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.

Ans. use mongo\_practice

switched to db mongo\_practice

> db.createCollection('movies')

{ "ok" : 1 }

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 Herman's Big Adventure"

... ... },

... ... {

... ... "title":"Avatar"

... ... }

... ])

Query / Find Documents query the movies collection to

1. get all documents

ans. db.movies.find().pretty()

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

ans. db.movies.find({"writer":"Quentin Tarantino"}).pretty()

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

ans. db.movies.find({"actors":"Brad Pitt"}).pretty()

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

ans. db.movies.find({"franchise":"The Hobbit"}).pretty()

1. get all movies released in the 90s

ans. db.movies.find({"year":{$gte:1990},"year":{$lte:2000}}).pretty()

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

ans. db.movies.find({$or:[{"year":{$lte:2000}},{"year":{$gte:2010}}]}).pretty()

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."

Ans. db.movies.save(

... {

... "\_id" : ObjectId("6159b9630fa85f26dcc99997"),

... "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 dragom 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."

Ans. db.movies.save(

... ... {

... ... "\_id" : ObjectId("6159b9630fa85f26dcc99998"),

... ... "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."

... }

... )

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

Ans. db.movies.update({"actors":["John Travolta","Uma Thurman"]},{$set:{"actors":["Samuel L. Jackson","John Travolta","Uma Thurman"]}})

Text Search

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

ans. db.movies.createIndex({synopsis:"text"})

{

"createdCollectionAutomatically" : false,

"numIndexesBefore" : 1,

"numIndexesAfter" : 2,

"ok" : 1

}

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

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

ans. db.movies.createIndex({synopsis:"text"})

{

"createdCollectionAutomatically" : false,

"numIndexesBefore" : 1,

"numIndexesAfter" : 2,

"ok" : 1

}

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

1. 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/}}]})

1. 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"}}]})

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

Ans. db.movies.remove({\_id: ObjectId("6159b9630fa85f26dcc9999a")})

2. delete the movie "Avatar"

Ans. db.movies.remove({\_id: ObjectId("6159b9630fa85f26dcc9999b")})

Relationships

Insert the following documents into a users collection

db.users.insert({\_id:1,username:"GoodGuyGreg", first\_name:"Good Guy", last\_name:"Greg"})

> db.users.insert({\_id:2, username:"ScumbagSteve", fullname:{first: "Scumbag", last:"Steve"}})

Insert the following documents into a posts collection

db.posts.insert({username:"GoodGuyGreg", title:"Passes out at Party", body:"Raises your credit score"})

> db.posts.insert({ username:"GoodGuyGreg", title:"Steals your identity", body:"Raises your credit score"})

> db.posts.insert({username:"GoodGuyGreg", title:"Reports a bug in your code", body:"Sends you a pull request"})

> db.posts.insert({ username:"ScumbagSteve", title:"Borrows something", body:"Sells it"})

> db.posts.insert({ username:"ScumbagSteve", title:"Borrows everything", body:"The end"})

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

Insert the following documents into a comments collection

db.comments.insert({ username:"GoodGuyGreg", comment:"Hope you got a good deal!", post:ObjectId("6159da6e0fa85f26dcc9999f")})

> db.comments.insert({username:"GoodGuyGreg", comment:"What's mine is yours!", post:ObjectId("6159da880fa85f26dcc999a0")})

> db.comments.insert({username:"GoodGuyGreg", comment:"Don't violate the licensing agreement!", post:ObjectId("6159da970fa85f26dcc999a1")})

> db.comments.insert({username:"ScumbagSteve", comment:"It still isn't clean", post:ObjectId("6159da310fa85f26dcc9999c")})

> db.comments.insert({username:"ScumbagSteve", comment:"Denied your PR cause I found a hack", post:ObjectId("6159da5b0fa85f26dcc9999e")})

Querying related collections

1. find all users

ans. db.users.find().pretty()

1. find all posts

ans. db.posts.find().pretty()

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

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

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

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

1. find all comments

ans.db.comments.find().pretty()

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

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

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

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

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

ans. db.comments.find({post:ObjectId("6159da5b0fa85f26dcc9999e")})