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

db.movies.insert({title:"Fight Club",writer:"Chuck Palahniuko",year:1999,actors:["John Travolta","Uma Thurman"]})

db.movies.insert({title:"Pulp Fiction",writer:"Quentin Tarantino",year:1994,actors:["Brad Pitt","Diane Kruger","Eli Roth"]})

db.movies.insert({title:"Inglorious Basterds",writer:"Quentin Tarantino",year:2004,actors:["Brad Pitt","Diane Kruger","Eli Roth"]})

db.movies.insert({title:"The Hobbit: An Unexpected Journey",writer:"J.R.R. Tolkein",year:2012,franchise:"The Hobbit"})

db.movies.insert({title:"The Hobbit: The Desolation of Smaug",writer:"J.R.R. Tolkein",year:2013,franchise:"The Hobbit"})

db.movies.insert({title:"The Hobbit: The Battle of the Five Armies",writer:"J.R.R. Tolkein",year:2012,franchise:"The Hobbit",synopsis:"Biblo 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 raising darkness."})

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

**Query / Find Documents**

query the movies collection to

1.get all documents

db.movies.find()

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

db.movies.find({"Quentin Tarantino"})

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

db.movies.find({actors:"Brad Pitt"})

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

db.movies.find({franchise:"The Hobbit"})

5. get all movies released in the 90s 6. get all movies released before the year 2000 or after 2010

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

**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.updateOne({title:"The Hobbit: An Unexpected Journey"},{$set:{synopsis:"A reluctant hobbit,Bilbo Baggins, sets out to the Lonely Mountain with 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.updateOne({title:"The Hobbit: The Desolation of Smaug"},{$set:{synopsis:"The dwarves, along with Bilbo Baggings and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggings 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 "}}

**Text Search**

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

db.movies.createIndex({“synopsis”:”text”})

db.movies.find({$text: {$search:"Bilbo"}},{title:1})

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

db.movies.find({$text: {$search:"Gandalf"}},{title:1})

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

({$text: {$search:"Bilbo -Gandalf"}},{title:1})

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

db.movies.find({$text: {$search:"dwarves hobbit"}},{title:1})

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

db.movies.find({$text:{$search:"\"gold\"\"dragon\""}},{title:1})

**Delete Documents**

1.delete the movie "Pee Wee Herman's Big Adventure"

db.movies.deleteOne({title:"Pee Wee Herman's Big Adventure"}

2. delete the movie "Avatar"

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

**Relationships**

Insert the following documents into a **users** collection

db.users.insert({username:"GoodGuyGreg" , first\_name:"Good Guy" , last\_name:"Greg"})

db.users.insert({username:"ScumbagSteve",full\_name:{first:"Scumbag",last:"Steve"}})

Insert the following documents into a **posts** collection

db.posts.insert({username:"GoodGuyGreg" , title:"Passes out a party" , body:"Wakes up early and cleans house"})

db.posts.insert({username:"GoodGuyGreg" , title:"Steals your identity" , body:"Raises your credit score"})

db.posts.insert({username:"GoodGuyGreg" , title:"Reports a bug in your code" , body:"Sends you a Pull Request"})

db.posts.insert({username:"ScumbagSteve" , title:"Borrows something" , body:"Sells it"})

db.posts.insert({username:"ScumbagSteve" , title:"Borrows everything" , body:"The end"})

db.posts.insert({username:"ScumbagSteve" , title:"Forks your repo on github" , body:"sets to private"})

Insert the following documents into a comments collection

db.comments.insert({username:"GoodGuyGerg",comment:"Hope you got a good deal!",post:ObjectId("61efa5787e62854e4fade79e")})

db.comments.insert({ username: "GoodGuyGerg", comment: "What's mine is yours!", post: ObjectId("61efa5917e62854e4fade79f") })

db.comments.insert({ username: "GoodGuyGreg", comment: "Don't violate the licensing agreement!", post: ObjectId("61efa5bb7e62854e4fade7a0") })

db.comments.insert({ username: "ScumbagSteve", comment: "It still isn't clean", post: ObjectId("61ef99677e62854e4fade799") })

db.comments.insert({ username: "ScumbagSteve", comment: "Denied your PR cause I found a hack", post: ObjectId("61efa5367e62854e4fade79d") })

**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 comments

db.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.find({title:"Reports a bug in your code"})

db.comments.find({post:ObjectId("61efa5367e62854e4fade79d")})