

Assignment 1

MongoDB Exercise in mongo shell

SHAIK MOHSIN S

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

```
C:\Windows\System32\cmd.exe - mongo
Microsoft Windows [Version 10.0.19044.1503]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Shahk Mohidin\Desktop>mongodb-windows-x86_64-5.0.6\mongodb-win32-x86_64-windows-5.0.6\bin>mongo
MongoDB shell version v5.0.6
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("0bee5274-elb6-4bee-9307-37bf6d356eab") }
MongoDB server version: 5.0.6

*****
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility. The "mongo" shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
*****

The server generated these startup warnings when booting:
  2022-02-07T16:43:59.559+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
  ---
  ---
  Enable MongoDB's free cloud-based monitoring service, which will then receive and display
  metrics about your deployment (disk utilization, CPU, operation statistics, etc).

  The monitoring data will be available on a MongoDB website with a unique URL accessible to you
  and anyone you share the URL with. MongoDB may use this information to make product
  improvements and to suggest MongoDB products and deployment options to you.

  To enable free monitoring, run the following command: db.enableFreeMonitoring()
  To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
  ---

use mongo_practice
switched to db mongo_practice
> db.movies.insert({title:"Fight Club", writer: " Chuck Palahniuk", year: "1999", actors:["Brad Pitt", "Edward Norton"]})
WriteResult({ "nInserted" : 1 })
> db.movies.insert({title:"Pulp Fiction", writer: "Quentin Tarantino", year: "2009", actors:["John Travolta", "Uma Thurman"]})
WriteResult({ "nInserted" : 1 })
> 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. Tolkein", year: "2013", franchise:"The Hobbit"})
WriteResult({ "nInserted" : 1 })
> db.movies.insert({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."})
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

```
C:\Windows\System32\cmd.exe - mongo
> db.movies.insert({title:"Avatar"})
writeResult({ "nInserted" : 1 })
> db.movies.find()
{ "_id" : ObjectId("62029fae095cd380e29c13bdd"), "title" : "Fight Club", "writer" : " Chuck Palahniuk", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }
{ "_id" : ObjectId("62029fae095cd380e29c13bdd"), "title" : "Pulp Fiction", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "John Travolta", "Uma Thurman" ] }
{ "_id" : ObjectId("6202a22a095cd380e29c13bdf"), "title" : "Inglorious Bastards", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "Brad Pitt", "Diane Kruger", "Eli Roth" ] }
{ "_id" : ObjectId("6202a397095cd380e29c13be0"), "title" : "The Hobbit:An unexpected Journey", "writer" : "J.R.R. Tolkein", "year" : "2012", "franchise" : "The Hobbit" }
{ "_id" : ObjectId("6202a40c095cd380e29c13be1"), "title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R. Tolkein", "year" : "2013", "franchise" : "The Hobbit" }
{ "_id" : ObjectId("6202a522095cd380e29c13be2"), "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("6202a5f6095cd380e29c13be3"), "title" : "Pee Wee Herman's Big Adventures" }
{ "_id" : ObjectId("6202a60d095cd380e29c13be4"), "title" : "Avatar" }
> db.movies.find({writer:"Quentin Tarantino"})
{ "_id" : ObjectId("6202a0d0095cd380e29c13bde"), "title" : "Pulp Fiction", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "John Travolta", "Uma Thurman" ] }
{ "_id" : ObjectId("6202a22a095cd380e29c13bdf"), "title" : "Inglorious Bastards", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "Brad Pitt", "Diane Kruger", "Eli Roth" ] }
> db.movies.find({actors:"Brad Pitt"})
{ "_id" : ObjectId("62029fae095cd380e29c13bdd"), "title" : "Fight Club", "writer" : " Chuck Palahniuk", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }
{ "_id" : ObjectId("6202a22a095cd380e29c13bdf"), "title" : "Inglorious Bastards", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "Brad Pitt", "Diane Kruger", "Eli Roth" ] }
> db.movies.find({franchise:"The Hobbit"})
{ "_id" : ObjectId("6202a397095cd380e29c13be0"), "title" : "The Hobbit:An unexpected Journey", "writer" : "J.R.R. Tolkein", "year" : "2012", "franchise" : "The Hobbit" }
{ "_id" : ObjectId("6202a40c095cd380e29c13be1"), "title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R. Tolkein", "year" : "2013", "franchise" : "The Hobbit" }
{ "_id" : ObjectId("6202a522095cd380e29c13be2"), "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." }
> db.movies.find({$gt:"1990",$lt:"2000"})
Error: error: {
  "ok" : 0,
  "errmsg" : "unknown top level operator: $gt. If you have a field name that starts with a '$' symbol, consider using $getField or $setfield.",
  "code" : 2,
  "codeName" : "BadValue"
}
> db.movies.find(year:{$gt:"1990",$lt:"2000"})
Uncaught exception: SyntaxError: missing ) after argument list :
$g(shell):1:19
> db.movies.find(year:{$gt:"1990",$lt:"2000"})
{ "_id" : ObjectId("62029fae095cd380e29c13bdd"), "title" : "Fight Club", "writer" : " Chuck Palahniuk", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }
> db.movies.find({$or:{{year:{$gt:"2010"}},{year:{$lt:"2000"}}}})
{ "_id" : ObjectId("62029fae095cd380e29c13bdd"), "title" : "Fight Club", "writer" : " Chuck Palahniuk", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }
{ "_id" : ObjectId("6202a397095cd380e29c13be0"), "title" : "The Hobbit:An unexpected Journey", "writer" : "J.R.R. Tolkein", "year" : "2012", "franchise" : "The Hobbit" }
{ "_id" : ObjectId("6202a40c095cd380e29c13be1"), "title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R. Tolkein", "year" : "2013", "franchise" : "The Hobbit" }
{ "_id" : ObjectId("6202a522095cd380e29c13be2"), "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." }
>
```

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"

```
C:\Windows\System32\cmd.exe - mongo
{ "id" : ObjectId("6202a40c05cd380e29c13be1"), "title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R. Tolkien", "year" : "2013", "franchise" : "The Hobbit" }
{ "id" : ObjectId("6202a52205cd380e29c13be2"), "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 engage in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of a rising darkness." }
> db.movies.update({'_id':ObjectId("6202a39705cd380e29c13be0")}, {$set:{synopsis:"A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home and the gold within it - from the dragon Smaug."}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.movies.update({'_id':ObjectId("6202a40c05cd380e29c13be1")}, {$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."}})
Error: invalid object id: length :
@shell:1:23
> db.movies.update({'_id':ObjectId("6202a40c05cd380e29c13be1")}, {$set:{synopsis:"The dwarves along with Bilbo Baggins and Gandalf the Grey,continue their quest to reclaim Erebor,their homeland,from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."}})
Error: invalid object id: length :
@shell:1:23
> db.movies.update({'_id':ObjectId("6202a40c05cd380e29c13be1")}, {$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.update({'_id':ObjectId("6202a0d005cd380e29c13bde")}, {$push:{actors:"Samuel L. Jackson"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

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"

```
C:\Windows\System32\cmd.exe - mongo
writeResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.movies.update({'_id:ObjectId("6202a0d005cd380e29c13bde")'}, {$push:{actors:"Samuel L. Jackson"}})
writeResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.movies.find({'synopsis':{'regex':"Bilbo"}})
( { "_id" : ObjectId("6202a39705cd380e29c13be0"), "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 drgon Smaug." }
( { "_id" : ObjectId("6202a4d0c05cd380e29c13be1"), "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 Erebon,their homeland,from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring." }
( { "_id" : ObjectId("6202a52205cd380e29c13be2"), "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." }
> db.movies.find({'synopsis':{'regex':"Gandalf"}})
( { "_id" : ObjectId("6202a4d0c05cd380e29c13be1"), "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 Erebon,their homeland,from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring." }
> db.movies.find({'and':{'synopsis':{'regex':"Bilbo"}},{synopsis:{!not:/Gandalf/}})})
( { "_id" : ObjectId("6202a39705cd380e29c13be0"), "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 drgon Smaug." }
( { "_id" : ObjectId("6202a52205cd380e29c13be2"), "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." }
> db.movies.find({'or':{'synopsis':{'regex':"dwarves"}},{synopsis:{!regex:"hobbit"}})})
( { "_id" : ObjectId("6202a39705cd380e29c13be0"), "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 drgon Smaug." }
( { "_id" : ObjectId("6202a4d0c05cd380e29c13be1"), "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 Erebon,their homeland,from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring." }
> db.movies.find({'and':{'synopsis':{'regex':"gold"}},{synopsis:{!regex:"dragon"}})})
> db.movies.find({'and':{'synopsis':{'regex':"gold"}},{synopsis:{!regex:"dragon"}})})
> db.movies.find({'or':{'synopsis':{'regex':"dwarves"}},{synopsis:{!regex:"hobbit"}})})
( { "_id" : ObjectId("6202a39705cd380e29c13be0"), "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 drgon Smaug." }
( { "_id" : ObjectId("6202a4d0c05cd380e29c13be1"), "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 Erebon,their homeland,from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring." }
> db.movies.find({'and':{'synopsis':{'regex':"gold"}},{synopsis:{!regex:"dragon"}})})
( { "_id" : ObjectId("6202a39705cd380e29c13be0"), "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 drgon Smaug." }
>
```

Delete Documents

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

```
C:\Windows\System32\cmd.exe - mongo
( { "_id" : ObjectId("6202a39705cd380e29c13be0"), "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 drgon Smaug." }
> db.movies.remove({'_id:ObjectId("6202a5fa05cd380e29c13be3")'})
writeResult({ "nRemoved" : 1 })
> db.movies.remove({'_id:ObjectId("6202a6d0d05cd380e29c13be4")'})
writeResult({ "nRemoved" : 1 })
>
```

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"

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

```

C:\Windows\System32\cmd.exe - mongo
mongo_practice 0.000GB
users          0.000GB
> db.users.insert(id:1,username:"GoodGuyGreg", first_name:"Good Guy", last_name:"Greg")
uncaught exception: SyntaxError: missing ) after argument list :
@shell:1:19
> use users
switched to db users
> db.users.insert(id:1,username:"GoodGuyGreg", first_name:"Good Guy", last_name:"Greg")
uncaught exception: SyntaxError: missing ) after argument list :
@shell:1:19
> use mongo_practice
switched to db mongo_practice
> db.users.insert(id:1,username:"GoodGuyGreg", first_name:"Good Guy", last_name:"Greg")
uncaught exception: SyntaxError: missing ) after argument list :
@shell:1:19
> db.createCollection("users")
{ "ok" : 1 }
> show collections
users
movies
> db.users.insert(id:1,username:"GoodGuyGreg", first_name:"Good Guy", last_name:"Greg")
uncaught exception: SyntaxError: missing ) after argument list :
@shell:1:19
> db.users.insert(id:2, username:"ScumbagSteve", fullname:{first: "Scumbag", last:"Steve"})
uncaught exception: SyntaxError: missing ) after argument list :
@shell:1:19
> db.post.insert({username:"GoodGuyGreg", title:"Passes out at Party", body:"Raises your credit score"})
WriteResult({"nInserted" : 1 })
> db.post.insert({username:"GoodGuyGreg", title:"Steals your identity", body:"Raises your credit score"})
WriteResult({"nInserted" : 1 })
> db.post.insert({username:"GoodGuyGreg", title:"Reports a bug in your code", body:"Sends you a pull request"})
WriteResult({"nInserted" : 1 })
> db.post.insert({username:"ScumbagSteve", title:"Borrows something", body:"Sells it"})
WriteResult({"nInserted" : 1 })
> db.post.insert({username:"ScumbagSteve", title:"Borrows everything", body:"The end"})
WriteResult({"nInserted" : 1 })
> db.post.insert({username:"ScumbagSteve", title:"Forkd your repo on github", body:"sets to private"})
WriteResult({"nInserted" : 1 })
>

```

```

C:\Windows\System32\cmd.exe - mongo
uncaught exception: SyntaxError: missing ) after argument list :
@shell:1:19
> db.post.insert({username:"GoodGuyGreg", title:"Passes out at Party", body:"Raises your credit score"})
WriteResult({"nInserted" : 1 })
> db.post.insert({username:"GoodGuyGreg", title:"Steals your identity", body:"Raises your credit score"})
WriteResult({"nInserted" : 1 })
> db.post.insert({username:"GoodGuyGreg", title:"Reports a bug in your code", body:"Sends you a pull request"})
WriteResult({"nInserted" : 1 })
> db.post.insert({username:"ScumbagSteve", title:"Borrows something", body:"Sells it"})
WriteResult({"nInserted" : 1 })
> db.post.insert({username:"ScumbagSteve", title:"Borrows everything", body:"The end"})
WriteResult({"nInserted" : 1 })
> db.post.insert({username:"ScumbagSteve", title:"Forkd 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

where [post_obj_id] is the ObjectId of the posts document: "Reports a bug in your Code"

```
C:\Windows\System32\cmd.exe - mongo
(kshell):1:18
> db.post.find()
{ "_id" : ObjectId("6202b86305cd380e29c13be5"), "username" : "GoodGuyGreg", "title" : "Passes out at Party", "body" : "Raises your credit score" }
{ "_id" : ObjectId("6202b89a95cd380e29c13be6"), "username" : "GoodGuyGreg", "title" : "Steals your identity", "body" : "Raises your credit score" }
{ "_id" : ObjectId("6202b8cf85cd380e29c13be7"), "username" : "GoodGuyGreg", "title" : "Reports a bug in your code", "body" : "Sends you a pull request" }
{ "_id" : ObjectId("6202b92505cd380e29c13be8"), "username" : "ScumbagSteve", "title" : "Borrows something", "body" : "Sells it" }
{ "_id" : ObjectId("6202b94a95cd380e29c13be9"), "username" : "ScumbagSteve", "title" : "Borrows everything", "body" : "The end" }
{ db.comments.insert({ username:"GoodGuyGrey", comment:"Hope you got a good deal", post:ObjectId("6202b92505cd380e29c13be8")})sets to private" }
writeResult({ "nInserted" : 1 })GoodGuyGrey", comment:"Hope you got a good deal", post:ObjectId("6202b92505cd380e29c13be8"))}
> db.comments.insert({username:"GoodGuyGrey", comment:"Hope you got a good deal", post:ObjectId("6202b92505cd380e29c13be8")})sets to private" }
writeResult({ "nInserted" : 1 })
> db.comments.insert({username:"GoodGuyGrey", comment:"What's mine is yours!", post:ObjectId("6202b94a95cd380e29c13be9")})db.comments.insert({username:"GoodGuyGrey", comment:"What's mine is yours!", post:ObjectId("6202b94a95cd380e29c13be9")})
writeResult({ "nInserted" : 1 })
> db.post.find()
{ "_id" : ObjectId("6202b86305cd380e29c13be5"), "username" : "GoodGuyGreg", "title" : "Passes out at Party", "body" : "Raises your credit score" }
{ "_id" : ObjectId("6202b89a95cd380e29c13be6"), "username" : "GoodGuyGreg", "title" : "Steals your identity", "body" : "Raises your credit score" }
{ "_id" : ObjectId("6202b8cf85cd380e29c13be7"), "username" : "GoodGuyGreg", "title" : "Reports a bug in your code", "body" : "Sends you a pull request" }
{ "_id" : ObjectId("6202b92505cd380e29c13be8"), "username" : "ScumbagSteve", "title" : "Borrows something", "body" : "Sells it" }
{ "_id" : ObjectId("6202b94a95cd380e29c13be9"), "username" : "ScumbagSteve", "title" : "Borrows everything", "body" : "The end" }
{ "_id" : ObjectId("6202b97905cd380e29c13bea"), "username" : "ScumbagSteve", "title" : "Forked your repo on github", "body" : "sets to private" }
{ db.comments.insert({username:"GoodGuyGreg", comment:"Don't violate the licensing agreement!", post:ObjectId("6202b97905cd380e29c13bea")})
writeResult({ "nInserted" : 1 })
> db.comments.insert({username:"ScumbagSteve", comment:"It still isn't clean", post:ObjectId("6202b86305cd380e29c13be5")})
writeResult({ "nInserted" : 1 })
> db.comments.insert({username:"ScumbagSteve", comment:"Denied your PR cause I found a hack", post:ObjectId("6202b8cf85cd380e29c13be7")})
writeResult({ "nInserted" : 1 })
```

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"

```

C:\Windows\System32\cmd.exe - mongo
writeResult({ "inserted" : 1 })
> db.users.find().pretty()
{
  "_id" : ObjectId("6202b86305cd380e29c13be5"),
  "username" : "GoodGuyGreg",
  "title" : "Passes out at Party",
  "body" : "Raises your credit score"
}

{
  "_id" : ObjectId("6202b89a05cd380e29c13be6"),
  "username" : "GoodGuyGreg",
  "title" : "Steals your identity",
  "body" : "Raises your credit score"
}

{
  "_id" : ObjectId("6202b8cf05cd380e29c13be7"),
  "username" : "GoodGuyGreg",
  "title" : "Reports a bug in your code",
  "body" : "Sends you a pull request"
}

{
  "_id" : ObjectId("6202b92505cd380e29c13be8"),
  "username" : "ScumbagSteve",
  "title" : "Borrows something",
  "body" : "Sells it"
}

{
  "_id" : ObjectId("6202b94a05cd380e29c13be9"),
  "username" : "ScumbagSteve",
  "title" : "Borrows everything",
  "body" : "The end"
}

{
  "_id" : ObjectId("6202b97905cd380e29c13bea"),
  "username" : "ScumbagSteve",
  "title" : "Forkd your repo on github",
  "body" : "sets to private"
}

db.post.find({username:"GoodGuyGreg"})
{
  "_id" : ObjectId("6202b86305cd380e29c13be5"), "username" : "GoodGuyGreg", "title" : "Passes out at Party", "body" : "Raises your credit score" }
{
  "_id" : ObjectId("6202b89a05cd380e29c13be6"), "username" : "GoodGuyGreg", "title" : "Steals your identity", "body" : "Raises your credit score" }
{
  "_id" : ObjectId("6202b8cf05cd380e29c13be7"), "username" : "GoodGuyGreg", "title" : "Reports a bug in your code", "body" : "Sends you a pull request" }
}
db.post.find({username:"ScumbagSteve"})
{
  "_id" : ObjectId("6202b92505cd380e29c13be8"), "username" : "ScumbagSteve", "title" : "Borrows something", "body" : "Sells it" }
{
  "_id" : ObjectId("6202b94a05cd380e29c13be9"), "username" : "ScumbagSteve", "title" : "Borrows everything", "body" : "The end" }
{
  "_id" : ObjectId("6202b97905cd380e29c13bea"), "username" : "ScumbagSteve", "title" : "Forkd your repo on github", "body" : "sets to private" }
}
db.comments.find().pretty()

```

```

C:\Windows\System32\cmd.exe - mongo
> db.post.find({username:"GoodGuyGreg"})
{
  "_id" : ObjectId("6202b86305cd380e29c13be5"), "username" : "GoodGuyGreg", "title" : "Passes out at Party", "body" : "Raises your credit score" }
{
  "_id" : ObjectId("6202b89a05cd380e29c13be6"), "username" : "GoodGuyGreg", "title" : "Steals your identity", "body" : "Raises your credit score" }
{
  "_id" : ObjectId("6202b8cf05cd380e29c13be7"), "username" : "GoodGuyGreg", "title" : "Reports a bug in your code", "body" : "Sends you a pull request" }
}
> db.post.find({username:"ScumbagSteve"})
{
  "_id" : ObjectId("6202b92505cd380e29c13be8"), "username" : "ScumbagSteve", "title" : "Borrows something", "body" : "Sells it" }
{
  "_id" : ObjectId("6202b94a05cd380e29c13be9"), "username" : "ScumbagSteve", "title" : "Borrows everything", "body" : "The end" }
{
  "_id" : ObjectId("6202b97905cd380e29c13bea"), "username" : "ScumbagSteve", "title" : "Forkd your repo on github", "body" : "sets to private" }
}
> db.comments.find().pretty()
{
  "_id" : ObjectId("6202bc1805cd380e29c13beb"),
  "username" : "GoodGuyGreg",
  "comment" : "Hope you got a good deal!",
  "post" : ObjectId("6202b92505cd380e29c13be8")
}

{
  "_id" : ObjectId("6202bc6f05cd380e29c13bec"),
  "username" : "GoodGuyGreg",
  "comment" : "Hope you got a good deal!",
  "post" : ObjectId("6202b92505cd380e29c13be8")
}

{
  "_id" : ObjectId("6202bd1005cd380e29c13bed"),
  "username" : "GoodGuyGreg",
  "comment" : "What's mine is yours!",
  "post" : ObjectId("6202b94a05cd380e29c13be9")
}

{
  "_id" : ObjectId("6202be6305cd380e29c13bee"),
  "username" : "GoodGuyGreg",
  "comment" : "Don't violate the licensing agreement!",
  "post" : ObjectId("6202b97905cd380e29c13bea")
}

{
  "_id" : ObjectId("6202bec305cd380e29c13bef"),
  "username" : "ScumbagSteve",
  "comment" : "It still isn't clean",
  "post" : ObjectId("6202b86305cd380e29c13be5")
}

{
  "_id" : ObjectId("6202bf4105cd380e29c13bfb"),
  "username" : "ScumbagSteve",
  "comment" : "Denied your PR cause I found a hack",
  "post" : ObjectId("6202b8cf05cd380e29c13be7")
}

db.comments.find({username:"GoodGuyGreg"})
{
  "_id" : ObjectId("6202bd1005cd380e29c13bed"), "username" : "GoodGuyGreg", "comment" : "What's mine is yours!", "post" : ObjectId("6202b94a05cd380e29c13be9") }
{
  "_id" : ObjectId("6202be6305cd380e29c13bee"), "username" : "GoodGuyGreg", "comment" : "Don't violate the licensing agreement!", "post" : ObjectId("6202b97905cd380e29c13bea") }
}
db.comments.find({username:"ScumbagSteve"})

```

