# 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.

C:\Users\BINDBR>mongosh "mongodb+srv://cluster1.2jhlm.mongodb.net/mongo\_demo" --username Bindushree

Enter password: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Current Mongosh Log ID: 61dd04f257e2447f77d47f1c

Connecting to: mongodb+srv://cluster1.2jhlm.mongodb.net/mongo\_demo

Using MongoDB: 4.4.11

Using Mongosh: 1.1.7

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

# Insert Documents

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

Atlas atlas-2v2ryx-shard-0 [primary] mongo\_demo> db.createCollection('movies')

{ ok: 1 }

Atlas atlas-2v2ryx-shard-0 [primary] mongo\_demo> show collections

movies

Atlas atlas-2v2ryx-shard-0 [primary] mongo\_demo>db.movies.insertMany([

{title:'Fight Club', writer: 'Chuck Palahniuko', year:1999, actors:['Brad Pitt','Edwrd 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 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'}])

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)

# Query / Find Documents

query the **movies** collection to

1. get all documents

1.db.movies.find()

Atlas atlas-2v2ryx-shard-0 [primary] mongo\_demo > db.movies.find()

[

{

\_id: ObjectId("61da79d7c4f4f44d993c57d6"),

title: 'Fight Club',

writer: 'Chuck Palahniuko',

year: 1999,

actors: [ 'Brad Pitt', 'Edwrd Norton' ]

},

{

\_id: ObjectId("61da79d7c4f4f44d993c57d7"),

title: 'Pulp Fiction',

writer: 'Quentin Tarantino',

year: 1994,

actors: [ 'John Travolta', 'Uma Thurman' ]

},

{

\_id: ObjectId("61da79d7c4f4f44d993c57d8"),

title: 'Inglorious Basterds',

writer: 'Quentin Tarantino',

year: 2009,

actors: [ 'Brad Pitt', 'Diane Kruger', 'Eli Roth' ]

},

{

\_id: ObjectId("61da79d7c4f4f44d993c57d9"),

title: 'The Hobbit: An Unexpected Journey',

writer: 'J.R.R Tolkein',

year: 2012,

franchise: 'The Hobbit'

},

{

\_id: ObjectId("61da79d7c4f4f44d993c57da"),

title: 'The Hobbit: The Desolation of Smaug',

writer: 'J.R.R Tolkein',

year: 2013,

franchise: 'The Hobbit'

},

{

\_id: ObjectId("61da79d7c4f4f44d993c57db"),

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 array of combatants and keep the Lonely Mountain from falling into the hands of a rising darkness'

},

{

\_id: ObjectId("61da79d7c4f4f44d993c57dc"),

title: "Pee Wee Herman's Big Adventure"

},

{ \_id: ObjectId("61da79d7c4f4f44d993c57dd"), title: 'Avatar' }

]

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

1.db.movies.find({writer: ”Quentin Tarantino”})

[

{

\_id: ObjectId("61da79d7c4f4f44d993c57d7"),

title: 'Pulp Fiction',

writer: 'Quentin Tarantino',

year: 1994,

actors: [ 'John Travolta', 'Uma Thurman' ]

},

{

\_id: ObjectId("61da79d7c4f4f44d993c57d8"),

title: 'Inglorious Basterds',

writer: 'Quentin Tarantino',

year: 2009,

actors: [ 'Brad Pitt', 'Diane Kruger', 'Eli Roth' ]

}

]

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

1.db.movies.find({actors: ”Brad Pitt”})

[

{

\_id: ObjectId("61da79d7c4f4f44d993c57d6"),

title: 'Fight Club',

writer: 'Chuck Palahniuko',

year: 1999,

actors: [ 'Brad Pitt', 'Edwrd Norton' ]

},

{

\_id: ObjectId("61da79d7c4f4f44d993c57d8"),

title: 'Inglorious Basterds',

writer: 'Quentin Tarantino',

year: 2009,

actors: [ 'Brad Pitt', 'Diane Kruger', 'Eli Roth' ]

}

]

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

1.db.movies.find({franchise: ”The Hobbit”})

[

{

\_id: ObjectId("61da79d7c4f4f44d993c57d9"),

title: 'The Hobbit: An Unexpected Journey',

writer: 'J.R.R Tolkein',

year: 2012,

franchise: 'The Hobbit'

},

{

\_id: ObjectId("61da79d7c4f4f44d993c57da"),

title: 'The Hobbit: The Desolation of Smaug',

writer: 'J.R.R Tolkein',

year: 2013,

franchise: 'The Hobbit'

},

{

\_id: ObjectId("61da79d7c4f4f44d993c57db"),

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 array of combatants and keep the Lonely Mountain from falling into the hands of a rising darkness'

}

]

1. get all movies released in the 90s

1.db.movies.find({year:{$lt:2000}})

[

{

\_id: ObjectId("61da79d7c4f4f44d993c57d6"),

title: 'Fight Club',

writer: 'Chuck Palahniuko',

year: 1999,

actors: [ 'Brad Pitt', 'Edwrd Norton' ]

},

{

\_id: ObjectId("61da79d7c4f4f44d993c57d7"),

title: 'Pulp Fiction',

writer: 'Quentin Tarantino',

year: 1994,

actors: [ 'John Travolta', 'Uma Thurman' ]

}

]

1. get all movies released before the year 2000 or after 2010
2. db.movies.find({$or:[{year:{$lt:2000}},{year:{$gt:2010}}]})
3. [
4. {
5. \_id: ObjectId("61da79d7c4f4f44d993c57d6"),
6. title: 'Fight Club',
7. writer: 'Chuck Palahniuko',
8. year: 1999,
9. actors: [ 'Brad Pitt', 'Edwrd Norton' ]
10. },
11. {
12. \_id: ObjectId("61da79d7c4f4f44d993c57d7"),
13. title: 'Pulp Fiction',
14. writer: 'Quentin Tarantino',
15. year: 1994,
16. actors: [ 'John Travolta', 'Uma Thurman' ]
17. },
18. {
19. \_id: ObjectId("61da79d7c4f4f44d993c57d9"),
20. title: 'The Hobbit: An Unexpected Journey',
21. writer: 'J.R.R Tolkein',
22. year: 2012,
23. franchise: 'The Hobbit'
24. },
25. {
26. \_id: ObjectId("61da79d7c4f4f44d993c57da"),
27. title: 'The Hobbit: The Desolation of Smaug',
28. writer: 'J.R.R Tolkein',
29. year: 2013,
30. franchise: 'The Hobbit'
31. },
32. {
33. \_id: ObjectId("61da79d7c4f4f44d993c57db"),
34. title: 'The Hobbit: The Battle of the Five Armies',
35. writer: 'J.R.R Tolkein',
36. year: 2012,
37. franchise: 'The Hobbit',
38. synopsis: 'Bilbo and Company are forced to engage in a war against array of combatants and keep the Lonely Mountain from falling into the hands of a rising darkness'
39. }
40. ]

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."

db.movies.updateOne({title:"the hobbit an unexpected journey"},{$set:{synopsis:"A reluctant hobbit, Bilbo Baggins, 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.updateOne({ title: "the hobbit the desolation of smaug" }, { $set: { synopsis: "The dwarves, along with BilboBaggins and Gandalf the Grey, and magical ring." } })

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

1. db.movies.update({title:'Pulp Fiction'}, {$push:{actors:"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"

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

[

{

\_id: ObjectId("61da79d7c4f4f44d993c57d9"),

title: 'The Hobbit: An Unexpected Journey',

writer: 'J.R.R Tolkein',

year: 2012,

franchise: 'The Hobbit',

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.'

},

{

\_id: ObjectId("61da79d7c4f4f44d993c57da"),

title: 'The Hobbit: The Desolation of Smaug',

writer: 'J.R.R Tolkein',

year: 2013,

franchise: 'The Hobbit',

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.'

},

{

\_id: ObjectId("61da79d7c4f4f44d993c57db"),

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 array of combatants and keep the Lonely Mountain from falling into the hands of a rising darkness'

}

]

1. find all movies that have a synopsis that contains the word "Gandalf"
2. db.movies.find({synopsis:{$regex:"Gandalf"}})

[

{

\_id: ObjectId("61da79d7c4f4f44d993c57da"),

title: 'The Hobbit: The Desolation of Smaug',

writer: 'J.R.R Tolkein',

year: 2013,

franchise: 'The Hobbit',

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

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

[

{

\_id: ObjectId("61da79d7c4f4f44d993c57d9"),

title: 'The Hobbit: An Unexpected Journey',

writer: 'J.R.R Tolkein',

year: 2012,

franchise: 'The Hobbit',

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.'

},

{

\_id: ObjectId("61da79d7c4f4f44d993c57db"),

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 array of combatants and keep the Lonely Mountain from falling into the hands of a rising darkness'

}

]

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

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

[

{

\_id: ObjectId("61da79d7c4f4f44d993c57d9"),

title: 'The Hobbit: An Unexpected Journey',

writer: 'J.R.R Tolkein',

year: 2012,

franchise: 'The Hobbit',

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.'

},

{

\_id: ObjectId("61da79d7c4f4f44d993c57db"),

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 array of combatants and keep the Lonely Mountain from falling into the hands of a rising darkness'

}

]

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

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

[

{

\_id: ObjectId("61da79d7c4f4f44d993c57d9"),

title: 'The Hobbit: An Unexpected Journey',

writer: 'J.R.R Tolkein',

year: 2012,

franchise: 'The Hobbit',

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.'

}

]

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"

1.db.movies.remove({\_id:ObjectId("61dac60b599a995de03c2dc7")})

1. delete the movie "Avatar"

1.db.movies.remove({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"

1.db.users.insertMany([{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

Db.posts.insertMany([{},{}])

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"

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

# Querying related collections

1. find all users

1.db.users.find() // db.users.find().pretty()

[

{

\_id: ObjectId("61da85b9f1e258629fdd6f5a"),

username: 'GoodGuyGreg',

first\_name: 'GoodGuy',

last\_name: 'Greg'

},

{

\_id: ObjectId("61da85b9f1e258629fdd6f5b"),

username: 'ScumbagSteve',

fullname: { first: 'Scumbag', last: 'Steve' }

}]

1. find all posts

1.db.posts.find()

[

{

\_id: ObjectId("61da896cf1e258629fdd6f5c"),

username: 'GoodGuyGreg',

title: 'Passes out at party',

body: 'Wakes up early and cleans house'

},

{

\_id: ObjectId("61da896cf1e258629fdd6f5d"),

username: 'GoodGuyGreg',

title: 'Steals your identity',

body: 'Raises your credit score'

},

{

\_id: ObjectId("61da896cf1e258629fdd6f5e"),

username: 'GoodGuyGreg',

title: 'Reports a bug in your code',

body: 'Sends you a pull request'

},

{

\_id: ObjectId("61da896cf1e258629fdd6f5f"),

username: 'ScumbagSteve',

title: 'Borrows something',

body: 'Sells it'

},

{

\_id: ObjectId("61da896cf1e258629fdd6f60"),

username: 'ScumbagSteve',

title: 'Borrows everything',

body: 'The end'

},

{

\_id: ObjectId("61da896cf1e258629fdd6f61"),

username: 'ScumbagSteve',

title: 'Forks your repo in github',

body: 'Sets to private'

}

]

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

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

[

{

\_id: ObjectId("61da896cf1e258629fdd6f5c"),

username: 'GoodGuyGreg',

title: 'Passes out at party',

body: 'Wakes up early and cleans house'

},

{

\_id: ObjectId("61da896cf1e258629fdd6f5d"),

username: 'GoodGuyGreg',

title: 'Steals your identity',

body: 'Raises your credit score'

},

{

\_id: ObjectId("61da896cf1e258629fdd6f5e"),

username: 'GoodGuyGreg',

title: 'Reports a bug in your code',

body: 'Sends you a pull request'

}

]

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

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

[

{

\_id: ObjectId("61da896cf1e258629fdd6f5f"),

username: 'ScumbagSteve',

title: 'Borrows something',

body: 'Sells it'

},

{

\_id: ObjectId("61da896cf1e258629fdd6f60"),

username: 'ScumbagSteve',

title: 'Borrows everything',

body: 'The end'

},

{

\_id: ObjectId("61da896cf1e258629fdd6f61"),

username: 'ScumbagSteve',

title: 'Forks your repo in github',

body: 'Sets to private'

}

]

1. find all comments

1.db.comments.find()

[

{

\_id: ObjectId("61da8b2ef1e258629fdd6f62"),

username: 'GoodGuyGreg',

comment: 'Hope you got a good deal!',

post: ObjectId("61da896cf1e258629fdd6f5f")

},

{

\_id: ObjectId("61da8cd5f1e258629fdd6f64"),

username: 'GoodGuyGreg',

comment: "What's mine is yours!",

post: ObjectId("61da896cf1e258629fdd6f60")

},

{

\_id: ObjectId("61da8d6ef1e258629fdd6f65"),

username: 'GoodGuyGreg',

comment: "Don't violate the licensing agreement!",

post: ObjectId("61da896cf1e258629fdd6f61")

},

{

\_id: ObjectId("61da8dfcf1e258629fdd6f66"),

username: 'ScumbagSteve',

comment: "It still isn't clean",

post: ObjectId("61da896cf1e258629fdd6f5c")

},

{

\_id: ObjectId("61da8e8bf1e258629fdd6f67"),

username: 'ScumbagSteve',

comment: 'Denied your PR cause I found a hack',

post: ObjectId("61da896cf1e258629fdd6f5e")

}

]

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

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

[

{

\_id: ObjectId("61da8b2ef1e258629fdd6f62"),

username: 'GoodGuyGreg',

comment: 'Hope you got a good deal!',

post: ObjectId("61da896cf1e258629fdd6f5f")

},

{

\_id: ObjectId("61da8cd5f1e258629fdd6f64"),

username: 'GoodGuyGreg',

comment: "What's mine is yours!",

post: ObjectId("61da896cf1e258629fdd6f60")

},

{

\_id: ObjectId("61da8d6ef1e258629fdd6f65"),

username: 'GoodGuyGreg',

comment: "Don't violate the licensing agreement!",

post: ObjectId("61da896cf1e258629fdd6f61")

}

]

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

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

[

{

\_id: ObjectId("61da8dfcf1e258629fdd6f66"),

username: 'ScumbagSteve',

comment: "It still isn't clean",

post: ObjectId("61da896cf1e258629fdd6f5c")

},

{

\_id: ObjectId("61da8e8bf1e258629fdd6f67"),

username: 'ScumbagSteve',

comment: 'Denied your PR cause I found a hack',

post: ObjectId("61da896cf1e258629fdd6f5e")

}

]

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

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

[

{

\_id: ObjectId("61da8e8bf1e258629fdd6f67"),

username: 'ScumbagSteve',

comment: 'Denied your PR cause I found a hack',

post: ObjectId("61da896cf1e258629fdd6f5e")

}

]

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

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@