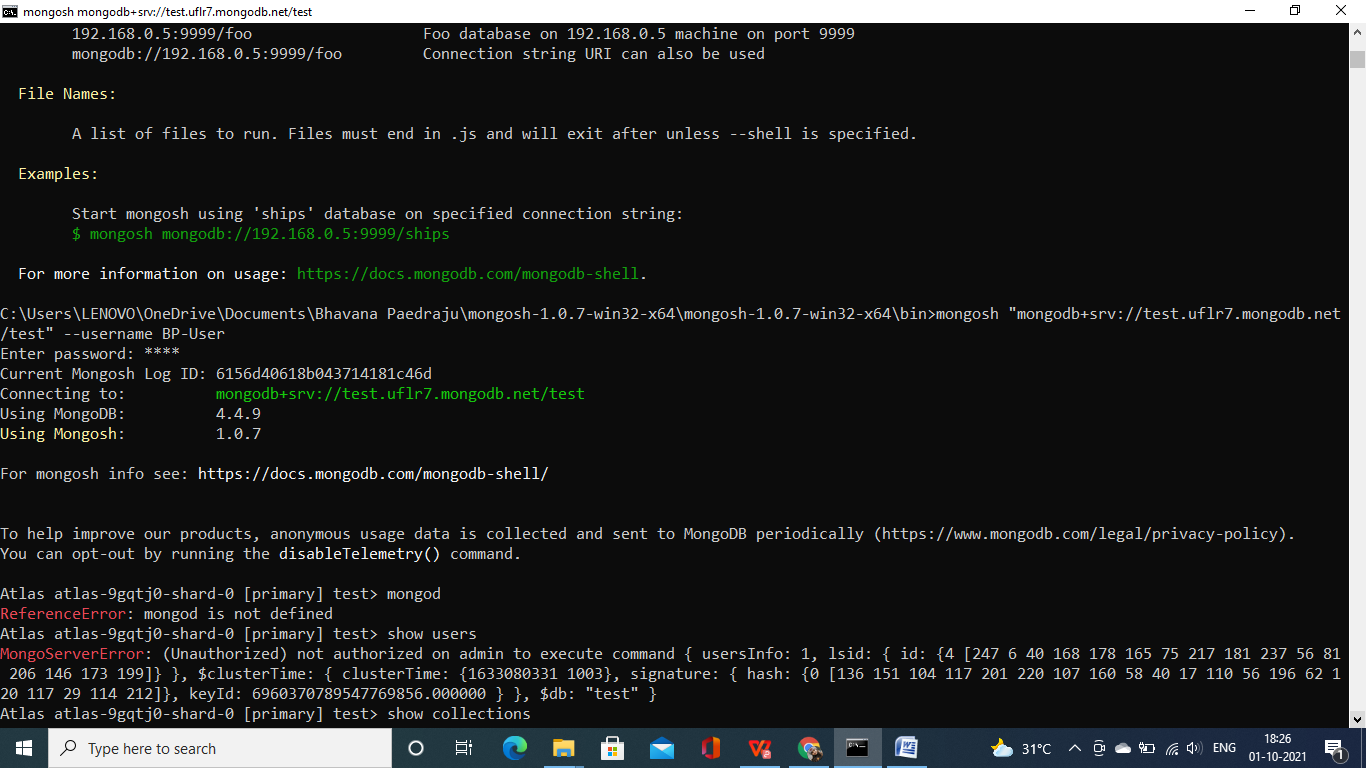
**MongoDB Lab Assignments -Day 1**

**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

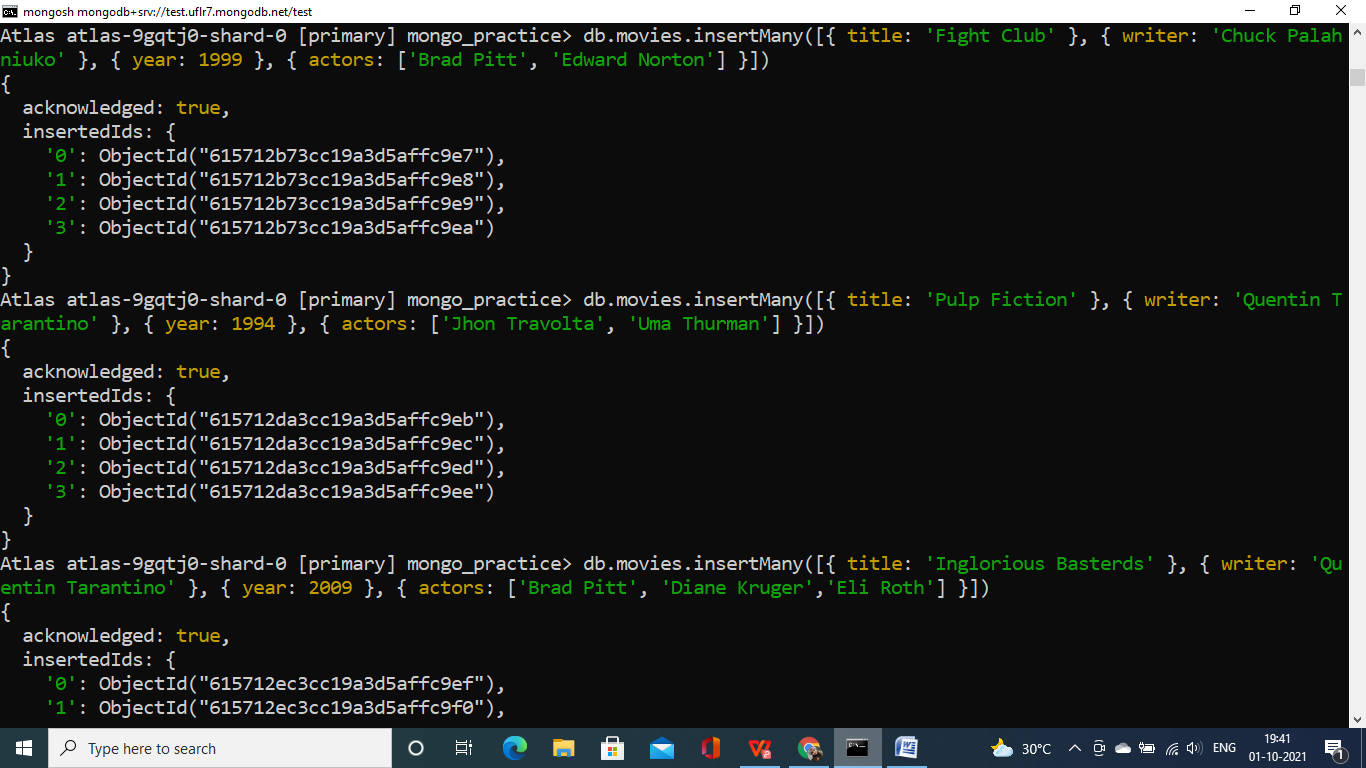
synopsis : Bilbo and Company are forced to engage in a war against an array of

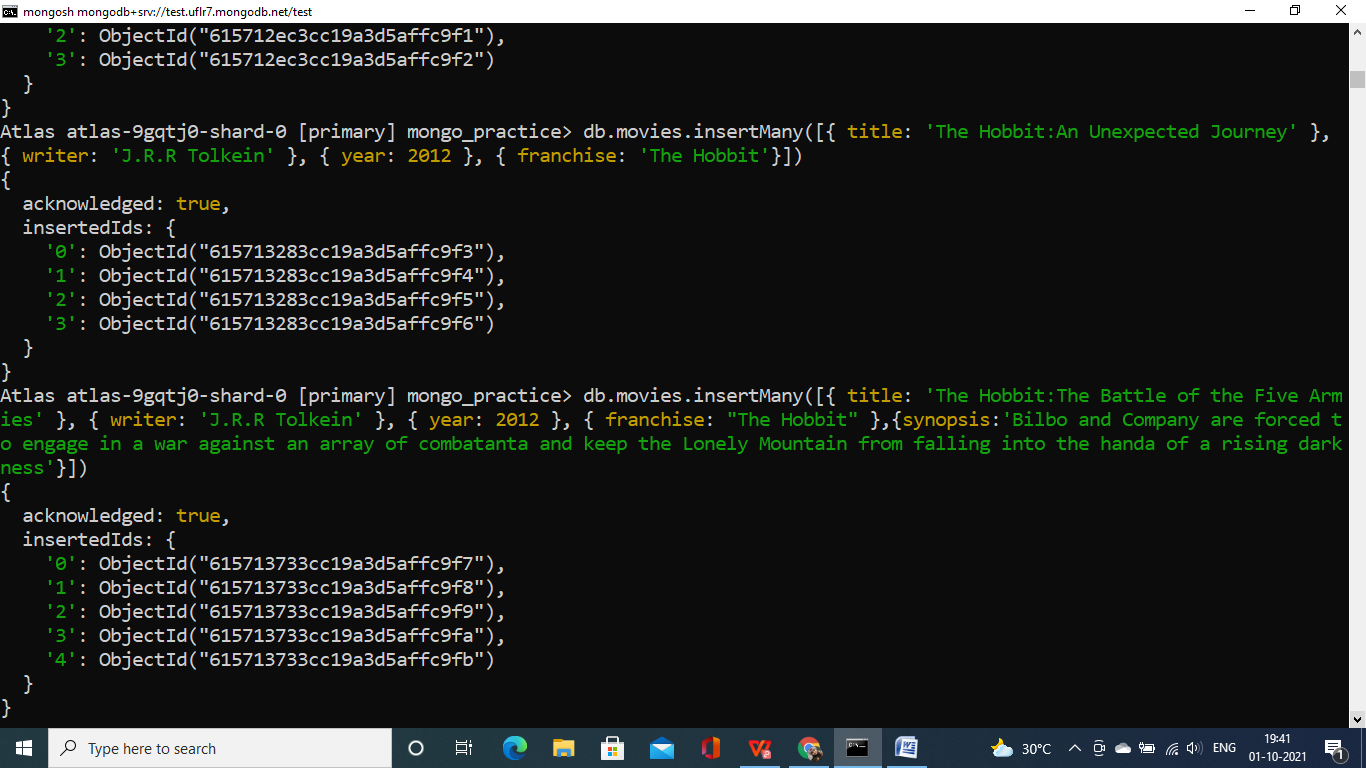
combatants and keep the Lonely Mountain from falling into the hands of a rising

darkness.

title : Pee Wee Herman's Big Adventure

title : Avatar





**Query / Find Documents**

query the **movies** collection to

1. get all documents

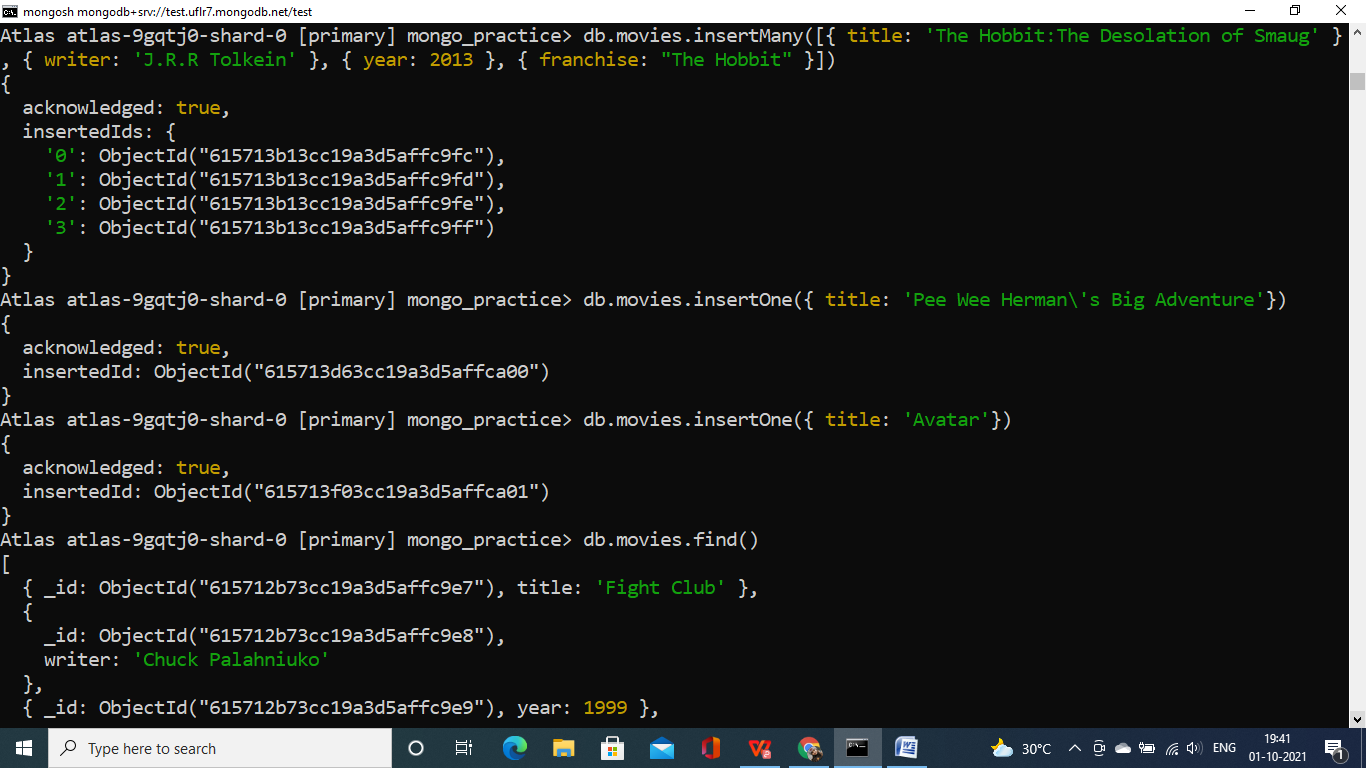
2. get all documents with writer set to "Quentin Tarantino"

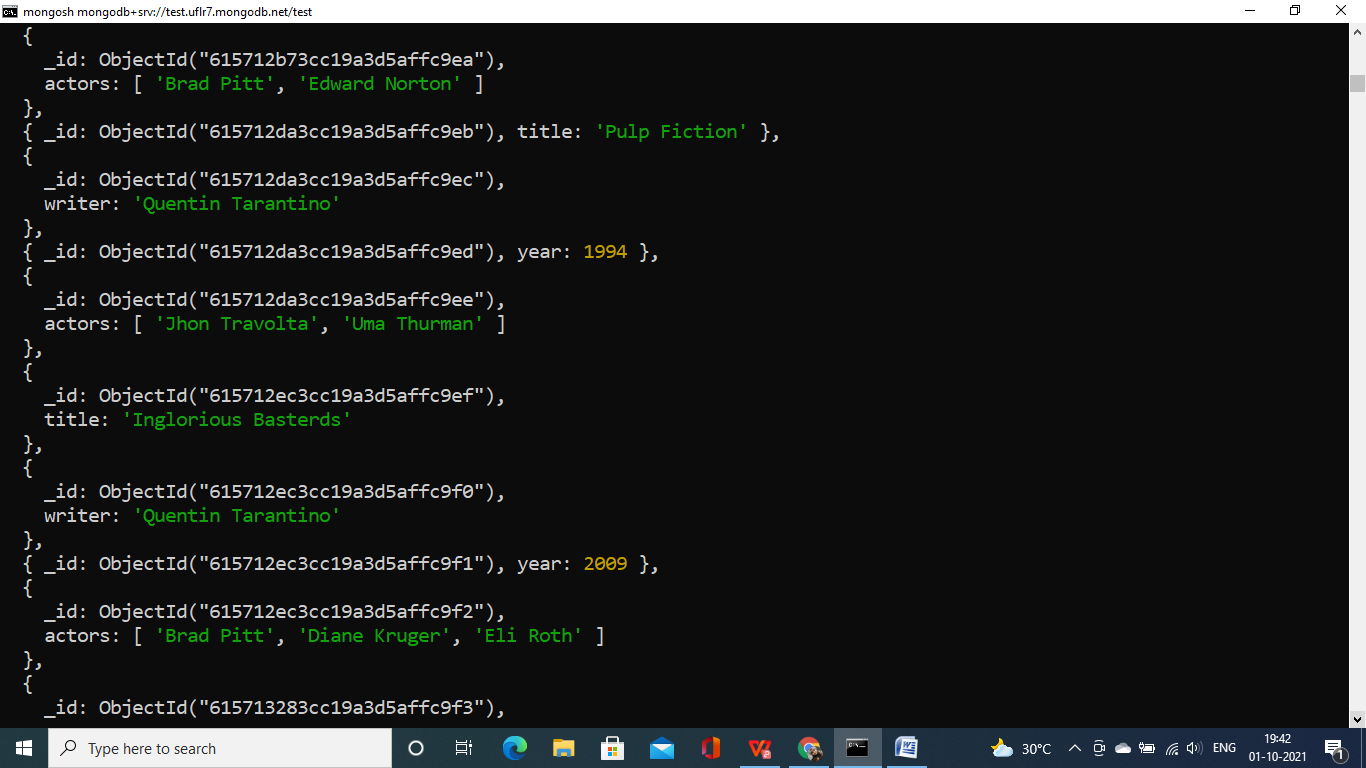
3. get all documents where actors include "Brad Pitt"

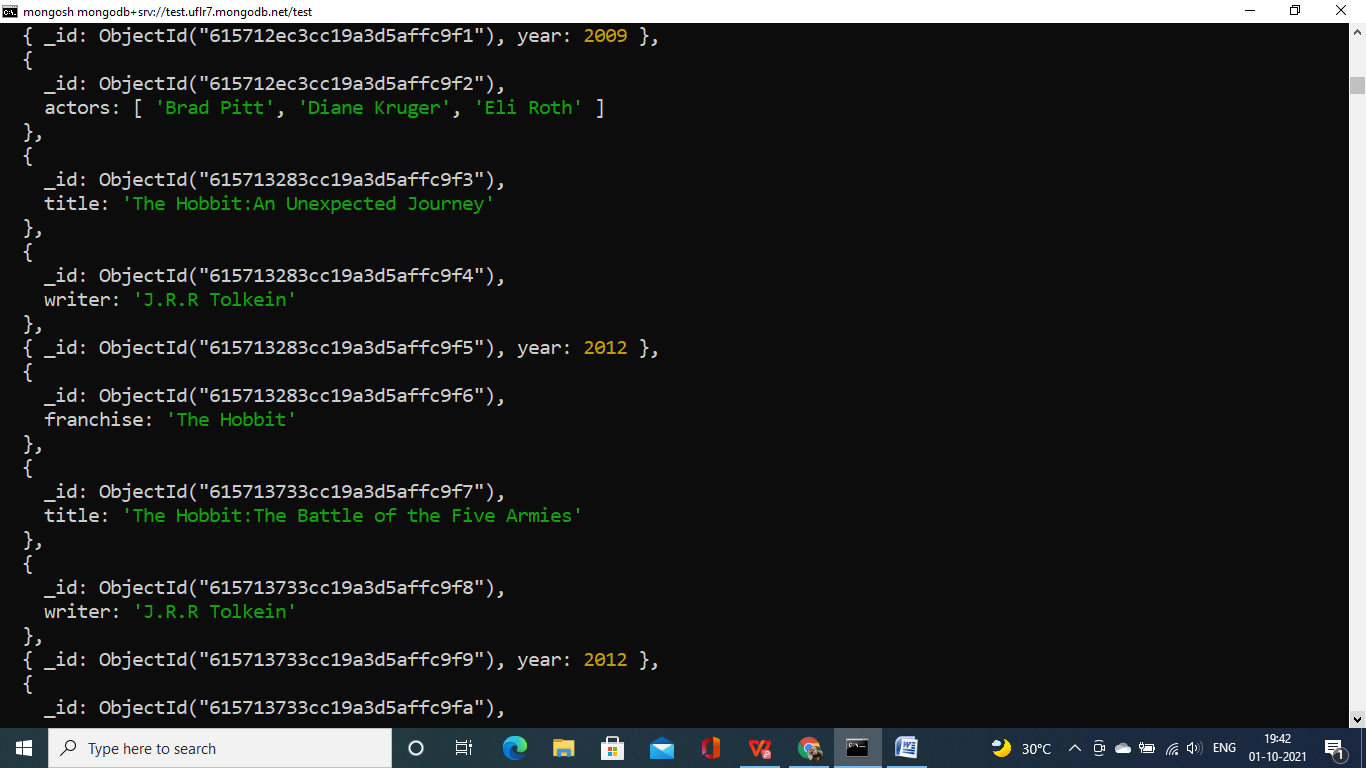
4. get all documents with franchise set to "The Hobbit"

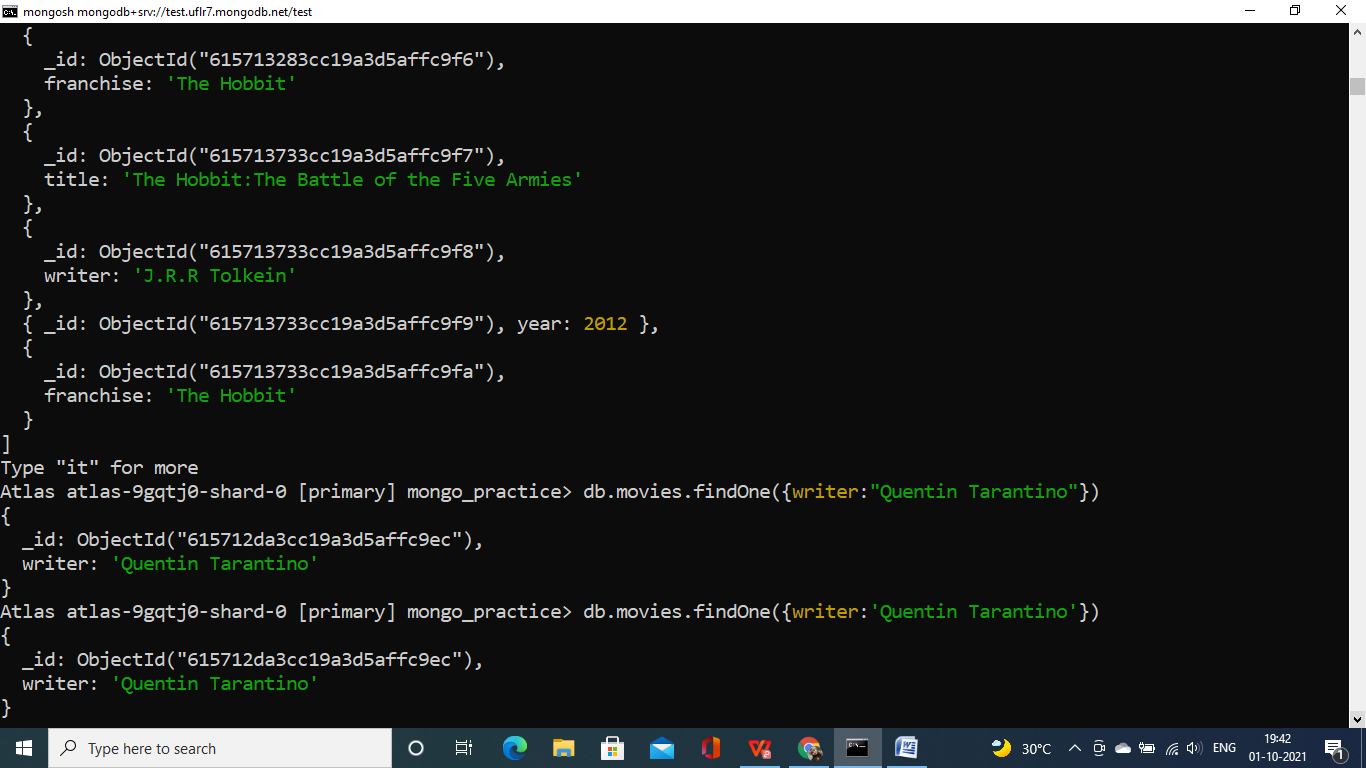
5. get all movies released in the 90s

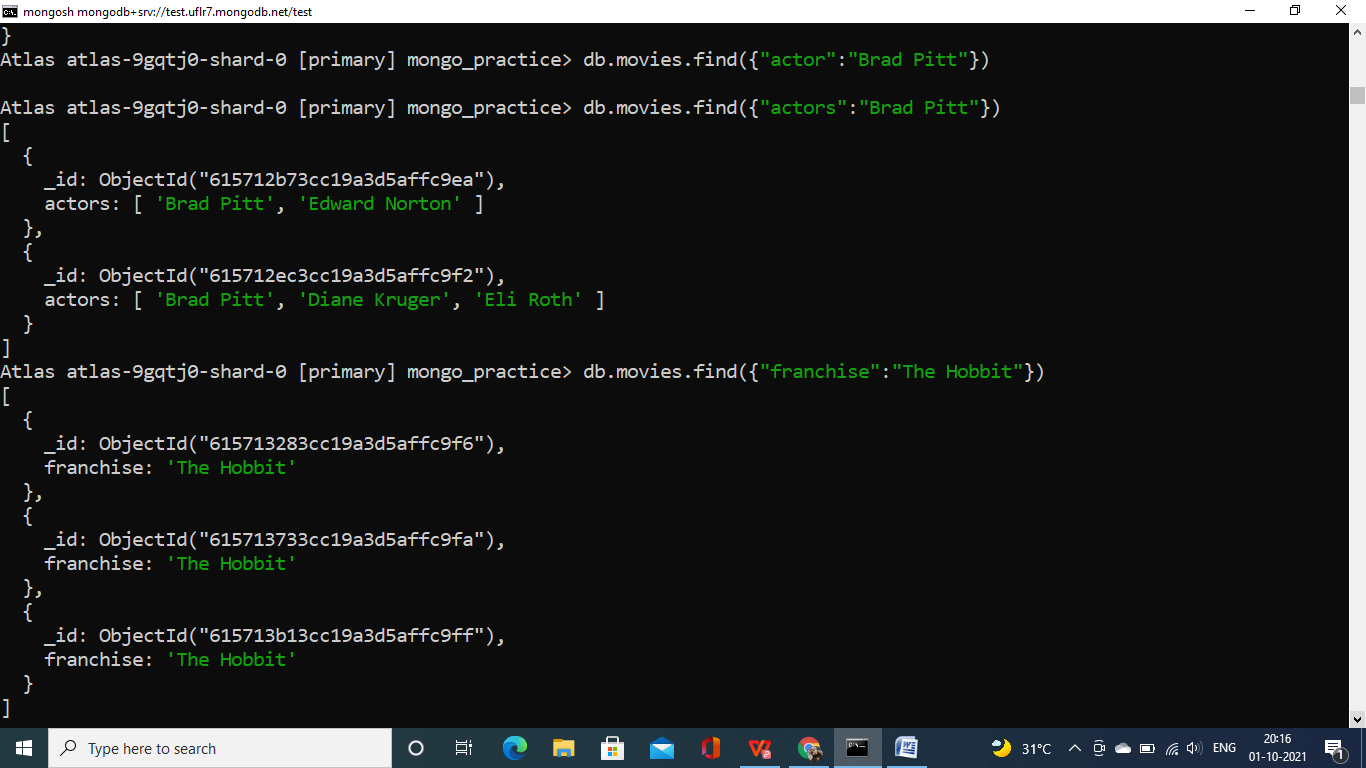
6. get all movies released before the year 2000 or after 2010

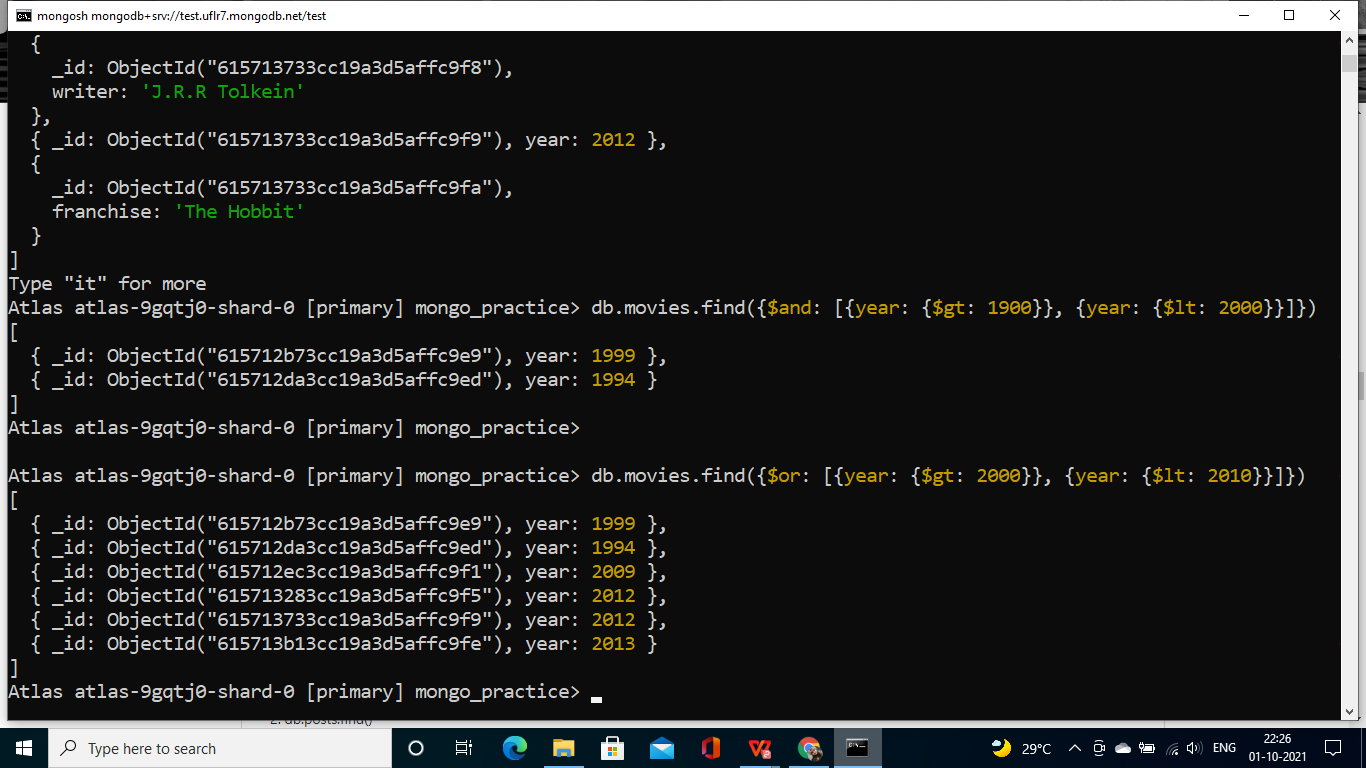
****

****

****

****





**Update Documents**

1. add a synopsis to "The Hobbit: An Unexpected Journey" : "A reluctant hobbit,

Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of

dwarves to reclaim their mountain home - and the gold within it - from the

dragon Smaug."

db.movies.update({title: "The Hobbit: An Unexpected Journey"}, {synopsis: "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug."})

1. add a synopsis to "The Hobbit: The Desolation of Smaug" : "The dwarves,

along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim

Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a

mysterious and magical ring.’

db.movies.update({title: "The Hobbit: The Desolation of Smaug"}, {synopsis: "The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."})

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

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

**Text Search**

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

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

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

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

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

word "Gandalf"

db.movies.find({ $and: [{ synopsis: ‘Bilbo’ }, { synopsis: { $not: ‘Gandalf’ } }] })

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

"hobbit"

db.movies.find({$or:[{$text: {$search: "dwarves”}},{$text:{$search:”hobbit"}}]})

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

"dragon"

db.movies.find({$or:[{$text: {$search: "gold”}},{$text:{$search:”dragon"}}]})

**Delete Documents**

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

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

1. delete the movie "Avatar"

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

**Relationships**

db.users.insertMany([  
{  
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  
db.posts.insertMany([  
{  
username: "GoodGuyGreg",  
title: "Passes out at party",  
body: "Wakes up early and cleans house"  
},  
{  
username: "GoodGuyGreg",  
title: "Steals your identity",  
body: "Raises your credit score"  
},  
{  
username: "GoodGuyGreg",  
title: "Reports a bug in your code",  
body: "Sends you a Pull Request"  
},  
{  
username: "ScumbagSteve",  
title: "Borrows something",  
body: "Sells it"  
},  
{  
username: "ScumbagSteve",  
title: "Borrows everything",  
body: "The end"  
},  
{  
username: "ScumbagSteve",  
title: "Forks your repo on github",  
body: "Sets to private"  
}  
]);

Insert the following documents into a comments collection  
db.comments.insertMany([  
{  
username: "GoodGuyGreg",  
comment: "Hope you got a good deal!",  
post: ObjectId("5f44d3a148197d7749864def")  
},  
{  
username: "GoodGuyGreg",  
comment: "Don't violate the licensing agreement!",  
post: ObjectId("5f44d3a148197d7749864df0")  
},  
{  
username: "GoodGuyGreg",  
comment: "Don't violate the licensing agreement!",  
post: ObjectId("5f44d3a148197d7749864df1")  
},  
{  
username: "ScumbagSteve",  
comment: "It still isn't clean",  
post: ObjectId("5f44d3a148197d7749864dec")  
},  
{  
username: "ScumbagSteve",  
comment: "Denied your PR cause I found a hack",  
post: ObjectId("5f44d3a148197d7749864dee")  
}  
]);

1. .find all users  
   db.users.find().pretty();

2.find all posts  
db.posts.find().pretty();

3.find all posts that was authored by "GoodGuyGreg"  
db.posts.find({ username: 'GoodGuyGreg' }).pretty();

4.find all posts that was authored by "ScumbagSteve"  
db.posts.find({ username: 'ScumbagSteve' }).pretty();

5.find all comments  
db.comments.find().pretty();

6.find all comments that was authored by "GoodGuyGreg"  
db.comments.find({ username: 'GoodGuyGreg' }).pretty();

7.find all comments that was authored by "ScumbagSteve"  
db.comments.find({ username: 'ScumbagSteve' }).pretty();

8.find all comments belonging to the post "Reports a bug in your code"  
db.posts.aggregate([  
{  
$match: { title: 'Reports a bug in your code' }  
},  
{  
$lookup: {  
from: 'comments',  
localField: '\_id',  
foreignField: 'post',  
as: 'comments'  
}  
}  
]).pretty();