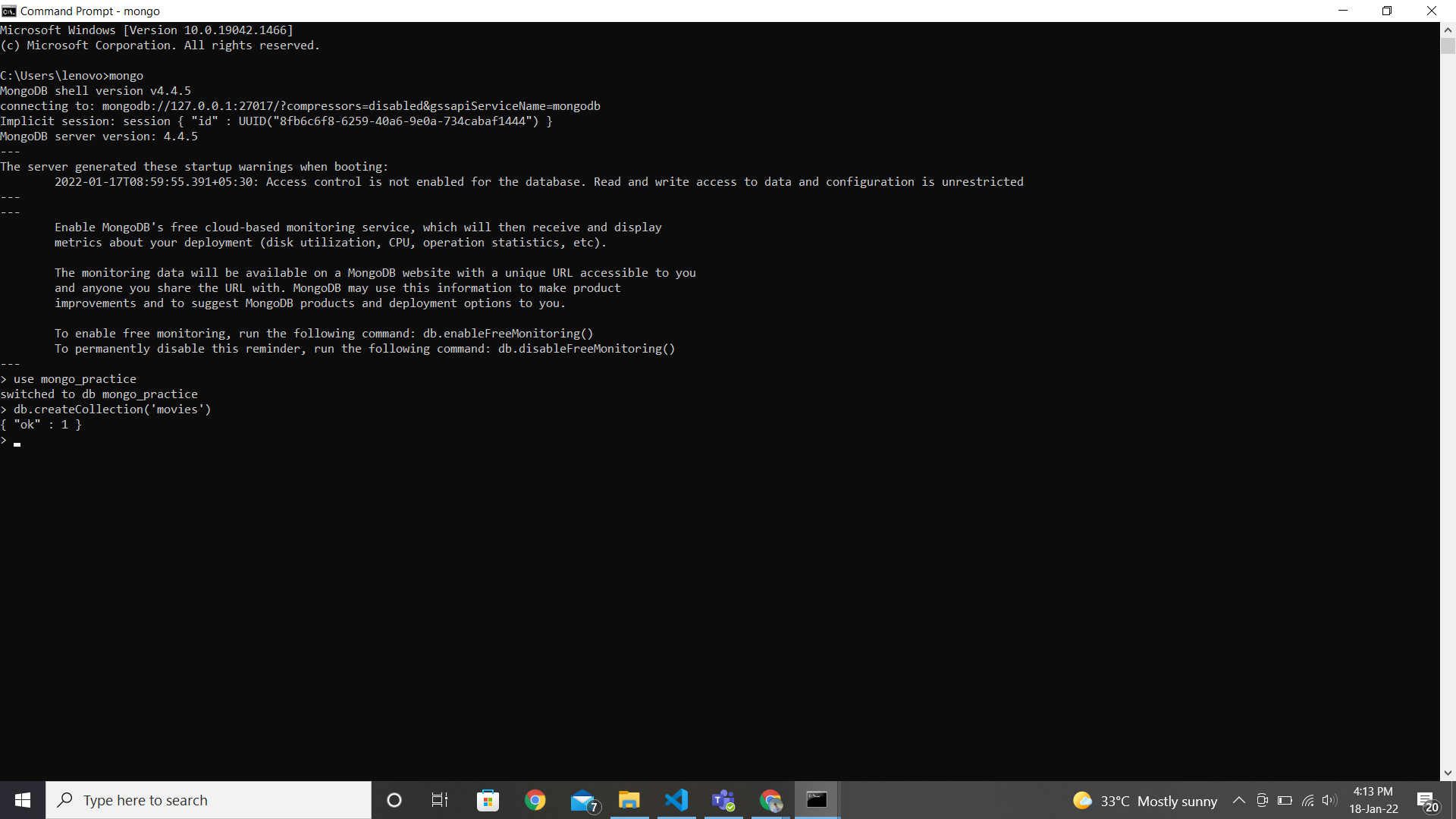
MongoDB Lab Assignments -Day 1

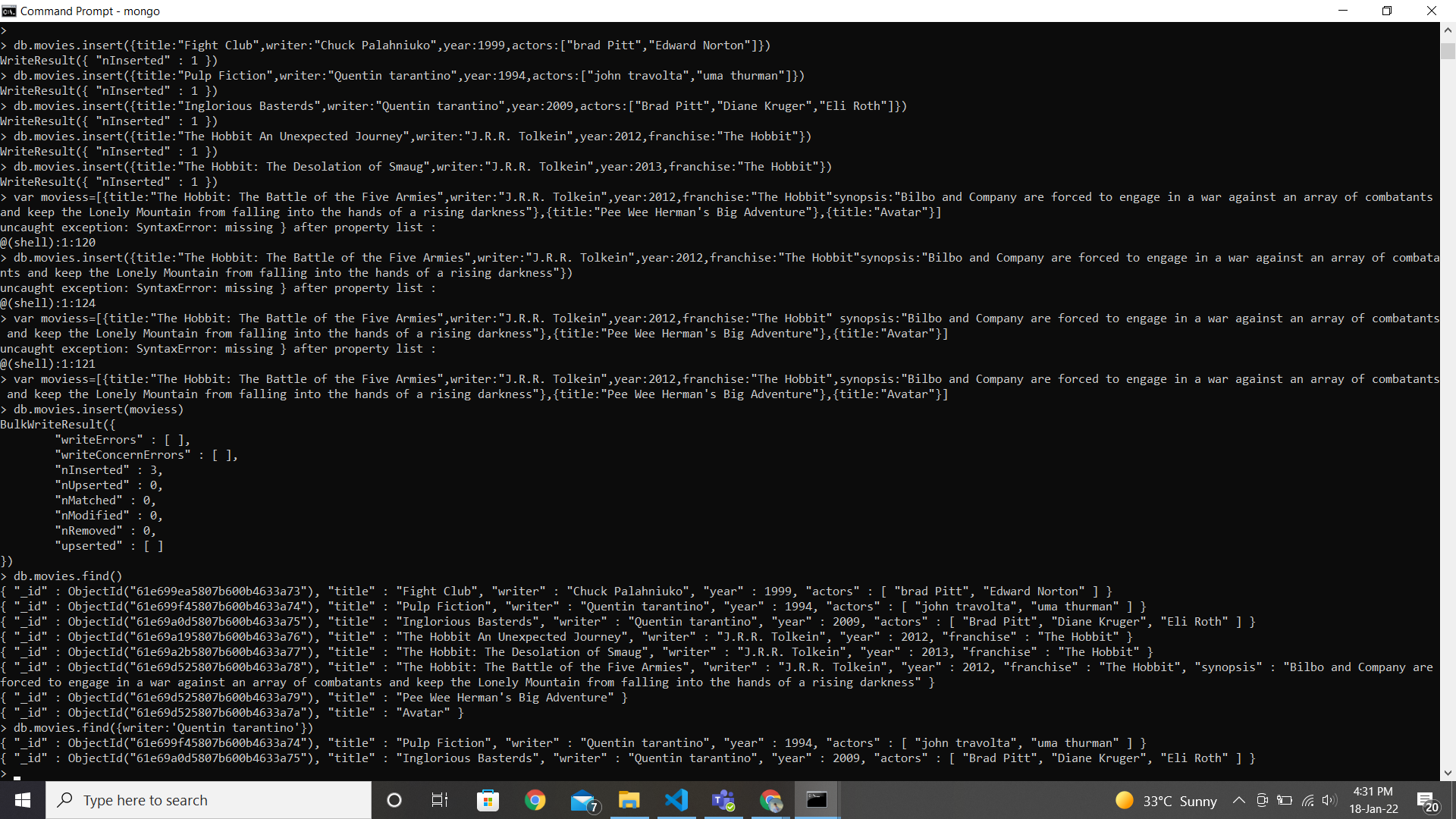
MongoDB Exercise in mongo shell

Connect to a running mongo instance, use a database named mongo\_practice.

* use mongo\_practice



insert into movies

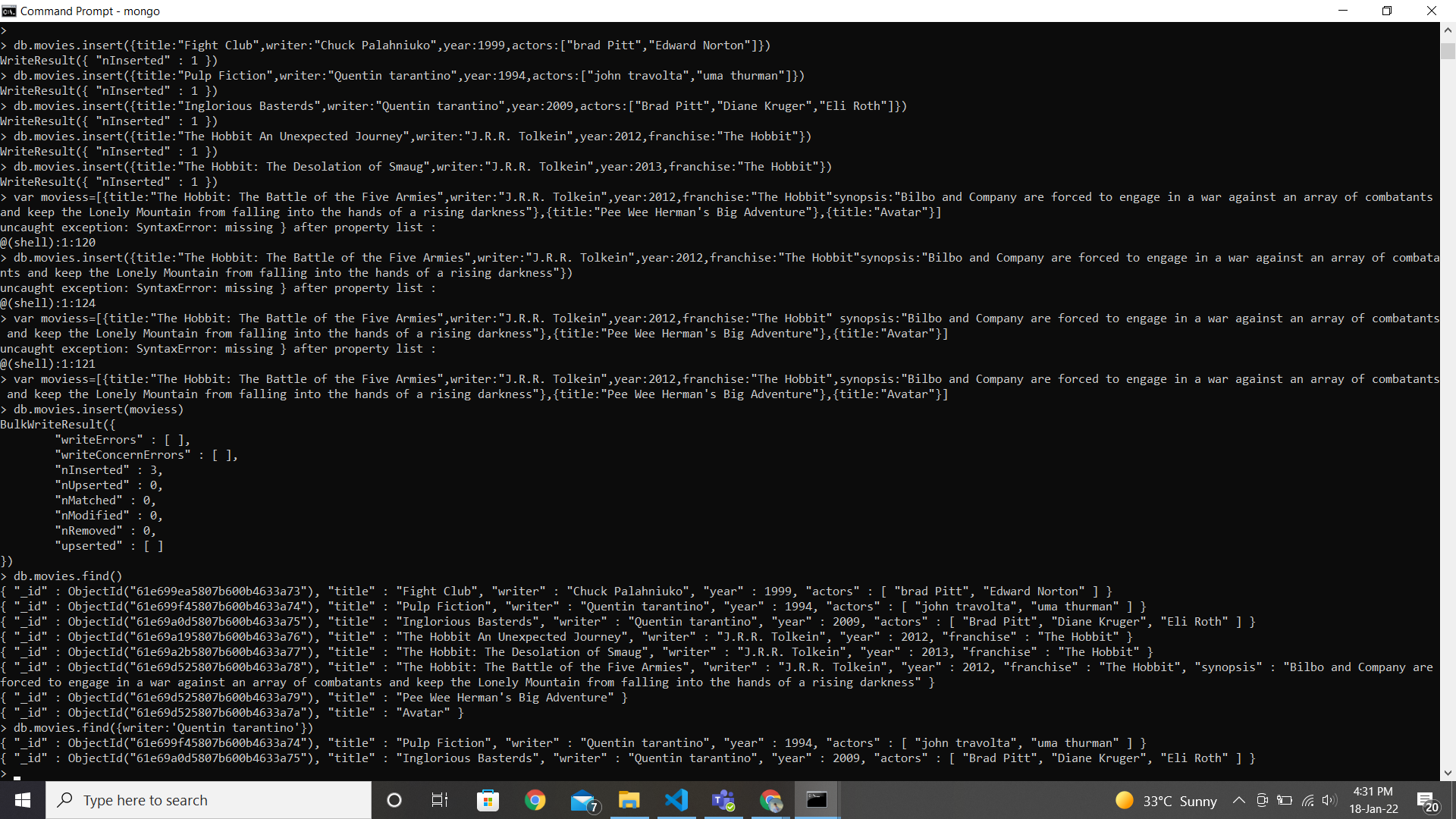


**Query / Find Documents**

**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$/i})

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: 1900}}, {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 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"}})

**Text Search**

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

* db.movies.find({ synopsis: /Bilbo/g }).pretty();

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

* db.movies.find({ synopsis: /Gandal/g }).pretty();

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

* db.movies.find({ $and: [{ synopsis: /Bilbo/g }, { synopsis: { $not: /Gandalf/g } }] }).pretty();

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

* db.movies.find({ $or: [ {synopsis: /dwarves/g}, {synopsis: /hobbit/g} ] }).pretty();

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

* db.movies.find({ $and: [ {synopsis: /gold/g}, {synopsis: /dragon/g} ] }).pretty();

**Delete Documents**

1.Delete "Pee Wee Herman's Big Adventure

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

2.Delete Avatar

* db.movies.remove({title:"Avatar"})

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

* db.users.insertMany([ { 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**

* 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"} ]);

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

* db.comments.insertMany([ { username: "GoodGuyGreg", comment: "Hope you got a good deal!", post: ObjectId("61e6d587946787c3d6403adb") }, { username: "GoodGuyGreg", comment: "What's mine is yours!", post: ObjectId("61e6d587946787c3d6403adc") }, { username: "GoodGuyGreg", comment: "Don't violate the licensing agreement!", post: ObjectId("61e6d587946787c3d6403add") }, { username: "ScumbagSteve", comment: "It still isn't clean", post: ObjectId("61e6d587946787c3d6403ad8") }, { username: "ScumbagSteve", comment: "Denied your PR cause I found a hack", post: ObjectId("61e6d587946787c3d6403ada") ]);

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.aggregate([

{

$match: { title: 'Reports a bug in your code' }

},

{ $lookup: {

from: 'comments',

localField: '\_id',

foreignField: 'post',

as: 'comments'} }]).pretty();