:"Passes out at party",body:"Wakes up early and cleans house"}, {username:"GudGuyGrey",title:"Steals your identity",body:"Raises your credit score"}, {username:"GudGuyGrey",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 everything",body:"The end"},**

**... {username:"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.insertOne({username:"GoodGuyGrey",comment:"Hope you got a good deal!",post:ObjectId("61767bc02d248ddc996b1b4e")})**

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.insertOne({username:"GoodGuyGrey",comment:"Whats's mine is yours!",post:ObjectId("61767bc02d248ddc996b1b4f")})**

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.insertOne({username:"GoodGuyGrey",comment:"Don't violate the licensing agreement!",post:ObjectId("61767bc02d248ddc996b1b50")})**

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.insertOne({username:"ScumbagSteve",comment:"It still isn't clean",post:ObjectId("61767bc02d248ddc996b1b4b")})**

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.insertOne({username:"ScumbagSteve",comment:"Denied your PR cause I found a hack",post:ObjectId("61767bc02d248ddc996b1b4d")})**

**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:"GoodGuyGrey"})**

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

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

1. find all comments

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

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

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

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

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

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

**MongoDB -Aggregation Exercises**

**Atlanta Population**

1. use db.zipcodes.find() to filter results to only the results where city is ATLANTA and state is GA.

**db.zipcodes.find({$and:[{city:"ATLANTA"},{state:"GA"}]} )**

1. use db.zipcodes.aggregate with $match to do the same as above.

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

1. use $group to count the number of zip codes in Atlanta.

**db.zipcodes.aggregate([{$group:{\_id:{city:"ALTANA"} } },{$count:"total"}])**

1. use $group to find the total population in Atlanta.

**Populations By State**

1. use aggregate to calculate the total population for each state

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

1. sort the results by population, highest first

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

1. limit the results to just the first 3 results. What are the top 3 states in population?

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

**Populations by City**

1. use aggregate to calculate the total population for each city (you have to use city/state combination). You can use a combination for the \_id of the $group: { city: '$city', state: '$state' }

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

1. sort the results by population, highest first

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

1. limit the results to just the first 3 results. What are the top 3 cities in population?

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

4. What are the top 3 cities in population in Texas?

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

Bonus

1. Write a query to get the average city population for each state.

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

1. What are the top 3 states in terms of average city population?

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

**MongoDB – Complex Queries**

**Exercise Questions**

1. Write a MongoDB query to display all the documents in the collection restaurants.

**db.addresses.find().pretty()**

1. Write a MongoDB query to display the fields restaurant\_id, name, borough and cuisine for all the documents in the collection restaurant.

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

1. Write a MongoDB query to display the fields restaurant\_id, name, borough and cuisine, but exclude the field \_id for all the documents in the collection restaurant.

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

1. Write a MongoDB query to display the fields restaurant\_id, name, borough and zip code, but exclude the field \_id for all the documents in the collection restaurant.

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

1. Write a MongoDB query to display the first 5 restaurant which is in the borough Bronx.

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

1. Write a MongoDB query to display all the restaurant which is in the borough Bronx.

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

1. Write a MongoDB query to display the next 5 restaurants after skipping first 5 which are in the borough Bronx.

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

1. Write a MongoDB query to find the restaurants who achieved a score more than 90.

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

1. Write a MongoDB query to find the restaurants that achieved a score, more than 80 but less than 100

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

1. Write a MongoDB query to find the restaurants which loc ate in latitude value less than -95.754168.

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

1. Write a MongoDB query to find the restaurants that do not prepare any cuisine of 'American' and their grade score more than 70 and latitude less than -65.754168.

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

1. Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American' and achieved a score more than 70 and located in the longitude less than -65.754168.

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

1. Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American ' and achieved a grade point 'A' not belongs to the borough Brooklyn. The document must be displayed according to the cuisine in descending order.

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

1. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Wil' as first three letters for its name.

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

1. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'ces' as last three letters for its name.

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

1. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Reg' as three letters somewhere in its name.

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

1. Write a MongoDB query to find the restaurants which belong to the borough Bronx and prepared either American or Chinese dish.

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