**Assignment 1**

Sample data:

#1

{

"Title" : "Fight Club",

"Writer" : "Chuck Palahniuko",

"Year" : 1999,

"Actors" : [

"Brad Pitt",

"Edward Norton"

]

}

#2

{

"Title" : "Pulp Fiction",

"Writer" : "Quentin Tarantino",

"Year" : 1994,

"Actors" : [

"John Travolta",

"Uma Thurman"

]

}

#3

db.movies.insertOne({

"Title" : "Inglorious Basterds",

"Writer" : "Quentin Tarantino",

"Year" : 2009,

"Actors" : [

"Brad Pitt",

"Diane Kruger",

"Eli Roth"

]

})

#4

db.movies.insertOne({

"Title" : "The Hobbit: An Unexpected Journey",

"Writer" : "J.R.R. Tolkein",

"Year" : 2012,

"Franchise" : "The Hobbit"

})

#5

db.movies.insertOne({

"Title" : "The Hobbit: The Desolation of Smaug",

"Writer" : "J.R.R. Tolkein",

"Year" : 2013,

"Franchise" : "The Hobbit"

})

#6

db.movies.insertOne({

"Title" : "The Hobbit: The Battle of the Five Armies",

"Writer" : "J.R.R. Tolkein",

"Year" : 2012,

"Franchise" : "The Hobbit",

"Synopsis" : "Bildo and Company are forced to engage in a war against an array ofcombatants and keep the Lonely Mountain from falling into the hands of a rising darkness."

})

#7

db.movies.insertOne({

"Title" : "Pee Wee Herman's Big Adventure"

})

#8

db.movies.insertOne({

"Title" : "Avatar"

})

query the movies collection to

1. get all documents

db.movies.find().pretty()

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

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

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

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

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

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

5. get all movies released in the 90s

db.movies.find({$and:[{Year:{$lt:2000}},{Year:{$gt:1900}}]}).pretty()

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

db.movies.find({$and:[{Year:{$lt:2000}},{Year:{$gt:2010}}]}).pretty()

Update document

#1

db.movies.updateOne({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."}})

#2

db.movies.updateOne({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."}})

#3

db.movies.updateOne({Title:"Pulp Fiction"},{$push:{Actors:"Samuel L.Jackson"}})

\*\*\*Text Search\*\*\*

Before we start with the text search, an index with text should be created

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

--check the indexes

db.movies.getIndexes()

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

db.movies.find({$text:{$search:"Bilbo"}}).pretty

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

db.movies.find({$text:{$search:"Gandalf"}})

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

db.movies.find({$text:{$search:"Bildo -Gandalf"}})

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

db.movies.find({$text:{$search:"dwarves hobbit"}})

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

db.movies.find({$text:{$search:"gold dragon"}})

Delete Documents

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

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

2. delete the movie "Avatar"

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

Relationships

created users collection

#1

db.users.insertOne({Username:"GoodGuyGreg", first\_name:"GoodGuy", last\_name:"Greg"})

#2

db.users.insertOne({Username:"ScumbagSteve", full\_name:{first:"Scumbag", last:"Steve"}})

created posts collections

#1

db.posts.insertOne({Username:"GoodGuyGreg",

title:"Passes out at party", body:"Wakes up early and cleans house"})

#2

db.posts.insertOne({Username:"GoodGuyGreg",

title:"Steals your identity", body:"Raises your credit score"})

#3

db.posts.insertOne({Username:"GoodGuyGreg",

title:"Reports a bug in your code", body:"Sends you a pull request"})

#4

db.posts.insertOne({Username:"ScumbagSteve",

title:"Borrows something", body:"Sells it"})

$5

db.posts.insertOne({Username:"ScumbagSteve",

title:"Borrows everything", body:"The end"})

#6

db.posts.insertOne({Username:"ScumbagSteve",

title:"Forks your repo on github", body:"Sets to private"})

created comments collection

#1

db.comments.insertOne({Username:"GoodGuyGreg",

comment:"Hope you got a good deal!", post:[ObjectId("60034c792223f4164425ee22")]})

#2

db.comments.insertOne({Username:"GoodGuyGreg",

comment:"What's mine is yours", post:[ObjectId("60034c792223f4164425ee22")]})

#3

db.comments.insertOne({Username:"GoodGuyGreg",

comment:"Don't violate the licensing agreement", post:[ObjectId("60034c892223f4164425ee23")]})

#4

db.comments.insertOne({Username:"ScumbagSteve",

comment:"It still isn't clean", post:[ObjectId("60034b072223f4164425ee1e")]})

#5

db.comments.insertOne({Username:"ScumbagSteve",

comment:"Denied your PR cause i found a hack", post:[ObjectId("60034c5b2223f4164425ee20")]})

Querying related collections

1. find all users

db.users.find().pretty()

2. find all posts

db.posts.find().pretty()

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

db.posts.find({Username:"GoodGuyGreg"}).pretty()

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

db.posts.find({Username:"ScumbagSteve"}).pretty()

5. find all comments

db.comments.find().pretty()

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

db.comments.find({Username:"GoodGuyGreg"}).pretty()

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

db.comments.find({Username:"ScumbagSteve"}).pretty()

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

db.comments.find({post:ObjectId("60034c5b2223f4164425ee20")).pretty()

**Assignment 2**

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

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

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

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

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

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

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

Out put:{ "\_id" : "ATLANTA", "count" : 41 }

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

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

Out put:{ "\_id" : "ATLANTA", "Totalpopulation" : 41 }

population by state

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

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

2. sort the results by population, highest first

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

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

db.zipcodes.aggregate([{$group:{\_id:"$state", populationbystate:{$sum:"$pop"}}},

{$sort:{populationbystate:-1}},{$limit:3}])

Output: { "\_id" : "CA", "populationbystate" : 29754890 }

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

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

population 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", Totalpopulation:{$sum:"$pop"}}}])

2. sort the results by population, highest first

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

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

db.zipcodes.aggregate([{$group:{\_id:"$city", populationofstate:{$sum:"$pop"}}},

{$sort:{populationofcity:-1}},{$limit:3}])

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

db.zipcodes.aggregate([{$match:{state:"TX"}},{$group:{\_id:"$city",

populationofcity:{$sum:"$pop"}}},{$sort:{populationofcity:-1}},{$limit:3}])

Output: { "\_id" : "HOUSTON", "populationofcity" : 2095918 }

{ "\_id" : "DALLAS", "populationofcity" : 940191 }

{ "\_id" : "SAN ANTONIO", "populationofcity" : 811792 }

Bonus

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

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

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

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

Output: { "\_id" : "DC", "averagepopulation" : 25287.5 }

{ "\_id" : "CA", "averagepopulation" : 19627.236147757256 }

{ "\_id" : "FL", "averagepopulation" : 15779.407960199005 }

**Assignment 3**

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

db.addresses.find().pretty()

2. 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,"borough":1,"cuisine":1}).pretty()

3. 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({},{"\_id":0,"restaurant\_id":1,"name":1,"borough":1,"cuisine":1}).pretty()

4. 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({},{"\_id":0,"restaurant\_id":1,"name":1,"borough":1,"address.zipcode":1}).pretty()

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

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

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

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

7. 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).pretty()

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

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

9. Write a MongoDB query to find the restaurants that achieved a score,

more than 80 but less than 100.

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

10. Write a MongoDB query to find the restaurants which locate in latitude value

less than -95.754168.

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

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

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

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

14. 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:/^Wil/},{"restaurant\_id":1,"name":1,"borough":1,"cuisine":1})

15. 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:/ces$/},{"restaurant\_id":1,"name":1,"borough":1,"cuisine":1})

16. 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})

17. 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"}]})

18. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for

those restaurants which belong to the borough Staten Island or Queens or Bronxor Brooklyn.

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

19. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for

those restaurants which are not belonging to the borough Staten Island or Queens or Bronxor Brooklyn.

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

20. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for

those restaurants which achieved a score which is not more than 10.

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

21. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for

those restaurants which prepared dish except 'American' and 'Chinees' or restaurant's

name begins with letter 'Wil'.

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

22. Write a MongoDB query to find the restaurant Id, name, and grades for those

restaurants which achieved a grade of "A" and scored 11 on an

ISODate "2014-08-11T00:00:00Z" among many of survey dates..

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

23. Write a MongoDB query to find the restaurant Id, name and grades for

those restaurants where the 2nd element of grades array contains a grade of "A"

and score 9 on an ISODate "2014-08-11T00:00:00Z"

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})

24. Write a MongoDB query to find the restaurant Id, name, address and geographical

location for those restaurants where 2nd element of coord array contains a value

which is more than 42 and upto 52..

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

25. Write a MongoDB query to arrange the name of the restaurants in ascending order

along with all the columns.

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

26. Write a MongoDB query to arrange the name of the restaurants in descending along

with all the columns.

b.addresses.find().sort({"name":-1})

27. Write a MongoDB query to arranged the name of the cuisine in ascending order and

for that same cuisine borough should be in descending order.

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

28. Write a MongoDB query to know whether all the addresses contains the street or not.

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

29. Write a MongoDB query which will select all documents in the restaurants collection

where the coord field value is Double.

db.addresses.find({"address.ccord":{$type:1}})

30. Write a MongoDB query which will select the restaurant Id, name and grades for those

restaurants which returns 0 as a remainder after dividing the score by 7.

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

31. Write a MongoDB query to find the restaurant name, borough, longitude and attitude

and cuisine for those restaurants which contains 'mon' as three letters somewhere in

its name.

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

32. Write a MongoDB query to find the restaurant name, borough, longitude and latitude

and cuisine for those restaurants which contain 'Mad' as first three letters of its name.

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