Assignment – 1

Ques) Connect to a running mongo instance, use a database named **mongo\_practice**.

Ans) step 1: Open command prompt(cmd) and run “mongod”, then wait for connection establishment

Step 2: Run now “mongo” now to coonect to mongo shell, we can start our queries

Step 3: Run “use mongo\_practice”

**Insert Documents**

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

Ans) db.movies.insertMany([{data 1},{data 2},…]). If collection movies is not there, it will be created automatically.

Here data 1, data 2 …. data n are the records that we wanted to enter. Array of documents.

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" ]},

{

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() or db.movies.find().pretty()

Pretty() – more easy format to read

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

Ans) db.movies.find({},{title:"$title",year:"$year",actors:"$actors",writer:"Quentin Tarantino"}).pretty() or db.movies.aggregate([{$match:{writer:"Quentin Tarantino"}}])

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

db.movies.find({},{title:"$title",year:"$year",actors:"Brad Pitt",writer:"$writr"}).pretty()

or

db.movies.aggregate([{$match: {actors:”Brad Pitt”}}])

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

Ans) db.movies.find({},{title:"$title",year:"$year",actors:"$actors",writer:"$writer",franchise:"The Hobbit"}).pretty() or db.movies.aggregate([

{$match : {franchise :"The Hobbit"}}

])

5. get all movies released in the 90s

Ans)db.movies.find({},{title:"$title",year:{$lt:["$year",2000]},actors:"$actors",writer:"$writer",franchise:"$franchise"}).pretty()

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

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

**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."

Ans) 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."

Ans) 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"

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

**Text Search**

Firstly create a new Index to make it easy

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

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

db.movies.find({$text:{$search:"Bilbo"}})

or

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

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

db.movies.find({$text:{$search:"Gandalf"}})

Or

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

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

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

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

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

5. 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"})

2. delete the movie "Avatar”

db.movies.remove({title:"Avatar"})

**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

db.comments.insertMany([

{ username:"GoodGuyGreg",

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

post:ObjectId("600019cf13cc6db7f0ef71ec")},

{

username : "GoodGuyGreg",

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

post : ObjectId("600019cf13cc6db7f0ef71ed")

},

{

username : "GoodGuyGreg",

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

post : ObjectId("600019cf13cc6db7f0ef71ee")

},{

username : "ScumbagSteve",

comment : "It still isn't clean",

post : ObjectId("600019cf13cc6db7f0ef71e9")

},

{

username : "ScumbagSteve",

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

post : ObjectId("600019cf13cc6db7f0ef71eb")

}])

**Querying related collections**

1. find all users

db.users.find()

2. find all posts

db.posts.find()

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

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

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

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

5. find all comments

db.comments.find()

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

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

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

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

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

db.posts.aggregate([

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

])