# 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

]

db.movies.insertOne({"Title":"Fight Club","Writer":"Chuck Palahniuko","year":"1999","Actor:["Brad Pitt","Edward Norton"]})

title : Pulp Fiction

writer : Quentin Tarantino year : 1994

actors : [

John Travolta Uma Thurman

]

db.movie.insertOne({ Title:"Plup Fiction" , Writer:"quentin Tarantino" ,year:"1994" ,Actor: [ "John Travolta" , "Uma Thurman"]})

title : Inglorious Basterds writer : Quentin Tarantino year : 2009

actors : [ Brad Pitt

Diane Kruger Eli Roth

]

db.movie.insertOne({ Title:"Inglorious BasterdsWriter:"quenTarantino" ,year:"2009" ,Actor: ["Brad Pitt", "Diane Kruger" , "Eli Roth"]})

title : The Hobbit: An Unexpected Journey writer : J.R.R. Tolkein

year : 2012

franchise : The Hobbit

db.movie.insertOne({ 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

db.movie.insertOne({ Title:"The Hobbit:The Desolation 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.

db.movie.insertOne({ 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 engagein 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

db.movie.find()

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

db.movie.find({Writer:"quentin Tarantino"})

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

db.movie.find({Actor:"Brad Pitt"})

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

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

1. get all movies released in the 90s

db.movie.find({Year:{$gt:"1990", $lt:"2000"}})

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

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

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.movie.update({"Title":"The Hobbit:An Unexpected Journey"}, {$set:{"Synopsis":"A relucant hobbbit,BilboBaggins,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.movie.update({"Title":"The Hobbit:The Desolation of Smaug"}, {$set:{"Synopsis":"The dwarves along with Bilbo Baggins and Gandalfthe gery,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.movie.update({"Title":"Plup Fiction"}, {$set:{"Actor":"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"

ANS-> db.movie.find({Synopsis:{$regex:"Bilbo"}})

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

ANS-> db.movie.find({Synopsis:{$regex:"Gandalf"}})

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

ANS-> db.movie.find({$and:[{Synopsis:{$regex:"Bilbo"}},{Synopsis:{$not:{$regex:"Gandalf"}}}]})

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

ANS->

db.movie.find({$or:[{Synopsis:{$regex:"dwarves"}},{Synopsis:{$regex:"hobbit"}}]})

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

ANS->

db.movie.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"

db.movie.deleteOne({"Title":"Bee Wee Harman's Big adventure"})

1. delete the movie "Avatar"

db.movie.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

use relation

username : GoodGuyGreg

first\_name : "Good Guy"

last\_name : "Greg"

db.users.insertOne({Username:"GoodGuyGreg",First\_Name:"GoodGuy",Last\_Name:"Grey"})

username : ScumbagSteve full\_name :

first : "Scumbag" last : "Steve"

db.users.insertOne({Username:"Scumbagsteve",Full\_Name:{First:"Scumbag",Last:"steve"}})

Insert the following documents into a **posts** collection username : GoodGuyGreg

title : Passes out at party

boy : Wakes up early and cleans house

db.posts.insertOne({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

db.posts.insertOne({Username:"GoodGuyGreg",Title:"Steals your identity",Body:"Raise your credit score"})

username : GoodGuyGreg

title : Reports a bug in your code body : Sends you a Pull

Request

db.posts.insertOne({Username:"GoodGuyGreg",Title:"Reports a bug in your code",Body:"Sends you a pull Request"})

username :ScumbagSteve title : Borrows something body : Sells it

db.posts.insertOne({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"

db.comments.insertOne({Username:"GoodGuyGreg",comments:"Hope you got a goog deal!",post:ObjectId("61da621e40c50180c0b3de97")})

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"

db.comments.insertOne({Username:"GoodGuyGreg",comments:"What's mine is yours!",post: ObjectId("61da648b40c50180c0b3de9a")})

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

db.comments.insertOne({Username:"GoodGuyGreg",comments:"Don't violate the licensing agreement!",post: ObjectId("61da628d40c50180c0b3de99")})

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"

db.comments.insertOne({Username:"Scumbagsteve",comments:"It still isn't clean",post: ObjectId("61da60e440c50180c0b3de94")}

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.insertOne({Username:"Scumbagsteve",comments:"Denied your PR cause I found a hack",post: ObjectId("61da61c240c50180c0b3de96")})

# 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"

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

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