# 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)

# Query:

# \_\_\_\_\_\_\_\_\_\_\_\_\_\_

# use mongo\_practice

# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 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
2. get all documents with writer set to "Quentin Tarantino"
3. get all documents where actors include "Brad Pitt"
4. get all documents with franchise set to "The Hobbit"
5. get all movies released in the 90s
6. get all movies released before the year 2000 or after 2010

# Show dbs //mongo\_practice

# db.movies.find()//objected

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

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

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

# db.movies.find({year:{$gt:”1990”,$lt:”2000”}})

# db.movies.find({$or:[{year:{$gt:”2000”}},{year:{$lt:”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."
2. 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."
3. db.movies.update({\_id:ObjectId("5c9f98e5e5c2dfe9b3729bfe")}, {$set:{synopsis:"A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug."}})
4. db.movies.update({\_id:ObjectId("5c9fa42ae5c2dfe9b3729c03")}, {$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."}})
5. db.movies.update({\_id:ObjectId("5c9f983ce5c2dfe9b3729bfc")}, {$push:{actors:"Samuel L. Jackson"}})
6. add an actor named "Samuel L. Jackson" to the movie "Pulp Fiction"

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"
2. find all movies that have a synopsis that contains the word "Gandalf"
3. find all movies that have a synopsis that contains the word "Bilbo" and not the word "Gandalf"
4. find all movies that have a synopsis that contains the word "dwarves" or "hobbit"
5. find all movies that have a synopsis that contains the word "gold" and "dragon"

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

2.db.movies.find({synopsis:{$regex:”Gandalf”}})

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

4.db.movies.find({$or:[{synopsis:{$regex:”dwaves”}},{synopsis:{$regex:”hobbit”}}]})

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

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"
2. delete the movie "Avatar"

1.db.movies.remove({\_id:ObjectId("5c9f992ae5c2dfe9b3729c00")})

2.db.movies.remove({\_id:ObjectId("5c9f9936e5c2dfe9b3729c01")})

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

db.users.insert({\_id:1,username:"GoodGuyGreg", first\_name:"Good Guy", last\_name:"Greg"})

db.users.insert({\_id:2, username:"ScumbagSteve", fullname:{first: "Scumbag", last:"Steve"}})

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.insert({ username:"GoodGuyGreg", title:"Steals your identity", body:"Raises your credit score"})

db.posts.insert({username:"GoodGuyGreg", title:"Reports a bug in your code", body:"Sends you a pull request"})

db.posts.insert({ username:"ScumbagSteve", title:"Borrows something", body:"Sells it"})

db.posts.insert({ username:"ScumbagSteve", title:"Borrows everything", body:"The end"})

db.posts.insert({username:"ScumbagSteve", title:"Forks your repo on github", body:"Sets to private"})

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 part

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:"GoodGuyGreg", comment:"Hope you got a good deal!", post:ObjectId("5ca0b7e96435f98b5901f463")})

db.comments.insert({username:"GoodGuyGreg", comment:"What's mine is yours!", post:ObjectId("5ca0b9706435f98b5901f46a")})

db.comments.insert({username:"GoodGuyGreg", comment:"Don't violate the licensing agreement!", post:ObjectId("5ca0b8766435f98b5901f467")})

db.comments.insert({username:"ScumbagSteve", comment:"It still isn't clean", post:ObjectId("5ca0b8546435f98b5901f466")})

db.comments.insert({username:"ScumbagSteve", comment:"Denied your PR cause I found a hack", post:ObjectId("5ca0b9256435f98b5901f469")})

# Querying related collections

1. find all users
2. find all posts
3. find all posts that was authored by "GoodGuyGreg"
4. find all posts that was authored by "ScumbagSteve"
5. find all comments
6. find all comments that was authored by "GoodGuyGreg"
7. find all comments that was authored by "ScumbagSteve"
8. d all comments belonfinging to the post "Reports a bug in your code"

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

2.db.posts.find().pretty()

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

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

5.db.comments.find().pretty()

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

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

8. db.comments.find(post:ObjectId("8ca0b6257435f98b5901f469"))

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

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