# 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.





Ans : db.movies.insertMany([

{

title: 'Fight Club',

writer: 'Chuck Palahniuk',

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'

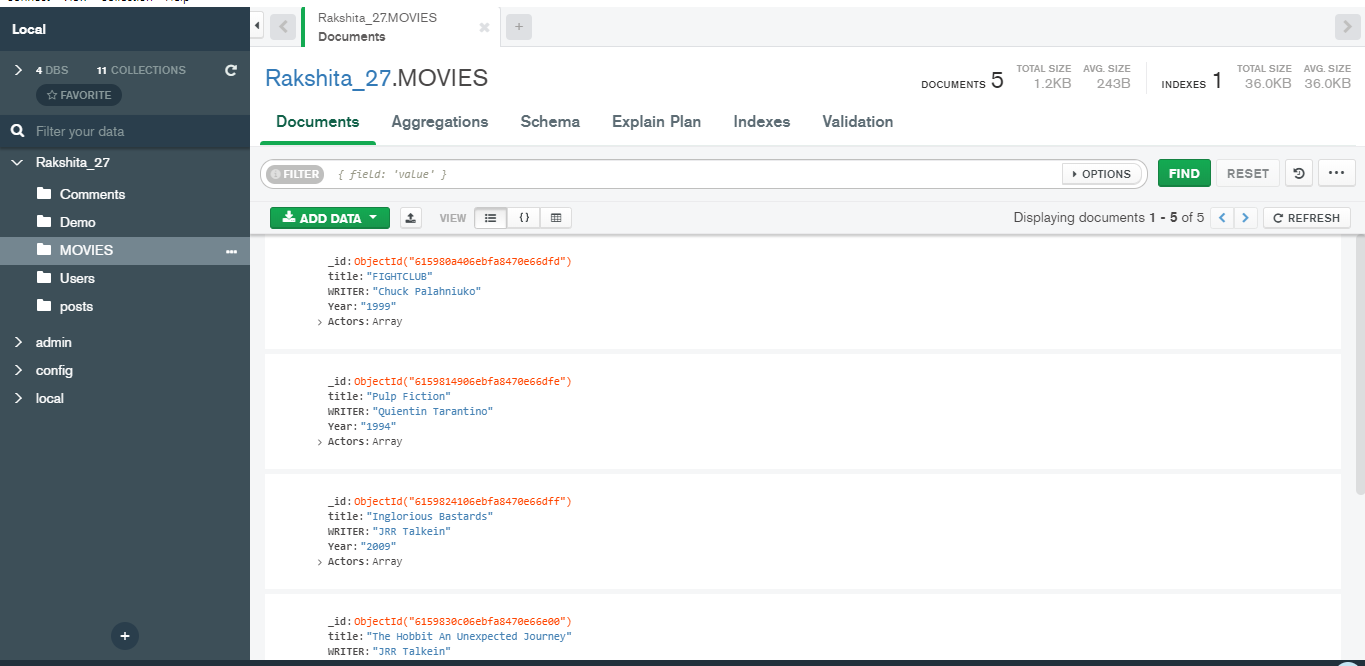
},

{

title: 'Avatar'

}

]);

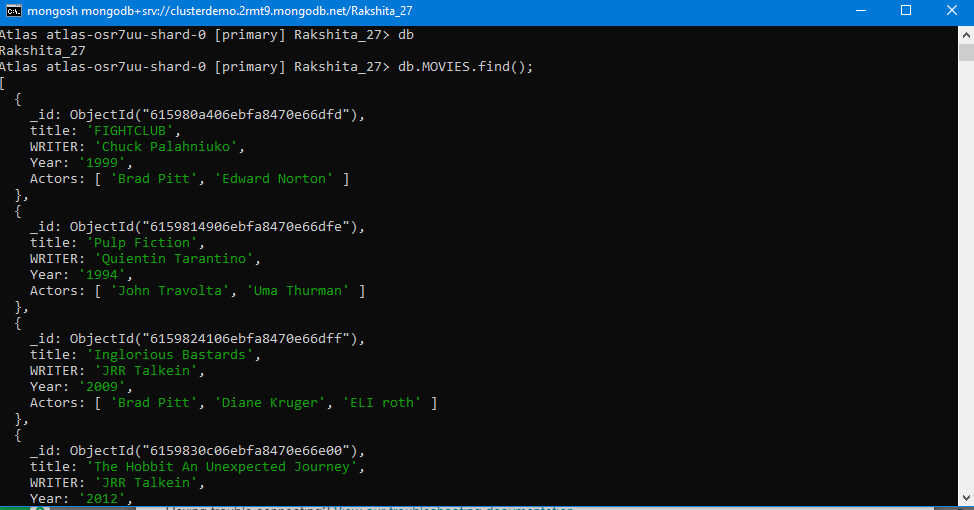


# Query / Find Documents

query the **movies** collection to

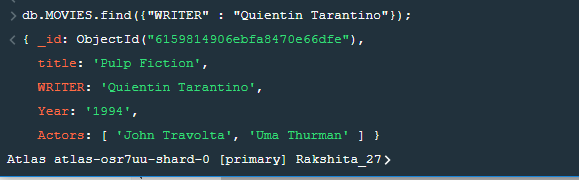
1. get all documents

Ans :db.movies.find().pretty();



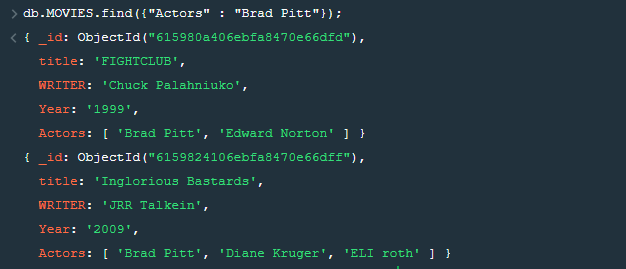
1. get all documents with writer set to "Quentin Tarantino"

Ans: db.movies.find({ writer: 'Quentin Tarantino' }).pretty();



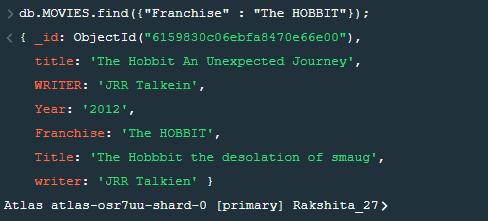
1. get all documents where actors include "Brad Pitt"

Ans : db.movies.find({ actors: 'Brad Pitt' }).pretty();



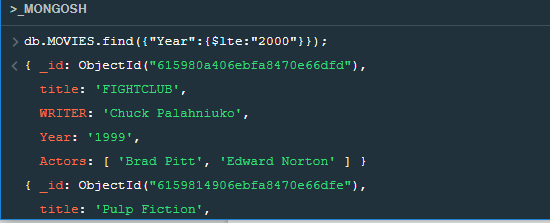
1. get all documents with franchise set to "The Hobbit"

Ans : db.movies.find({ franchise: 'The Hobbit' }).pretty();



1. get all movies released in the 90s

Ans : db.movies.find({ year: { $gte: 1990, $lte: 1999 } }).pretty();



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

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

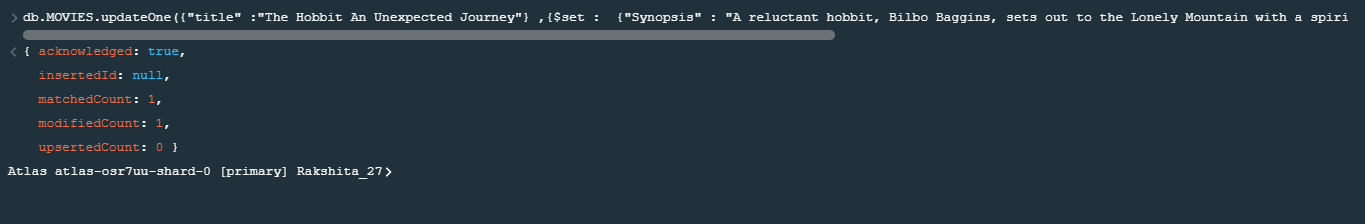


Reference: https[://ww](http://www.tutorialspoint.com/mongodb/mongodb_query_document.htm)w.t[utorialspoint.com/mongodb/mongodb\_query\_document.htm](http://www.tutorialspoint.com/mongodb/mongodb_query_document.htm)

# Update Documents

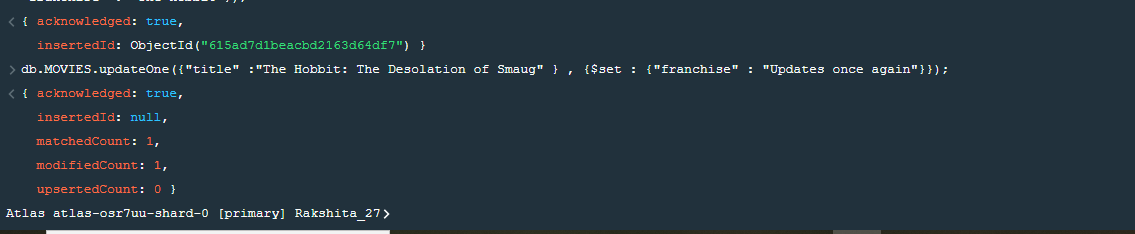
1. add a synopsis to "The Hobbit: An Unexpected Journey" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug."

db.movies.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." } });



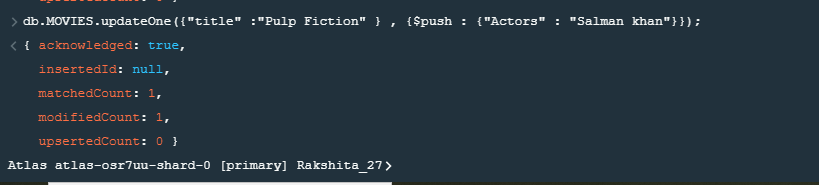
1. add a synopsis to "The Hobbit: The Desolation of Smaug" : "The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."

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"

db.movies.update({ title: 'Pulp Fiction' }, { $push: { actors: 'Samuel L. Jackson' } });

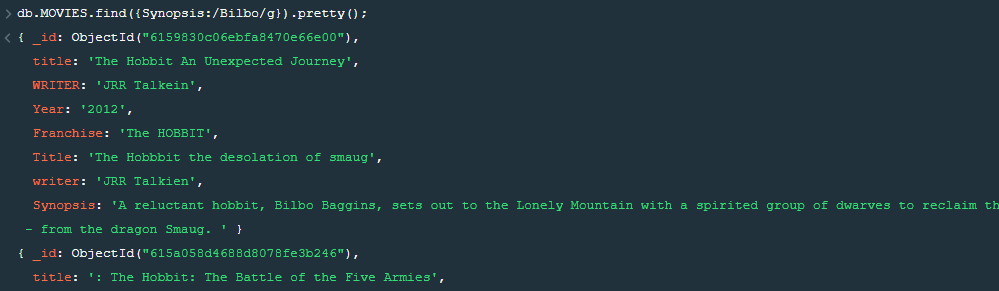


Reference: https[://ww](http://www.tutorialspoint.com/mongodb/mongodb_update_document.htm)w.t[utorialspoint.com/mongodb/mongodb\_update\_document.htm](http://www.tutorialspoint.com/mongodb/mongodb_update_document.htm)

# Text Search

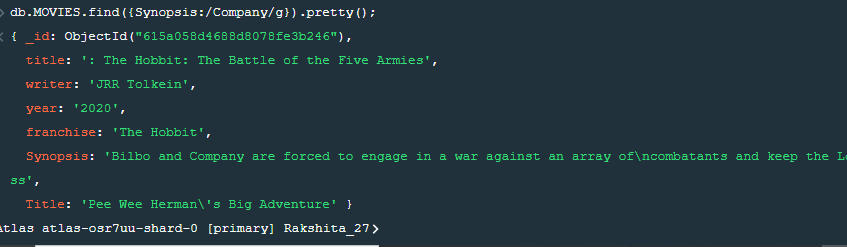
1. find all movies that have a synopsis that contains the word "Bilbo"

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



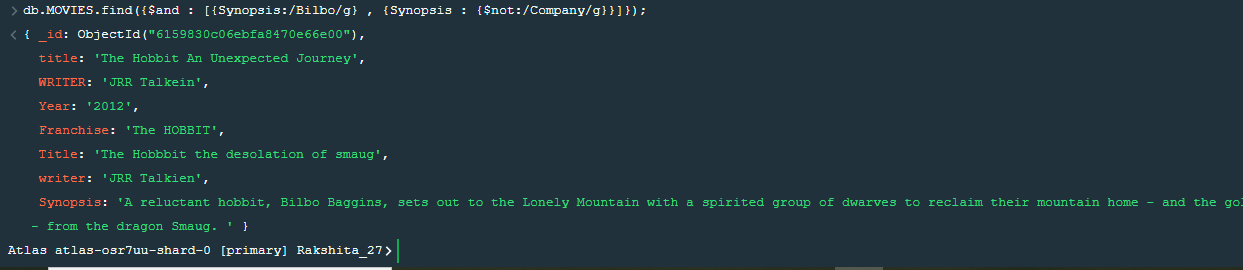
1. find all movies that have a synopsis that contains the word "Gandalf"

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: /Bilbo/g }, { synopsis: { $not: /Gandalf/g } }] }).pretty();



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

db.movies.find({ synopsis: /(dwarves|hobbit)/g }).pretty();

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

db.movies.find({ synopsis: /(gold|dragon)/g }).pretty();

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.deleteMany({ title: "Pee Wee Herman's Big Adventure" });

1. delete the movie "Avatar"

db.movies.deleteMany({ 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

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





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 everything",

body: "The end"

},

{

username: "ScumbagSteve",

title: "Forks your repo on github",

body: "Sets to private"

}

]);

Insert the following documents into a **comments** collection



where [post\_obj\_id] is the ObjectId of the posts document: "Borrows something"



where [post\_obj\_id] is the ObjectId of the posts document: "Borrows everything"



where [post\_obj\_id] is the ObjectId of the posts document: "Forks your repo on github



db.comments.insertMany([

{

username: "GoodGuyGreg",

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

post: ObjectId("5f44d3a148197d7749864def")

},

{

username: "GoodGuyGreg",

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

post: ObjectId("5f44d3a148197d7749864df0")

},

{

username: "GoodGuyGreg",

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

post: ObjectId("5f44d3a148197d7749864df1")

},

{

username: "ScumbagSteve",

comment: "It still isn't clean",

post: ObjectId("5f44d3a148197d7749864dec")

},

{

username: "ScumbagSteve",

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

post: ObjectId("5f44d3a148197d7749864dee")

}

]);

where [post\_obj\_id] is the ObjectId of the posts document: "Passes out at party"



where [post\_obj\_id] is the ObjectId of the posts document: "Reports a bug in your code"

db.comments.insertMany([

{

username: "GoodGuyGreg",

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

post: ObjectId("5f44d3a148197d7749864def")

},

{

username: "GoodGuyGreg",

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

post: ObjectId("5f44d3a148197d7749864df0")

},

{

username: "GoodGuyGreg",

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

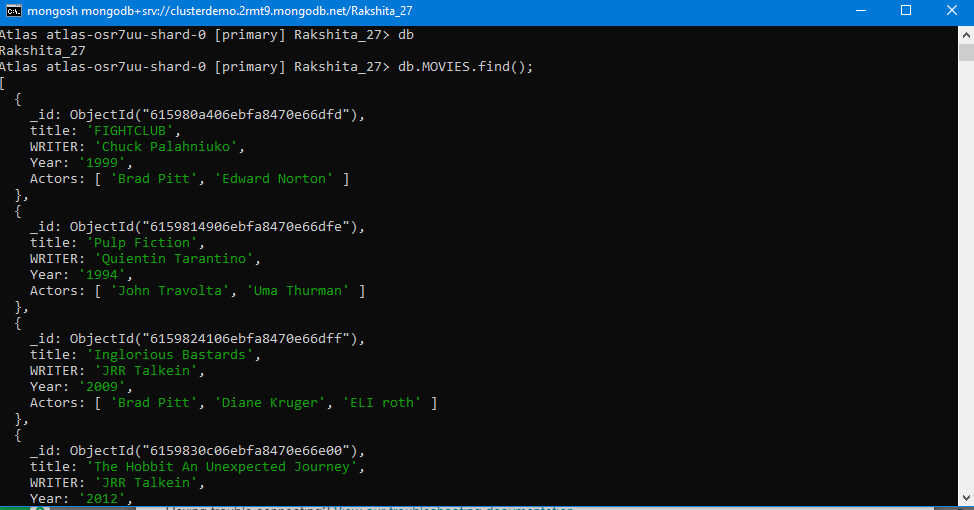
post: ObjectId("5f44d3a148197d7749864df1")

}]);

# 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: 'GoodGuyGreg' }).pretty();

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

db.posts.find({ username: 'ScumbagSteve' }).pretty();

1. find all comments

db.comments.find().pretty();

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

db.comments.find({ username: 'GoodGuyGreg' }).pretty();

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

db.comments.find({ username: 'ScumbagSteve' }).pretty();

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

db.posts.aggregate([

{

$match: { title: 'Reports a bug in your code' }

},

{

$lookup: {

from: 'comments',

localField: '\_id',

foreignField: 'post',

as: 'comments'

}

}

]).pretty();

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

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