MongoDB Assignment - 1

Insert Documents

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

db.movies.insertMany([{data 1},{data 2},…])

Here data 1, data 2 …. Data n are the records that we wanted to enter. Array of documents.

db.movies.insertMany(

```
[ {
    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"
    }
])
```

Q1. 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({},{title:"$title",year:"$year",actors:"$actors",writer:"Quentin Tarantino"}).pretty()

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

db.movies.find({},{title:"$title",year:"$year",actors:"Brad Pitt",writer:"$writr"}).pretty()

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

db.movies.find({},{title:"$title",year:"$year",actors:"$actors",writer:"$writer",franchise:"The Hobbit"}).pretty()

5. get all movies released in the 90s

db.movies.find({},{title:"$title",year:{$lt:["$year",2000]},actors:"$actors",writer:"$writer",franchise:"$franchise"}).pretty()

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

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

Q2. 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 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. 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 their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."}})

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

db.movies.update({title:"Pulp Fiction"},{$push:{actors:"Samuel L. Jackson"}})

Q3. Text Search

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

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

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

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

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/}}]})
```

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"}}]})
```

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"}}]})
```

Q4. 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

Q5. Querying related collections

1. **find all users**

```
db.users.find()
```

2. **find all posts**

```
db.posts.find()
```

3. **find all posts that was authored by "GoodGuyGreg"**

```
db.posts.find({username:"GoodGuyGreg"})
```

4. **find all posts that was authored by "ScumbagSteve"**

```
db.posts.find({username:" ScumbagSteve "})
```

5. **find all commentsdb.comments.find()**
6. **find all comments that was authored by "GoodGuyGreg"**

```
db.comments.find({username:"GoodGuyGreg"})
```

7. **find all comments that was authored by "ScumbagSteve"**

```
db.comments.find({username:"ScumbagSteve "})
```

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"
   }}
])
```