ASSIGNMENT-1

Insert documents

> use mongo\_practice

switched to db mongo\_practice

> 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 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"}])

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("6158d55fda29e983bcd27c95"),

ObjectId("6158d55fda29e983bcd27c96"),

ObjectId("6158d55fda29e983bcd27c97"),

ObjectId("6158d55fda29e983bcd27c98"),

ObjectId("6158d55fda29e983bcd27c99"),

ObjectId("6158d55fda29e983bcd27c9a"),

ObjectId("6158d55fda29e983bcd27c9b"),

ObjectId("6158d55fda29e983bcd27c9c")

]

}

QUERY/FIND DOCUMENTS

1. > db.Movies.find()
2. > db.Movies.find({writer:"quentin tarantino"})
3. > db.Movies.find({actors:"brad pitt"})
4. > db.Movies.find({franchise:"the hobbit"})
5. > db.Movies.find({year:{$lt:"2000"}})
6. > db.Movies.find({$or:[{year:{$lt:"2000"}},{year:{$gt:"2010"}}]})

UPDATE DOCUMENTS

1. > db.Movies.updateOne({title:"the hobbit:an unexpected journey"},{$set:{synopsis:"a reluctant hobbit, bilbo baggins, setd 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."}})

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

1. > db.Movies.updateOne({title:"the hobbit:the desolation of smaug"},{$set:{synopsis:"the dwarves,along with bilbo baggins and the gandalf the grey, continue their quest to reclaim erebor, their homeland, from smaug. bilbo baggins is in possession of a mysterious and magical ring."}})

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

TEXT SEARCH

1. > db.Movies.find({synopsis:{$regex:"bilbo"}})
2. > db.Movies.find({synopsis:{$regex:"Gandalf”}})
3. > db.Movies.find({$and:[{synopsis:{$regex:"bilbo"}},{synopsis:{$not:{$regex:"gandalf"}}}]})
4. > db.Movies.find({$or:[{synopsis:{$regex:"dwarves"}},{synopsis:{$regex:"hobbit"}}]})
5. > db.Movies.find({$and:[{synopsis:{$regex:"gold"}},{synopsis:{$regex:"dragon"}}]})

DELETE DOCUMENTS

1. > db.Movies.remove({title:"pee wee herman's big adventure"})

WriteResult({ "nRemoved" : 1 })

1. > db.Movies.remove({title:"avatar"})

WriteResult({ "nRemoved" : 1 })

RELATIONSHIPS

1. > db.users.find().pretty()
2. > db.posts.find()
3. > db.posts.find({username:"GoodGuyGreg"})
4. > db.posts.find({username:"ScumbagSteve"})
5. > db.comments.find()
6. > db.comments.find({username:"ScumbagSteve"})
7. > db.comments.find({post: ObjectId("61598d8b1d7976e02bb440de")})

ASSIGNMENT-2

Atlanta Population

1. > db.zipcodes.find({$and:[{city:"ATLANTA"},{state:"GA"}]})
2. > db.zipcodes.aggregate([{$match:{city:"ATLANTA",state:"GA"}}])
3. > db.zipcodes.aggregate([{$group:{\_id:{city:"ATLANTA"},count:{$sum:1}}}])
4. > db.zipcodes.aggregate([{$group:{\_id:{city:"ATLANTA"},population:{$sum:'$pop'}}}])

POPULATION BY STATE

1. > db.zipcodes.aggregate([{$group:{\_id:{state:"$state"},population:{$sum:'$pop'}}}])
2. > db.zipcodes.aggregate([{$group:{\_id:{state:"$state"},population:{$sum:'$pop'}}},{$sort:{population:-1}}])
3. > db.zipcodes.aggregate([{$group:{\_id:{state:"$state"},population:{$sum:'$pop'}}},{$sort:{population:-1}},{$limit:3}])

Population by city

1. > db.zipcodes.aggregate([{$group:{\_id:{city:"$city",state:"$state"},population:{$sum:'$pop'}}}])
2. > db.zipcodes.aggregate([{$group:{\_id:{city:"$city",state:"$state"},population:{$sum:'$pop'}}},{$sort:{population:-1}}])
3. > db.zipcodes.aggregate([{$group:{\_id:{city:"$city",state:"$state"},population:{$sum:'$pop'}}},{$sort:{population:-1}},{$limit:3}])
4. > db.zipcodes.aggregate([{$group:{\_id:{city:"TEXAS",state:"$state"},population:{$sum:'$pop'}}},{$sort:{population:-1}},{$limit:3}])

ASSIGNMENT 3

1. > db.addresses.find()
2. > db.addresses.aggregate([{$project:{restaurant\_id:1,name:1,borough:1,cuisine:1}}])
3. > db.addresses.aggregate([{$project:{restaurant\_id:1,name:1,borough:1,cuisine:1,\_id:0}}])
4. > db.addresses.aggregate([{$project:{restaurant\_id:1,name:1,borough:1,zipcode:1,\_id:0}}])
5. > db.addresses.find({},{"borough":"Bronx"}).limit(5)
6. > db.addresses.find({},{borough:"Bronx"})
7. > db.addresses.find({},{borough:"Bronx"}).skip(5).limit(5)
8. > db.addresses.find({"grades.score":{$gt:90}})
9. > db.addresses.find({"grades.score":{$gt:80,$lt:100}})
10. > db.addresses.find({"address.coord.0":{$lt:-95.754168}})
11. >db.addresses.find({$and:[{"cuisine":{$ne:"American"}},{"grades.score":{$lte:70}},{"address.coord.0":{$lt:-65.754168}}]})
12. >db.addresses.find({$and:[{"cuisine":{$ne:"American"}},{"grades.score":{$lte:70}},{"address.coord.0":{$lt:-65.754168}}]})
13. >db.addresses.find({$and:[{"cuisine":{$ne:"American"}},{"grades.grade":"A"},{"borough":{$ne:"Brooklyn"}}]}).sort({"cuisine":1})
14. >db.addresses.find({"name":{$regex:/^Wil/i}},{"\_id":0,"restaurant\_id":1,"name":1,"borough":1,"cuisine":1})
15. >db.addresses.find({"name":{$regex:/.ces$/}},{"\_id":0,"restaurant\_id":1,"name":1,"borough":1,"cuisine":1})
16. >db.addresses.find({"name":{$regex:"Reg"}},{"\_id":0,"restaurant\_id":1,"name":1,"borough":1,"cuisine":1})
17. > db.addresses.find({"borough":"Bronx"},{"cuisine":{$ne:["American","Chinese"]}})

20.>db.addresses.find({"grades.score":{$lte:10}},{"\_id":0,"restaurant\_id":1,"name":1,"borough":1,"cuisine":1}) 21.<db.addresses.find({$nor:[{"cuisine":["American","Chinese"]},{"name":{$regex:/^Wil/i}}]},{"\_id":0,"restaurant\_id":1,"borough":1,"name":1,"cuisine":1})

22. > db.addresses.find({$and:[{"grades.date":ISODate("2014-08-11T00:00:00Z")},{"grades.grade":"A"},{"grades.score":11}]},{"\_id":0,"restaurant\_id":1,"name":1,"grades.grade":1})

23. > db.addresses.find({$and:[{"grades.date":ISODate("2014-08-11T00:00:00Z")},{"grades.1.grade":"A"},{"grades.score":9}]},{"\_id":0,"restaurant\_id":1,"name":1,"grades.grade":1})

24. > db.addresses.find().sort({"name":-1})

25. > db.addresses.find().sort({"name":-1})

26. > db.addresses.find().sort({name:-1})

27. > db.addresses.find().sort({cuisine:-1},{borough:1})

27. > db.addresses.find().sort({cuisine:1},{borough:-1})

28. > db.addresses.find({"address.street":{$exists:true}})

29. > db.addresses.find({"address.coord":{$type:"double"}})

30. > db.addresses.find({"grades.score":{$mod:[7,0]}},{\_id:0,restaurant\_id:1,name:1,grades:1})

31. > db.addresses.find({name:{$regex:"mon"}},{\_id:0,cuisine:1,name:1,"address.coord":1,borough:1})

32. > db.addresses.find({name:{$regex:/^Mad/}},{\_id:0,cuisine:1,name:1,"address.coord":1,borough:1})