NOSQL & MONGODB ASSIGNMENT

PROMPT:

>mongo

>show dbs

>use mongo_practice

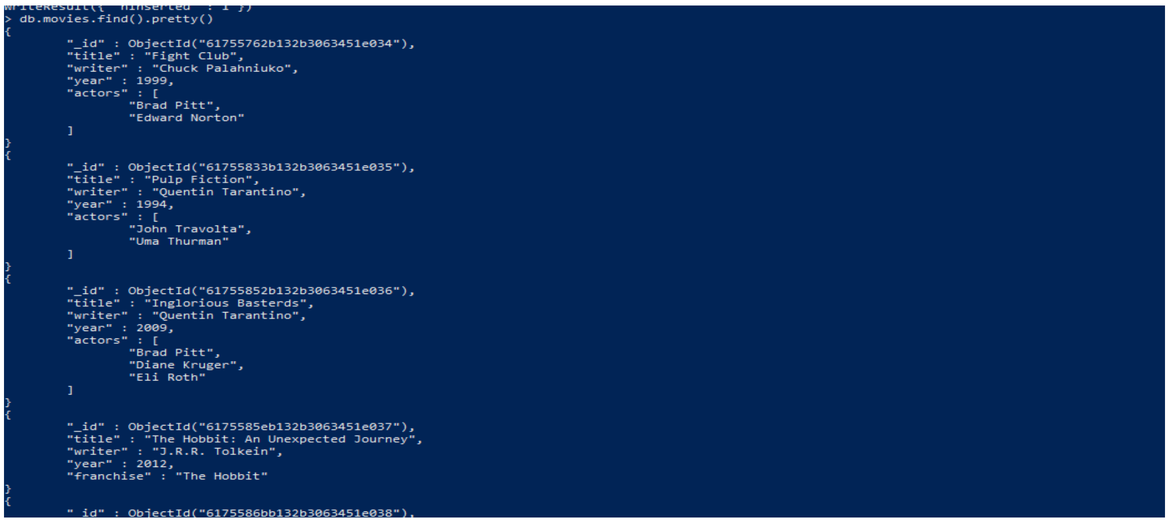
>db.createCollections('movies')

<u>INSERSTION:</u>

```
db.movies.insert({
title : "Fight Club",
writer : "Chuck Palahniuko",
year : 1999,
actors : ['Brad Pitt','Edward Norton']
})

db.movies.insert({
title : "Pulp Fiction",
writer : "Quentin Tarantino",
year : 1994,
actors : ['John Travolta','Uma Thurman']
})

db.movies.insert({
    title : "Inglorious Basterds",
    writer : "Quentin Tarantino",
    year : 2009,
    actors : ['Brad Pitt','Diane Kruger','Eli Roth']
})

db.movies.insert({
    title : "The Hobbit: An Unexpected Journey",
    writer : "J.R.R. Tolkein",
    year : 2012,
    franchise : "The Hobbit"
})

db.movies.insert({
    title : "The Hobbit: The Desolation of Smaug",
    writer : "J.R.R. Tolkein",
    year : 2013,
    franchise : "The Hobbit"

})

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."
})

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

db.movies.insert({
    title : "Avatar"
})
```

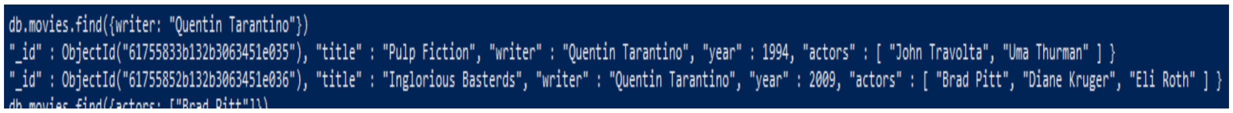
QUERY DOCUMENTS:

1. getall documents



```
WriteResult({ "nInserted" : 1 })
> db.movies.find().pretty()
{
        "_id" : ObjectId("61755762b132b3063451e034"),
        "title" : "Fight Club",
        "writer" : "Chuck Palahniuko",
        "year" : 1999,
        "actors" : [
                "Brad Pitt",
                "Edward Norton"
        ]
}
{
        "_id" : ObjectId("61755833b132b3063451e035"),
        "title" : "Pulp Fiction",
        "writer" : "Quentin Tarantino",
        "year" : 1994,
        "actors" : [
                "John Travolta",
                "Uma Thurman"
        ]
}
{
        "_id" : ObjectId("61755852b132b3063451e036"),
        "title" : "Inglorious Basterds",
        "writer" : "Quentin Tarantino",
        "year" : 2009,
        "actors" : [
                "Brad Pitt",
                "Diane Kruger",
                "Eli Roth"
        ]
}
{
        "_id" : ObjectId("6175585eb132b3063451e037"),
        "title" : "The Hobbit: An Unexpected Journey",
        "writer" : "J.R.R. Tolkein",
        "year" : 2012,
        "franchise" : "The Hobbit"
}
{
        "_id" : ObjectId("6175586bb132b3063451e038"),
```

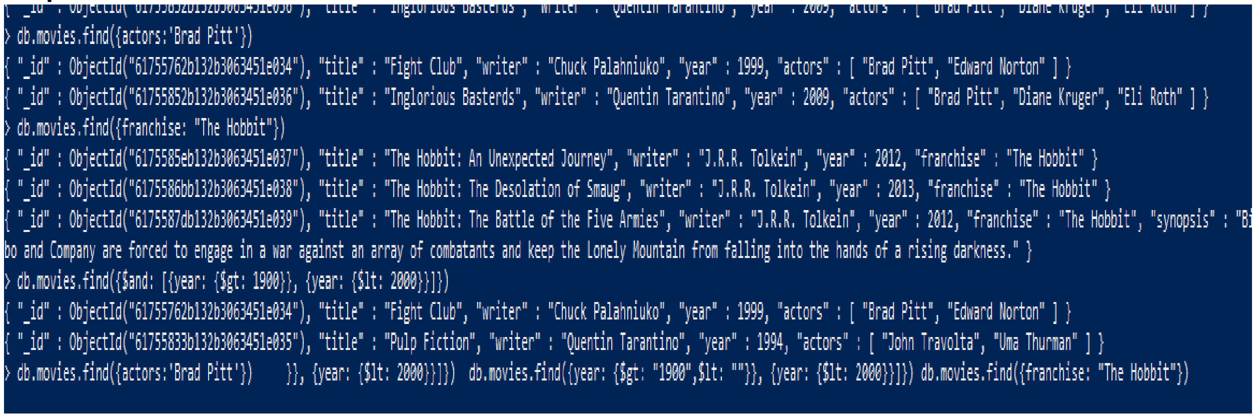
2. get all documents with writer set to "Quentin Tarantino"



```
db.movies.find({writer: "Quentin Tarantino"})
"_id" : ObjectId("61755833b132b3063451e035"), "title" : "Pulp Fiction", "writer" : "Quentin Tarantino", "year" : 1994, "actors" : [ "John Travolta", "Uma Thurman" ] }
"_id" : ObjectId("61755852b132b3063451e036"), "title" : "Inglorious Basterds", "writer" : "Quentin Tarantino", "year" : 2009, "actors" : [ "Brad Pitt", "Diane Kruger", "Eli Roth" ] }
db.movies.find({actors: ["Brad Pitt"]})
```

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

Steps: 3 to 6

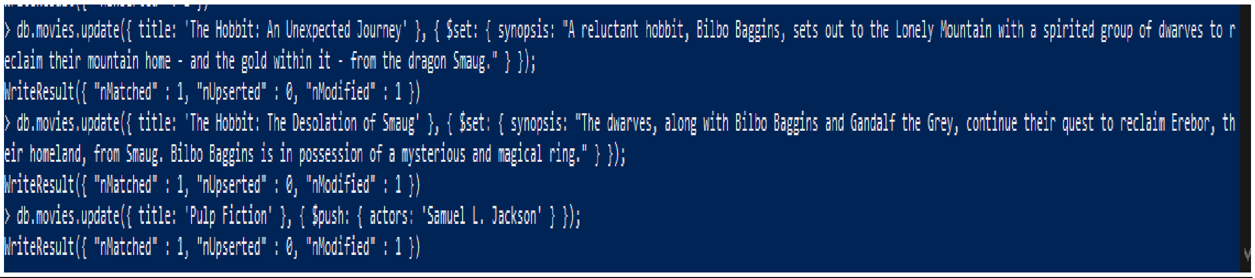


```
> db.movies.find({actors:'Brad Pitt'})
{ "_id" : ObjectId("61755762b132b3063451e034"), "title" : "Fight Club", "writer" : "Chuck Palahniuko", "year" : 1999, "actors" : [ "Brad Pitt", "Edward Norton" ] }
{ "_id" : ObjectId("61755852b132b3063451e036"), "title" : "Inglorious Basterds", "writer" : "Quentin Tarantino", "year" : 2009, "actors" : [ "Brad Pitt", "Diane Kruger", "Eli Roth" ] }
> db.movies.find({franchise: "The Hobbit"})
{ "_id" : ObjectId("6175585eb132b3063451e037"), "title" : "The Hobbit: An Unexpected Journey", "writer" : "J.R.R. Tolkein", "year" : 2012, "franchise" : "The Hobbit" }
{ "_id" : ObjectId("6175586bb132b3063451e038"), "title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R. Tolkein", "year" : 2013, "franchise" : "The Hobbit" }
{ "_id" : ObjectId("6175587db132b3063451e039"), "title" : "The Hobbit: The Battle of the Five Armies", "writer" : "J.R.R. Tolkein", "year" : 2012, "franchise" : "The Hobbit", "synopsis" : "Bi
bo 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({$and: [{year: {$gt: 1900}}, {year: {$lt: 2000}}]})
{ "_id" : ObjectId("61755762b132b3063451e034"), "title" : "Fight Club", "writer" : "Chuck Palahniuko", "year" : 1999, "actors" : [ "Brad Pitt", "Edward Norton" ] }
{ "_id" : ObjectId("61755833b132b3063451e035"), "title" : "Pulp Fiction", "writer" : "Quentin Tarantino", "year" : 1994, "actors" : [ "John Travolta", "Uma Thurman" ] }
> db.movies.find({actors:'Brad Pitt'})     }}, {year: {$lt: 2000}}]})  db.movies.find({year: {$gt: "1900",$lt: ""}}, {year: {$lt: 2000}}]}) db.movies.find({franchise: "The Hobbit"})
```

UPDATE DOCUMENTS:

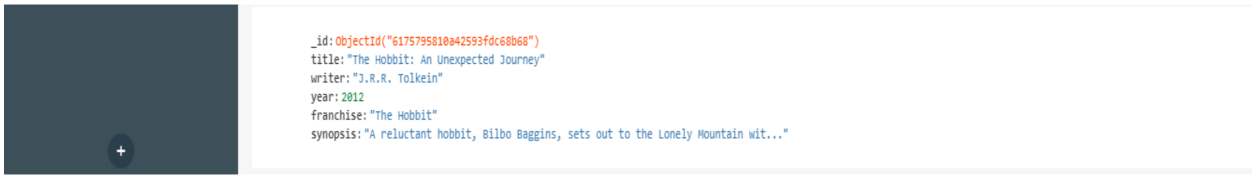
1. add a synopsis to "The Hobbit: An Unexpected Journey" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug."
2. add a synopsis to "The Hobbit: The Desolation of Smaug"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

Steps: 1 to 3



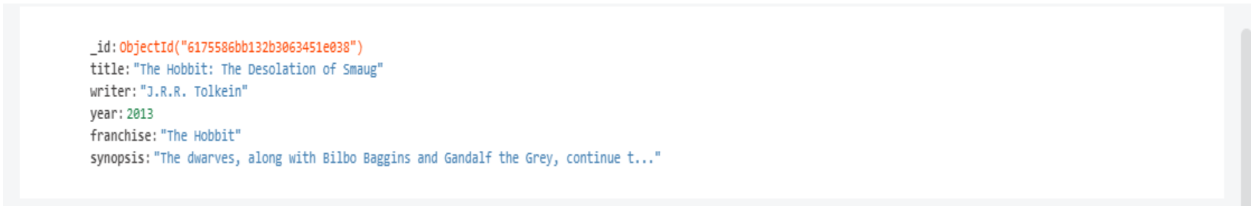
```
> db.movies.update({ title: 'The Hobbit: An Unexpected Journey' }, { $set: { synopsis: "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to r
eclaim their mountain home - and the gold within it - from the dragon Smaug." } });
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> 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, th
eir homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring." } });
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.movies.update({ title: 'Pulp Fiction' }, { $push: { actors: 'Samuel L. Jackson' } });
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

Step1:



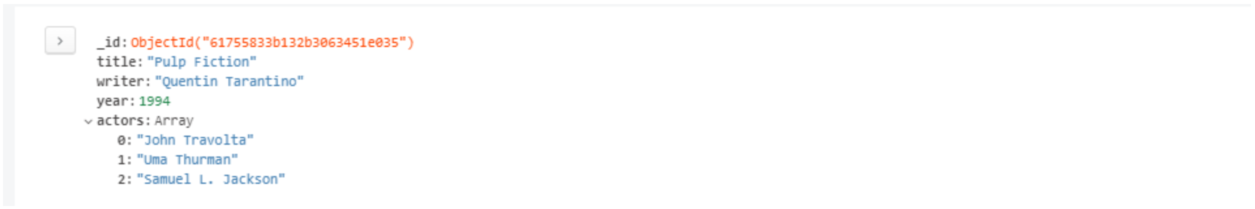
```
_id: ObjectId("6175795810a42593fdc68b68")
title: "The Hobbit: An Unexpected Journey"
writer: "J.R.R. Tolkein"
year: 2012
franchise: "The Hobbit"
synopsis: "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain wit..."
```

Step2:



```
_id: ObjectId("6175586bb132b3063451e038")
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 t..."
```

Step3:

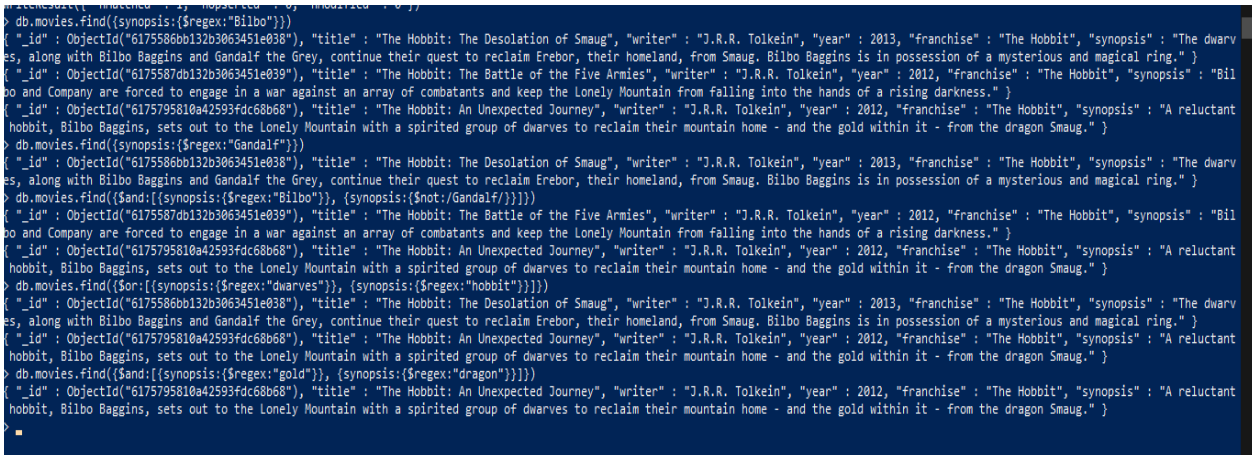


```
_id: ObjectId("61755833b132b3063451e035")
title: "Pulp Fiction"
writer: "Quentin Tarantino"
year: 1994
actors: Array
   0: "John Travolta"
   1: "Uma Thurman"
   2: "Samuel L. Jackson"
```

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"

Steps: 1 to 5

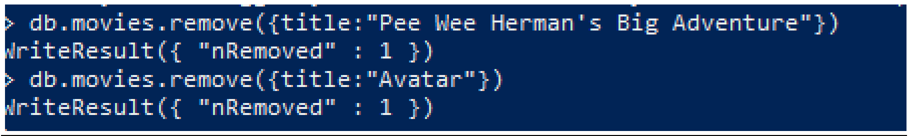


```
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 0 })
> db.movies.find({synopsis:{$regex:"Bilbo"}})
{ "_id" : ObjectId("6175586bb132b3063451e038"), "title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R. Tolkein", "year" : 2013, "franchise" : "The Hobbit", "synopsis" : "The dwarv
es, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring." }
{ "_id" : ObjectId("6175587db132b3063451e039"), "title" : "The Hobbit: The Battle of the Five Armies", "writer" : "J.R.R. Tolkein", "year" : 2012, "franchise" : "The Hobbit", "synopsis" : "Bil
bo and Company are forced to engage in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of a rising darkness." }
{ "_id" : ObjectId("6175795810a42593fdc68b68"), "title" : "The Hobbit: An Unexpected Journey", "writer" : "J.R.R. Tolkein", "year" : 2012, "franchise" : "The Hobbit", "synopsis" : "A reluctant
 hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug." }
> db.movies.find({synopsis:{$regex:"Gandalf"}})
{ "_id" : ObjectId("6175586bb132b3063451e038"), "title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R. Tolkein", "year" : 2013, "franchise" : "The Hobbit", "synopsis" : "The dwarv
es, 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.find({$and:[{synopsis:{$regex:"Bilbo"}}, {synopsis:{$not:/Gandalf/}}]})
{ "_id" : ObjectId("6175587db132b3063451e039"), "title" : "The Hobbit: The Battle of the Five Armies", "writer" : "J.R.R. Tolkein", "year" : 2012, "franchise" : "The Hobbit", "synopsis" : "Bil
bo and Company are forced to engage in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of a rising darkness." }
{ "_id" : ObjectId("6175795810a42593fdc68b68"), "title" : "The Hobbit: An Unexpected Journey", "writer" : "J.R.R. Tolkein", "year" : 2012, "franchise" : "The Hobbit", "synopsis" : "A reluctant
 hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug." }
> db.movies.find({$or:[{synopsis:{$regex:"dwarves"}}, {synopsis:{$regex:"hobbit"}}]})
{ "_id" : ObjectId("6175586bb132b3063451e038"), "title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R. Tolkein", "year" : 2013, "franchise" : "The Hobbit", "synopsis" : "The dwarv
es, 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("6175795810a42593fdc68b68"), "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." }
> db.movies.find({$and:[{synopsis:{$regex:"gold"}}, {synopsis:{$regex:"dragon"}}]})
{ "_id" : ObjectId("6175795810a42593fdc68b68"), "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." }
>
```

DELETE DOCUMENTS :

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

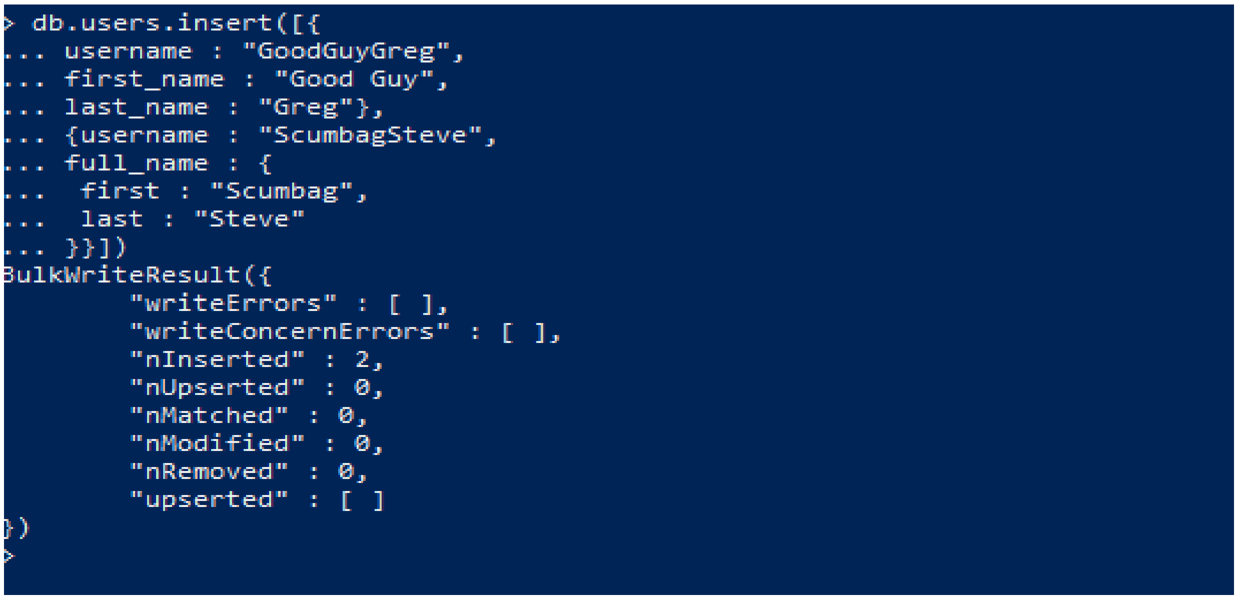
Steps: 1 to 2



```
> db.movies.remove({title:"Pee Wee Herman's Big Adventure"})
WriteResult({ "nRemoved" : 1 })
> db.movies.remove({title:"Avatar"})
WriteResult({ "nRemoved" : 1 })
```

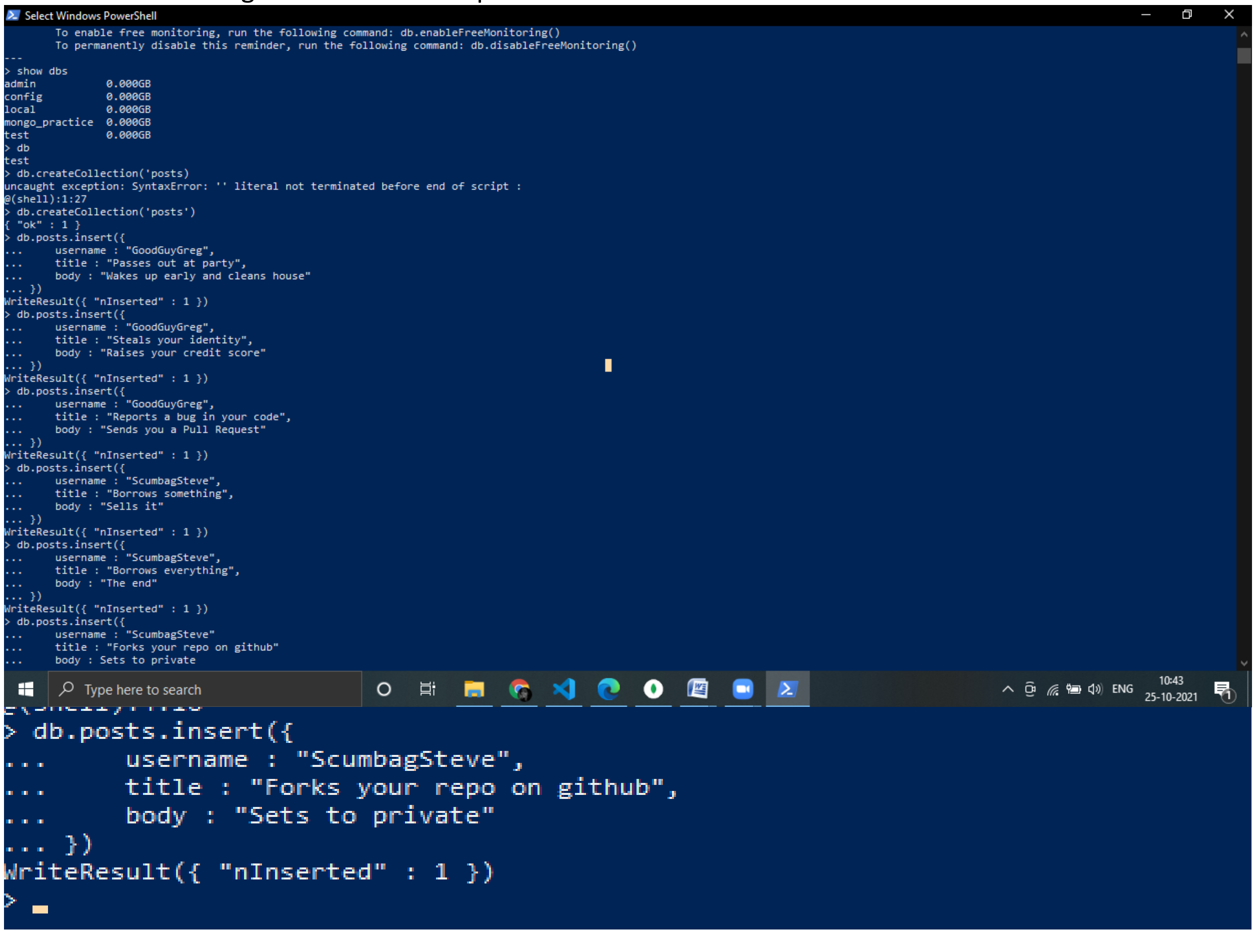
RELATIONSHIPS:

1.Insert the following documents into a users collection



```
> db.users.insert([{
... username : "GoodGuyGreg",
... first_name : "Good Guy",
... last_name : "Greg"},
... {username : "ScumbagSteve",
... full_name : {
...  first : "Scumbag",
...  last : "Steve"
... }}])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
>
```

2.Insert the following documents into a posts collection



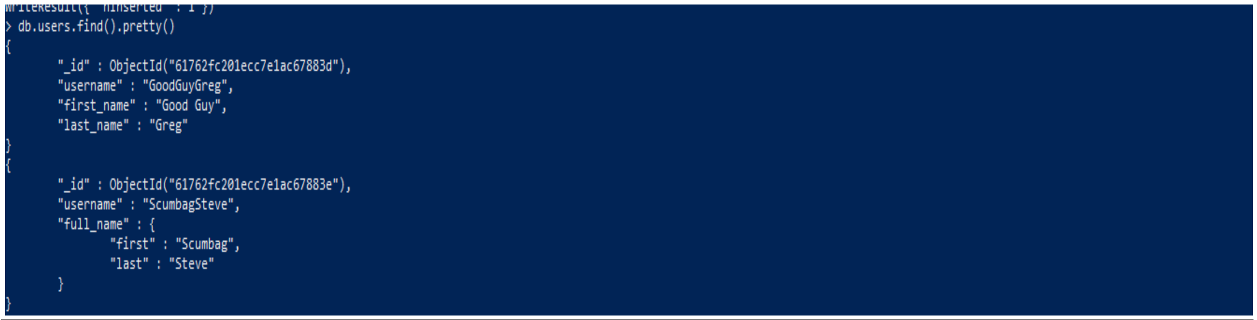
```
        To enable free monitoring, run the following command: db.enableFreeMonitoring()
        To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> show dbs
admin          0.000GB
config         0.000GB
local          0.000GB
mongo_practice 0.000GB
test           0.000GB
> db
test
> db.createCollection('posts)
uncaught exception: SyntaxError: '' literal not terminated before end of script :
@(shell):1:27
> db.createCollection('posts')
{ "ok" : 1 }
> db.posts.insert({
...     username : "GoodGuyGreg",
...     title : "Passes out at party",
...     body : "Wakes up early and cleans house"
... })
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
```

```
> db.posts.insert({
...     username : "ScumbagSteve",
...     title : "Forks your repo on github",
...     body : "Sets to private"
... })
WriteResult({ "nInserted" : 1 })
>
```

3.Insert the following documents into a comments collection

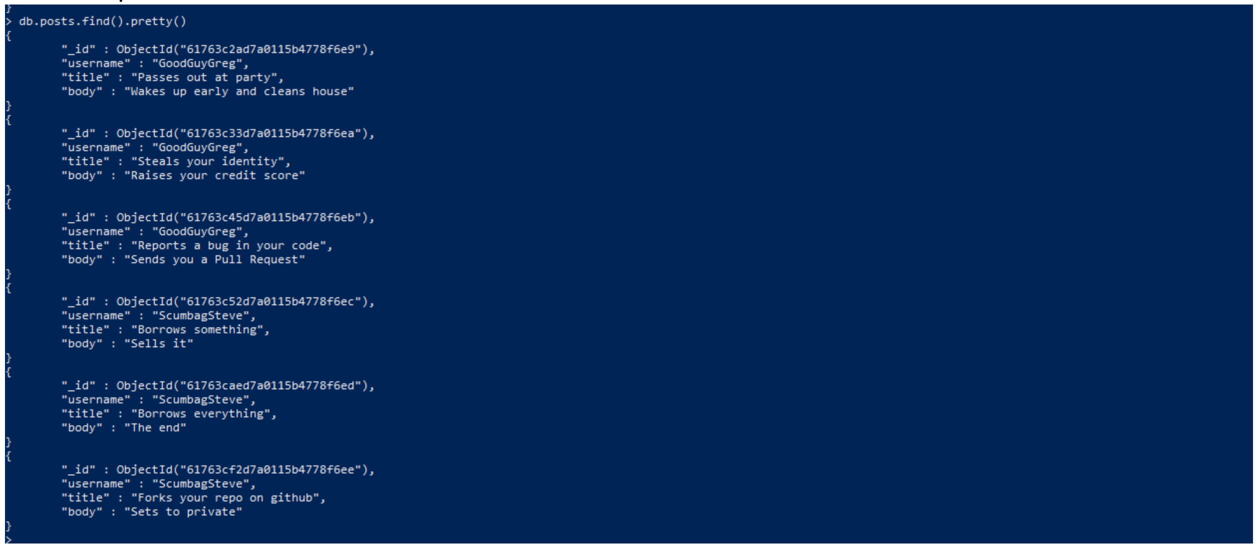
Querying related collections:

1. find all users



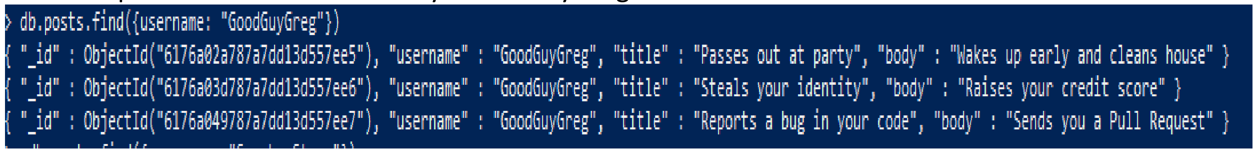
```
> db.users.find().pretty()
{
        "_id" : ObjectId("61762fc201ecc7e1ac67883d"),
        "username" : "GoodGuyGreg",
        "first_name" : "Good Guy",
        "last_name" : "Greg"
}
{
        "_id" : ObjectId("61762fc201ecc7e1ac67883e"),
        "username" : "ScumbagSteve",
        "full_name" : {
                "first" : "Scumbag",
                "last" : "Steve"
        }
}
```

2. find all posts



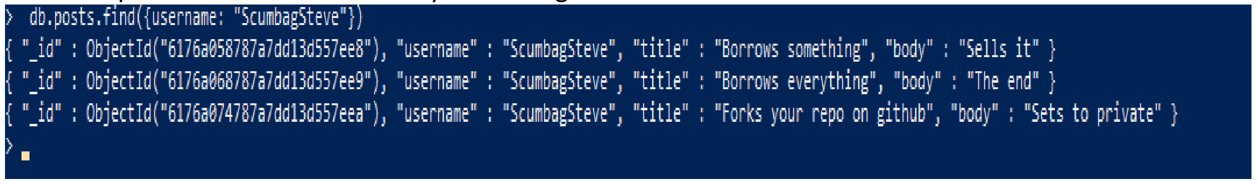
```
> db.posts.find().pretty()
{
        "_id" : ObjectId("61763c2ad7a0115b4778f6e9"),
        "username" : "GoodGuyGreg",
        "title" : "Passes out at party",
        "body" : "Wakes up early and cleans house"
}
{
        "_id" : ObjectId("61763c33d7a0115b4778f6ea"),
        "username" : "GoodGuyGreg",
        "title" : "Steals your identity",
        "body" : "Raises your credit score"
}
{
        "_id" : ObjectId("61763c45d7a0115b4778f6eb"),
        "username" : "GoodGuyGreg",
        "title" : "Reports a bug in your code",
        "body" : "Sends you a Pull Request"
}
{
        "_id" : ObjectId("61763c52d7a0115b4778f6ec"),
        "username" : "ScumbagSteve",
        "title" : "Borrows something",
        "body" : "Sells it"
}
{
        "_id" : ObjectId("61763caed7a0115b4778f6ed"),
        "username" : "ScumbagSteve",
        "title" : "Borrows everything",
        "body" : "The end"
}
{
        "_id" : ObjectId("61763cf2d7a0115b4778f6ee"),
        "username" : "ScumbagSteve",
        "title" : "Forks your repo on github",
        "body" : "Sets to private"
}
>
```

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



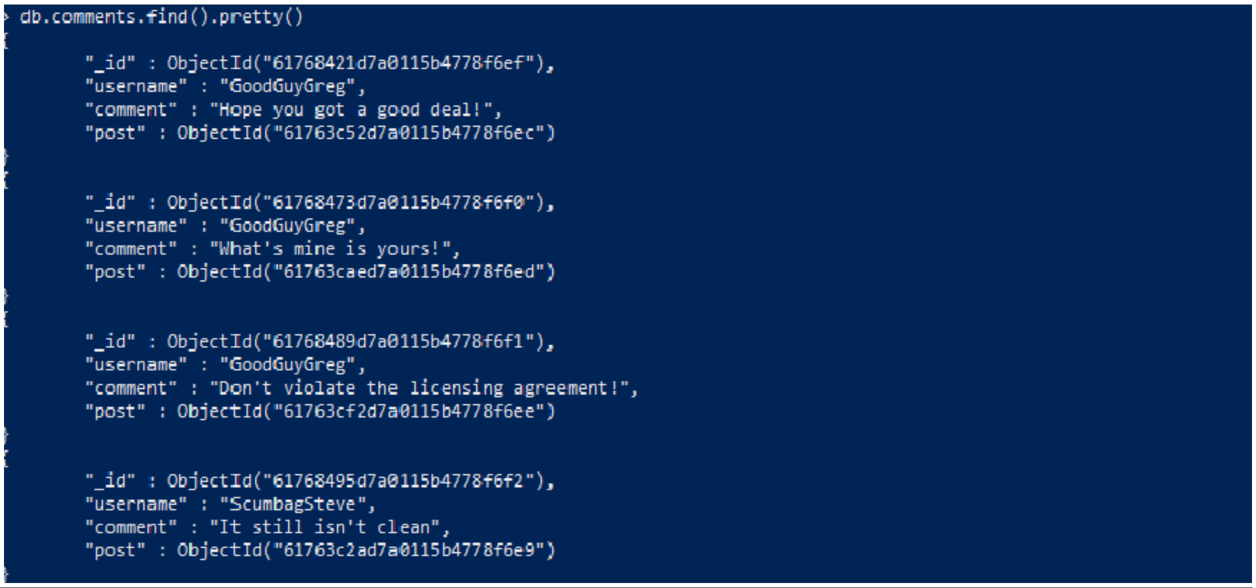
```
> db.posts.find({username: "GoodGuyGreg"})
{ "_id" : ObjectId("6176a02a787a7dd13d557ee5"), "username" : "GoodGuyGreg", "title" : "Passes out at party", "body" : "Wakes up early and cleans house" }
{ "_id" : ObjectId("6176a03d787a7dd13d557ee6"), "username" : "GoodGuyGreg", "title" : "Steals your identity", "body" : "Raises your credit score" }
{ "_id" : ObjectId("6176a049787a7dd13d557ee7"), "username" : "GoodGuyGreg", "title" : "Reports a bug in your code", "body" : "Sends you a Pull Request" }
```

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



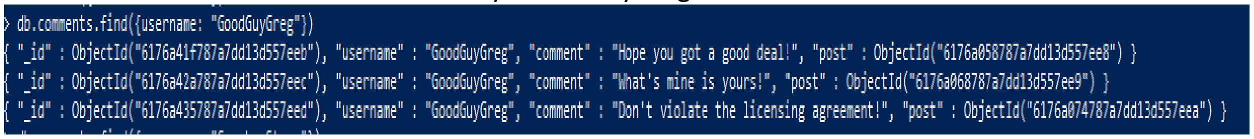
```
>  db.posts.find({username: "ScumbagSteve"})
{ "_id" : ObjectId("6176a058787a7dd13d557ee8"), "username" : "ScumbagSteve", "title" : "Borrows something", "body" : "Sells it" }
{ "_id" : ObjectId("6176a068787a7dd13d557ee9"), "username" : "ScumbagSteve", "title" : "Borrows everything", "body" : "The end" }
{ "_id" : ObjectId("6176a074787a7dd13d557eea"), "username" : "ScumbagSteve", "title" : "Forks your repo on github", "body" : "Sets to private" }
>
```

5. find all comments



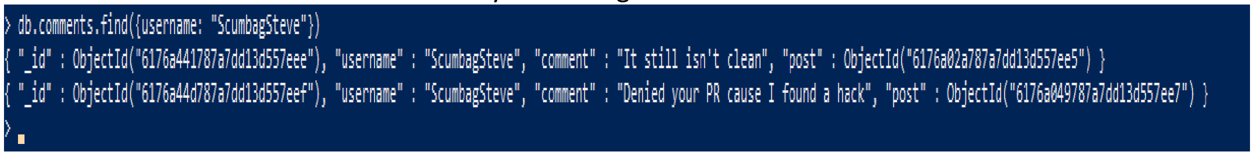
```
> db.comments.find().pretty()
{
        "_id" : ObjectId("61768421d7a0115b4778f6ef"),
        "username" : "GoodGuyGreg",
        "comment" : "Hope you got a good deal!",
        "post" : ObjectId("61763c52d7a0115b4778f6ec")
}
{
        "_id" : ObjectId("61768473d7a0115b4778f6f0"),
        "username" : "GoodGuyGreg",
        "comment" : "What's mine is yours!",
        "post" : ObjectId("61763caed7a0115b4778f6ed")
}
{
        "_id" : ObjectId("61768489d7a0115b4778f6f1"),
        "username" : "GoodGuyGreg",
        "comment" : "Don't violate the licensing agreement!",
        "post" : ObjectId("61763cf2d7a0115b4778f6ee")
}
{
        "_id" : ObjectId("61768495d7a0115b4778f6f2"),
        "username" : "ScumbagSteve",
        "comment" : "It still isn't clean",
        "post" : ObjectId("61763c2ad7a0115b4778f6e9")
}
```

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



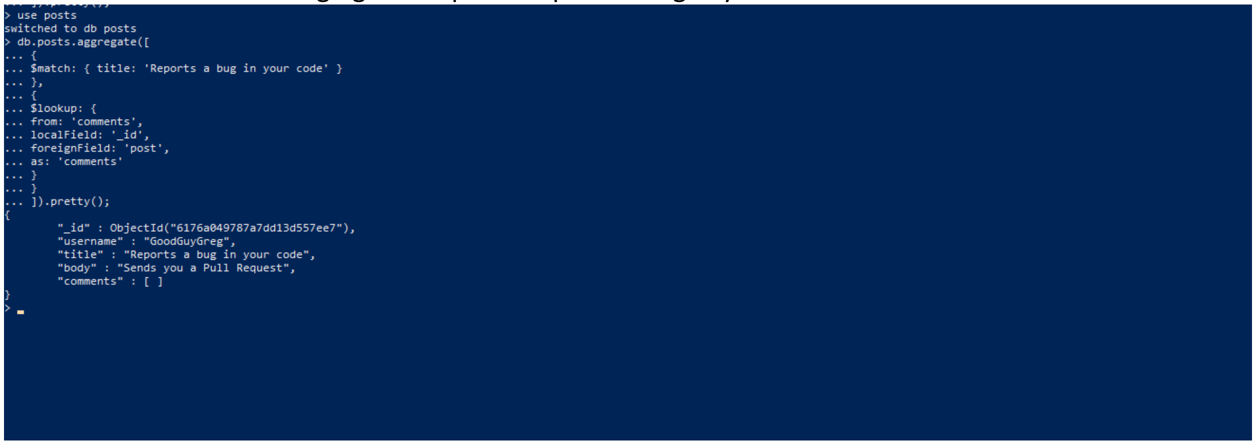
```
> db.comments.find({username: "GoodGuyGreg"})
{ "_id" : ObjectId("6176a41f787a7dd13d557eeb"), "username" : "GoodGuyGreg", "comment" : "Hope you got a good deal!", "post" : ObjectId("6176a058787a7dd13d557ee8") }
{ "_id" : ObjectId("6176a42a787a7dd13d557eec"), "username" : "GoodGuyGreg", "comment" : "What's mine is yours!", "post" : ObjectId("6176a068787a7dd13d557ee9") }
{ "_id" : ObjectId("6176a435787a7dd13d557eed"), "username" : "GoodGuyGreg", "comment" : "Don't violate the licensing agreement!", "post" : ObjectId("6176a074787a7dd13d557eea") }
```

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



```
> db.comments.find({username: "ScumbagSteve"})
{ "_id" : ObjectId("6176a441787a7dd13d557eee"), "username" : "ScumbagSteve", "comment" : "It still isn't clean", "post" : ObjectId("6176a02a787a7dd13d557ee5") }
{ "_id" : ObjectId("6176a44d787a7dd13d557eef"), "username" : "ScumbagSteve", "comment" : "Denied your PR cause I found a hack", "post" : ObjectId("6176a049787a7dd13d557ee7") }
>
```

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



```
> use posts
switched to db posts
> db.posts.aggregate([
... {
... $match: { title: 'Reports a bug in your code' }
... },
... {
... $lookup: {
... from: 'comments',
... localField: '_id',
... foreignField: 'post',
... as: 'comments'
... }
... }
... ]).pretty();
{
        "_id" : ObjectId("6176a049787a7dd13d557ee7"),
        "username" : "GoodGuyGreg",
        "title" : "Reports a bug in your code",
        "body" : "Sends you a Pull Request",
        "comments" : [ ]
}
>
```