Mongodb Assignment - 1

Query the movies collection to

--

**db.Movies.insertMany([{**

**'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 Hermans Big Adventure'**

**... },**

**... {**

**... 'title':'Avatar'**

**... }])**

**{**

**"acknowledged" : true,**

**"insertedIds" : [**

**ObjectId("61582c061767ee1b1b4cb7a4"),**

**ObjectId("61582c061767ee1b1b4cb7a5"),**

**ObjectId("61582c061767ee1b1b4cb7a6"),**

**ObjectId("61582c061767ee1b1b4cb7a7"),**

**ObjectId("61582c061767ee1b1b4cb7a8"),**

**ObjectId("61582c061767ee1b1b4cb7a9"),**

**ObjectId("61582c061767ee1b1b4cb7aa"),**

**ObjectId("61582c061767ee1b1b4cb7ab")**

**]**

**}**

**1.get all the documents?**

Ans. db.Movies.find()

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

Ans. db.Movies.find({"writer":"Quentin Tarantino"})

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

Ans. db.Movies.find({"actors":"Brad Pitt"})

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

Ans. db.Movies.find({"franchise":"The Hobbit"})

5.get all movies released in the 90s?

Ans. db.Movies.find({year:{$gt:1989,$lt:2000}})

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

Ans.db.movies.find({$or:[{year:{$gt:"2010"}},{year: {$lt:"2000"}}]})

Use insertMany command to insert query into the database

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 reclain their mountain home - and the gold within it - from the dragon Smaug."?**

Ans. 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 reclaim their mountain home - and the gold within it - from the dragon Smaug."}})

**2. add a synopsis to" The Hobbit:The Desolation 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 possesion of a mysterious and magical ring".**

**A**ns. db.movies.update({title:"The Hobbit:The Desolation Smaug"}, {$set:{synopsis:"The Hobbit:The Desolation 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 possesion of a mysterious and magical rin"}})

**3.add an actor named "Samuel L. Jackson" to the movies "Pulp Fiction"**

**A**ns. db.movies.update({title:' Pulp Fiction'},{$set:{"actors.2": 'Samuel L. Jackson'}})

Text Search

**1.Find all movies that have a synopsis that contains the word "Bilbo".?**

**Ans**. db.Movies.find({synopsis:{$regex:"Bilbo"}})

**2.Find all movies that have a synopsis that contains the word "Gandalf".?**

**Ans** db.Movies.find({synopsis:{$regex:"Gandalf"}})

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

**Ans.** db.Movies.find({$and:[{synopsis:{$regex:"Bilbo"}}, {synopsis:{$not:/Gandalf/}}]})

**4.Find all movies that have a synopsis that contains the word "dwarves" or "hobbit".?**

**Ans** . db.Movies.find({$or:[{synopsis:{$regex:"dwarves"}}, {synopsis:{$regex:"hobbit"}}]})

**5.Find all movies that have a synopsis that contains the word "gold" and "dragon".?**

**Ans**. db.Movies.find({$and:[{synopsis:{$regex:"gold"}}, {synopsis:{$regex:"dragon"}}]})

delete Documents

**1.delete the movies "Pee Wee Hermans Big adventure"**

**Ans.** db.movies.remove({\_id:ObjectId("5c9f992ae5c2dfe9b3729c00")})

or

db.Movies.deleteOne({title:"Pee Wee Hermans Big adventure"})

**2.delete the movie "Avatar"?**

**Ans.** db.Movies.remove({title:"Avatar"})

**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

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"

Querying related collections

**1.find all users?**

**Ans.** db.users.find().pretty()

**2.find all posts?**

**Ans.** db.posts.find().pretty()

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

**Ans.** db.posts.find({username:"GoodGuyGreg"})

**4.find all posts that was authored by "Scumbagsteve"?**

**Ans.** db.posts.find({username:"ScumbagSteve"})

**5.find all comments?**

**Ans .** db.comments.find().pretty()

**6.find all comments that was authored by "GoodGuyGre"g?**

**Ans.** db.comments.find({username:"GoodGuyGreg"})

**7.find all posts that was authored by "ScumbagSteve"?**

**Ans.** db.comments.find({username:"ScumbagSteve"})

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

**Ans.** db.comments.find({"post":ObjectId("615849771767ee1b1b4cb7b2")})