**Inserting the data into Database named monogo\_practice inside collection named movies**

To create database name monogo\_practice

User mongo\_practice

To create collection name movies

Db.createCollection(“movies”)

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"}])

**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({writer:"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

db.movies.find ({$and: [{year: {$gt: 1990}}, {year: {$lt: 2000}}]})

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.update({title:'The Hobbit: An Unexpected Journey'},{$set: {synopsis: 'A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited roup 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 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'}})

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:'Bilbo -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.deleteOne({title: "Pee Wee Herman's Big Adventure"})

2. delete the movie "Avatar"

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

**Relationships**

Creating collection ‘users’

db.createCollection('users')

Inserting document into users collection

db.users.insertOne({username : "GoodGuyGreg", first\_name : "Good Guy", last\_name : "Greg", username : "ScumbagSteve", full\_name :{first : "Scumbag", last : "Steve"}})

Creating collection ‘posts’

Db.createCollection(‘posts’)

Inserting document into posts collection

db.posts.insertMany([{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"}])

**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.comments.find({post : "617ab363724fe21946c5af64"})