Akshay. A. R

MongoDB Lab Assignments -Day 1

MongoDB Exercise in mongo shell

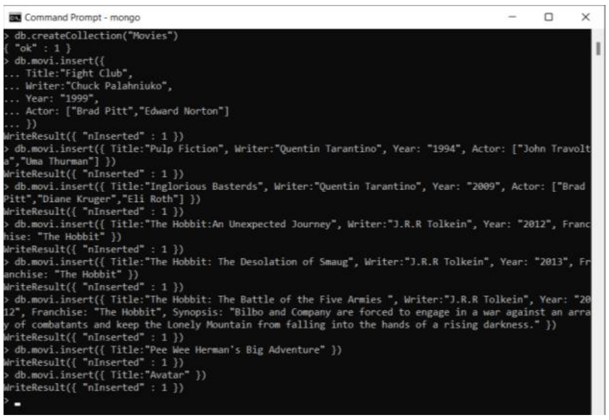
Connect to a running mongo instance, use a database named **mongo_practice**.
Document all your queries in a javascript file to use as a reference.

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

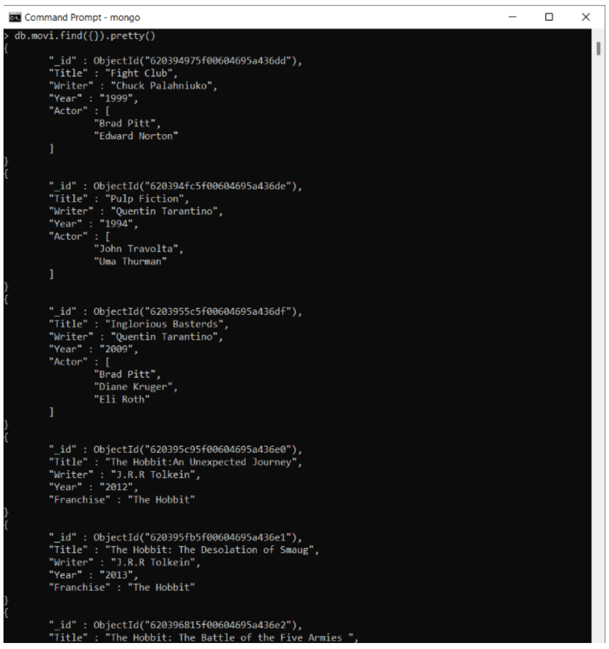
year : 2013
franchise : The Hobbit
title : The Hobbit: The Battle of the Five Armies
writer : J.R.R. Tolkein
year : 2012
franchise : The Hobbit
synopsis : Bilbo and Company are forced to engage in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of a rising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



Command Prompt - mongo

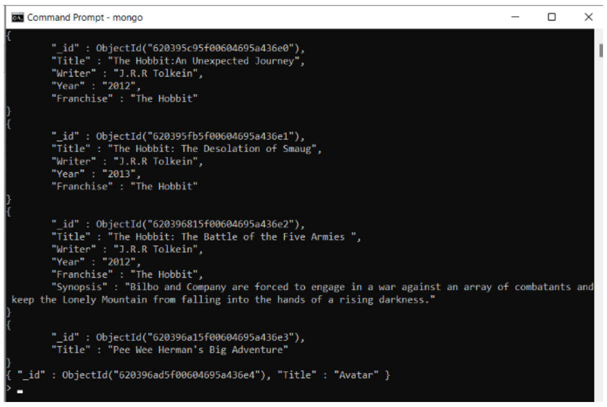
```
> db.createCollection("Movies")
{ "ok" : 1 }
> db.movi.insert({
... Title:"Fight Club",
... Writer:"Chuck Palahniuko",
... Year: "1999",
... Actor: ["Brad Pitt","Edward Norton"]
... })
WriteResult({ "nInserted" : 1 })
> db.movi.insert({ Title:"Pulp Fiction", Writer:"Quentin Tarantino", Year: "1994", Actor: ["John Travolt
a","Uma Thurman"] })
WriteResult({ "nInserted" : 1 })
> db.movi.insert({ Title:"Inglorious Basterds", Writer:"Quentin Tarantino", Year: "2009", Actor: ["Brad
Pitt","Diane Kruger","Eli Roth"] })
WriteResult({ "nInserted" : 1 })
> db.movi.insert({ Title:"The Hobbit:An Unexpected Journey", Writer:"J.R.R Tolkein", Year: "2012", Franc
hise: "The Hobbit" })
WriteResult({ "nInserted" : 1 })
> db.movi.insert({ Title:"The Hobbit: The Desolation of Smaug", Writer:"J.R.R Tolkein", Year: "2013", Fr
anchise: "The Hobbit" })
WriteResult({ "nInserted" : 1 })
> db.movi.insert({ Title:"The Hobbit: The Battle of the Five Armies ", Writer:"J.R.R Tolkein", Year: "20
12", Franchise: "The Hobbit", Synopsis: "Bilbo and Company are forced to engage in a war against an arra
y of combatants and keep the Lonely Mountain from falling into the hands of a rising darkness." })
WriteResult({ "nInserted" : 1 })
> db.movi.insert({ Title:"Pee Wee Herman's Big Adventure" })
WriteResult({ "nInserted" : 1 })
> db.movi.insert({ Title:"Avatar" })
WriteResult({ "nInserted" : 1 })
>
```

Query / Find Documents
query the **movies** collection to
1. get all documents



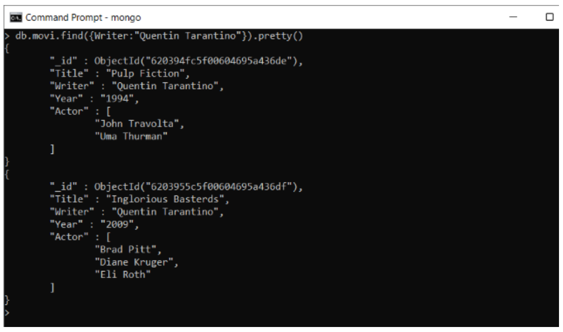
Command Prompt - mongo

```
> db.movi.find({}).pretty()
{
        "_id" : ObjectId("620394975f00604695a436dd"),
        "Title" : "Fight Club",
        "Writer" : "Chuck Palahniuko",
        "Year" : "1999",
        "Actor" : [
                "Brad Pitt",
                "Edward Norton"
        ]
}
{
        "_id" : ObjectId("620394fc5f00604695a436de"),
        "Title" : "Pulp Fiction",
        "Writer" : "Quentin Tarantino",
        "Year" : "1994",
        "Actor" : [
                "John Travolta",
                "Uma Thurman"
        ]
}
{
        "_id" : ObjectId("6203955c5f00604695a436df"),
        "Title" : "Inglorious Basterds",
        "Writer" : "Quentin Tarantino",
        "Year" : "2009",
        "Actor" : [
                "Brad Pitt",
                "Diane Kruger",
                "Eli Roth"
        ]
}
{
        "_id" : ObjectId("620395c95f00604695a436e0"),
        "Title" : "The Hobbit:An Unexpected Journey",
        "Writer" : "J.R.R Tolkein",
        "Year" : "2012",
        "Franchise" : "The Hobbit"
}
{
        "_id" : ObjectId("620395fb5f00604695a436e1"),
        "Title" : "The Hobbit: The Desolation of Smaug",
        "Writer" : "J.R.R Tolkein",
        "Year" : "2013",
        "Franchise" : "The Hobbit"
}
{
        "_id" : ObjectId("620396815f00604695a436e2"),
        "Title" : "The Hobbit: The Battle of the Five Armies ",
```



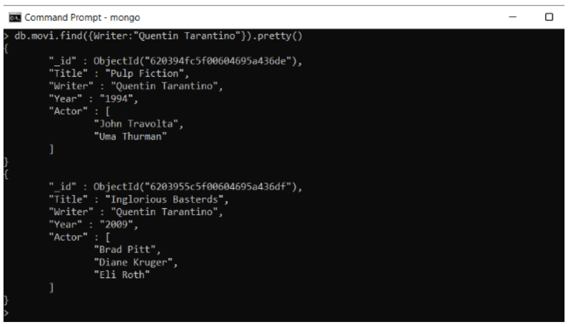
```
{
        "_id" : ObjectId("620395c95f00604695a436e0"),
        "Title" : "The Hobbit:An Unexpected Journey",
        "Writer" : "J.R.R Tolkein",
        "Year" : "2012",
        "Franchise" : "The Hobbit"
}
{
        "_id" : ObjectId("620395fb5f00604695a436e1"),
        "Title" : "The Hobbit: The Desolation of Smaug",
        "Writer" : "J.R.R Tolkein",
        "Year" : "2013",
        "Franchise" : "The Hobbit"
}
{
        "_id" : ObjectId("620396815f00604695a436e2"),
        "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."
}
{
        "_id" : ObjectId("620396a15f00604695a436e3"),
        "Title" : "Pee Wee Herman's Big Adventure"
}
{ "_id" : ObjectId("620396ad5f00604695a436e4"), "Title" : "Avatar" }
> _
```

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



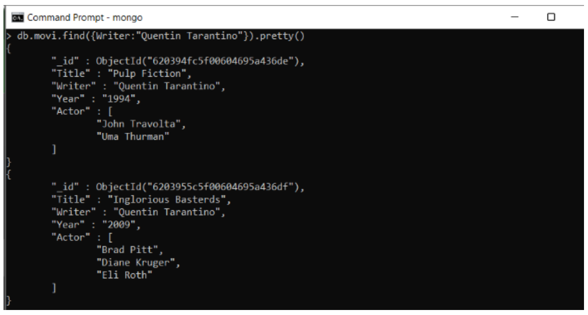
```
> db.movi.find({Writer:"Quentin Tarantino"}).pretty()
{
        "_id" : ObjectId("620394fc5f00604695a436de"),
        "Title" : "Pulp Fiction",
        "Writer" : "Quentin Tarantino",
        "Year" : "1994",
        "Actor" : [
                "John Travolta",
                "Uma Thurman"
        ]
}
{
        "_id" : ObjectId("6203955c5f00604695a436df"),
        "Title" : "Inglorious Basterds",
        "Writer" : "Quentin Tarantino",
        "Year" : "2009",
        "Actor" : [
                "Brad Pitt",
                "Diane Kruger",
                "Eli Roth"
        ]
}
>
```

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



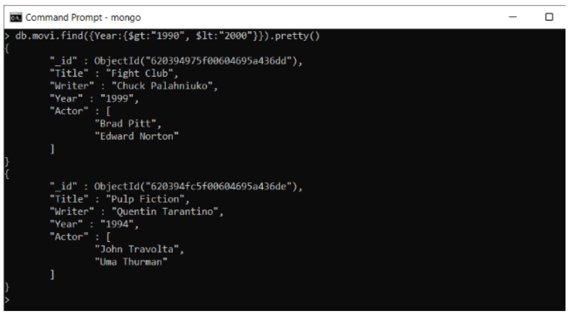
```
> db.movi.find({Writer:"Quentin Tarantino"}).pretty()
{
        "_id" : ObjectId("620394fc5f00604695a436de"),
        "Title" : "Pulp Fiction",
        "Writer" : "Quentin Tarantino",
        "Year" : "1994",
        "Actor" : [
                "John Travolta",
                "Uma Thurman"
        ]
}
{
        "_id" : ObjectId("6203955c5f00604695a436df"),
        "Title" : "Inglorious Basterds",
        "Writer" : "Quentin Tarantino",
        "Year" : "2009",
        "Actor" : [
                "Brad Pitt",
                "Diane Kruger",
                "Eli Roth"
        ]
}
>
```

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



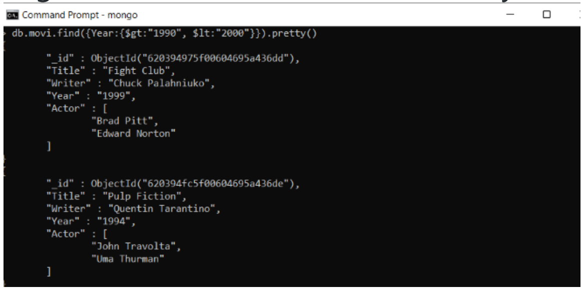
```
> db.movi.find({Writer:"Quentin Tarantino"}).pretty()
{
        "_id" : ObjectId("620394fc5f00604695a436de"),
        "Title" : "Pulp Fiction",
        "Writer" : "Quentin Tarantino",
        "Year" : "1994",
        "Actor" : [
                "John Travolta",
                "Uma Thurman"
        ]
}
{
        "_id" : ObjectId("6203955c5f00604695a436df"),
        "Title" : "Inglorious Basterds",
        "Writer" : "Quentin Tarantino",
        "Year" : "2009",
        "Actor" : [
                "Brad Pitt",
                "Diane Kruger",
                "Eli Roth"
        ]
}
```

5. get all movies released in the 90s



```
Command Prompt - mongo
> db.movi.find({Year:{$gt:"1990", $lt:"2000"}}).pretty()
{
        "_id" : ObjectId("620394975f00604695a436dd"),
        "Title" : "Fight Club",
        "Writer" : "Chuck Palahniuko",
        "Year" : "1999",
        "Actor" : [
                "Brad Pitt",
                "Edward Norton"
        ]
}
{
        "_id" : ObjectId("620394fc5f00604695a436de"),
        "Title" : "Pulp Fiction",
        "Writer" : "Quentin Tarantino",
        "Year" : "1994",
        "Actor" : [
                "John Travolta",
                "Uma Thurman"
        ]
}
>
```

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



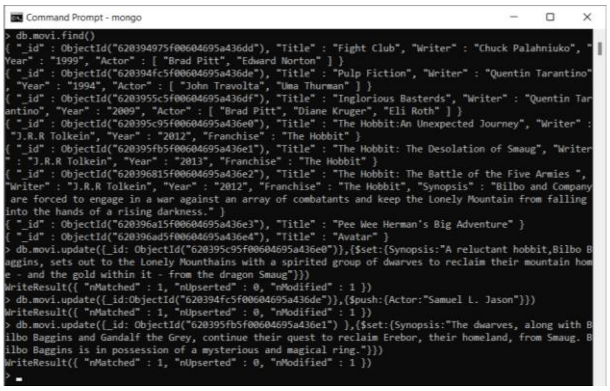
```
Command Prompt - mongo
> db.movi.find({Year:{$gt:"1990", $lt:"2000"}}).pretty()
{
        "_id" : ObjectId("620394975f00604695a436dd"),
        "Title" : "Fight Club",
        "Writer" : "Chuck Palahniuko",
        "Year" : "1999",
        "Actor" : [
                "Brad Pitt",
                "Edward Norton"
        ]
}
{
        "_id" : ObjectId("620394fc5f00604695a436de"),
        "Title" : "Pulp Fiction",
        "Writer" : "Quentin Tarantino",
        "Year" : "1994",
        "Actor" : [
                "John Travolta",
                "Uma Thurman"
        ]
```

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."

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"

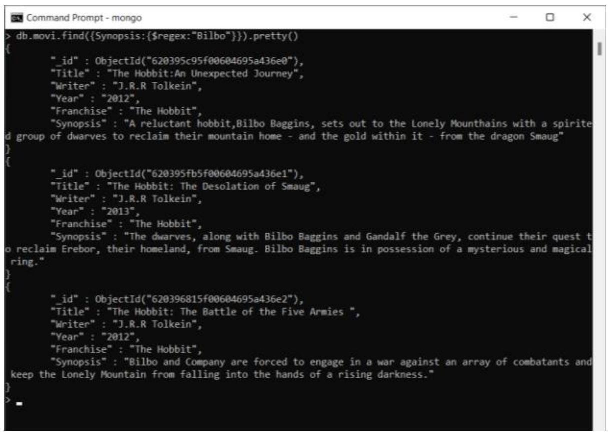


```
Command Prompt - mongo
> db.movi.find()
{ "_id" : ObjectId("620394975f00604695a436dd"), "Title" : "Fight Club", "Writer" : "Chuck Palahniuko", "
Year" : "1999", "Actor" : [ "Brad Pitt", "Edward Norton" ] }
{ "_id" : ObjectId("620394fc5f00604695a436de"), "Title" : "Pulp Fiction", "Writer" : "Quentin Tarantino"
, "Year" : "1994", "Actor" : [ "John Travolta", "Uma Thurman" ] }
{ "_id" : ObjectId("6203955c5f00604695a436df"), "Title" : "Inglorious Basterds", "Writer" : "Quentin Tar
antino", "Year" : "2009", "Actor" : [ "Brad Pitt", "Diane Kruger", "Eli Roth" ] }
{ "_id" : ObjectId("620395c95f00604695a436e0"), "Title" : "The Hobbit:An Unexpected Journey", "Writer" :
 "J.R.R Tolkein", "Year" : "2012", "Franchise" : "The Hobbit" }
{ "_id" : ObjectId("620395fb5f00604695a436e1"), "Title" : "The Hobbit: The Desolation of Smaug", "Writer
" : "J.R.R Tolkein", "Year" : "2013", "Franchise" : "The Hobbit" }
{ "_id" : ObjectId("620396815f00604695a436e2"), "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." }
{ "_id" : ObjectId("620396a15f00604695a436e3"), "Title" : "Pee Wee Herman's Big Adventure" }
{ "_id" : ObjectId("620396ad5f00604695a436e4"), "Title" : "Avatar" }
> db.movi.update({_id: ObjectId("620395c95f00604695a436e0")},{$set:{Synopsis:"A reluctant hobbit,Bilbo B
aggins, sets out to the Lonely Mounthains with a spirited group of dwarves to reclaim their mountain hom
e - and the gold within it - from the dragon Smaug"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.movi.update({_id:ObjectId("620394fc5f00604695a436de")},{$push:{Actor:"Samuel L. Jason"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.movi.update({_id: ObjectId("620395fb5f00604695a436e1") },{$set:{Synopsis:"The dwarves, along with B
ilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. B
ilbo Baggins is in possession of a mysterious and magical ring."}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> _
```

Reference: https://www.tutorialspoint.com/mongodb/mongodb_update_document.htm

Text Search

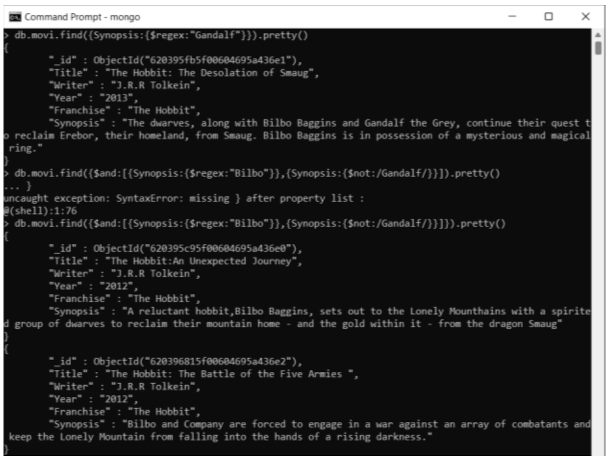
1. find all movies that have a synopsis that contains the word "Bilbo"



```
> db.movi.find({Synopsis:{$regex:"Bilbo"}}).pretty()
{
        "_id" : ObjectId("620395c95f00604695a436e0"),
        "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 Mounthains with a spirite
d group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug"
}
{
        "_id" : ObjectId("620395fb5f00604695a436e1"),
        "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 t
o reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical
 ring."
}
{
        "_id" : ObjectId("620396815f00604695a436e2"),
        "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."
}
> 
```

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

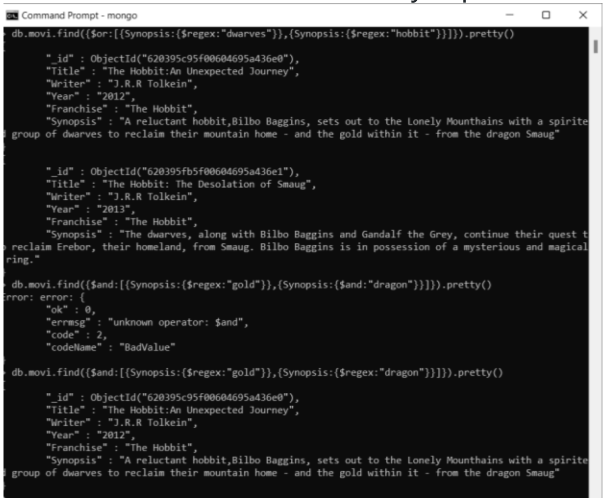
3. find all movies that have a synopsis that contains the word "Bilbo" and not the word "Gandalf"



```
> db.movi.find({Synopsis:{$regex:"Gandalf"}}).pretty()
{
        "_id" : ObjectId("620395fb5f00604695a436e1"),
        "Title" : "The Hobbit: The Desolation of Smaug",
        "Writer" : "J.R.R Tolkein",
        "Year" : "2013",
        "Franchise" : "The Hobbit",
        "Synopsis" : "The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest t
o reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical
 ring."
}
> db.movi.find({$and:[{Synopsis:{$regex:"Bilbo"}},{Synopsis:{$not:/Gandalf/}}]).pretty()
... }
uncaught exception: SyntaxError: missing } after property list :
@(shell):1:76
> db.movi.find({$and:[{Synopsis:{$regex:"Bilbo"}},{Synopsis:{$not:/Gandalf/}}]}).pretty()
{
        "_id" : ObjectId("620395c95f00604695a436e0"),
        "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 Mounthains with a spirite
d group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug"
}
{
        "_id" : ObjectId("620396815f00604695a436e2"),
        "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."
}
```

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"



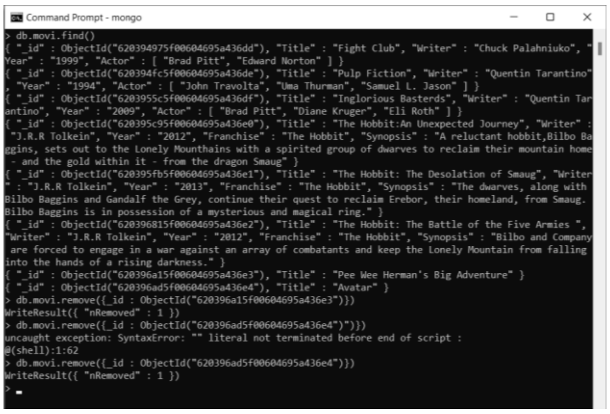
```
 db.movi.find({$or:[{Synopsis:{$regex:"dwarves"}},{Synopsis:{$regex:"hobbit"}}]}).pretty()

        "_id" : ObjectId("620395c95f00604695a436e0"),
        "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 Mounthains with a spirite
 group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug"


        "_id" : ObjectId("620395fb5f00604695a436e1"),
        "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 t
 reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical
ring."

 db.movi.find({$and:[{Synopsis:{$regex:"gold"}},{Synopsis:{$and:"dragon"}}]}).pretty()
rror: error: {
        "ok" : 0,
        "errmsg" : "unknown operator: $and",
        "code" : 2,
        "codeName" : "BadValue"

 db.movi.find({$and:[{Synopsis:{$regex:"gold"}},{Synopsis:{$regex:"dragon"}}]}).pretty()

        "_id" : ObjectId("620395c95f00604695a436e0"),
        "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 Mounthains with a spirite
 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.htm

Delete Documents

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

2. delete the movie "Avatar"



```
> db.movi.find()
{ "_id" : ObjectId("620394975f00604695a436dd"), "Title" : "Fight Club", "Writer" : "Chuck Palahniuko", "
Year" : "1999", "Actor" : [ "Brad Pitt", "Edward Norton" ] }
{ "_id" : ObjectId("620394fc5f00604695a436de"), "Title" : "Pulp Fiction", "Writer" : "Quentin Tarantino"
, "Year" : "1994", "Actor" : [ "John Travolta", "Uma Thurman", "Samuel L. Jason" ] }
{ "_id" : ObjectId("6203955c5f00604695a436df"), "Title" : "Inglorious Basterds", "Writer" : "Quentin Tar
antino", "Year" : "2009", "Actor" : [ "Brad Pitt", "Diane Kruger", "Eli Roth" ] }
{ "_id" : ObjectId("620395c95f00604695a436e0"), "Title" : "The Hobbit:An Unexpected Journey", "Writer" :
 "J.R.R Tolkein", "Year" : "2012", "Franchise" : "The Hobbit", "Synopsis" : "A reluctant hobbit,Bilbo Ba
ggins, sets out to the Lonely Mounthains with a spirited group of dwarves to reclaim their mountain home
 - and the gold within it - from the dragon Smaug" }
{ "_id" : ObjectId("620395fb5f00604695a436e1"), "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("620396815f00604695a436e2"), "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." }
{ "_id" : ObjectId("620396a15f00604695a436e3"), "Title" : "Pee Wee Herman's Big Adventure" }
{ "_id" : ObjectId("620396ad5f00604695a436e4"), "Title" : "Avatar" }
> db.movi.remove({_id : ObjectId("620396a15f00604695a436e3")})
WriteResult({ "nRemoved" : 1 })
> db.movi.remove({_id : ObjectId("620396ad5f00604695a436e4")")})
uncaught exception: SyntaxError: "" literal not terminated before end of script :
@(shell):1:62
> db.movi.remove({_id : ObjectId("620396ad5f00604695a436e4")})
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"

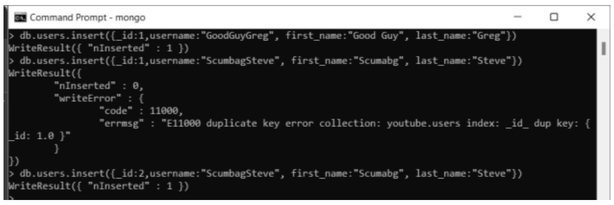
last_name : "Greg"

username : ScumbagSteve

full_name :

first : "Scumbag"

last : "Steve"



```
> db.users.insert({_id:1,username:"GoodGuyGreg", first_name:"Good Guy", last_name:"Greg"})
WriteResult({ "nInserted" : 1 })
> db.users.insert({_id:1,username:"ScumbagSteve", first_name:"Scumabg", last_name:"Steve"})
WriteResult({
        "nInserted" : 0,
        "writeError" : {
                "code" : 11000,
                "errmsg" : "E11000 duplicate key error collection: youtube.users index: _id_ dup key: {
_id: 1.0 }"
        }
})
> db.users.insert({_id:2,username:"ScumbagSteve", first_name:"Scumabg", last_name:"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

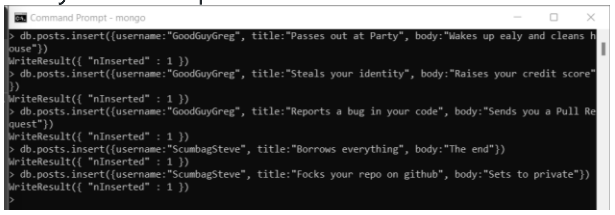
username : ScumbagSteve

title : Borrows something

body : Sells it

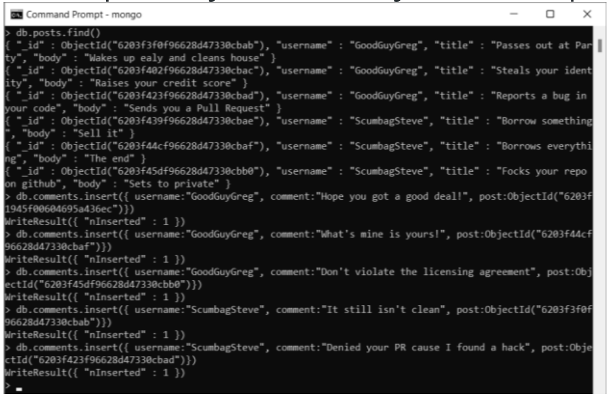
username : ScumbagSteve

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:"Wakes up ealy and cleans h
ouse"})
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 Re
quest"})
WriteResult({ "nInserted" : 1 })
> db.posts.insert({username:"ScumbagSteve", title:"Borrows everything", body:"The end"})
WriteResult({ "nInserted" : 1 })
> db.posts.insert({username:"ScumbagSteve", title:"Focks your repo on github", body:"Sets to private"})
WriteResult({ "nInserted" : 1 })
>
```

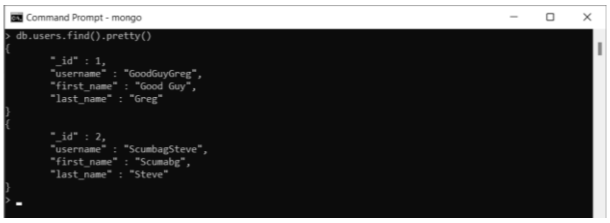
Insert the following documents into a **comments** collection
username : GoodGuyGreg
comment : Hope you got a good deal!
post : [post_obj_id]
where [post_obj_id] is the ObjectId of the posts document: "Borrows something"
username : GoodGuyGreg
comment : What's mine is yours!
post : [post_obj_id]
where [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.posts.find()
{ "_id" : ObjectId("6203f3f0f96628d47330cbab"), "username" : "GoodGuyGreg", "title" : "Passes out at Par
ty", "body" : "Wakes up ealy and cleans house" }
{ "_id" : ObjectId("6203f402f96628d47330cbac"), "username" : "GoodGuyGreg", "title" : "Steals your ident
ity", "body" : "Raises your credit score" }
{ "_id" : ObjectId("6203f423f96628d47330cbad"), "username" : "GoodGuyGreg", "title" : "Reports a bug in
your code", "body" : "Sends you a Pull Request" }
{ "_id" : ObjectId("6203f439f96628d47330cbae"), "username" : "ScumbagSteve", "title" : "Borrow something
", "body" : "Sell it" }
{ "_id" : ObjectId("6203f44cf96628d47330cbaf"), "username" : "ScumbagSteve", "title" : "Borrows everythi
ng", "body" : "The end" }
{ "_id" : ObjectId("6203f45df96628d47330cbb0"), "username" : "ScumbagSteve", "title" : "Focks your repo
on github", "body" : "Sets to private" }
> db.comments.insert({ username:"GoodGuyGreg", comment:"Hope you got a good deal!", post:ObjectId("6203f
1945f00604695a436ec")})
WriteResult({ "nInserted" : 1 })
> db.comments.insert({ username:"GoodGuyGreg", comment:"What's mine is yours!", post:ObjectId("6203f44cf
96628d47330cbaf")})
WriteResult({ "nInserted" : 1 })
> db.comments.insert({ username:"GoodGuyGreg", comment:"Don't violate the licensing agreement", post:Obj
ectId("6203f45df96628d47330cbb0")})
WriteResult({ "nInserted" : 1 })
> db.comments.insert({ username:"ScumbagSteve", comment:"It still isn't clean", post:ObjectId("6203f3f0f
96628d47330cbab")})
WriteResult({ "nInserted" : 1 })
> db.comments.insert({ username:"ScumbagSteve", comment:"Denied your PR cause I found a hack", post:Obje
ctId("6203f423f96628d47330cbad")})
WriteResult({ "nInserted" : 1 })
> _
```

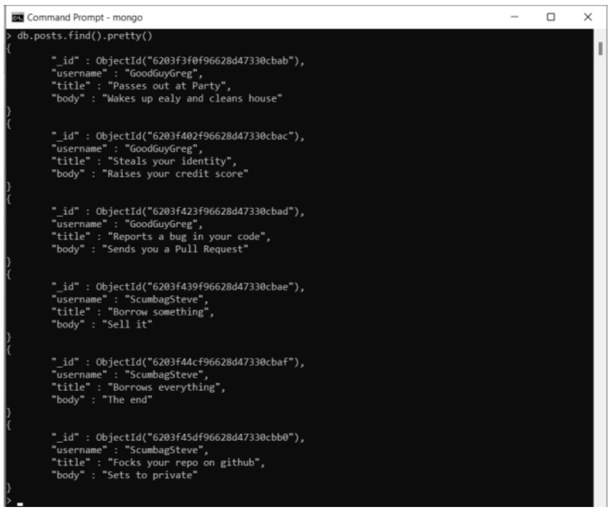
Querying related collections

1. find all users



```
> db.users.find().pretty()
{
        "_id" : 1,
        "username" : "GoodGuyGreg",
        "first_name" : "Good Guy",
        "last_name" : "Greg"
}
{
        "_id" : 2,
        "username" : "ScumbagSteve",
        "first_name" : "Scumabg",
        "last_name" : "Steve"
}
> 
```

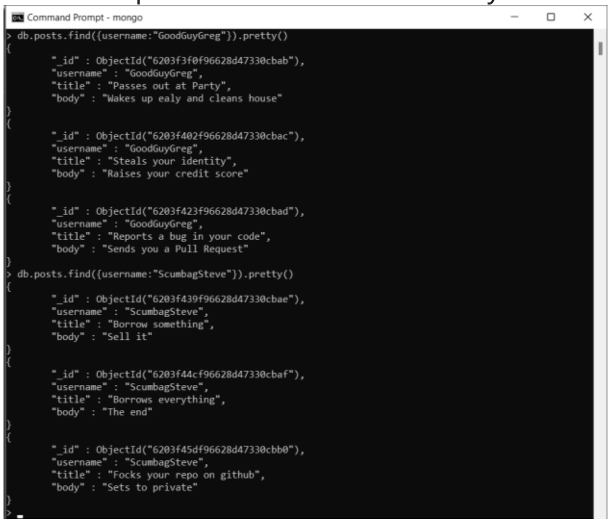
2. find all posts



```
> db.posts.find().pretty()
{
        "_id" : ObjectId("6203f3f0f96628d47330cbab"),
        "username" : "GoodGuyGreg",
        "title" : "Passes out at Party",
        "body" : "Wakes up ealy and cleans house"
}
{
        "_id" : ObjectId("6203f402f96628d47330cbac"),
        "username" : "GoodGuyGreg",
        "title" : "Steals your identity",
        "body" : "Raises your credit score"
}
{
        "_id" : ObjectId("6203f423f96628d47330cbad"),
        "username" : "GoodGuyGreg",
        "title" : "Reports a bug in your code",
        "body" : "Sends you a Pull Request"
}
{
        "_id" : ObjectId("6203f439f96628d47330cbae"),
        "username" : "ScumbagSteve",
        "title" : "Borrow something",
        "body" : "Sell it"
}
{
        "_id" : ObjectId("6203f44cf96628d47330cbaf"),
        "username" : "ScumbagSteve",
        "title" : "Borrows everything",
        "body" : "The end"
}
{
        "_id" : ObjectId("6203f45df96628d47330cbb0"),
        "username" : "ScumbagSteve",
        "title" : "Focks your repo on github",
        "body" : "Sets to private"
}
> 
```

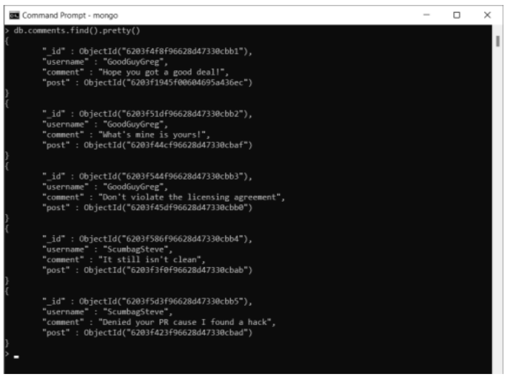
3. find all posts that was authored by "GoodGuyGreg"

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



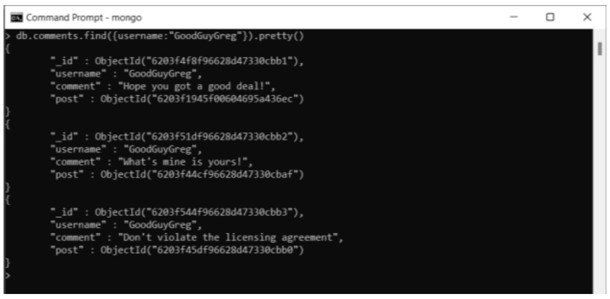
```
> db.posts.find({username:"GoodGuyGreg"}).pretty()
{
        "_id" : ObjectId("6203f3f0f96628d47330cbab"),
        "username" : "GoodGuyGreg",
        "title" : "Passes out at Party",
        "body" : "Wakes up ealy and cleans house"
}
{
        "_id" : ObjectId("6203f402f96628d47330cbac"),
        "username" : "GoodGuyGreg",
        "title" : "Steals your identity",
        "body" : "Raises your credit score"
}
{
        "_id" : ObjectId("6203f423f96628d47330cbad"),
        "username" : "GoodGuyGreg",
        "title" : "Reports a bug in your code",
        "body" : "Sends you a Pull Request"
}
> db.posts.find({username:"ScumbagSteve"}).pretty()
{
        "_id" : ObjectId("6203f439f96628d47330cbae"),
        "username" : "ScumbagSteve",
        "title" : "Borrow something",
        "body" : "Sell it"
}
{
        "_id" : ObjectId("6203f44cf96628d47330cbaf"),
        "username" : "ScumbagSteve",
        "title" : "Borrows everything",
        "body" : "The end"
}
{
        "_id" : ObjectId("6203f45df96628d47330cbb0"),
        "username" : "ScumbagSteve",
        "title" : "Focks your repo on github",
        "body" : "Sets to private"
}
> 
```

5. find all comments



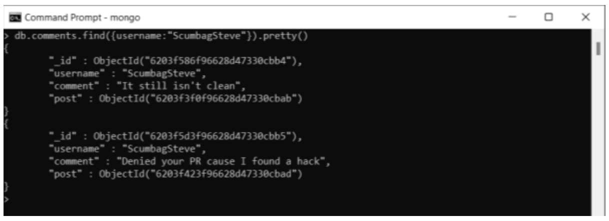
```
> db.comments.find().pretty()
{
        "_id" : ObjectId("6203f4f8f96628d47330cbb1"),
        "username" : "GoodGuyGreg",
        "comment" : "Hope you got a good deal!",
        "post" : ObjectId("6203f1945f00604695a436ec")
}
{
        "_id" : ObjectId("6203f51df96628d47330cbb2"),
        "username" : "GoodGuyGreg",
        "comment" : "What's mine is yours!",
        "post" : ObjectId("6203f44cf96628d47330cbaf")
}
{
        "_id" : ObjectId("6203f544f96628d47330cbb3"),
        "username" : "GoodGuyGreg",
        "comment" : "Don't violate the licensing agreement",
        "post" : ObjectId("6203f45df96628d47330cbb0")
}
{
        "_id" : ObjectId("6203f586f96628d47330cbb4"),
        "username" : "ScumbagSteve",
        "comment" : "It still isn't clean",
        "post" : ObjectId("6203f3f0f96628d47330cbab")
}
{
        "_id" : ObjectId("6203f5d3f96628d47330cbb5"),
        "username" : "ScumbagSteve",
        "comment" : "Denied your PR cause I found a hack",
        "post" : ObjectId("6203f423f96628d47330cbad")
}
> 
```

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



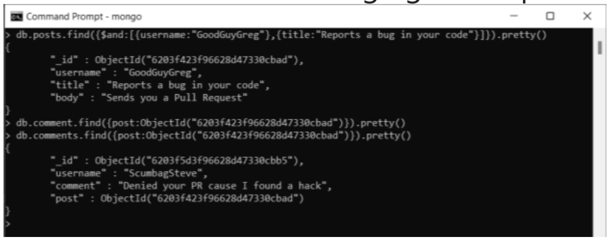
```
> db.comments.find({username:"GoodGuyGreg"}).pretty()
{
        "_id" : ObjectId("6203f4f8f96628d47330cbb1"),
        "username" : "GoodGuyGreg",
        "comment" : "Hope you got a good deal!",
        "post" : ObjectId("6203f1945f00604695a436ec")
}
{
        "_id" : ObjectId("6203f51df96628d47330cbb2"),
        "username" : "GoodGuyGreg",
        "comment" : "What's mine is yours!",
        "post" : ObjectId("6203f44cf96628d47330cbaf")
}
{
        "_id" : ObjectId("6203f544f96628d47330cbb3"),
        "username" : "GoodGuyGreg",
        "comment" : "Don't violate the licensing agreement",
        "post" : ObjectId("6203f45df96628d47330cbb0")
}
>
```

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



```
> db.comments.find({username:"ScumbagSteve"}).pretty()
{
        "_id" : ObjectId("6203f586f96628d47330cbb4"),
        "username" : "ScumbagSteve",
        "comment" : "It still isn't clean",
        "post" : ObjectId("6203f3f0f96628d47330cbab")
}
{
        "_id" : ObjectId("6203f5d3f96628d47330cbb5"),
        "username" : "ScumbagSteve",
        "comment" : "Denied your PR cause I found a hack",
        "post" : ObjectId("6203f423f96628d47330cbad")
}
>
```

8. find all comments belonging to the post "Reports a bug in your code"



```
> db.posts.find({$and:[{username:"GoodGuyGreg"},{title:"Reports a bug in your code"}]}).pretty()
{
        "_id" : ObjectId("6203f423f96628d47330cbad"),
        "username" : "GoodGuyGreg",
        "title" : "Reports a bug in your code",
        "body" : "Sends you a Pull Request"
}
> db.comment.find({post:ObjectId("6203f423f96628d47330cbad")}).pretty()
> db.comments.find({post:ObjectId("6203f423f96628d47330cbad")}).pretty()
{
        "_id" : ObjectId("6203f5d3f96628d47330cbb5"),
        "username" : "ScumbagSteve",
        "comment" : "Denied your PR cause I found a hack",
        "post" : ObjectId("6203f423f96628d47330cbad")
}
>
```

References: https://docs.mongodb.com/manual/reference/method/db.collection.find/
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@