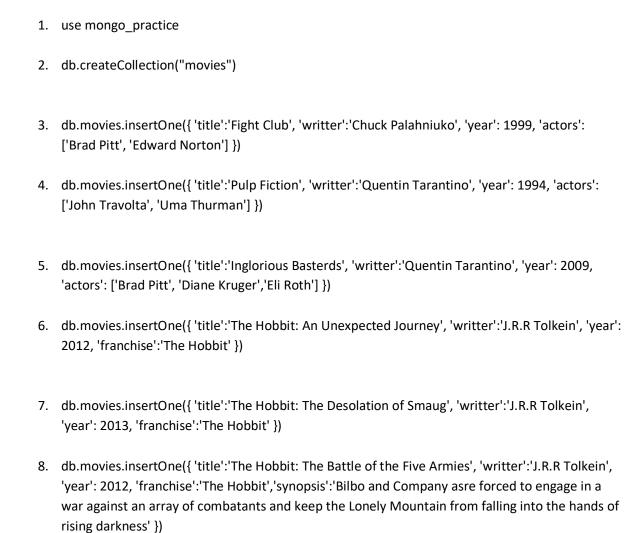
MongoDB Lab Assignment - Day 1

Insert Documents:

10. db.movies.find().pretty()



9. db.movies.insertMany([{ title: "Pee Wee Herman's Big Adventure" }, { title: "Avatar" }])

Query/ Find Documents:

- 1. db.movies.find()
- 2. db.movies.find({"writter":"Quentin Tarantino"}).pretty()
- 3. db.movies.find({"actors":"Brad Pitt"}).pretty()
- 4. db.movies.find({"franchise":"The Hobbit"}).pretty()
- 5. db.movies.find({ \$and:[{"year": {\$gte:1900}}, {"year": {\$lt:2000}}] }).pretty()
- 6. db.movies.find({ \$or:[{"year": {\$lt:2000}}, {"year": {\$gt:2010}}] }). pretty()

Update Documents:

- 1. 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'} })
- 2. 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.'} })
- 3. db.movies.update({'title':'Pulp Fiction'},{\$set:{'actors':'Samuel L.Jackson'}})

Text Search:

- db.movies.createIndex({synopsis:"text"})
 db.movies.find({\$text:{\$search:"Bilbo"}}).pretty()
- 2. db.movies.find({\$text:{\$search:"Gandalf"}}).pretty()
- db.movies.find({\$text:{\$search:"Bilbo -Gandalf"}}).pretty()
- 4. db.movies.find({\$text:{\$search:"dwarves hobbit"}}).pretty()
- 5. db.movies.find({\$text:{\$search:"gold dragon"}}).pretty()

Delete Documents:

- 1. db.movies.remove({"title":"Pee Wee Herman's Big Adventure"})
- 2. db.movies.remove({"title":"Avatar"})

Relationships:

<u>Users collection:</u>

- db.users.insertOne({"username":"GoodGuyGreg","first_name":"Good Guy","last_name":"greg"})
- 2. db.users.insertOne({"username":"ScumbagSteve","full_name":{"first":"Scumbag","last":"Steve"}
 })

posts collection:

comments collection:

- 1. db.comments.insertOne({username:"GoodGuyGreg",comment:"Hope you got a good deal!", post: ObjectId("617438978548d20d7bb74f1c")})
- 2. db.comments.insertOne({username:"GoodGuyGreg",comment:"What's mine is yours!", post: ObjectId("617438978548d20d7bb74f1d")})
- 3. db.comments.insertOne({username:"GoodGuyGreg",comment:"Don't violate the licensing agreement!", post: ObjectId("617438978548d20d7bb74f1e")})
- 4. db.comments.insertOne({username:"ScumbagSteve",comment:"It still isn't clean", post: ObjectId("617437088548d20d7bb74f19")})
- 5. db.comments.insertOne({username:"ScumbagSteve",comment:"Denied your PR cause I found a hack", post: ObjectId("617438978548d20d7bb74f1b")})

Querying related collections:

- db.users.find().pretty()
- 2. db.posts.find().pretty()
- 3. db.posts.find({"username":"GoodGuyGreg"}).pretty()
- 4. db.posts.find({"username":"ScumbagSteve"}).pretty()
- 5. db.comments.find().pretty()
- 6. db.comments.find({"username":"GoodGuyGreg"}).pretty()
- 7. db.comments.find({"username":"ScumbagSteve"}).pretty()
- $8. \quad db. comments. find (\{post: \{ sin: [ObjectId("617438978548d20d7bb74f1b")] \} \}). pretty() \\$