Practical-1

- **Insert the following documents into a movies collection.**

```
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. Tolkien",
  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: 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"
```

}]);

1. Write a MongoDB query to get all documents.
   ➔ db.movies.find({})
2. Write a MongoDB query to get all documents with writer set to "Quentin Tarantino".
   ➔ db.movies.find({ writer: "Quentin Tarantino" })
3. Write a MongoDB query to get all documents where actors include "Brad Pitt"
   ➔ db.movies.find({ actors: { $in: ["Brad Pitt"] } })
4. Write a MongoDB query to get all documents with franchise set to "The Hobbit"
   ➔ db.movies.find({ franchise: "The Hobbit" })
5. Write a MongoDB query to get all movies released in the 90s
   ➔ db.movies.find({ year: { $gte: 1990, $lt: 2000 } })
6. Write a MongoDB query to get all movies released before the year 2000 or after 2010
   ➔ db.movies.find({$or: [{year: {$lt: 2000}}, {year: {$gt: 2010}}]})

Update Documents:

7. 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." } });

8. 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." } });

9. add an actor named "Samuel L. Jackson" to the movie "Pulp Fiction"
   ➔ db.movies.updateOne({ title: "Pulp Fiction" }, { $push: { actors: "Samuel L. Jackson" } });
10. Write a MongoDB query to find all movies that have a synopsis that contains the word "Bilbo"
    ➔ db.movies.find({ synopsis: { $regex: /Bilbo/i } })
11. Write a MongoDB query to find all movies that have a synopsis that contains the word "Gandalf"
    ➔ db.movies.find({ synopsis: { $regex: /Gandalf/i } })
12. Write a MongoDB query to find all movies that have a synopsis that contains the word "Bilbo" and not the word "Gandalf"
    ➔ db.movies.find({ synopsis: { $regex: /Bilbo/i, $not: { $regex: /Gandalf/i } } })
13. Write a MongoDB query to find all movies that have a synopsis that contains the word "dwarves" or "hobbit"
    ➔ db.movies.find({ synopsis: { $regex: /dwarves/i, $or: { $regex: /hobbit/i } } })
14. Write a MongoDB query to find all movies that have a synopsis that contains the word "gold" and "dragon"
    ➔ db.movies.find({ synopsis: { $regex: /gold/i, $regex: /dragon/i } })

Delete Documents:

15. delete the movie "Pee Wee Herman's Big Adventure"
    ➔ db.movies.deleteOne({ title: "Pee Wee Herman's Big Adventure" });
16. delete the movie "Avatar"

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

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.insertOne({ username: "GoodGuyGreg", first_name: "Good Guy", last_name: "Greg" });
➔ db.users.insertOne({ username: "ScumbagSteve", full_name: { first: "Scumbag", last: "Steve" }});

Create a collection name with posts and Insert the following documents.

Postid: 1
username: GoodGuy Greg
title: Passes out at party
body : Wakes up early and cleans house
Postid: 2
username: GoodGuyGreg
title: Steals your identity
body : Raises your credit score
Postid: 3
username: GoodGuy Greg
title : Reports a bug in your code
body: Sends you a Pull Request
Postid: 4
username: ScumbagSteve
title : Borrows something
body: Sells it
Postid: 5
username: ScumbagSteve
title : Borrows everything
body: The end
Postid: 6
username: ScumbagSteve
title : Forks your repo on github
body: Sets to private

➔ db.createCollection("posts");
➔ db.posts.insertMany([{ Postid: 1, username: "GoodGuyGreg", title: "Passes out at party", body: "Wakes up early and cleans house" }, { Postid: 2, username: "GoodGuyGreg", title: "Steals your identity", body: "Raises your credit score" }, { Postid: 3, username: "GoodGuyGreg", title: "Reports a bug in your code", body: "Sends you a Pull Request" }, { Postid: 4, username: "ScumbagSteve", title: "Borrows something", body: "Sells it" }, { Postid: 5, username: "ScumbagSteve", title: "Borrows everything", body: "The end" }, { Postid: 6, username: "ScumbagSteve", title: "Forks your repo on github", body: "Sets to private" }]);

Create a collection name with comments and Insert the following documents.

username: GoodGuy Greg
comment : Hope you got a good deal!
postid: 4
username: GoodGuyGreg

comment : What's mine is yours!
postid: 4
username: GoodGuy Greg
comment : Don't violate the licensing agreement!
postid: 4
username : ScumbagSteve
comment : It still isn't clean
postid: 1
username : ScumbagSteve
comment : Denied your PR cause I found a hack
postid: 3

- ➔ db.createCollection("comments");
- ➔ db.comments.insertMany([{ username: "GoodGuyGreg", comment: "Hope you got a good deal!", postid: 4 }, { username: "GoodGuyGreg", comment: "What's mine is yours!", postid: 4 }, { username: "GoodGuyGreg", comment: "Don't violate the licensing agreement!", postid: 4 }, { username: "ScumbagSteve", comment: "It still isn't clean", postid: 1 }, { username: "ScumbagSteve", comment: "Denied your PR cause I found a hack", postid: 3 },]);

Now perform the below queries against users, posts, and comments dataset:

Query Documents:

17. Write a MongoDB query to find all users
    - ➔ db.users.find({})
18. Write a MongoDB query to find all posts
    - ➔ db.users.find({})
19. Write a MongoDB query to find all posts that was authored by "GoodGuyGreg"
    - ➔ db.posts.find({ username: "GoodGuyGreg" })
20. Write a MongoDB query to find all posts that was authored by "ScumbagSteve"
    - ➔ db.posts.find({ username: "ScumbagSteve" })
21. Write a MongoDB query to find all comments
    - ➔ db.posts.find({ _id: 1 }, { comments: 1 })
22. Write a MongoDB query to find all comments that was authored by "GoodGuyGreg"
    - ➔ db.comments.find({ username: "GoodGuyGreg" })
23. Write a MongoDB query to find all comments that was authored by "ScumbagSteve"
    - ➔ db.comments.find({ username: "ScumbagSteve" }).sort({ _id: -1 }).limit(5)
24. Write a MongoDB query to find all comments belonging to the post "Reports a bug in your code"
    - ➔ db.posts.find({ title: "Reports a bug in your code" })