MongoDB

Assignment-1

Assignment 1

Sample data:

#1

```
{
      "Title" : "Fight Club",
      "Writer" : "Chuck Palahniuko",
      "Year" : 1999,
      "Actors" : [
            "Brad Pitt",
            "Edward Norton"
      ]
}
```

#2

```
{
 "Title" : "Pulp Fiction",
 "Writer" : "Quentin Tarantino",
 "Year" : 1994,
 "Actors" : [
 "John Travolta",
 "Uma Thurman"
 ]
}
```

#3

```
db.movies.insertOne({
      "Title" : "Inglorious Basterds",
      "Writer" : "Quentin Tarantino",
      "Year" : 2009,
      "Actors" : [
```

```
            "Brad Pitt",
            "Diane Kruger",
                        "Eli Roth"
        ]
})
#4
db.movies.insertOne({
    "Title" : "The Hobbit: An Unexpected Journey",
    "Writer" : "J.R.R. Tolkein",
    "Year" : 2012,
    "Franchise" : "The Hobbit"
})
#5
db.movies.insertOne({
    "Title" : "The Hobbit: The Desolation of Smaug",
    "Writer" : "J.R.R. Tolkein",
    "Year" : 2013,
    "Franchise" : "The Hobbit"
})
#6
db.movies.insertOne({
    "Title" : "The Hobbit: The Battle of the Five Armies",
    "Writer" : "J.R.R. Tolkein",
    "Year" : 2012,
    "Franchise" : "The Hobbit",
        "Synopsis" : "Bildo and Company are forced to engage in a war against an array ofcombatants and keep the Lonely Mountain from falling into the hands of a rising darkness."
})

#7
db.movies.insertOne({
        "Title" : "Pee Wee Herman's Big Adventure"
})
```

#8

```
db.movies.insertOne({
        "Title" : "Avatar"
})
```

query the movies collection to

1. get all documents

```
db.movies.find().pretty()
```

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

```
db.movies.find({Writer:"Quentin Tarantino"}).pretty()
```

3. 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()
```

5. get all movies released in the 90s

```
db.movies.find({$and:[{Year:{$lt:2000}},{Year:{$gt:1900}}]}).pretty()
```

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

```
db.movies.find({$and:[{Year:{$lt:2000}},{Year:{$gt:2010}}]}).pretty()
```

Update document

#1

```
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

```
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

db.movies.updateOne({Title:"Pulp Fiction"},{$push:{Actors:"Samuel L.Jackson"}})

Text Search

Before we start with the text search, an index with text should be created

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

--check the indexes

db.movies.getIndexes()

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"}})

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

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

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

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

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

db.movies.find({$text:{$search:"gold 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

created users collection

#1

db.users.insertOne({Username:"GoodGuyGreg", first_name:"GoodGuy", last_name:"Greg"})

#2

db.users.insertOne({Username:"ScumbagSteve", full_name:{first:"Scumbag", last:"Steve"}})

created posts collections

#1

db.posts.insertOne({Username:"GoodGuyGreg",

title:"Passes out at party", body:"Wakes up early and cleans house"})

#2

db.posts.insertOne({Username:"GoodGuyGreg",

title:"Steals your identity", body:"Raises your credit score"})

#3

db.posts.insertOne({Username:"GoodGuyGreg",

title:"Reports a bug in your code", body:"Sends you a pull request"})

#4

db.posts.insertOne({Username:"ScumbagSteve",

title:"Borrows something", body:"Sells it"})

$5

db.posts.insertOne({Username:"ScumbagSteve",

title:"Borrows everything", body:"The end"})

#6

db.posts.insertOne({Username:"ScumbagSteve",

title:"Forks your repo on github", body:"Sets to private"})

created comments collection

#1

db.comments.insertOne({Username:"GoodGuyGreg",

comment:"Hope you got a good deal!", post:[ObjectId("60034c792223f4164425ee22")]})

#2

db.comments.insertOne({Username:"GoodGuyGreg",

comment:"What's mine is yours", post:[ObjectId("60034c792223f4164425ee22")]})

#3

db.comments.insertOne({Username:"GoodGuyGreg",

comment:"Don't violate the licensing agreement", post:[ObjectId("60034c892223f4164425ee23")]})

#4

db.comments.insertOne({Username:"ScumbagSteve",

comment:"It still isn't clean", post:[ObjectId("60034b072223f4164425ee1e")]})

#5

db.comments.insertOne({Username:"ScumbagSteve",

comment:"Denied your PR cause i found a hack", post:[ObjectId("60034c5b2223f4164425ee20")]})

Querying related collections

1. find all users

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.comments.find({post:ObjectId("60034c5b2223f4164425ee20")).pretty()