MongoDB Lab Assignment 1

S.yuvan karthick

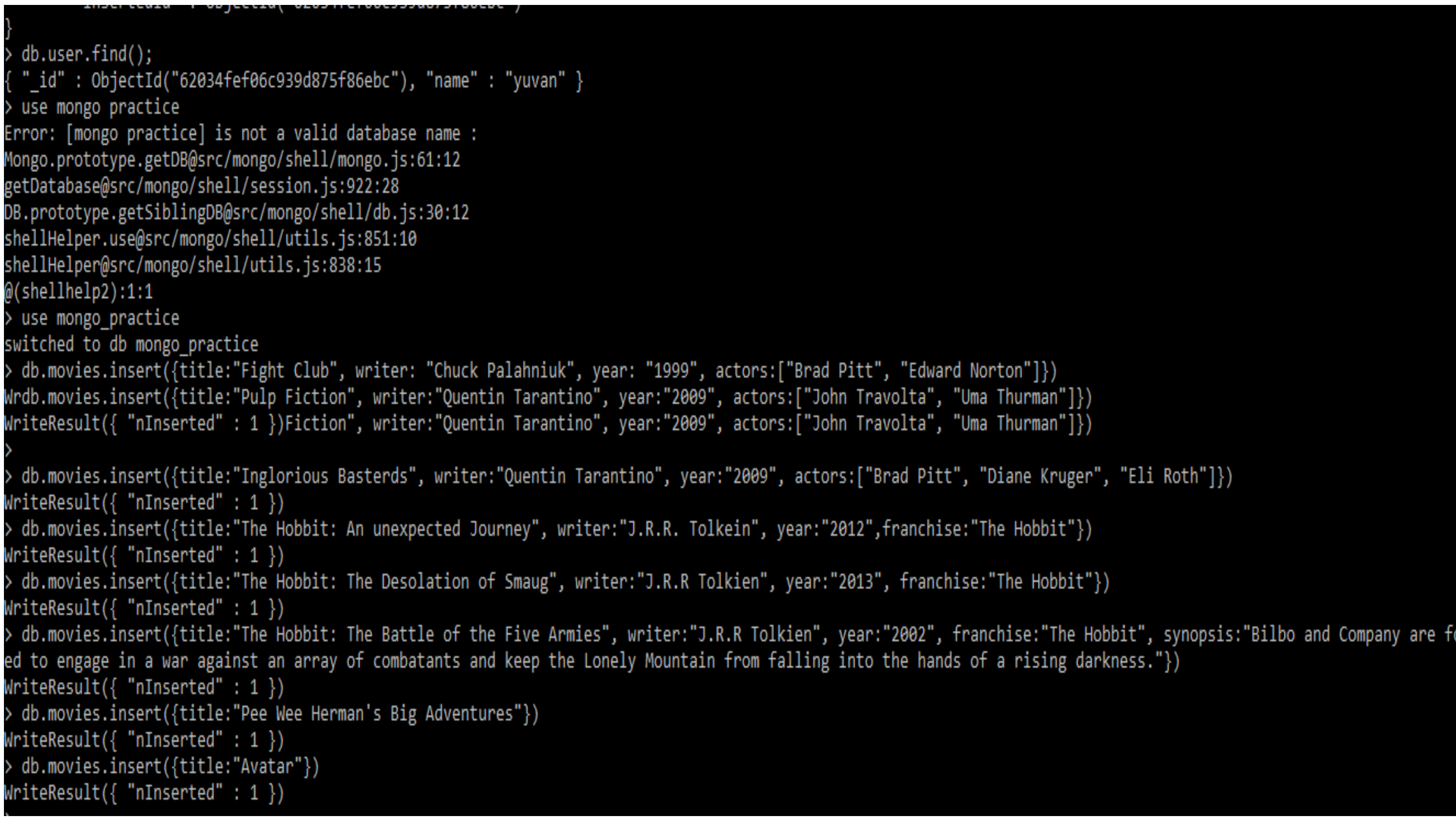
MongoDB Exercise in mongo shell

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.

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

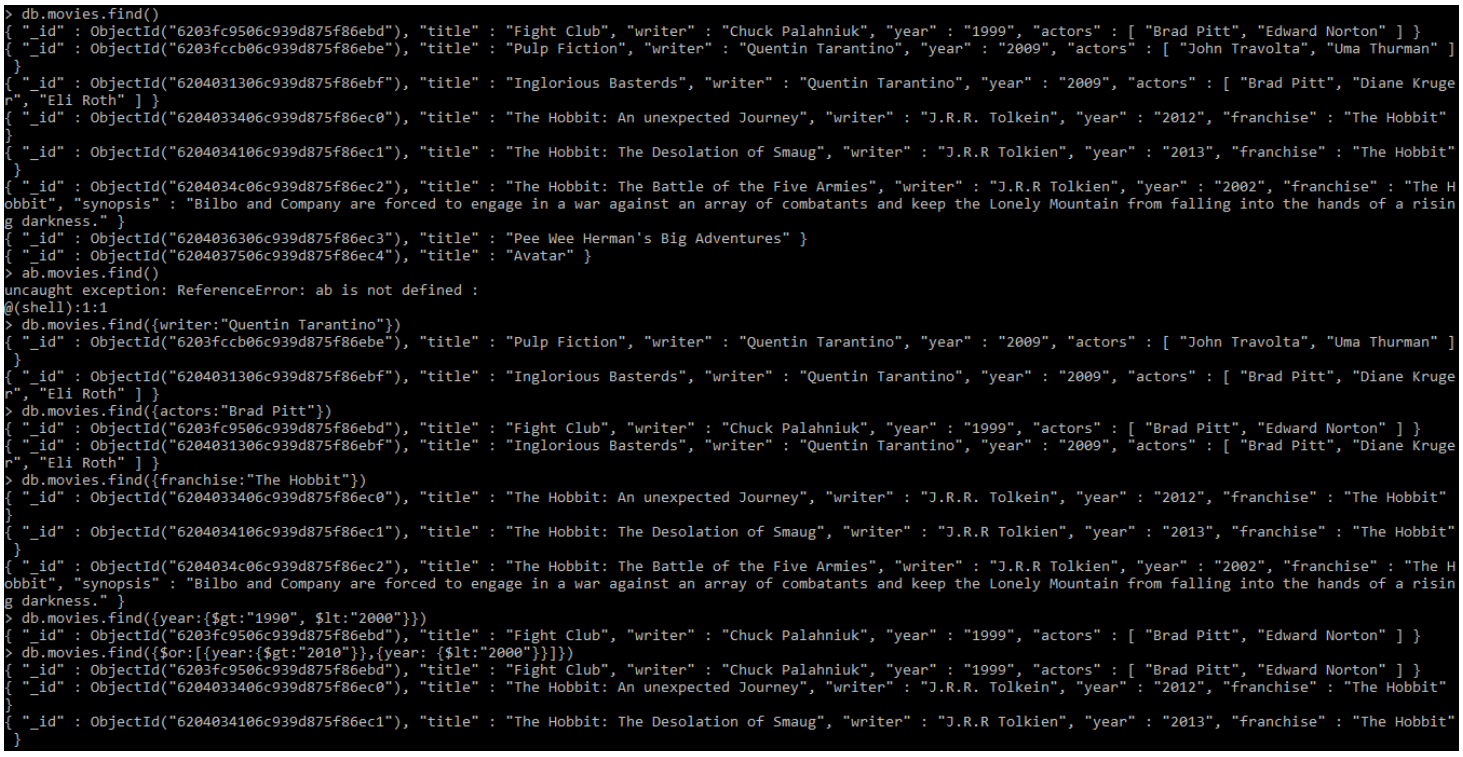
year : 2013
franchise : The Hobbit
title : The Hobbit: The Battle of the Five Armies
writer : J.R.R. Tolkein
year : 2012
franchise : The Hobbit
synopsis : Bilbo and Company are forced to engage in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of a rising darkness.
title : Pee Wee Herman's Big Adventure
title : Avatar



```
}
> db.user.find();
{ "_id" : ObjectId("62034fef06c939d875f86ebc"), "name" : "yuvan" }
> use mongo practice
Error: [mongo practice] is not a valid database name :
Mongo.prototype.getDB@src/mongo/shell/mongo.js:61:12
getDatabase@src/mongo/shell/session.js:922:28
DB.prototype.getSiblingDB@src/mongo/shell/db.js:30:12
shellHelper.use@src/mongo/shell/utils.js:851:10
shellHelper@src/mongo/shell/utils.js:838:15
@(shellhelp2):1:1
> use mongo_practice
switched to db mongo_practice
> db.movies.insert({title:"Fight Club", writer: "Chuck Palahniuk", year: "1999", actors:["Brad Pitt", "Edward Norton"]})
Wrdb.movies.insert({title:"Pulp Fiction", writer:"Quentin Tarantino", year:"2009", actors:["John Travolta", "Uma Thurman"]})
WriteResult({ "nInserted" : 1 })Fiction", writer:"Quentin Tarantino", year:"2009", actors:["John Travolta", "Uma Thurman"]})
>
> db.movies.insert({title:"Inglorious Basterds", writer:"Quentin Tarantino", year:"2009", actors:["Brad Pitt", "Diane Kruger", "Eli Roth"]})
WriteResult({ "nInserted" : 1 })
> db.movies.insert({title:"The Hobbit: An unexpected Journey", writer:"J.R.R. Tolkein", year:"2012",franchise:"The Hobbit"})
WriteResult({ "nInserted" : 1 })
> db.movies.insert({title:"The Hobbit: The Desolation of Smaug", writer:"J.R.R Tolkien", year:"2013", franchise:"The Hobbit"})
WriteResult({ "nInserted" : 1 })
> db.movies.insert({title:"The Hobbit: The Battle of the Five Armies", writer:"J.R.R Tolkien", year:"2002", franchise:"The Hobbit", synopsis:"Bilbo and Company are f
ed to engage in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of a rising darkness."})
WriteResult({ "nInserted" : 1 })
> db.movies.insert({title:"Pee Wee Herman's Big Adventures"})
WriteResult({ "nInserted" : 1 })
> db.movies.insert({title:"Avatar"})
WriteResult({ "nInserted" : 1 })
```

Query / Find Documents query

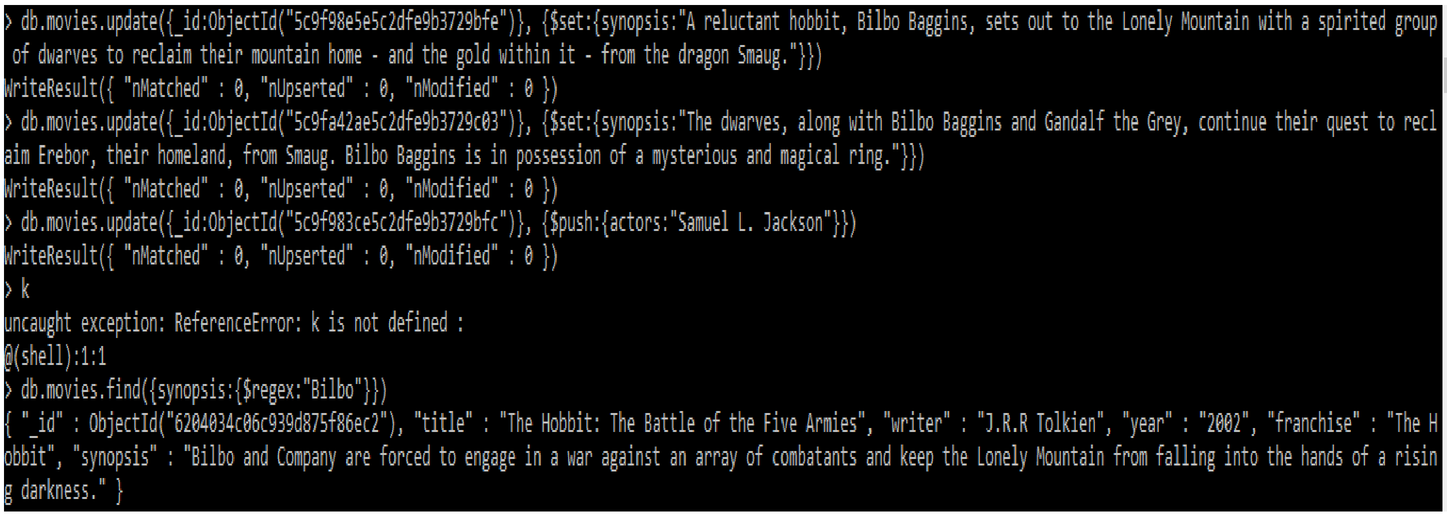
The movies collection to 1. get all documents 2. get all documents with writer set to "Quentin Tarantino" 3. get all documents where actors include "Brad Pitt" 4. get all documents with franchise set to "The Hobbit" 5. get all movies released in the 90s 6. get all movies released before the year 2000 or after 2010.



```
> db.movies.find()
{ "_id" : ObjectId("6203fc9506c939d875f86ebd"), "title" : "Fight Club", "writer" : "Chuck Palahniuk", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }
{ "_id" : ObjectId("6203fccb06c939d875f86ebe"), "title" : "Pulp Fiction", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "John Travolta", "Uma Thurman" ]
 }
{ "_id" : ObjectId("6204031306c939d875f86ebf"), "title" : "Inglorious Basterds", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "Brad Pitt", "Diane Kruge
r", "Eli Roth" ] }
{ "_id" : ObjectId("6204033406c939d875f86ec0"), "title" : "The Hobbit: An unexpected Journey", "writer" : "J.R.R. Tolkein", "year" : "2012", "franchise" : "The Hobbit"
}
{ "_id" : ObjectId("6204034106c939d875f86ec1"), "title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R Tolkien", "year" : "2013", "franchise" : "The Hobbit"
 }
{ "_id" : ObjectId("6204034c06c939d875f86ec2"), "title" : "The Hobbit: The Battle of the Five Armies", "writer" : "J.R.R Tolkien", "year" : "2002", "franchise" : "The H
obbit", "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 risin
g darkness." }
{ "_id" : ObjectId("6204036306c939d875f86ec3"), "title" : "Pee Wee Herman's Big Adventures" }
{ "_id" : ObjectId("6204037506c939d875f86ec4"), "title" : "Avatar" }
> ab.movies.find()
uncaught exception: ReferenceError: ab is not defined :
@(shell):1:1
> db.movies.find({writer:"Quentin Tarantino"})
{ "_id" : ObjectId("6203fccb06c939d875f86ebe"), "title" : "Pulp Fiction", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "John Travolta", "Uma Thurman" ]
 }
{ "_id" : ObjectId("6204031306c939d875f86ebf"), "title" : "Inglorious Basterds", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "Brad Pitt", "Diane Kruge
r", "Eli Roth" ] }
> db.movies.find({actors:"Brad Pitt"})
{ "_id" : ObjectId("6203fc9506c939d875f86ebd"), "title" : "Fight Club", "writer" : "Chuck Palahniuk", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }
{ "_id" : ObjectId("6204031306c939d875f86ebf"), "title" : "Inglorious Basterds", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "Brad Pitt", "Diane Kruge
r", "Eli Roth" ] }
> db.movies.find({franchise:"The Hobbit"})
{ "_id" : ObjectId("6204033406c939d875f86ec0"), "title" : "The Hobbit: An unexpected Journey", "writer" : "J.R.R. Tolkein", "year" : "2012", "franchise" : "The Hobbit"
}
{ "_id" : ObjectId("6204034106c939d875f86ec1"), "title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R Tolkien", "year" : "2013", "franchise" : "The Hobbit"
 }
{ "_id" : ObjectId("6204034c06c939d875f86ec2"), "title" : "The Hobbit: The Battle of the Five Armies", "writer" : "J.R.R Tolkien", "year" : "2002", "franchise" : "The H
obbit", "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 risin
g darkness." }
> db.movies.find({year:{$gt:"1990", $lt:"2000"}})
{ "_id" : ObjectId("6203fc9506c939d875f86ebd"), "title" : "Fight Club", "writer" : "Chuck Palahniuk", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }
> db.movies.find({$or:[{year:{$gt:"2010"}},{year: {$lt:"2000"}}]})
{ "_id" : ObjectId("6203fc9506c939d875f86ebd"), "title" : "Fight Club", "writer" : "Chuck Palahniuk", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }
{ "_id" : ObjectId("6204033406c939d875f86ec0"), "title" : "The Hobbit: An unexpected Journey", "writer" : "J.R.R. Tolkein", "year" : "2012", "franchise" : "The Hobbit"
}
{ "_id" : ObjectId("6204034106c939d875f86ec1"), "title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R Tolkien", "year" : "2013", "franchise" : "The Hobbit"
 }
```

Update Documents

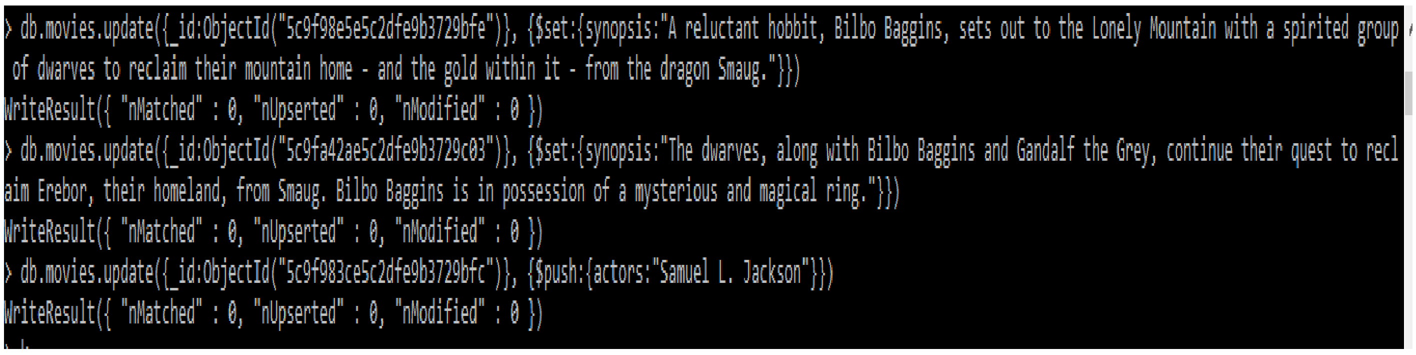
1. add a synopsis to "The Hobbit: An Unexpected Journey" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug."
2. add a synopsis to "The Hobbit: The Desolation of Smaug" : "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."
3. add an actor named "Samuel L. Jackson" to the movie "Pulp Fiction"



```
> db.movies.update({_id:ObjectId("5c9f98e5e5c2dfe9b3729bfe")}, {$set:{synopsis:"A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group
 of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug."}})
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
> db.movies.update({_id:ObjectId("5c9fa42ae5c2dfe9b3729c03")}, {$set:{synopsis:"The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to recl
aim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."}})
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
> db.movies.update({_id:ObjectId("5c9f983ce5c2dfe9b3729bfc")}, {$push:{actors:"Samuel L. Jackson"}})
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
> k
uncaught exception: ReferenceError: k is not defined :
@(shell):1:1
> db.movies.find({synopsis:{$regex:"Bilbo"}})
{ "_id" : ObjectId("6204034c06c939d875f86ec2"), "title" : "The Hobbit: The Battle of the Five Armies", "writer" : "J.R.R Tolkien", "year" : "2002", "franchise" : "The H
obbit", "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 risin
g darkness." }
```

Text Search

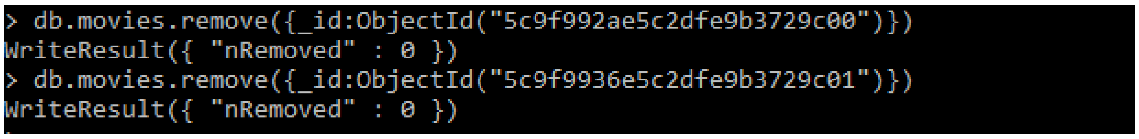
1. find all movies that have a synopsis that contains the word "Bilbo"
2. find all movies that have a synopsis that contains the word "Gandalf"
3. find all movies that have a synopsis that contains the word "Bilbo" and not the word "Gandalf"
4. find all movies that have a synopsis that contains the word "dwarves" or "hobbit"
5. find all movies that have a synopsis that contains the word "gold" and "dragon"



```
> db.movies.update({_id:ObjectId("5c9f98e5e5c2dfe9b3729bfe")}, {$set:{synopsis:"A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group
 of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug."}})
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
> db.movies.update({_id:ObjectId("5c9fa42ae5c2dfe9b3729c03")}, {$set:{synopsis:"The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to recl
aim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."}})
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
> db.movies.update({_id:ObjectId("5c9f983ce5c2dfe9b3729bfc")}, {$push:{actors:"Samuel L. Jackson"}})
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
```

Delete Documents

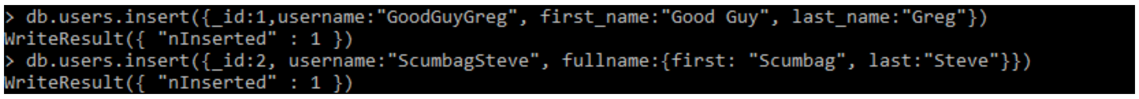
1. delete the movie "Pee Wee Herman's Big Adventure"
2. delete the movie "Avatar



```
> db.movies.remove({_id:ObjectId("5c9f992ae5c2dfe9b3729c00")})
WriteResult({ "nRemoved" : 0 })
> db.movies.remove({_id:ObjectId("5c9f9936e5c2dfe9b3729c01")})
WriteResult({ "nRemoved" : 0 })
```

Relationships
Insert the following documents into a users collection

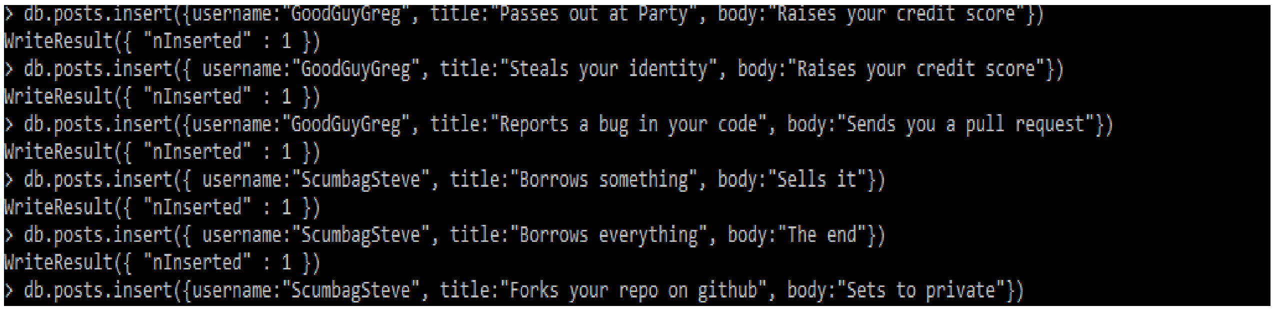
username : GoodGuyGreg
first_name : "Good Guy"
last_name : "Greg"
username : ScumbagSteve
full_name :
 first : "Scumbag"
 last : "Steve"



```
> db.users.insert({_id:1,username:"GoodGuyGreg", first_name:"Good Guy", last_name:"Greg"})
WriteResult({ "nInserted" : 1 })
> db.users.insert({_id:2, username:"ScumbagSteve", fullname:{first: "Scumbag", last:"Steve"}})
WriteResult({ "nInserted" : 1 })
```

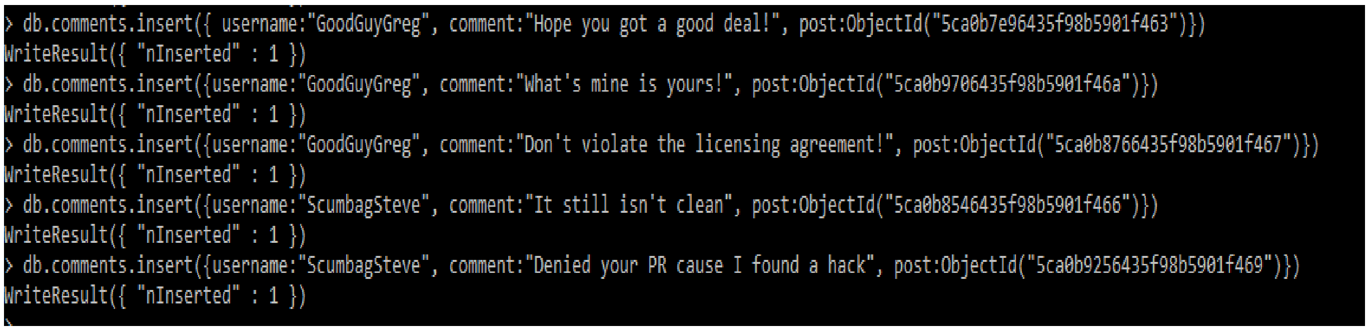
Insert the following documents into a posts collection

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.insert({username:"GoodGuyGreg", title:"Passes out at Party", body:"Raises your credit score"})
WriteResult({ "nInserted" : 1 })
> db.posts.insert({ username:"GoodGuyGreg", title:"Steals your identity", body:"Raises your credit score"})
WriteResult({ "nInserted" : 1 })
> db.posts.insert({username:"GoodGuyGreg", title:"Reports a bug in your code", body:"Sends you a pull request"})
WriteResult({ "nInserted" : 1 })
> db.posts.insert({ username:"ScumbagSteve", title:"Borrows something", body:"Sells it"})
WriteResult({ "nInserted" : 1 })
> db.posts.insert({ username:"ScumbagSteve", title:"Borrows everything", body:"The end"})
WriteResult({ "nInserted" : 1 })
> db.posts.insert({username:"ScumbagSteve", title:"Forks your repo on github", body:"Sets to private"})
```

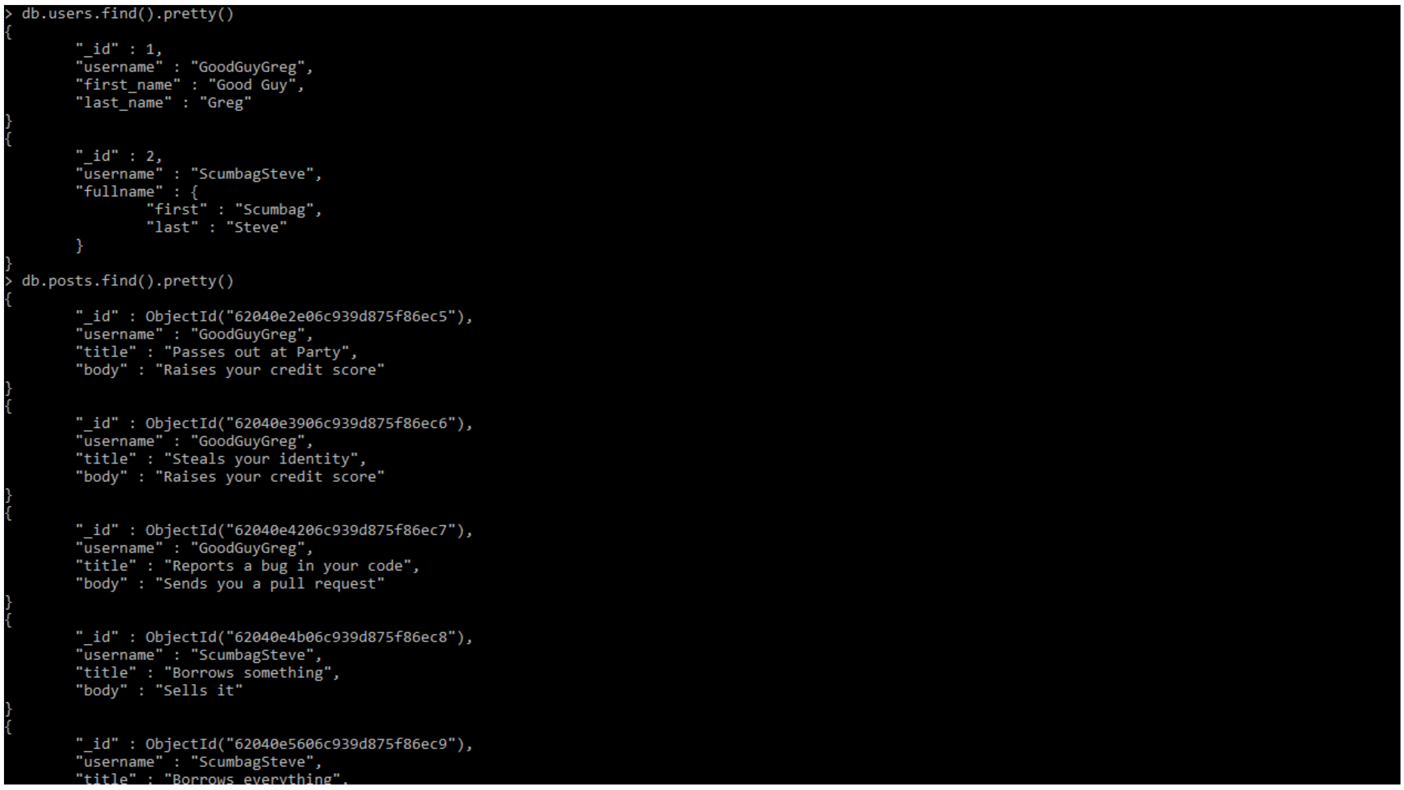
Insert the following documents into a comments collection
username : GoodGuyGreg
comment : Hope you got a good deal!
post : [post_obj_id]
where [post_obj_id] is the ObjectId of the posts document: "Borrows something"
username : GoodGuyGreg
comment : What's mine is yours!
post : [post_obj_id]
where [post_obj_id] 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



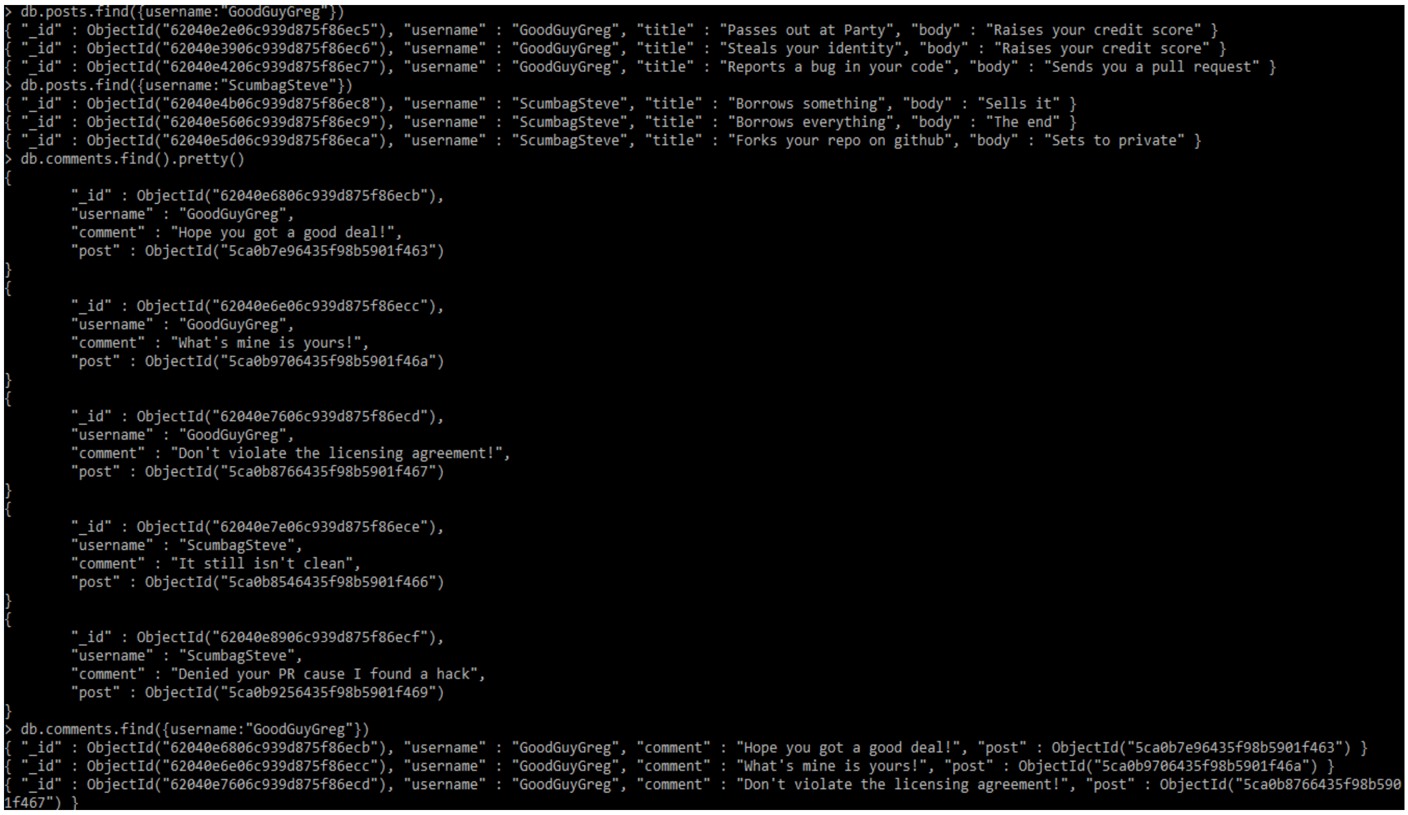
```
> db.comments.insert({ username:"GoodGuyGreg", comment:"Hope you got a good deal!", post:ObjectId("5ca0b7e96435f98b5901f463")})
WriteResult({ "nInserted" : 1 })
> db.comments.insert({username:"GoodGuyGreg", comment:"What's mine is yours!", post:ObjectId("5ca0b9706435f98b5901f46a")})
WriteResult({ "nInserted" : 1 })
> db.comments.insert({username:"GoodGuyGreg", comment:"Don't violate the licensing agreement!", post:ObjectId("5ca0b8766435f98b5901f467")})
WriteResult({ "nInserted" : 1 })
> db.comments.insert({username:"ScumbagSteve", comment:"It still isn't clean", post:ObjectId("5ca0b8546435f98b5901f466")})
WriteResult({ "nInserted" : 1 })
> db.comments.insert({username:"ScumbagSteve", comment:"Denied your PR cause I found a hack", post:ObjectId("5ca0b9256435f98b5901f469")})
WriteResult({ "nInserted" : 1 })
>
```

Querying related collections

1. find all users
2. find all posts
3. find all posts that was authored by "GoodGuyGreg"
4. find all posts that was authored by "ScumbagSteve"
5. find all comments
6. find all comments that was authored by "GoodGuyGreg"
7. find all comments that was authored by "ScumbagSteve"
8. find all comments belonging to the post "Reports a bug in your code"



```
> db.users.find().pretty()
{
        "_id" : 1,
        "username" : "GoodGuyGreg",
        "first_name" : "Good Guy",
        "last_name" : "Greg"
}
{
        "_id" : 2,
        "username" : "ScumbagSteve",
        "fullname" : {
                "first" : "Scumbag",
                "last" : "Steve"
        }
}
> db.posts.find().pretty()
{
        "_id" : ObjectId("62040e2e06c939d875f86ec5"),
        "username" : "GoodGuyGreg",
        "title" : "Passes out at Party",
        "body" : "Raises your credit score"
}
{
        "_id" : ObjectId("62040e3906c939d875f86ec6"),
        "username" : "GoodGuyGreg",
        "title" : "Steals your identity",
        "body" : "Raises your credit score"
}
{
        "_id" : ObjectId("62040e4206c939d875f86ec7"),
        "username" : "GoodGuyGreg",
        "title" : "Reports a bug in your code",
        "body" : "Sends you a pull request"
}
{
        "_id" : ObjectId("62040e4b06c939d875f86ec8"),
        "username" : "ScumbagSteve",
        "title" : "Borrows something",
        "body" : "Sells it"
}
{
        "_id" : ObjectId("62040e5606c939d875f86ec9"),
        "username" : "ScumbagSteve",
        "title" : "Borrows everything",
```



```
> db.posts.find({username:"GoodGuyGreg"})
{ "_id" : ObjectId("62040e2e06c939d875f86ec5"), "username" : "GoodGuyGreg", "title" : "Passes out at Party", "body" : "Raises your credit score" }
{ "_id" : ObjectId("62040e3906c939d875f86ec6"), "username" : "GoodGuyGreg", "title" : "Steals your identity", "body" : "Raises your credit score" }
{ "_id" : ObjectId("62040e4206c939d875f86ec7"), "username" : "GoodGuyGreg", "title" : "Reports a bug in your code", "body" : "Sends you a pull request" }
> db.posts.find({username:"ScumbagSteve"})
{ "_id" : ObjectId("62040e4b06c939d875f86ec8"), "username" : "ScumbagSteve", "title" : "Borrows something", "body" : "Sells it" }
{ "_id" : ObjectId("62040e5606c939d875f86ec9"), "username" : "ScumbagSteve", "title" : "Borrows everything", "body" : "The end" }
{ "_id" : ObjectId("62040e5d06c939d875f86eca"), "username" : "ScumbagSteve", "title" : "Forks your repo on github", "body" : "Sets to private" }
> db.comments.find().pretty()
{
        "_id" : ObjectId("62040e6806c939d875f86ecb"),
        "username" : "GoodGuyGreg",
        "comment" : "Hope you got a good deal!",
        "post" : ObjectId("5ca0b7e96435f98b5901f463")
}
{
        "_id" : ObjectId("62040e6e06c939d875f86ecc"),
        "username" : "GoodGuyGreg",
        "comment" : "What's mine is yours!",
        "post" : ObjectId("5ca0b9706435f98b5901f46a")
}
{
        "_id" : ObjectId("62040e7606c939d875f86ecd"),
        "username" : "GoodGuyGreg",
        "comment" : "Don't violate the licensing agreement!",
        "post" : ObjectId("5ca0b8766435f98b5901f467")
}
{
        "_id" : ObjectId("62040e7e06c939d875f86ece"),
        "username" : "ScumbagSteve",
        "comment" : "It still isn't clean",
        "post" : ObjectId("5ca0b8546435f98b5901f466")
}
{
        "_id" : ObjectId("62040e8906c939d875f86ecf"),
        "username" : "ScumbagSteve",
        "comment" : "Denied your PR cause I found a hack",
        "post" : ObjectId("5ca0b9256435f98b5901f469")
}
> db.comments.find({username:"GoodGuyGreg"})
{ "_id" : ObjectId("62040e6806c939d875f86ecb"), "username" : "GoodGuyGreg", "comment" : "Hope you got a good deal!", "post" : ObjectId("5ca0b7e96435f98b5901f463") }
{ "_id" : ObjectId("62040e6e06c939d875f86ecc"), "username" : "GoodGuyGreg", "comment" : "What's mine is yours!", "post" : ObjectId("5ca0b9706435f98b5901f46a") }
{ "_id" : ObjectId("62040e7606c939d875f86ecd"), "username" : "GoodGuyGreg", "comment" : "Don't violate the licensing agreement!", "post" : ObjectId("5ca0b8766435f98b590
1f467") }
```