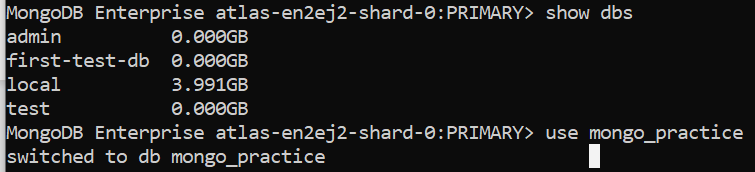
**MongoDB Lab Assignments -Day 1**

**MongoDB Exercise in mongo shell**

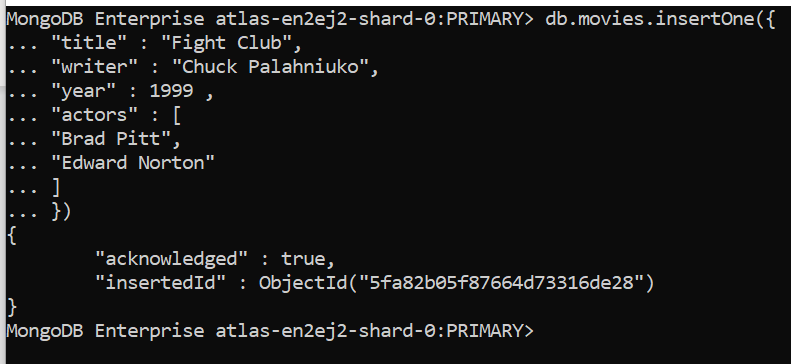
Connect to a running mongo instance, use a database named **mongo\_practice**.



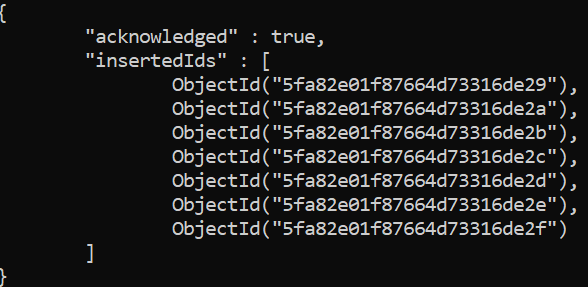
Document all your queries in a javascript file to use as a reference.

**Insert Documents**

* **InsertOne Records**



* **insertMany Records**



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://www.tutorialspoint.com/mongodb/mongodb\_insert\_document.htm

**Query / Find Documents**

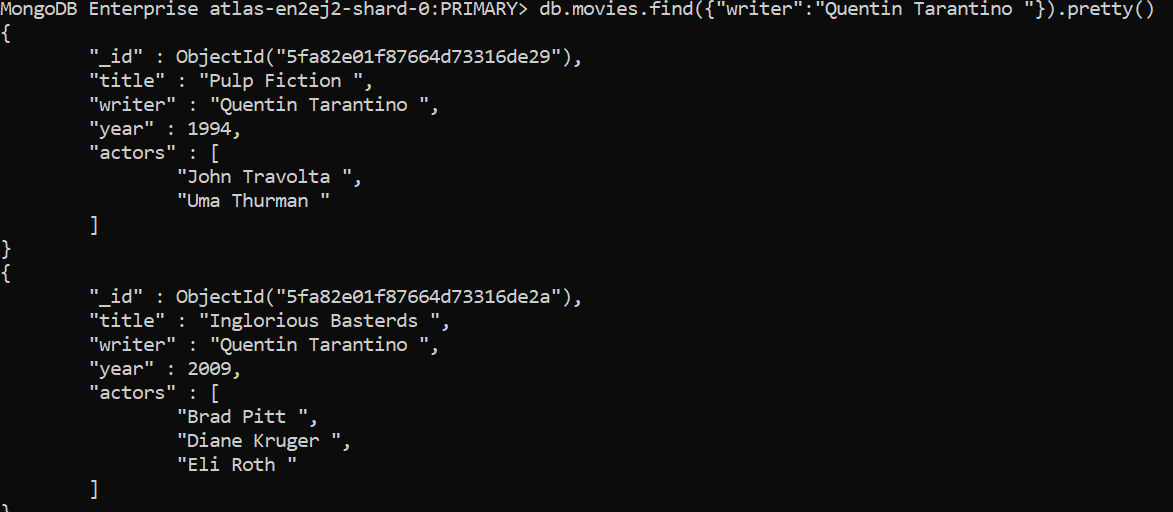
query the **movies** collection to

1. get all documents

> db.movies.find()

2. 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": "Brad Pitt " }).pretty()

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

> db.movies.find({"franchise": "The Hobbit " }).pretty()

1. get all movies released in the 90s

> db.movies.find({"year": {$lt: 2000} }).pretty()

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

> db.movies.find(

{

"$or": [

{"year": {$lt: 2000}},

{"year": {$gt: 2010} }

]

}

).pretty()

Reference: https://www.tutorialspoint.com/mongodb/mongodb\_query\_document.htm

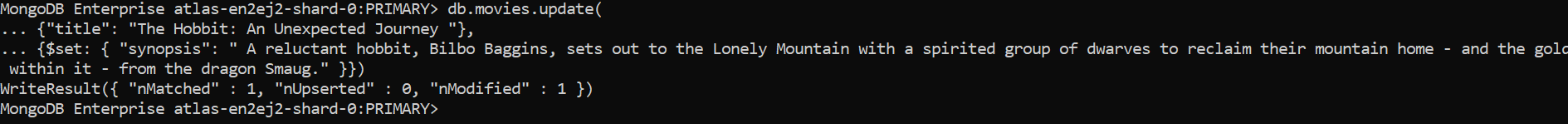
**Update Documents**

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.updateOne(

{"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." }})



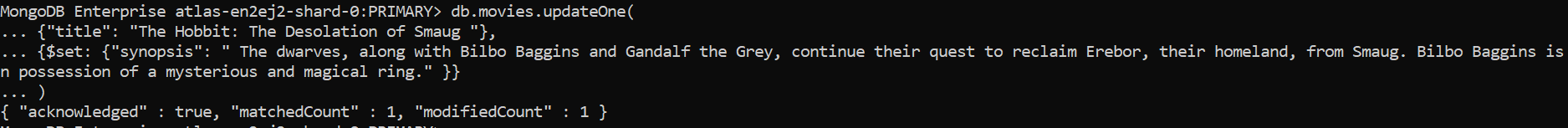
2. add a synopsis to "The Hobbit: The Desolation of Smaug" : "The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."

> db.movies.updateOne(

{"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." }}

)



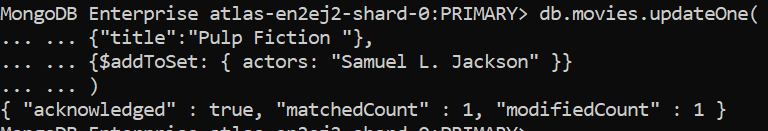
3. add an actor named "Samuel L. Jackson" to the movie "Pulp Fiction"

db.movies.updateOne(

{"title": "Pulp Fiction "},

{$addToSet: {actors: "Samuel L. Jackson"}}

)



Reference: https://www.tutorialspoint.com/mongodb/mongodb\_update\_document.htm

**Text Search**

* **Index created on** synopsis filed

MongoDB Enterprise atlas-en2ej2-shard-0:PRIMARY> db.movies.createIndex({synopsis:"text"})

{

"createdCollectionAutomatically" : false,

"numIndexesBefore" : 1,

"numIndexesAfter" : 2,

"ok" : 1,

"$clusterTime" : {

"clusterTime" : Timestamp(1604862077, 2),

"signature" : {

"hash" : BinData(0,"9Ge644DCTsou4RDQj1pdgLR5gII="),

"keyId" : NumberLong("6884935607458463747")

}

},

"operationTime" : Timestamp(1604862077, 2)

}

* Get All the index Details:

MongoDB Enterprise atlas-en2ej2-shard-0:PRIMARY> db.movies.getIndexes()

[

{

"v" : 2,

"key" : {

"\_id" : 1

},

"name" : "\_id\_",

"ns" : "mongo\_practice.movies"

},

{

"v" : 2,

"key" : {

"\_fts" : "text",

"\_ftsx" : 1

},

"name" : "synopsis\_text",

"ns" : "mongo\_practice.movies",

"weights" : {

"synopsis" : 1

},

"default\_language" : "english",

"language\_override" : "language",

"textIndexVersion" : 3

}

]

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

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

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

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

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

> db.movies.find({$text:{$search:'Bilbo -Gandalf'}}).pretty()

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

> db.movies.find({$text:{$search:'dwarves, hobbit'}}).pretty()

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

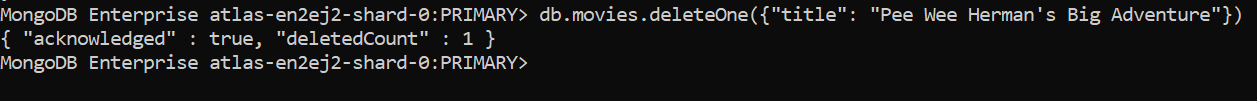
> db.movies.find({$text:{$search:'gold, dragon'}}).pretty()

Reference: https://www.tutorialspoint.com/mongodb/mongodb\_text\_search.htm

**Delete Documents**

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

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



2. delete the movie "Avatar"

> db.movies.deleteOne({"title": "Avatar"})



Reference: <https://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"

> 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

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

> 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

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 party"

>

**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"

> MongoDB Enterprise atlas-en2ej2-shard-0:PRIMARY> db.posts.find({'username':'GoodGuyGreg'})

{ "\_id" : ObjectId("5fa8471eeb4b293d8bae88bd"), "username" : "GoodGuyGreg", "title" : "Passes out at party", "body" : "Wakes up early and cleans house" }

{ "\_id" : ObjectId("5fa8471eeb4b293d8bae88be"), "username" : "GoodGuyGreg", "title" : "Steals your identity", "body" : "Raises your credit score" }

{ "\_id" : ObjectId("5fa8471eeb4b293d8bae88bf"), "username" : "GoodGuyGreg", "title" : "Reports a bug in your code", "body" : "Sends you a Pull Request" }

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

MongoDB Enterprise atlas-en2ej2-shard-0:PRIMARY> db.posts.find({'username':"ScumbagSteve"})

{ "\_id" : ObjectId("5fa8471eeb4b293d8bae88c0"), "username" : "ScumbagSteve", "title" : "Borrows something", "body" : "Sells it" }

{ "\_id" : ObjectId("5fa8471eeb4b293d8bae88c1"), "username" : "ScumbagSteve", "title" : "Borrows everything", "body" : "The end" }

{ "\_id" : ObjectId("5fa8471eeb4b293d8bae88c2"), "username" : "ScumbagSteve", "title" : "Forks your repo on github", "body" : "Sets to private" }

MongoDB Enterprise atlas-en2ej2-shard-0:PRIMARY> 5. find all comments

> db.comments.find()

6. 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"})

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

db.comments.find({post: "5fa8471eeb4b293d8bae88bf"})

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

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