MongoDB Lab Assignments -Day 1

MongoDB Exercise in mongo shell

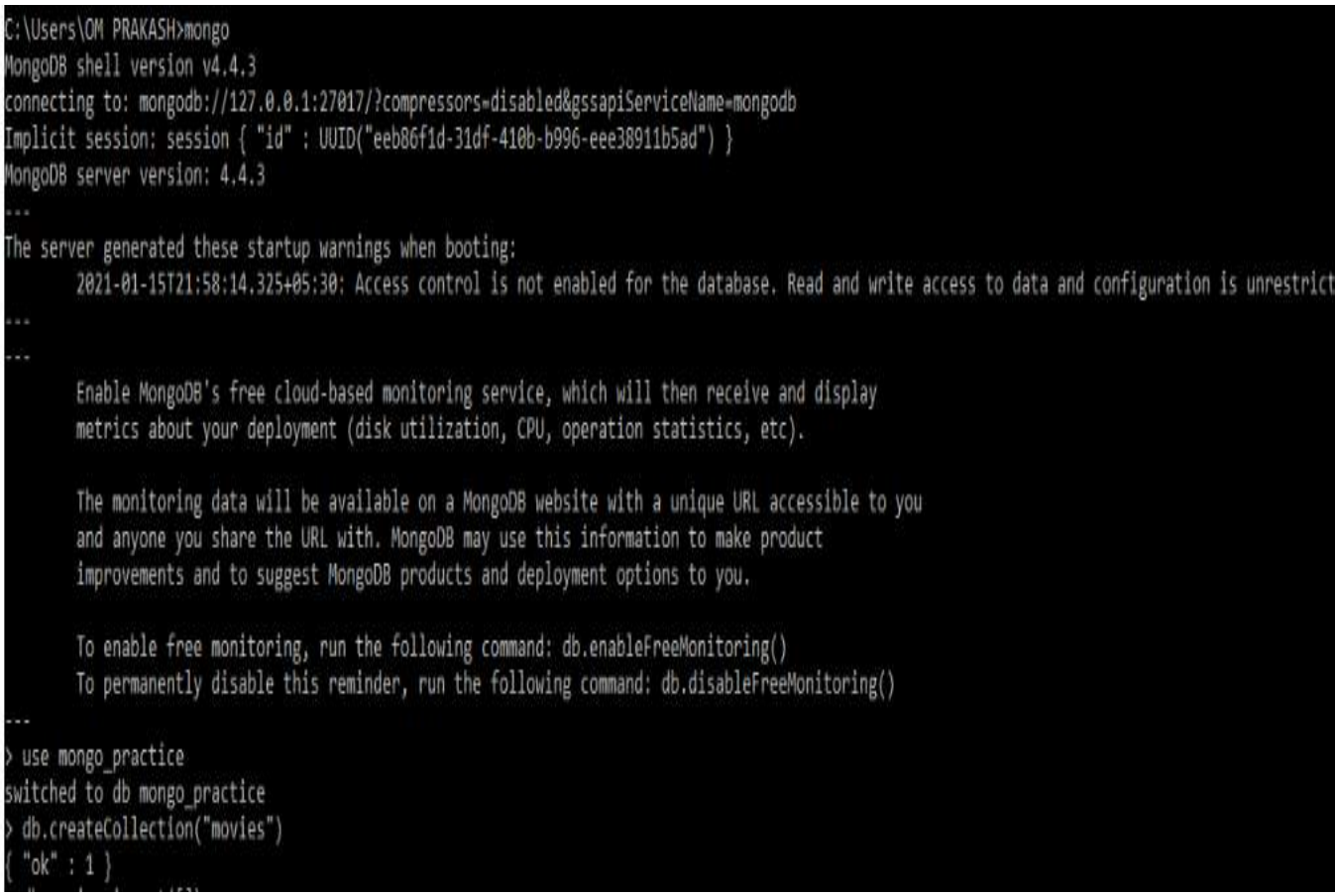
Connect to a running mongo instance, use a database named **mongo_practice**

Insert Documents

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



```
> db.movies.insert([
... {
... 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
hands of a rising darkness"
... },
... {
... title: "Pee Wee Herman's Big Adventure",
... title: "Avatar"
... }
... ])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 7,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
>
```

Query / Find Documents

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()
{
        "_id" : ObjectId("6003211a0b765fe585f992b7"),
        "title" : "Pulp Fiction",
        "writer" : "Quentin Tarantino",
        "year" : 1994,
        "actors" : [
                "John Travolta",
                "Uma Thurman"
        ]
}
{
        "_id" : ObjectId("6003211a0b765fe585f992b8"),
        "title" : "Inglorious Basterds",
        "writer" : "Quentin Tarantino",
        "year" : 2009,
        "actors" : [
                "Brad Pitt",
                "Diane Kruger",
                "Eli Roth"
        ]
}
>
```

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

```
> db.movies.find({actors: "Brad Pitt"}).pretty()
{
        "_id" : ObjectId("6003211a0b765fe585f992b6"),
        "title" : "Fight Club",
        "writer" : "Chuck Palahniuko",
        "year" : 1999,
        "actors" : [
                "Brad Pitt",
                "Edward Norton"
        ]
}
{
        "_id" : ObjectId("6003211a0b765fe585f992b8"),
        "title" : "Inglorious Basterds",
        "writer" : "Quentin Tarantino",
        "year" : 2009,
        "actors" : [
                "Brad Pitt",
                "Diane Kruger",
                "Eli Roth"
        ]
}
>
```

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

```
> db.movies.find({franchise: "The Hobbit"}).pretty()
{
        "_id" : ObjectId("6003211a0b765fe585f992b9"),
        "title" : "The Hobbit: An Unexpected Journey",
        "writer" : "J.R.R. Tolkein",
        "year" : 2012,
        "franchise" : "The Hobbit"
}
{
        "_id" : ObjectId("6003211a0b765fe585f992ba"),
        "title" : "The Hobbit: The Desolation of Smaug",
        "writer" : "J.R.R. Tolkein",
        "year" : 2013,
        "franchise" : "The Hobbit"
}
>
```

5. get all movies released in the 90s

```
> db.movies.find({$and: [{year:{$gt: 1990}},{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()
```

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.update({'title':'The Hobbit: An Unexpected Journey'},{$set:{'synopsis': 'A reluctant hobbit, Bilbo Baggins, sets to the Lonely Mountain with a sprit
 of dwarves to reclaim their mountain home - and the gold within it - form the dragon Smaug.'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

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.update({'title':'The Hobbit: The Desolation of Smaug'},{$set:{'synopsis': 'The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue the
to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring.'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

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

```
> db.movies.update({'title':'Pulp Fiction'},{$push:{'actors':'Samuel L. Jackson'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

Reference:
https://www.tutorialspoint.com/mongodb/mongodb_update_document.htm

Text Search

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

```
> db.movies.createIndex({synopsis:"text"})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
> db.movies.find({$text:{$search:"Bilbo"}}).pretty()
```

OR

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

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

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

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/}}]}).pretty()
```

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"}}]}).pretty()
```

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"}}]}).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.remove({title:"Pee Wee Herman's Big Adventure"})
WriteResult({ "nRemoved" : 1 })
```

2. delete the movie "Avatar"

```
> db.movies.remove({title:"Avatar"})
WriteResult({ "nRemoved" : 1 })
>
```

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"

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.insert([
... {
... 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"
... }])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 6,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
>
```

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"
username : ScumbagSteve
comment : Denied your PR cause I found a hack
post : [post_obj_id]

```
> db.createCollection("comments")
{ "ok" : 1 }
> db.comments.insert([{
... username: "GoodGuyGreg",
... comment: "Hope you got a good deal!",
... post: ObjectId("60046c7183f24cee901002d4"),
... },
... {
... username: "GoodGuyGreg",
... comment: "What's mine is yours!",
... post: ObjectId("60046c7183f24cee901002d5"),
... },
... {
... username: "GoodGuyGreg",
... comment: "Don't violate the licensing agreement!",
... post: ObjectId("60046c7183f24cee901002d6"),
... },
... {
... username: "ScumbagSteve",
... comment: "It still isn't clean",
... post: ObjectId("60046c7183f24cee901002d1"),
... },
... {
... username: "ScumbagSteve",
... comment: "Denied your PR cause I found a hack",
... post: ObjectId("60046c7183f24cee901002d3"),
... }])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 5,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
>
```

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.posts.find({'title': 'Reports a bug in your code'}).pretty()
```