**MongoDB Exercise in mongo shell**

1. Connect to a running mongo instance

* **C:\Users\hp>mongo**

1. use a database named mongo\_practice.

* **use mongo\_practice**

**Insert Documents**

Insert the following documents into a movies collection.

1. 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

]

* **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']},**

**{title : 'Inglorious Basterds', writer : 'Quentin Tarantino', year : 2009, actors : [ 'Brad Pitt', 'Diane Kruger', 'Eli Roth' ]}])**

1. 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.

* **db.movies.insertMany([{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.'}])**

1. title : Pee Wee Herman's Big Adventure

* **db.movies.insertOne({title:"Pee Wee Herman's Big Adventure"})**

1. title : Avatar

* **db.movies.insertOne({title:"Avatar"})**

**Query / Find Documents**

query the movies collection to

1. get all documents

* **db.movies.find()**

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

* **db.movies.find({writer:'Quentin Tarantino'})**

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

* **db.movies.find({actors:{$all:['Brad Pitt']}})**

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

* **db.movies.find({franchise:'The Hobbit'})**

1. get all movies released in the 90s

* **db.movies.find({$and:[{year:{$gt:1899}},{year:{$lt:2000}}]})**

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

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

**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'},{$addToSet:{actors:'Samuel L. Jackson'}})**

**Text Search**

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

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

* **db.movies.find({$text:{$search:'Bilbo'}})**
* **db.movies.find({synopsis:{$regex:"Bilbo"}})**

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

* **db.movies.find({$text:{$search:'Gandalf'}})**
* **db.movies.find({synopsis:{$regex:"Gandalf"}})**

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

* **db.movies.find({$text:{$search:'Bilbo -Gandalf'}})**
* **db.movies.find({$and:[{synopsis:{$regex:"Bilbo"}}, {synopsis:{$not:/Gandalf/}}]})**

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

* **db.movies.find({$text:{$search:'dwarves hobbit'}})**
* **db.movies.find({$or:[{synopsis:{$regex:"dwarves"}}, {synopsis:{$regex:"hobbit"}}]})**

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

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

**Delete Documents**

1. delete the movie "Pee Wee Herman's Big Adventure"

* **db.movies.remove({title:"Pee Wee Herman's Big Adventure"})**

1. delete the movie "Avatar"

* **db.movies.remove({\_id:ObjectId("5ffeb64d106ec63db0a09eb3")})**

**Relationships**

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

1. username : GoodGuyGreg

first\_name : "Good Guy"

last\_name : "Greg"

* **db.users.insert({username:'GoodGuyGreg', first\_name:'Good Guy', last\_name:'Greg'})**

1. username : ScumbagSteve

full\_name :

first : "Scumbag"

last : "Steve"

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

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

1. username : GoodGuyGreg

title : Passes out at party

body : Wakes up early and cleans house

* **db.posts.insert({username:"GoodGuyGreg", title:"Passes out at Party", body:"Raises your credit score"})**

1. username : GoodGuyGreg

title : Steals your identity

body : Raises your credit score

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

1. username : GoodGuyGreg

title : Reports a bug in your code

body : Sends you a Pull Request

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

1. username : ScumbagSteve

title : Borrows something

body : Sells it

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

1. username : ScumbagSteve

title : Borrows everything

body : The end

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

1. username : ScumbagSteve

title : Forks your repo on github

body : Sets to private

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

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

1. 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.insert({ username:"GoodGuyGreg", comment:"Hope you got a good deal!", post:ObjectId("5fff20d315ca2b605f2fff3b")})**

1. 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.insert({username:"GoodGuyGreg", comment:"What's mine is yours!", post:ObjectId("5fff20dc15ca2b605f2fff3c")})**

1. 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.insert({username:"GoodGuyGreg", comment:"Don't violate the licensing agreement!", post:ObjectId("5fff20e515ca2b605f2fff3d")})**

1. 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.insert({username:"ScumbagSteve", comment:"It still isn't clean", post:ObjectId("5ffedff84ff1675522810543")})**

1. 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("5fff209715ca2b605f2fff3a")})**

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"})**

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

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

1. find all comments

* **db.comments.find().pretty()**

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({post:ObjectId("5fff209715ca2b605f2fff3a")})**