# MongoDB Lab Assignments -Day 1

**MongoDB Exercise in mongo shell**

Connect to a running mongo instance, use a database named **mongo\_practice**. Document all your queries in a javascript file to use as a reference.

# Insert Documents

Insert the following documents into a **movies** collection.

title : Fight Club

writer : Chuck Palahniuko year : 1999

actors : [ Brad Pitt

Edward Norton

]

title : Pulp Fiction

writer : Quentin Tarantino year : 1994

actors : [

John Travolta Uma Thurman

]

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

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

Reference https[://ww](http://www.tutorialspoint.com/mongodb/mongodb_insert_document.htm)w.t[utorialspoint.com/mongodb/mongodb\_insert\_document.htm](http://www.tutorialspoint.com/mongodb/mongodb_insert_document.htm)

**Command Used:**

1. use mongo\_practice
2. db.movies.insertMany([{

title: 'Fight Club',

writer: 'Chuck Palahniuko',

year: '1999',

actors: 'Brad Pitt, Edward Norton'

}, {

title: 'Pulp Fiction',

writer: 'Quentin Tarantino',

year: '1994',

actors: 'John Travolta, Uma Thurman'

}])

db.movies.insertMany([{

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'

}, {

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'

}])

# Query / Find Documents

query the **movies** collection to

1. get all documents

Atlas atlas-10chxn-shard-0 [primary] mongo\_practice> db.movies.find().pretty()

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

Atlas atlas-10chxn-shard-0 [primary] mongo\_practice> db.movies.find({"writer": 'Quentin Tarantino'}).pretty()

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

Atlas atlas-10chxn-shard-0 [primary] mongo\_practice> db.movies.find({actors:{$eq:"Brad Pitt"}})

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

Atlas atlas-10chxn-shard-0 [primary] mongo\_practice> db.movies.find({franchise:{$eq:"The Hobbit"}})

1. get all movies released in the 90s

db.movies.find({$and:[{"year":{$gte:1900}},{"year":{$lt:2000}}]})

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

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

Reference: https[://ww](http://www.tutorialspoint.com/mongodb/mongodb_query_document.htm)w.t[utorialspoint.com/mongodb/mongodb\_query\_document.htm](http://www.tutorialspoint.com/mongodb/mongodb_query_document.htm)

# 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."

Atlas atlas-10chxn-shard-0 [primary] mongo\_practice> 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'}})

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."

Atlas atlas-10chxn-shard-0 [primary] mongo\_practice> 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.'}})

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

Atlas atlas-10chxn-shard-0 [primary] mongo\_practice> db.movies.update({title: 'Pulp Fiction'}, {$set: {actors:['John Travolta', 'Uma Thurman', 'Samuel L. Jackson']}})

Reference: https[://ww](http://www.tutorialspoint.com/mongodb/mongodb_update_document.htm)w.t[utorialspoint.com/mongodb/mongodb\_update\_document.htm](http://www.tutorialspoint.com/mongodb/mongodb_update_document.htm)

# Text Search

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

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

db.movies.aggregate({$match:{$text:{$search:"Bilbo"}}},{$project:{title:1}}) (or)

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

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

db.movies.aggregate({$match:{$text:{$search:"\"Gandalf\""}}},{$project:{title:1}})

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

db.movies.aggregate({$match:{$text:{$search:"Bilbo -Gandalf"}}},{$project:{title:1}})

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

db.movies.aggregate({$match:{$text:{$search:"dwarves hobbit"}}},{$project:{title:1}})

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

db.movies.aggregate({$match:{$text:{$search:"\"gold\" \”dragon\”"}}},{$project:{title:1}})

Reference: https[://w](http://www.tutorialspoint.com/mongodb/mongodb_text_search.htm)ww[.t](http://www.tutorialspoint.com/mongodb/mongodb_text_search.htm)u[torialspoint.com/mongodb/mongodb\_text\_search.htm](http://www.tutorialspoint.com/mongodb/mongodb_text_search.htm)

# 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"})

Reference: https[://ww](http://www.tutorialspoint.com/mongodb/mongodb_delete_document.htm)w.t[utorialspoint.com/mongodb/mongodb\_delete\_document.htm](http://www.tutorialspoint.com/mongodb/mongodb_delete_document.htm)

# 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"

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

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"

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"

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

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"

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"

# Querying related collections

1. find all users

db.users.find()

1. find all posts

db.posts.find()

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()

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.comments.find({5:"Reports a bug in your code" })

References: <https://docs.mongodb.com/manual/reference/method/db.collection.find/>

**MongoDB -Aggregation Exercises**

Import the zips.json file into your MongoDB. Database name is "population" and collection name is "zipcodes".

mongoimport --db population --collection zipcodes --file zips.json

**Atlanta Population**

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

{city:'ATLANTA',state:'GA'}

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

In aggregation tab select $match

$match

{

city:'ATLANTA',

state:'GA'

}

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

$group

{

\_id: "Atlanta",

count: {

$sum: 1

}

}

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

$group

{

\_id: "Atlanta",

total: {

$sum: "$pop"

}

}

**Populations By State**

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

$group

{

\_id: "$state",

total: {

$sum: "$pop"

}

}

1. sort the results by population, highest first

$sort

{

pop: -1

}

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

$sort

{

pop: -1

}

Save and add a next stage for limits

Limit is 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' }

$group

{

\_id:{

city:'$city',

state:'$state'

},

total: {

$sum: "$pop"

}

}

1. sort the results by population, highest first

$sort

{

total: -1

}

And add a stage for limit

3

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

{

total: -1

}

Save and add a next stage for limits

Limit is 3

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

$group

{

\_id: {city:'$city',state:'Texas'},

total: {

$sum: '$pop'

}

}

$sort

{

total: -1

}

$limit

3

**Bonus**

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

$group

{

\_id: {city:'$city',

state:'$state'},

avgpop: {

$avg: "$pop"

}

}

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

$group

{

\_id: {city:'$city',

state:'$state'},

avgpop: {

$avg: "$pop"

}

}

$sort

{

avgpop:-1

}

$limit

3

**MongoDB – Complex Queries**

**Mongo DB Exercises - With the Restaurants Data Set**

1. Download the restaurants.zip file

2. Unzip the file, you will see restaurants.json file

3. Run the mongod server

4. Run the following command to import the json file provided. It will load the json file into the mongodb with database name - restaurants, collections name - addresses

**mongoimport --db restaurants --collection addresses --file restaurants.json**

5. Run mongo shell command

6. show databases

7. use restaurants

8. db.addresses.find() should print entire json data

9. Then start working on the following exercises and submit your queries as the answers to the questions

**Query Reference Links and Cheat sheets**

1. https://docs.mongodb.com/manual/crud/

**Exercise Questions**

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

db.restaurants.find()

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

$group

{

\_id: {restaurant\_id:'$restaurant\_id',name:'$name',borough:'$borough',cuisine:'$cuisine'

}

}

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.restaurants.find({},{"restaurant\_id" : 1,"name":1,"borough":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.restaurants.find({},{"restaurant\_id" : 1,"name":1,"borough":1,"address.zipcode" :1,"\_id":0});

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

$match

{

borough:"Bronx"

}

$limit

5

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

{

borough:"Bronx"

}

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

$match

{

borough:"Bronx"

}

$skip

5

$limit

5

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

[

{

'$match': {

'grades.score': {

'$gt': 90

}

}

}

]

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

[{

$match: {

'grades.score': {

$gt: 80,

$lt: 100

}

}

}]

10. Write a MongoDB query to find the restaurants which locate in latitude value less than -95.754168.

[{

$match: {

'address.coord.0': {

$lt: -95.754168

}

}

}]

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 th[{

$match: {

cuisine: {

$not: RegExp('American')

},

'grades.score': {

$gt: 70

},

'address.coord': {

$lt: -65.754168

}

}

}]

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.

[{

$match: {

cuisine: {

$not: RegExp('American')

},

'grades.score': {

$gt: 70

},

'address.coord': {

$lt: -65.754168

}

}

}]

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.

[{

$match: {

cusine: {

$not: RegExp('^American.\*')

},

'grades.grade': 'A',

borough: {

$not: RegExp('^Brooklyn.\*')

}

}

}, {

$sort: {

cusine: -1

}

}]

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.restaurants.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.restaurants.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.restaurants.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.

{

"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.restaurants.find(

{"borough" :{$in :["Staten Island","Queens","Bronx","Brooklyn"]}},

{

"restaurant\_id" : 1,

"name":1,"borough":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.restaurants.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.restaurants.find(

{"grades.score" :

{ $not:

{$gt : 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.restaurants.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.restaurants.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.restaurants.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.restaurants.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.restaurants.find().sort({"name":1});

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

db.restaurants.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.restaurants.find().sort(

{"cuisine":1,"borough" : -1,}

);

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

db.restaurants.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.restaurants.find(

{"address.coord" :

{$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.restaurants.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.restaurants.find(

{ name :

{ $regex : "mon.\*", $options: "i" }

},

{

"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.restaurants.find(

{ name :

{ $regex : /^Mad/i, }

},

{

"name":1,

"borough":1,

"address.coord":1,

"cuisine" :1 }

);