**ASSIGNMENT 1**

**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"]},**

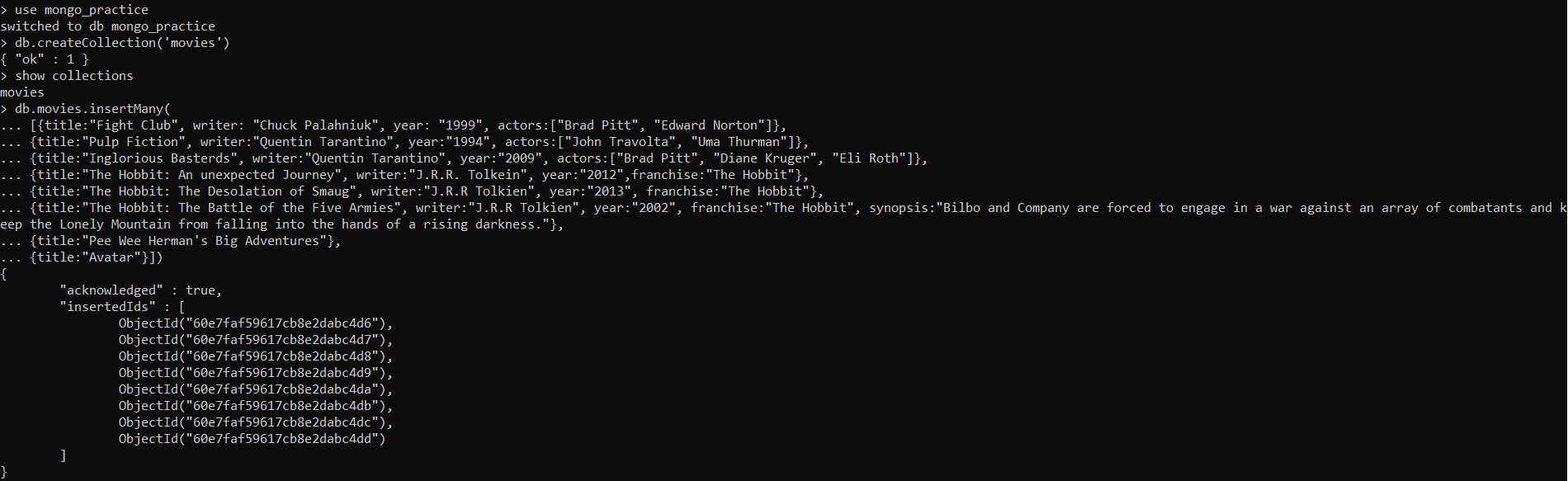
**{title:"The Hobbit: An unexpected Journey", writer:"J.R.R. Tolkein", year:"2012",franchise:"The Hobbit"},**

**{title:"The Hobbit: The Desolation of Smaug", writer:"J.R.R Tolkien", year:"2013", franchise:"The Hobbit"},**

**{title:"The Hobbit: The Battle of the Five Armies", writer:"J.R.R Tolkien", year:"2002", 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 Adventures"},**

**{title:"Avatar"}])**



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:"Brad Pitt"})**

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

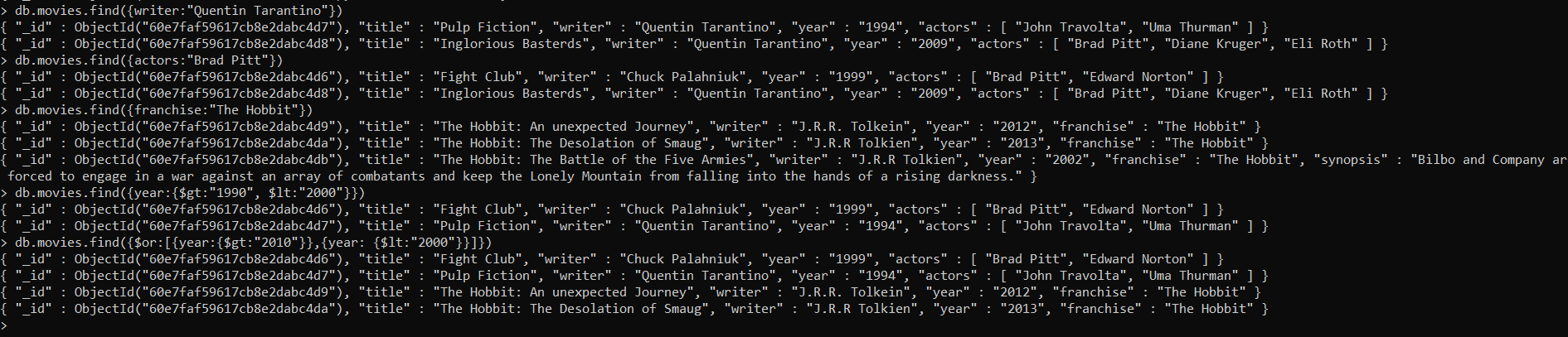
**db.movies.find({franchise:"The Hobbit"})**

1. get all movies released in the 90s

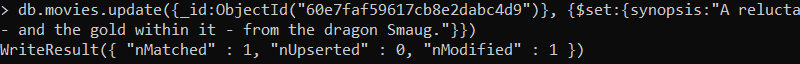
**db.movies.find({year:{$gt:"1990", $lt:"2000"}})**

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

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

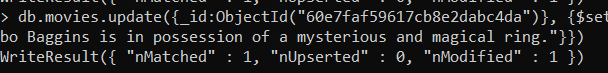


1. add a synopsis to "The Hobbit: An Unexpected Journey" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug."



db.movies.update({\_id:ObjectId("60e7faf59617cb8e2dabc4d9")}, {$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."



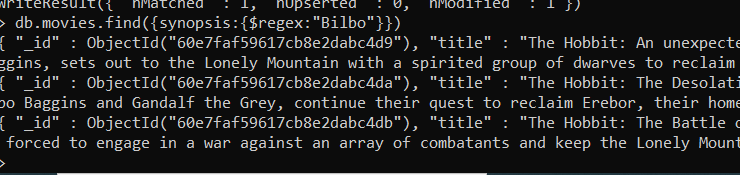
db.movies.update({\_id:ObjectId("60e7faf59617cb8e2dabc4da")}, {$set:{synopsis:"The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."}})

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



db.movies.update({\_id:ObjectId("60e7faf59617cb8e2dabc4d7")}, {$push:{actors:"Samuel L. Jackson"}})

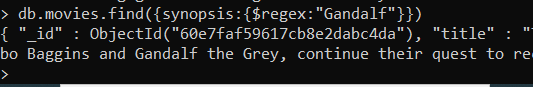
1. find all movies that have a synopsis that contains the word "Bilbo"



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

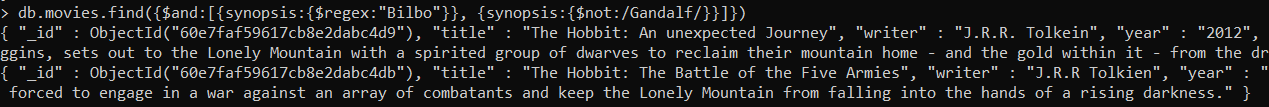
1. find all movies that have a synopsis that contains the word "Gandalf"

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



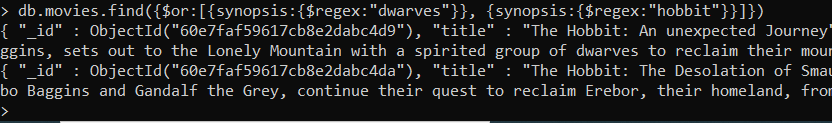
1. find all movies that have a synopsis that contains the word "Bilbo" and not the word "Gandalf"

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



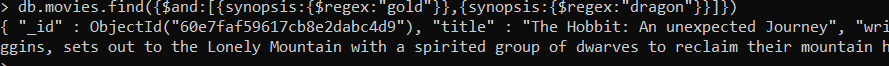
1. find all movies that have a synopsis that contains the word "dwarves" or "hobbit"

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

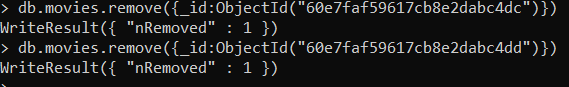


1. find all movies that have a synopsis that contains the word "gold" and "dragon"

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



DELETE DOCS



**INSERT**

> db.createCollection('users')

{ "ok" : 1 }

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

WriteResult({ "nInserted" : 1 })

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

WriteResult({ "nInserted" : 1 })

> db.createCollection('posts')

{ "ok" : 1 }

> db.posts.insert({\_id:1,username:"GoodGuyGreg", title:"Passes out at Party", body:"Raises your credit score"})

WriteResult({ "nInserted" : 1 })

> db.posts.remove({\_id:1})

WriteResult({ "nRemoved" : 1 })

> db.posts.insert({\_id:1,username:"GoodGuyGreg", title:"Passes out at Party", body:"Wakes up early and cleans house"})

WriteResult({ "nInserted" : 1 })

> db.posts.insert({\_id:2,username:"GoodGuyGreg", title:"Steals your identity", body:"Raises your credit score"})

WriteResult({ "nInserted" : 1 })

> db.posts.insert({\_id:3,username:"GoodGuyGreg", title:"Reports a bug in your code", body:"Sends you a pull request"})

WriteResult({ "nInserted" : 1 })

> db.posts.insert({\_id:4,username:"ScumbagSteve", title:"Borrows something", body:"Sells it"})

WriteResult({ "nInserted" : 1 })

> db.posts.insert({\_id:5,username:"ScumbagSteve", title:"Borrows everything", body:"The end"})

WriteResult({ "nInserted" : 1 })

> db.posts.insert({\_id:6,username:"ScumbagSteve", title:"Forks your repo on github", body:"Sets to private"})

WriteResult({ "nInserted" : 1 })

> db.createCollection('comments')

{ "ok" : 1 }

> db.comments.insert({username:"GoodGuyGreg", comment:"Hope you got a good deal!", post:4})

WriteResult({ "nInserted" : 1 })

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

WriteResult({ "nInserted" : 1 })

> db.comments.insert({username:"GoodGuyGreg", comment:"Don't violate the licensing agreement!", post:6})

WriteResult({ "nInserted" : 1 })

> db.comments.insert({username:"ScumbagSteve", comment:"It still isn't clean", post:1})

WriteResult({ "nInserted" : 1 })

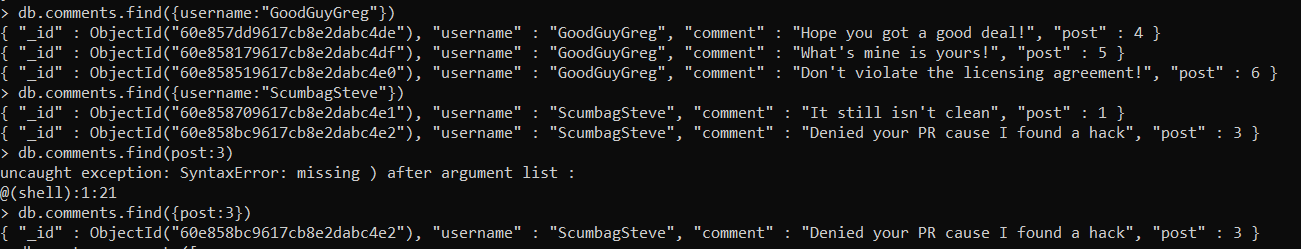
> db.comments.insert({username:"ScumbagSteve", comment:"Denied your PR cause I found a hack", post:3})

WriteResult({ "nInserted" : 1 })

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

//2.find all posts  
db.posts.find().pretty();

//3.find all posts that was authored by "GoodGuyGreg"  
db.posts.find({ username: 'GoodGuyGreg' }).pretty();



//4.find all posts that was authored by "ScumbagSteve"  
db.posts.find({ username: 'ScumbagSteve' }).pretty();

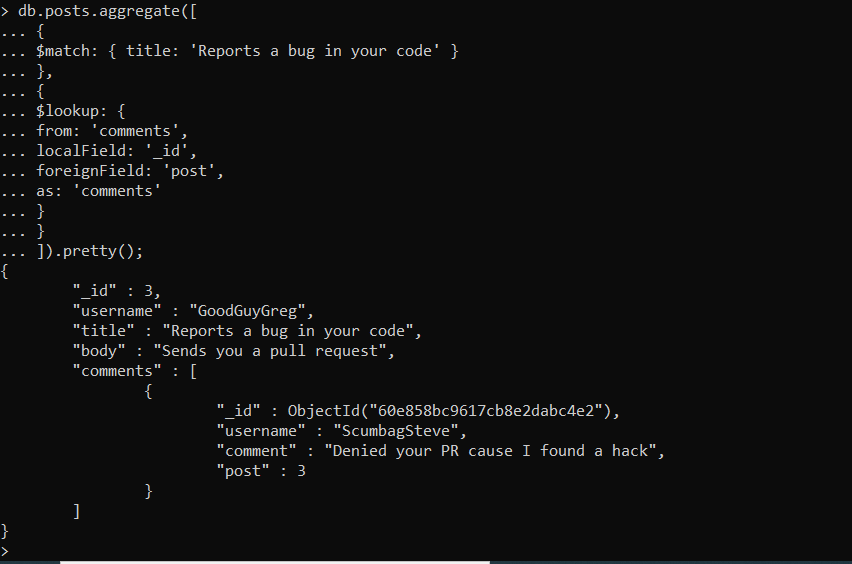
//5.find all comments  
db.comments.find().pretty();

//6.find all comments that was authored by "GoodGuyGreg"  
db.comments.find({ username: 'GoodGuyGreg' }).pretty();

//7.find all comments that was authored by "ScumbagSteve"  
db.comments.find({ username: 'ScumbagSteve' }).pretty();

**//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' }  
},  
{  
$lookup: {  
from: 'comments',  
localField: '\_id',  
foreignField: 'post',  
as: 'comments'  
}  
}  
]).pretty();**



1. db.addresses.find()
2. db.addresses.find({},{restaurant\_id:1,name:1,borough:1,cuisine:1})
3. db.addresses.find({},{restaurant\_id:1,name:1,borough:1,cuisine:1,\_id:0})
4. db.addresses.find({},{restaurant\_id:1,name:1,borough:1,”address.zipcode”:1,\_id:0})
5. db.addresses.find({borough:”Bronx”}).limit(5)
6. db.addresses.find({borough:”Bronx”})
7. db.addresses.find({borough:”Bronx”}).skip(5).limit(5)
8. db.addresses.find({“grades”:{$elemMatch:{score:{$gt:90}}}})
9. db.addresses.find({“grades”:{$elemMatch:{score:{$gt:80,$lt:100}}}})
10. db.addresses.find({"address.coord.0":{$lt:-95.754168}})
11. **db.addresses.find({$and:[{cuisine:{$not:/American/}},{"grades.score":{$gt:70}},{"address.coord.0":{$lt:-65.754168}}]}).pretty()**
12. db.addresses.find({$and:[{cuisine:{$not:{$regex:"American"}}},{"grades.score":{$gt:70}},{"address.coord.1":{$lt:-65.754168}}]})
13. db.addresses.find({$and:[{cuisine:{$not:{$regex:"American"}}},{"grades.grade":"A"},{borough:{$not:{$regex:"Brooklyn"}}}]}).sort({cuisine:-1})
14. db.addresses.find({name:/^Wil/},{restaurant\_id:1,name:1,borough:1,cuisine:1})
15. db.addresses.find({name:/ces$/},{restaurant\_id:1,name:1,borough:1,cuisine:1})
16. db.addresses.find({name:/.\*Reg.\*/},{restaurant\_id:1,name:1,borough:1,cuisine:1})
17. db.addresses.find({borough:"Bronx",$or:[{cuisine:"American"},{cuisine:"Chinese"}]})
18. db.addresses.find({$or:[{borough:"Staten Island"},{borough:"Queens"},{borough:"Bronx"},{borough:"Brooklyn"}]},{restaurant\_id:1,name:1,borough:1,cuisine:1})
19. db.addresses.find({$and:[{borough:{$not:/Bronx/}},{borough:{$not:/Brooklyn/}},{borough:{$not:/Staten Island/}},{borough:{$not:/Queens/}}]},{restaurant\_id:1,name:1,borough:1,cuisine:1})
20. db.addresses.find({"grades.score":{$lte:10}},{restaurant\_id:1,name:1,borough:1,cuisine:1})
21. db.addresses.find({$or:[{name:/^Wil/},{$and:[{cuisine:{$not:/American/}},{cuisine:{$not:/Chinese/}}]}]},{restaurant\_id:1,name:1,borough:1,cuisine:1})
22. db.addresses.find({"grades.grade":"A","grades.score":11,"grades.date":ISODate('2014-08-11T00:00:00Z')}, {restaurant\_id:1,name:1,grades:1})
23. db.addresses.find({"grades.1.grade":"A","grades.1.score":9,"grades.1.date":ISODate('2014-08-11T00:00:00Z')}, {restaurant\_id:1,name:1,grades:1})
24. db.addresses.find({$and:[{"address.coord.1":{$gt:42}},{"address.coord.1":{$lte:52}}]},{restaurant\_id:1,name:1,"address":1})
25. db.addresses.find().sort({name:1})
26. db.addresses.find().sort({name:-1})
27. db.addresses.find().sort({cuisine:1,borough:-1})
28. db.addresses.aggregate([{$match:{“address.street”:{$exists:false}}}]).toArray().length
29. db.addresses.find({“address.coord”:{$type:1}})
30. db.addresses.find({"grades.score":{$mod:[7,0]}}, {restaurant\_id:1,name:1,grades:1})
31. db.addresses.find({name:/.\*mon.\*/}, {name:1,borough:1,cuisine:1,"address.coord":1})
32. db.addresses.find({name:/^Mad/}, {name:1,borough:1,cuisine:1,"address.coord":1})