**MongoDB Lab Assignments**

# MongoDB Exercise in mongo shell

Connect to a running mongo instance, use a database named **mongo\_practice**. Document all your queries in a javascript file to use as a reference.

> show dbs

admin 0.000GB

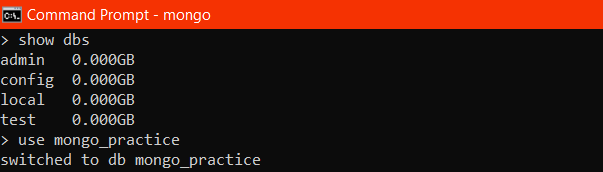
config 0.000GB

local 0.000GB

test 0.000GB

> use mongo\_practice

switched to db mongo\_practice



# Insert Documents

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

|  |
| --- |
| 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  Reference https://www.tutorialspoint.com/mongodb/mongodb\_insert\_document.html |

> 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. Tolkien",

... "year":2012,

... "franchise":"The Hobbit",

... "synopsis":"Bilbo and Company are forced to engae in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of rising darkness."

... },

... {

... "title":"Pee Wee Herman's Big Adventure"

... },

... {

... "title":"Avatar"

... }

... ])

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("6200c7fefe7c8f93ae7321b2"),

ObjectId("6200c7fefe7c8f93ae7321b3"),

ObjectId("6200c7fefe7c8f93ae7321b4"),

ObjectId("6200c7fefe7c8f93ae7321b5"),

ObjectId("6200c7fefe7c8f93ae7321b6"),

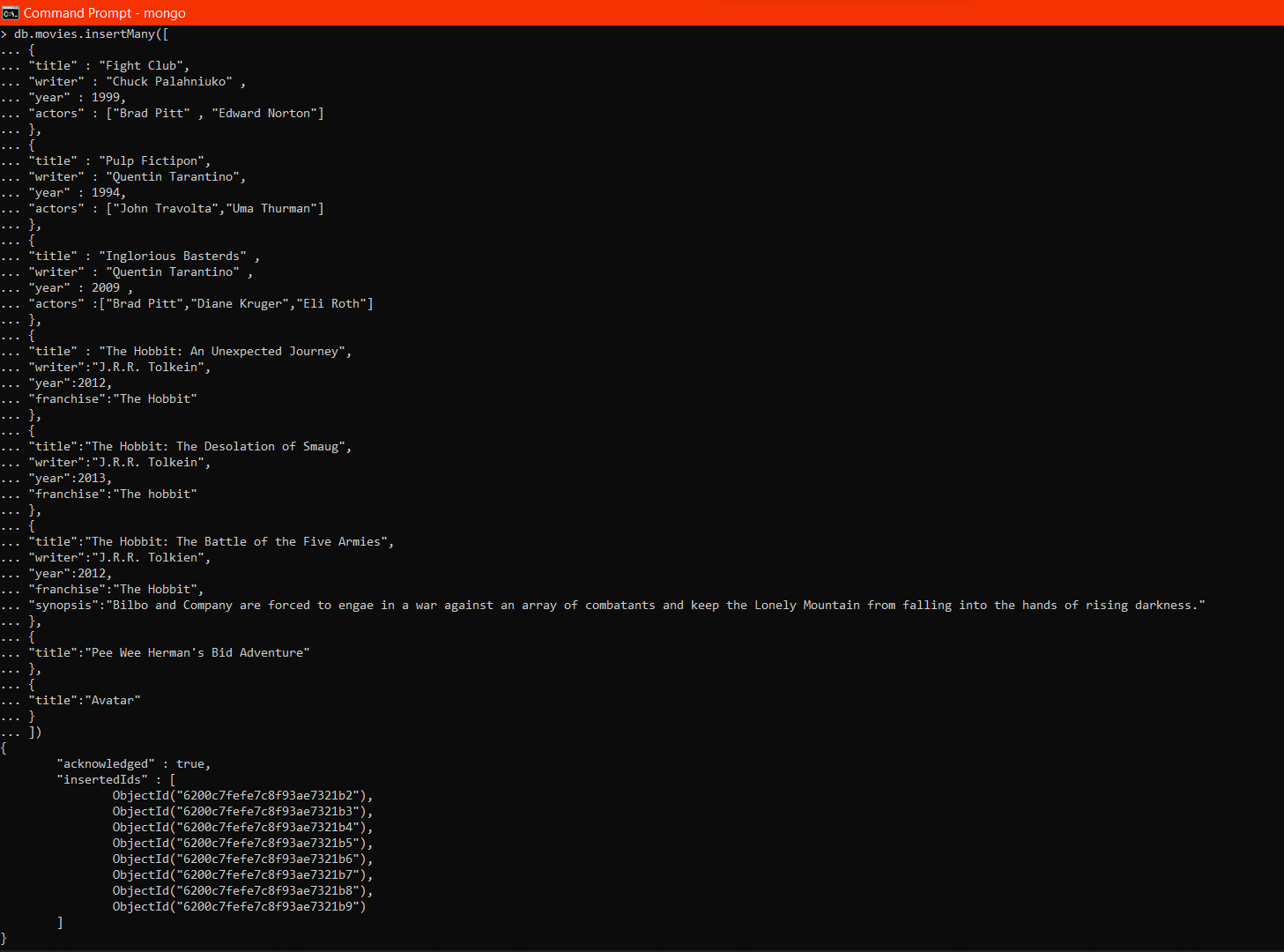
ObjectId("6200c7fefe7c8f93ae7321b7"),

ObjectId("6200c7fefe7c8f93ae7321b8"),

ObjectId("6200c7fefe7c8f93ae7321b9")

]

}



# Query / Find Documents

query the **movies** collection to

1.get all documents

> db.movies.find().pretty()

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b2"),

"title" : "Fight Club",

"writer" : "Chuck Palahniuko",

"year" : 1999,

"actors" : [

"Brad Pitt",

"Edward Norton"

]

}

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b3"),

"title" : "Pulp Fictipon",

"writer" : "Quentin Tarantino",

"year" : 1994,

"actors" : [

"John Travolta",

"Uma Thurman"

]

}

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b4"),

"title" : "Inglorious Basterds",

"writer" : "Quentin Tarantino",

"year" : 2009,

"actors" : [

"Brad Pitt",

"Diane Kruger",

"Eli Roth"

]

}

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b5"),

"title" : "The Hobbit: An Unexpected Journey",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit"

}

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b6"),

"title" : "The Hobbit: The Desolation of Smaug",

"writer" : "J.R.R. Tolkein",

"year" : 2013,

"franchise" : "The hobbit"

}

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b7"),

"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 engae in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of rising darkness."

}

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b8"),

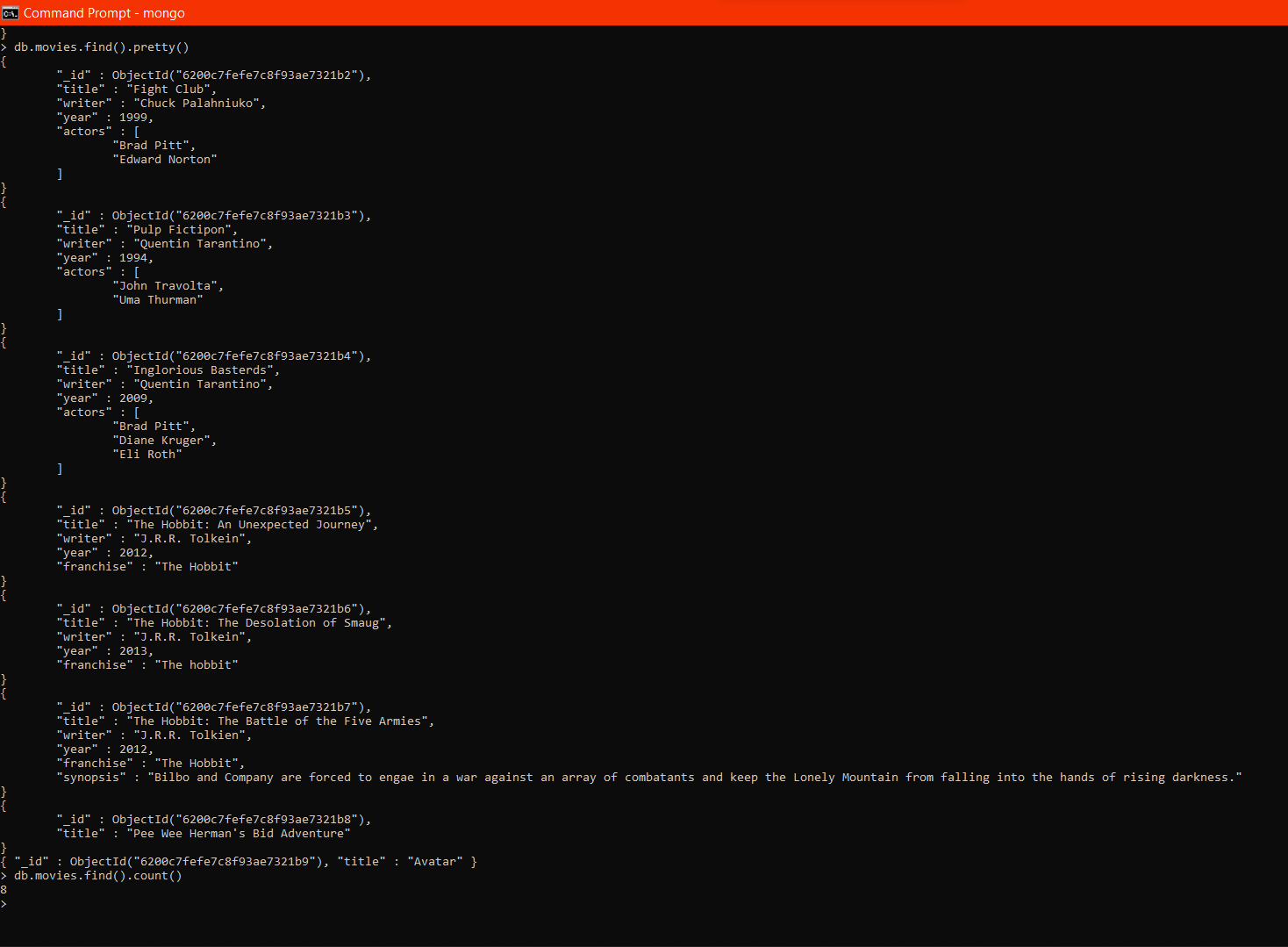
"title" : "Pee Wee Herman's Bid Adventure"

}

{ "\_id" : ObjectId("6200c7fefe7c8f93ae7321b9"), "title" : "Avatar" }

> db.movies.find().count()

8



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

> db.movies.find({"writer":"Quentin Tarantino"}).pretty()

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b3"),

"title" : "Pulp Fictipon",

"writer" : "Quentin Tarantino",

"year" : 1994,

"actors" : [

"John Travolta",

"Uma Thurman"

]

}

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b4"),

"title" : "Inglorious Basterds",

"writer" : "Quentin Tarantino",

"year" : 2009,

"actors" : [

"Brad Pitt",

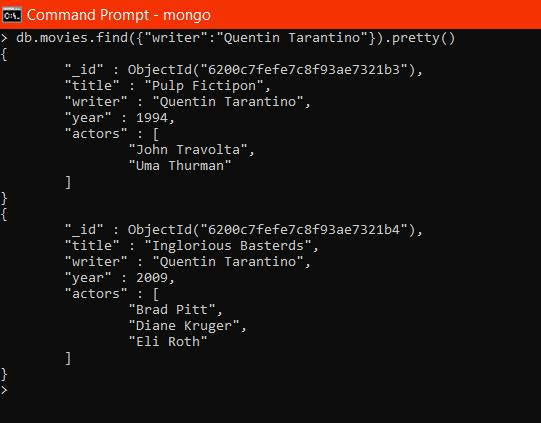
"Diane Kruger",

"Eli Roth"

]

}

>



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

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

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b2"),

"title" : "Fight Club",

"writer" : "Chuck Palahniuko",

"year" : 1999,

"actors" : [

"Brad Pitt",

"Edward Norton"

]

}

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b4"),

"title" : "Inglorious Basterds",

"writer" : "Quentin Tarantino",

"year" : 2009,

"actors" : [

"Brad Pitt",

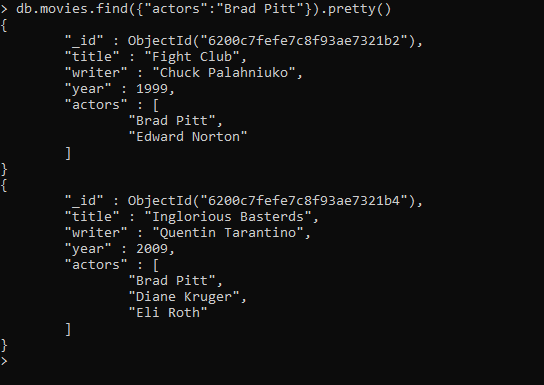
"Diane Kruger",

"Eli Roth"

]

}

>



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

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

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b5"),

"title" : "The Hobbit: An Unexpected Journey",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit"

}

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b7"),

"title" : "The Hobbit: The Battle of the Five Armies",

"writer" : "J.R.R. Tolkien",

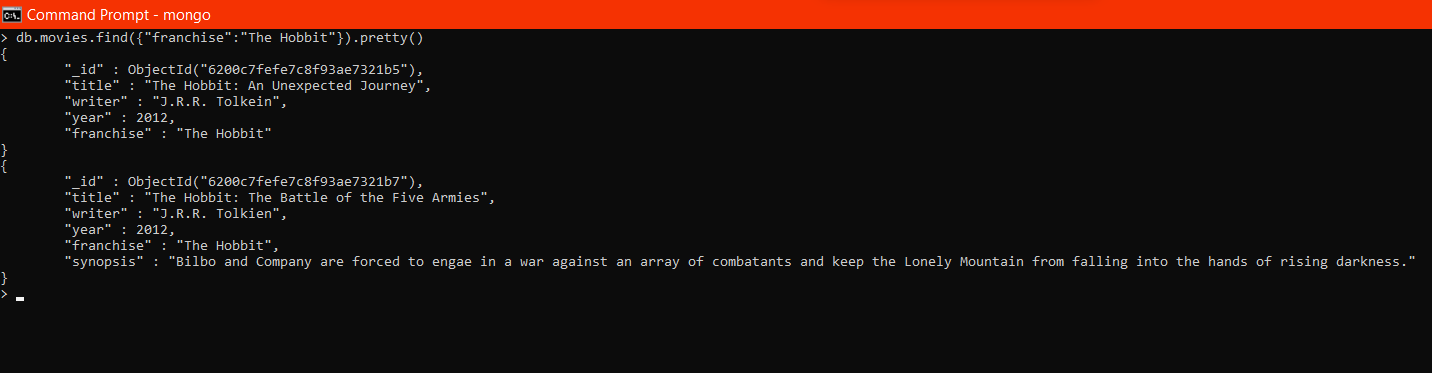
"year" : 2012,

"franchise" : "The Hobbit",

"synopsis" : "Bilbo and Company are forced to engae in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of rising darkness."

}

>



5.get all movies released in the 90s

> db.movies.find({"year":{$gte:1990,$lt :2000}}).pretty()

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b2"),

"title" : "Fight Club",

"writer" : "Chuck Palahniuko",

"year" : 1999,

"actors" : [

"Brad Pitt",

"Edward Norton"

]

}

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b3"),

"title" : "Pulp Fictipon",

"writer" : "Quentin Tarantino",

"year" : 1994,

"actors" : [

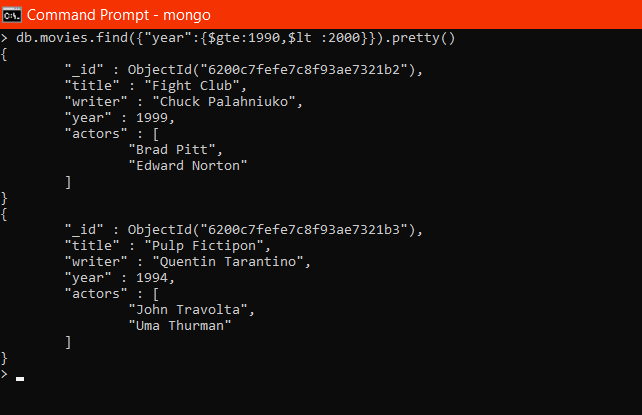
"John Travolta",

"Uma Thurman"

]

}

>



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

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

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b2"),

"title" : "Fight Club",

"writer" : "Chuck Palahniuko",

"year" : 1999,

"actors" : [

"Brad Pitt",

"Edward Norton"

]

}

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b3"),

"title" : "Pulp Fictipon",

"writer" : "Quentin Tarantino",

"year" : 1994,

"actors" : [

"John Travolta",

"Uma Thurman"

]

}

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b5"),

"title" : "The Hobbit: An Unexpected Journey",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit"

}

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b6"),

"title" : "The Hobbit: The Desolation of Smaug",

"writer" : "J.R.R. Tolkein",

"year" : 2013,

"franchise" : "The hobbit"

}

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b7"),

"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 engae in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of rising darkness."

}



Reference:

https://www.tutorialspoint.com/mongodb/mongodb\_query\_document.htm

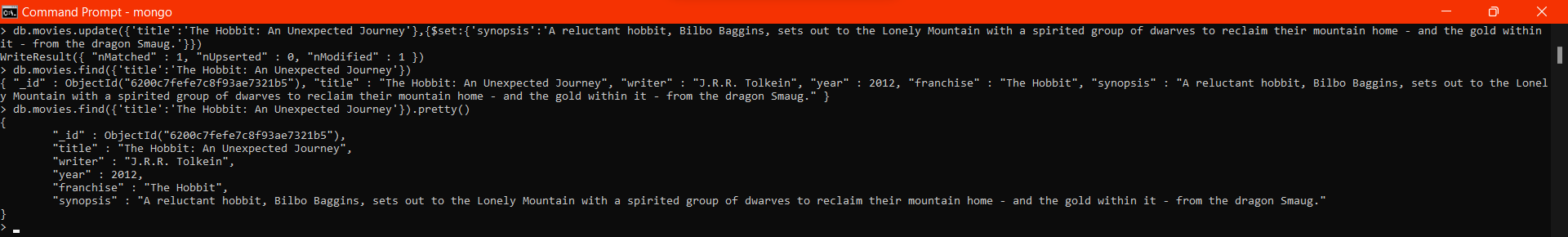
# Update Documents

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.'}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

>



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.'}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.movies.find({'title':'The Hobbit: The Desolation of Smaug'}).pretty()

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b6"),

"title" : "The Hobbit: The Desolation of Smaug",

"writer" : "J.R.R. Tolkein",

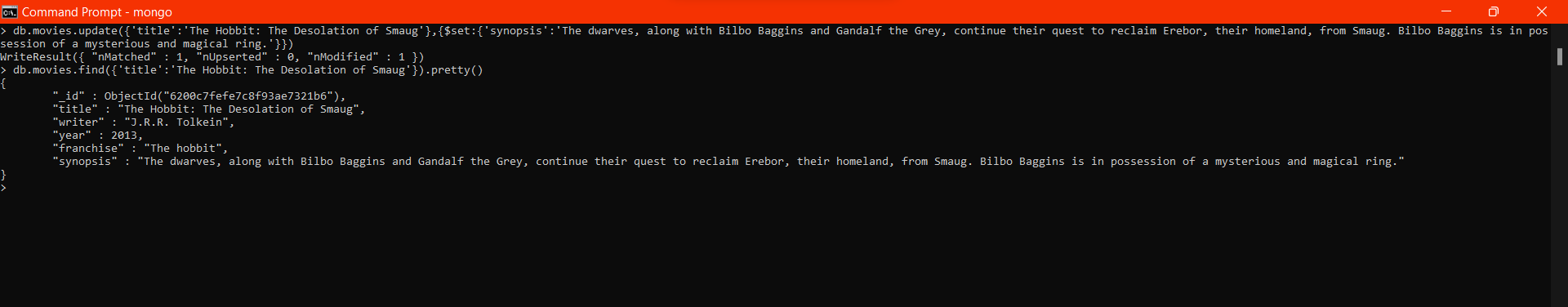
"year" : 2013,

"franchise" : "The hobbit",

"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."

}

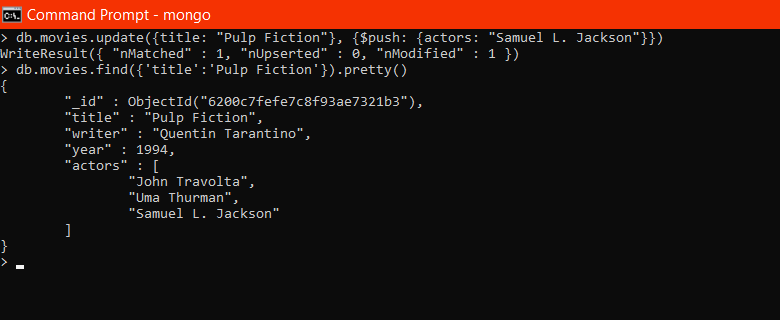
>



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

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

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })



Reference:

https://www.tutorialspoint.com/mongodb/mongodb\_update\_document.htm

# Text Search

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

> db.movies.createIndex({'synopsis':'text'})

{

"numIndexesBefore" : 1,

"numIndexesAfter" : 2,

"createdCollectionAutomatically" : false,

"ok" : 1

}

> db.movies.find({$text: {$search: "Bilbo"}}).pretty()

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b6"),

"title" : "The Hobbit: The Desolation of Smaug",

"writer" : "J.R.R. Tolkein",

"year" : 2013,

"franchise" : "The hobbit",

"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."

}

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b7"),

"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 engae in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of rising darkness."

}

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b5"),

"title" : "The Hobbit: An Unexpected Journey",

"writer" : "J.R.R. Tolkein",

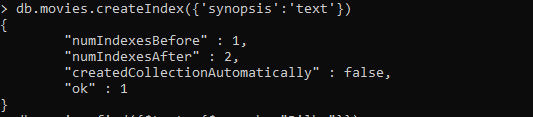
"year" : 2012,

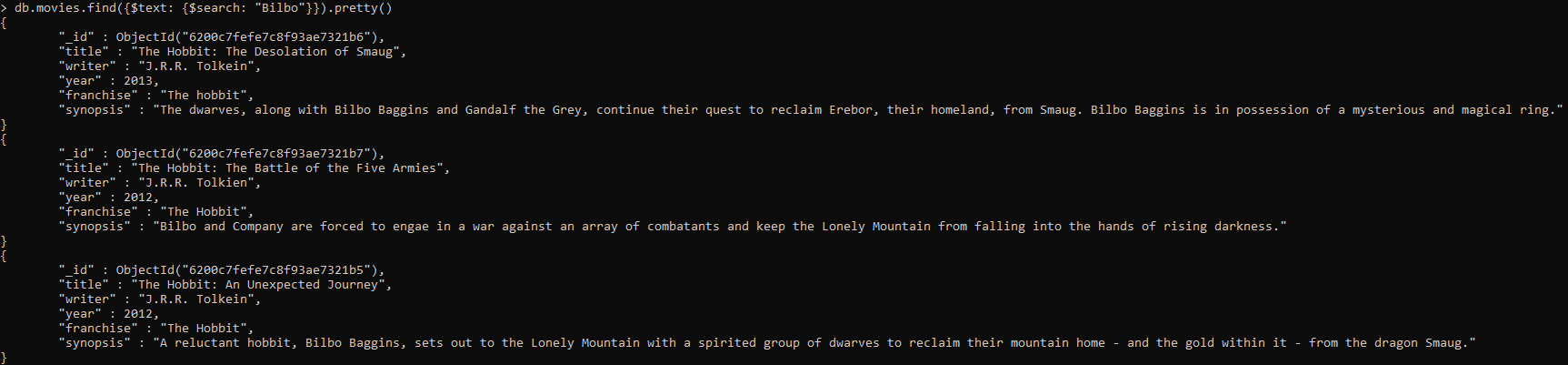
"franchise" : "The Hobbit",

"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."

}

>





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

> db.movies.find({$text: {$search: "Gandalf"}}).pretty()

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b6"),

"title" : "The Hobbit: The Desolation of Smaug",

"writer" : "J.R.R. Tolkein",

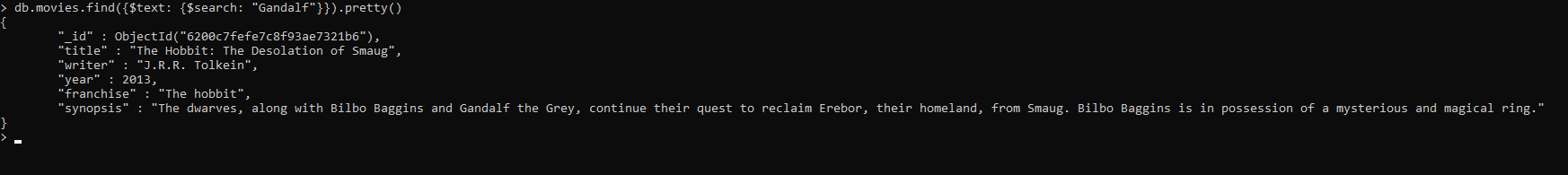
"year" : 2013,

"franchise" : "The hobbit",

"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."

}

>



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

> db.movies.find({$text: {$search: "Bilbo -Gandalf"}}).pretty()

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b7"),

"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 engae in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of rising darkness."

}

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b5"),

"title" : "The Hobbit: An Unexpected Journey",

"writer" : "J.R.R. Tolkein",

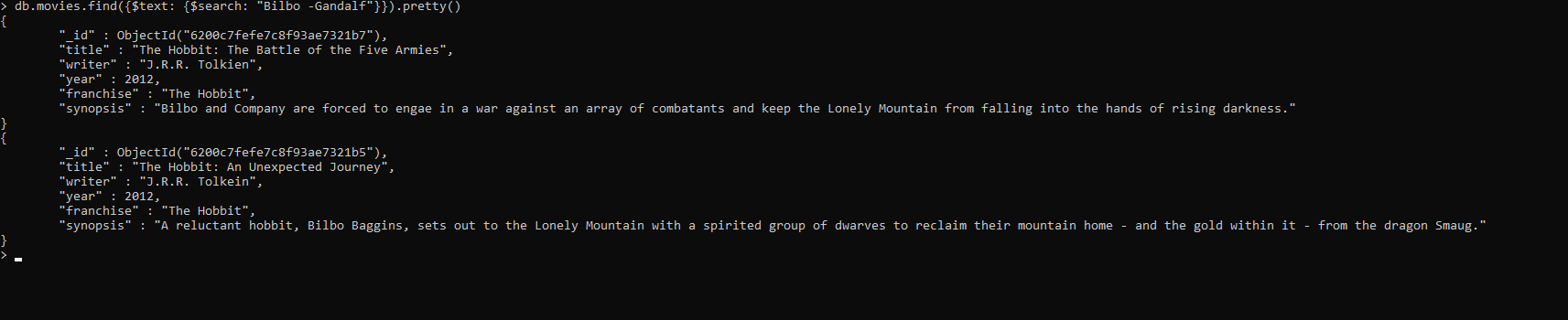
"year" : 2012,

"franchise" : "The Hobbit",

"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."

}

>



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

> db.movies.find({$text: {$search: "dwarves hobbit"}}).pretty()

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b5"),

"title" : "The Hobbit: An Unexpected Journey",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit",

"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."

}

{

"\_id" : ObjectId("6200c7fefe7c8f93ae7321b6"),

"title" : "The Hobbit: The Desolation of Smaug",

"writer" : "J.R.R. Tolkein",

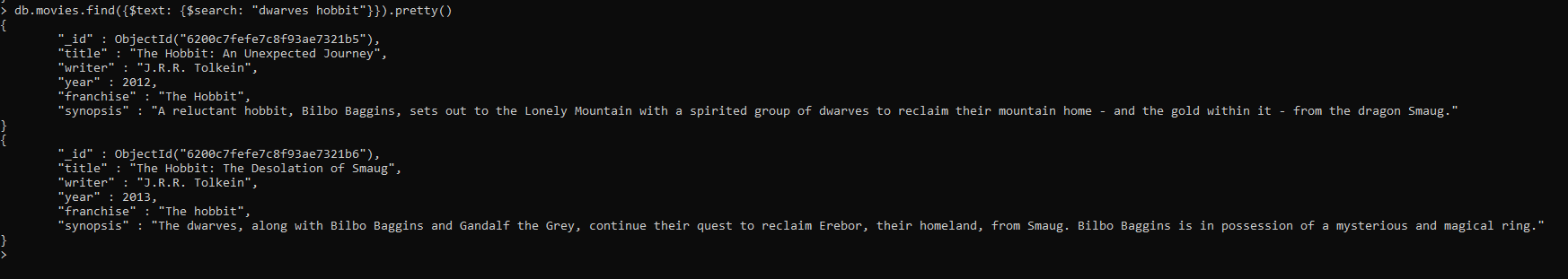
"year" : 2013,

"franchise" : "The hobbit",

"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."

}

>



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

> db.movies.find({$text:{$search:'"gold" "dragon"'}}).pretty()

{

"\_id" : ObjectId("620317aa3a8794ff0d6c5b66"),

"title" : "The Hobbit: An Unexpected Journey",

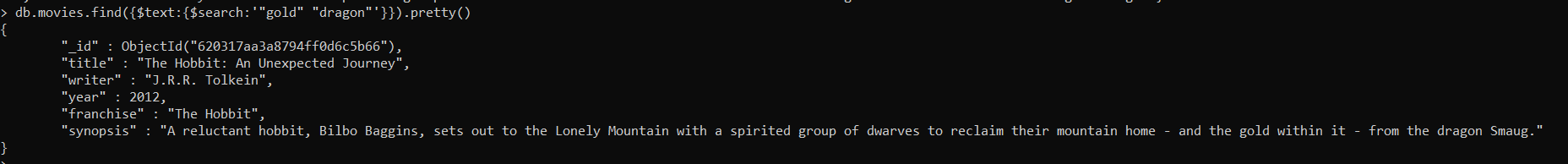
"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit",

"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."

}



Reference: https://www.tutorialspoint.com/mongodb/mongodb\_text\_search.html

# Delete Documents

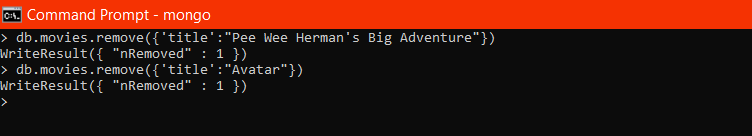
1. delete the movie "Pee Wee Herman's Big Adventure"
2. delete the movie "Avatar"

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

WriteResult({ "nRemoved" : 1 })

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

WriteResult({ "nRemoved" : 1 })



Reference: https://www.tutorialspoint.com/mongodb/mongodb\_delete\_document.htm

# Relationships

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.insertMany(

... [

... {

... "username" : "GoodGuyGreg" ,

... "first\_name" : "Good Guy" ,

... "last\_name" : "Greg" ,

... },

... {

... "username" : "ScumbagSteve",

... "full\_name" : [

... {"first" : "Scumbag"},

... {"last" : "Steve"}

... ]

... }

... ]

... )

{

"acknowledged" : true,

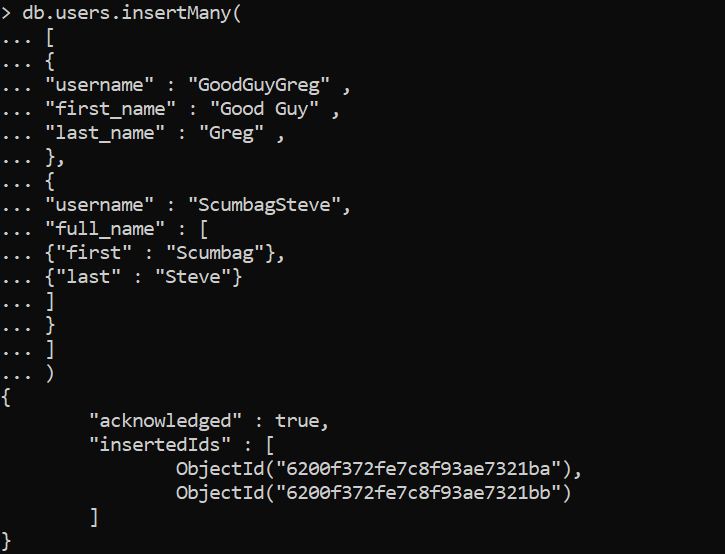
"insertedIds" : [

ObjectId("6200f372fe7c8f93ae7321ba"),

ObjectId("6200f372fe7c8f93ae7321bb")

]

}



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

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 |

> 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"

... }

... ])

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("6200f4f1fe7c8f93ae7321bc"),

ObjectId("6200f4f1fe7c8f93ae7321bd"),

ObjectId("6200f4f1fe7c8f93ae7321be"),

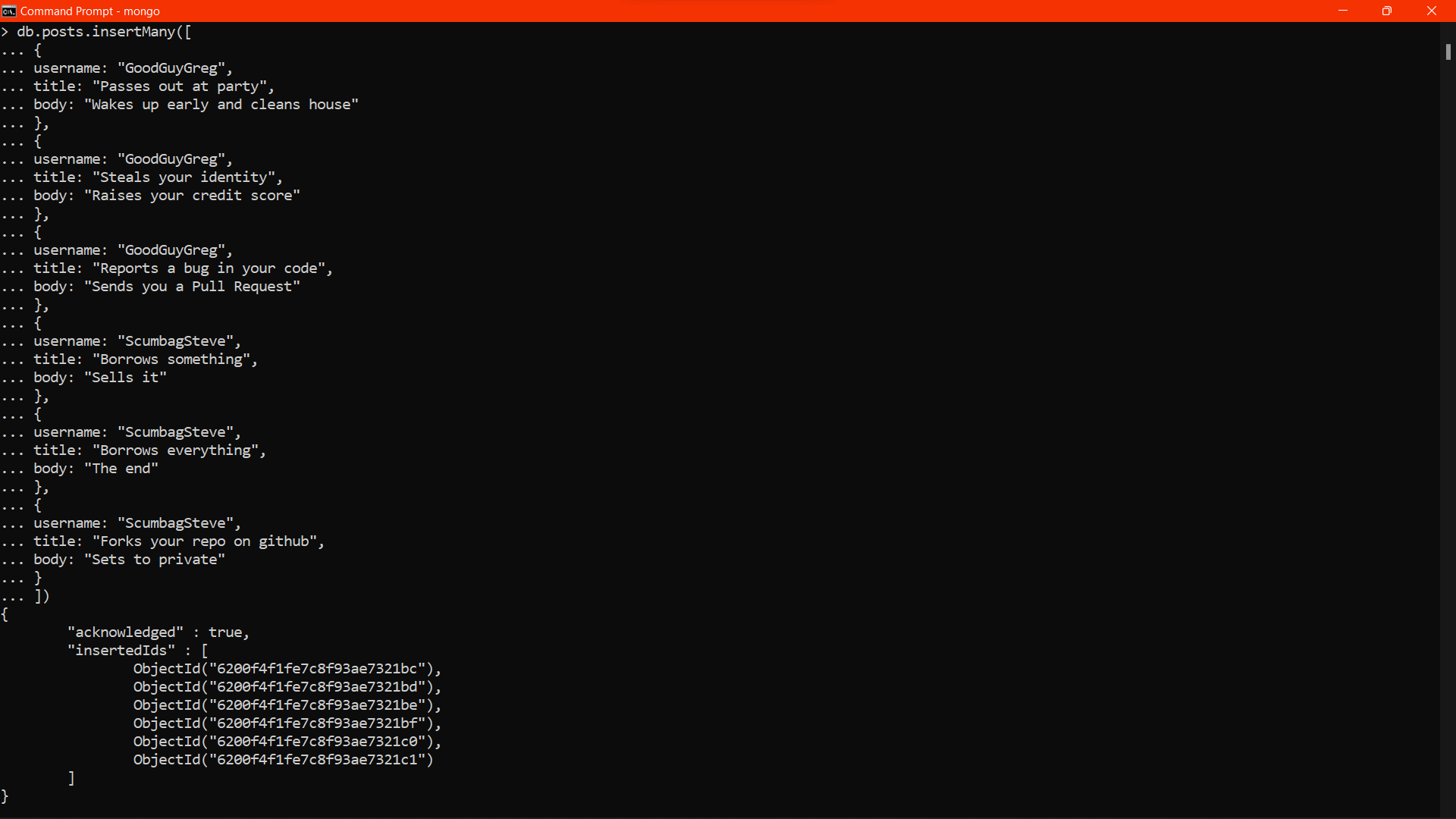
ObjectId("6200f4f1fe7c8f93ae7321bf"),

ObjectId("6200f4f1fe7c8f93ae7321c0"),

ObjectId("6200f4f1fe7c8f93ae7321c1")

]

}



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

username : GoodGuyGreg comment : Hope you got a good deal!

post : [post\_obj\_id]

where [post\_obj\_id] is the ObjectId of the posts document: "Borrows something"

username : GoodGuyGreg comment : What's mine is yours!

post : [post\_obj\_id]

where [post\_obj\_id] is the ObjectId of the posts document: "Borrows everything"

username : GoodGuyGreg comment : Don't violate the licensing agreement!

post : [post\_obj\_id]

where [post\_obj\_id] is the ObjectId of the posts document: "Forks your repo on github

username : ScumbagSteve

comment : It still isn't clean post : [post\_obj\_id]

where [post\_obj\_id] is the ObjectId of the posts document: "Passes out at party" username : ScumbagSteve comment : Denied your PR cause I found a hack post : [post\_obj\_id]

where [post\_obj\_id] is the ObjectId of the posts document: "Reports a bug in your code"

> db.comments.insertMany([

... {

... username: "GoodGuyGreg",

... comment: "Hope you got a good deal!",

... post: ObjectId("6200f4f1fe7c8f93ae7321bf")

... },

... {

... username: "GoodGuyGreg",

... comment: "Don't violate the licensing agreement!",

... post: ObjectId("6200f4f1fe7c8f93ae7321c0"),

... },

... {

... username: "GoodGuyGreg",

... comment: "Don't violate the licensing agreement!",

... post: ObjectId("6200f4f1fe7c8f93ae7321c1")

... },

... {

... username: "ScumbagSteve",

... comment: "It still isn't clean",

... post: ObjectId("6200f4f1fe7c8f93ae7321bc")

... },

... {

... username: "ScumbagSteve",

... comment: "Denied your PR cause I found a hack",

... post: ObjectId("6200f4f1fe7c8f93ae7321be"),

... }

... ])

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("6200f9b4fe7c8f93ae7321c7"),

ObjectId("6200f9b4fe7c8f93ae7321c8"),

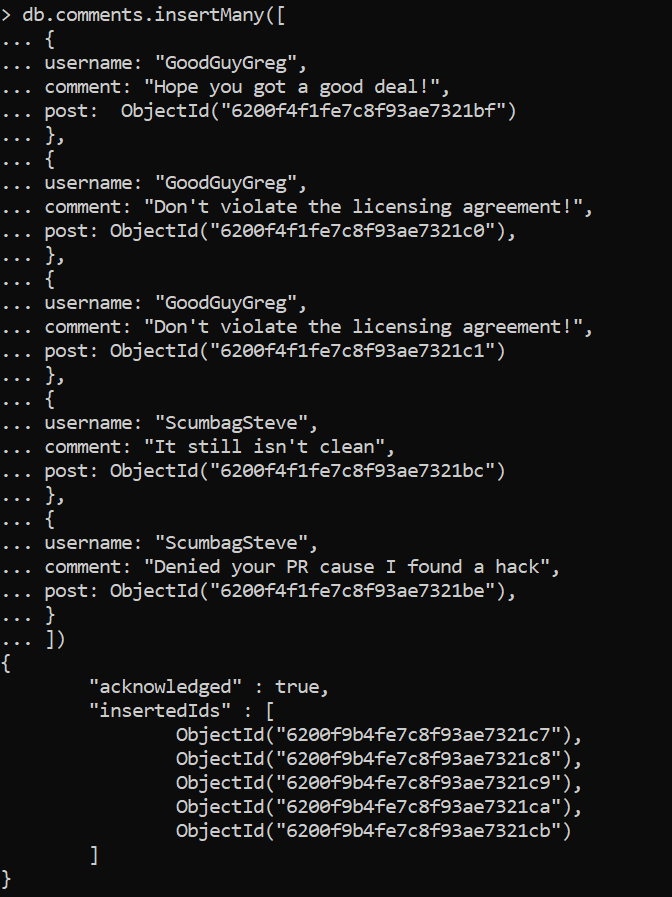
ObjectId("6200f9b4fe7c8f93ae7321c9"),

ObjectId("6200f9b4fe7c8f93ae7321ca"),

ObjectId("6200f9b4fe7c8f93ae7321cb")

]

}



# Querying related collections

1. find all users

> db.users.find().pretty()

{

"\_id" : ObjectId("6200f372fe7c8f93ae7321ba"),

"username" : "GoodGuyGreg",

"first\_name" : "Good Guy",

"last\_name" : "Greg"

}

{

"\_id" : ObjectId("6200f372fe7c8f93ae7321bb"),

"username" : "ScumbagSteve",

"full\_name" : [

{

"first" : "Scumbag"

},

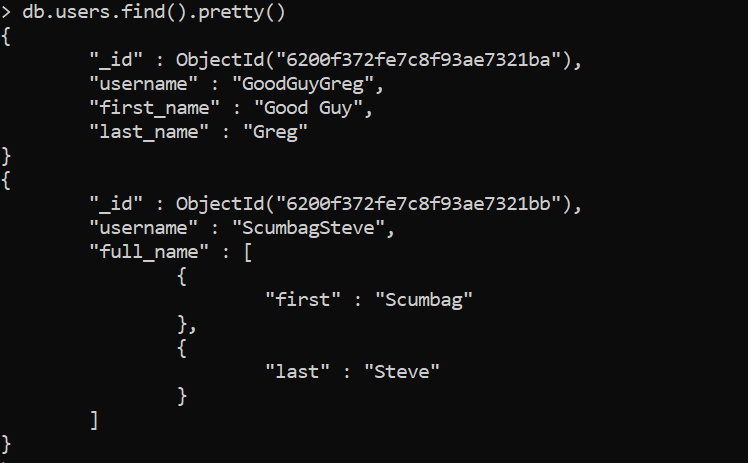
{

"last" : "Steve"

}

]

}



1. find all posts

> db.posts.find().pretty()

{

"\_id" : ObjectId("6200f4f1fe7c8f93ae7321bc"),

"username" : "GoodGuyGreg",

"title" : "Passes out at party",

"body" : "Wakes up early and cleans house"

}

{

"\_id" : ObjectId("6200f4f1fe7c8f93ae7321bd"),

"username" : "GoodGuyGreg",

"title" : "Steals your identity",

"body" : "Raises your credit score"

}

{

"\_id" : ObjectId("6200f4f1fe7c8f93ae7321be"),

"username" : "GoodGuyGreg",

"title" : "Reports a bug in your code",

"body" : "Sends you a Pull Request"

}

{

"\_id" : ObjectId("6200f4f1fe7c8f93ae7321bf"),

"username" : "ScumbagSteve",

"title" : "Borrows something",

"body" : "Sells it"

}

{

"\_id" : ObjectId("6200f4f1fe7c8f93ae7321c0"),

"username" : "ScumbagSteve",

"title" : "Borrows everything",

"body" : "The end"

}

{

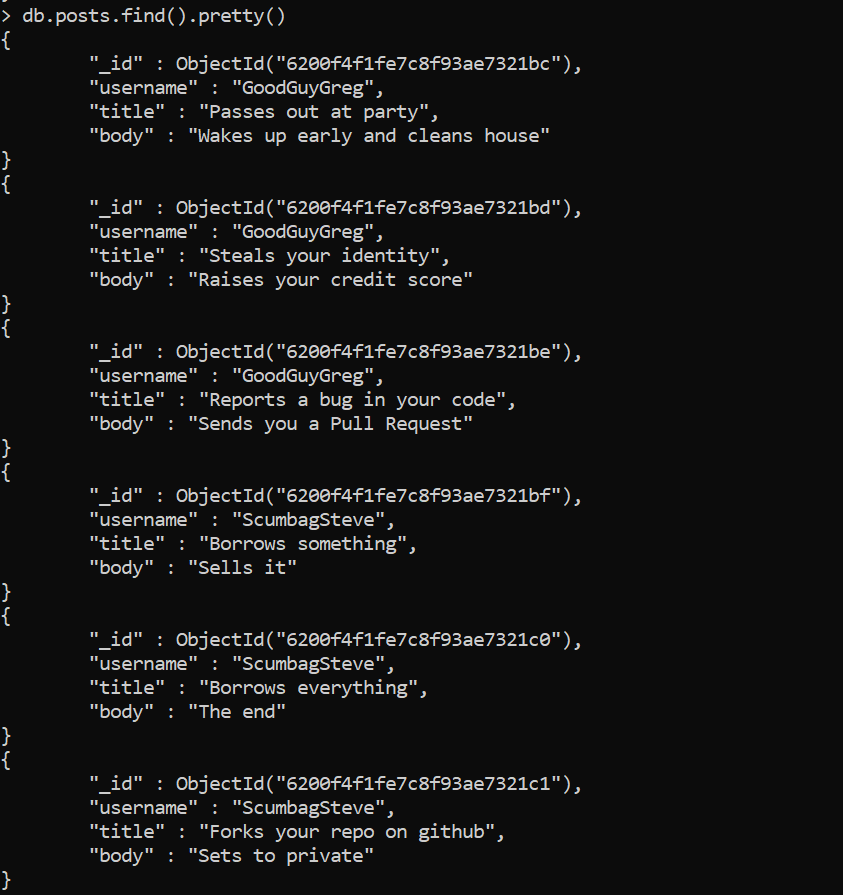
"\_id" : ObjectId("6200f4f1fe7c8f93ae7321c1"),

"username" : "ScumbagSteve",

"title" : "Forks your repo on github",

"body" : "Sets to private"

}



1. find all posts that was authored by "GoodGuyGreg"

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

{

"\_id" : ObjectId("6200f4f1fe7c8f93ae7321bc"),

"username" : "GoodGuyGreg",

"title" : "Passes out at party",

"body" : "Wakes up early and cleans house"

}

{

"\_id" : ObjectId("6200f4f1fe7c8f93ae7321bd"),

"username" : "GoodGuyGreg",

"title" : "Steals your identity",

"body" : "Raises your credit score"

}

{

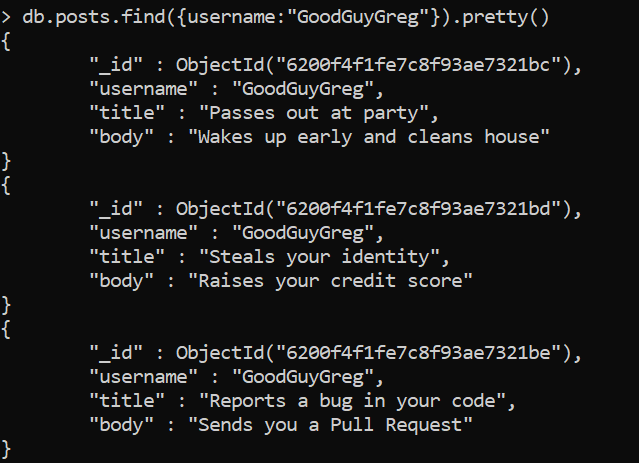
"\_id" : ObjectId("6200f4f1fe7c8f93ae7321be"),

"username" : "GoodGuyGreg",

"title" : "Reports a bug in your code",

"body" : "Sends you a Pull Request"

}



1. find all posts that was authored by "ScumbagSteve"

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

{

"\_id" : ObjectId("6200f4f1fe7c8f93ae7321bf"),

"username" : "ScumbagSteve",

"title" : "Borrows something",

"body" : "Sells it"

}

{

"\_id" : ObjectId("6200f4f1fe7c8f93ae7321c0"),

"username" : "ScumbagSteve",

"title" : "Borrows everything",

"body" : "The end"

}

{

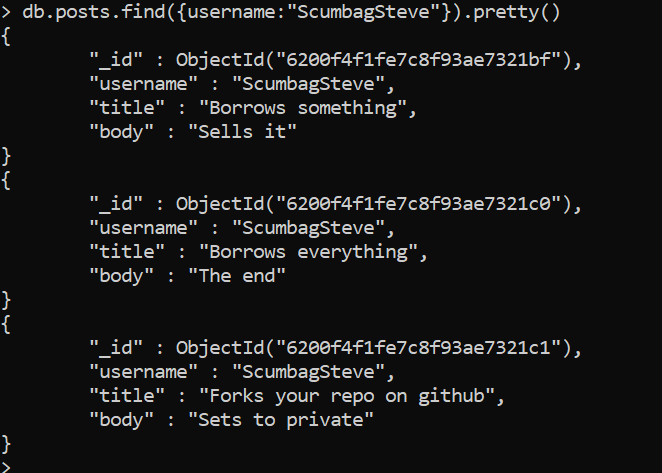
"\_id" : ObjectId("6200f4f1fe7c8f93ae7321c1"),

"username" : "ScumbagSteve",

"title" : "Forks your repo on github",

"body" : "Sets to private"

}



1. find all comments

> db.comments.find().pretty()

{

"\_id" : ObjectId("6200f9b4fe7c8f93ae7321c7"),

"username" : "GoodGuyGreg",

"comment" : "Hope you got a good deal!",

"post" : ObjectId("6200f4f1fe7c8f93ae7321bf")

}

{

"\_id" : ObjectId("6200f9b4fe7c8f93ae7321c8"),

"username" : "GoodGuyGreg",

"comment" : "Don't violate the licensing agreement!",

"post" : ObjectId("6200f4f1fe7c8f93ae7321c0")

}

{

"\_id" : ObjectId("6200f9b4fe7c8f93ae7321c9"),

"username" : "GoodGuyGreg",

"comment" : "Don't violate the licensing agreement!",

"post" : ObjectId("6200f4f1fe7c8f93ae7321c1")

}

{

"\_id" : ObjectId("6200f9b4fe7c8f93ae7321ca"),

"username" : "ScumbagSteve",

"comment" : "It still isn't clean",

"post" : ObjectId("6200f4f1fe7c8f93ae7321bc")

}

{

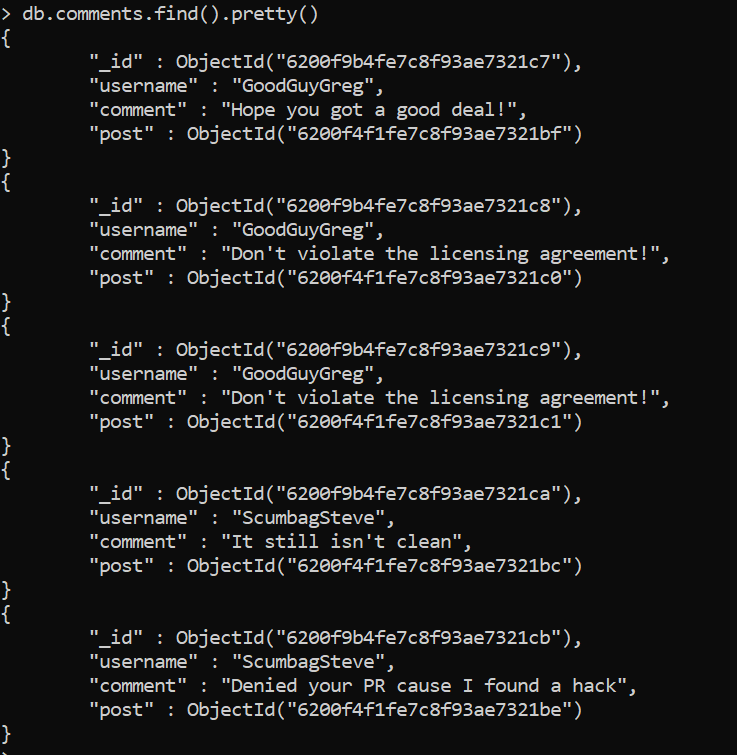
"\_id" : ObjectId("6200f9b4fe7c8f93ae7321cb"),

"username" : "ScumbagSteve",

"comment" : "Denied your PR cause I found a hack",

"post" : ObjectId("6200f4f1fe7c8f93ae7321be")

}



1. find all comments that was authored by "GoodGuyGreg"

> db.comments.find("username":"GoodGuyGreg").pretty()

uncaught exception: SyntaxError: missing ) after argument list :

@(shell):1:27

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

{

"\_id" : ObjectId("6200f9b4fe7c8f93ae7321c7"),

"username" : "GoodGuyGreg",

"comment" : "Hope you got a good deal!",

"post" : ObjectId("6200f4f1fe7c8f93ae7321bf")

}

{

"\_id" : ObjectId("6200f9b4fe7c8f93ae7321c8"),

"username" : "GoodGuyGreg",

"comment" : "Don't violate the licensing agreement!",

"post" : ObjectId("6200f4f1fe7c8f93ae7321c0")

}

{

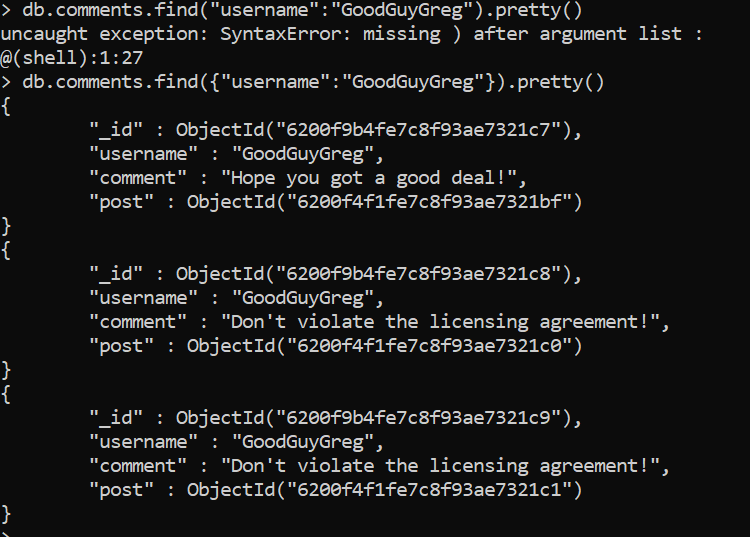
"\_id" : ObjectId("6200f9b4fe7c8f93ae7321c9"),

"username" : "GoodGuyGreg",

"comment" : "Don't violate the licensing agreement!",

"post" : ObjectId("6200f4f1fe7c8f93ae7321c1")

}



1. find all comments that was authored by "ScumbagSteve"

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

{

"\_id" : ObjectId("6200f9b4fe7c8f93ae7321ca"),

"username" : "ScumbagSteve",

"comment" : "It still isn't clean",

"post" : ObjectId("6200f4f1fe7c8f93ae7321bc")

}

{

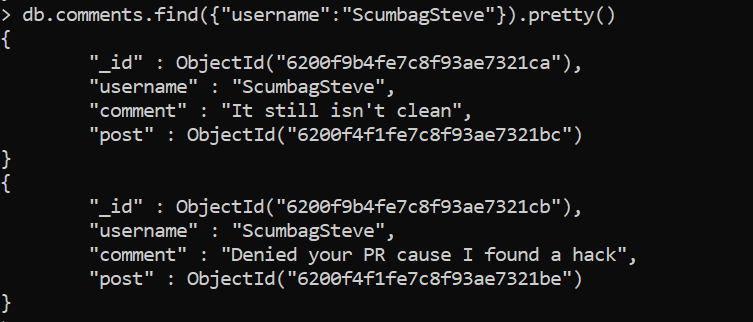
"\_id" : ObjectId("6200f9b4fe7c8f93ae7321cb"),

"username" : "ScumbagSteve",

"comment" : "Denied your PR cause I found a hack",

"post" : ObjectId("6200f4f1fe7c8f93ae7321be")

}



1. 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()

{

"\_id" : ObjectId("6200f4f1fe7c8f93ae7321be"),

"username" : "GoodGuyGreg",

"title" : "Reports a bug in your code",

"body" : "Sends you a Pull Request",

"comments" : [

{

"\_id" : ObjectId("6200f9b4fe7c8f93ae7321cb"),

"username" : "ScumbagSteve",

"comment" : "Denied your PR cause I found a hack",

"post" : ObjectId("6200f4f1fe7c8f93ae7321be")

}

]

}



References:

<https://docs.mongodb.com/manual/reference/method/db.collection.find/>

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@