1. Db.movies.find().pretty()
2. db.movies.find({"writer": "Quentin Tarantino"}).pretty()
3. db.movies.find({"actors": "Brad Pitt"}).pretty()
4. db.movies.find({"franchise": "The Hobbit"}).pretty()
5. db.movies.find({$and: [{year: {$gt: 1900}}, {year: {$lt: 2000}}]})
6. db.movies.find({$or: [{year: {$lt: 2000}}, {year: {$gt: 2010}}]})

Update Documents

1. db.movies.update({title: "The Hobbit: An Unexpected Journey"}, {$set: {synopsis: "A reluctant… "}})
2. same as above
3. db.movies.update({title: "Pulp Fiction"}, {$push: {actors: "Samuel L. Jackson"}})

Text Search: <https://docs.mongodb.com/manual/reference/operator/query/text/>

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

1. db.movies.find({$text: {$search: "Bilbo"}}).pretty()
2. db.movies.find({$text: {$search: "Gandalf"}}).pretty()

Prefixing a word with a hyphen-minus (-) negates a word:

1. db.movies.find({$text: {$search: "Bilbo -Gandalf"}}).pretty()

If the search string is a space-delimited string, [$text](https://docs.mongodb.com/manual/reference/operator/query/text/#op._S_text) operator performs a logical OR search on each term and returns documents that contains any of the terms.

1. db.movies.find({$text: {$search: "dwarves hobbit"}}).pretty()
2. db.movies.find({$text: {$search: ‘"gold" "dragon"’}}).pretty()

Delete Documents

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

Which to follow: delete with objectId or just with title or some other features?

Relationships

1. db.posts.find({username: "GoodGuyGreg"}).pretty()
2. db.posts.find({username: "ScumbagSteve"}).pretty()
3. db.comments.find({username: "GoodGuyGreg"}).pretty()
4. db.comments.find({username: "ScumbagSteve"}).pretty()
5. db.comments.find({post: db.posts.findOne({title: "Reports a bug in your code"}).\_id}).pretty()

**MongoDB -Aggregation Exercises**

1. db.zipcodes.find({city: "ATLANTA", state: "GA"}).pretty()
2. db.zipcodes.aggregate({$match : {city: "ATLANTA", state: "GA"}}).pretty()
3. db.zipcodes.aggregate( [ {$match: {city: 'ATLANTA'} }, {$group: { \_id: '$city', Count: {$sum: 1}}}]) : 41
4. db.zipcodes.aggregate([{$match: {city: 'ATLANTA'}}, {$group: {\_id: {city: '$city'}, Population: {$sum: '$pop'}}}]) : 630046

**Populations 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}])

**Populations by City =>**

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

**Bonus =>**

1. db.zipcodes.aggregate([{$group: {\_id: {state: '$state', city: '$city'}, Pop: {$avg: '$pop'}}}])
2. db.zipcodes.aggregate([{$group: {\_id: {state: '$state'}, AvgPop: {$avg: '$pop'}}}, {$sort: {AvgPop: -1}}, {$limit: 3}])

**MongoDB – Complex Queries**

1. db.addresses.find().pretty()
2. db.addresses.find({}, {"restaurant\_id": 1, "name": 1, "borough": 1, "cuisine": 1}).pretty()
3. db.addresses.find({}, {"\_id": 0,"restaurant\_id": 1, "name": 1, "borough": 1, "cuisine": 1}).pretty()
4. db.addresses.find({}, {"\_id": 0,"restaurant\_id": 1, "name": 1, "borough": 1, "address.zipcode": 1}).pretty()
5. db.addresses.find({borough: 'Bronx'}).limit(5).pretty()
6. db.addresses.find({borough: 'Bronx'}).pretty()
7. db.addresses.find({borough: 'Bronx'}).skip(5).limit(5).pretty()
8. db.addresses.find({"grades.score": {$gt: 90}}).pretty()

or

db.addresses.find({grades : { $elemMatch:{"score":{$gt : 90}}}}).pretty()

1. db.addresses.find({grades : { $elemMatch:{"score":{$gt : 80 , $lt :100}}}}).pretty()
2. db.addresses.find({"address.coord.0": {$lt: -95.754168}}).pretty()
3. db.addresses.find({$and:[{"cuisine" : {$ne :"American "}},{"grades.score" : {$gt : 70}},{"address.coord.0" : {$lt : -65.754168}}]}).pretty()
4. db.addresses.find({$and:[{"cuisine" : {$ne :"American "}},{"grades.score" : {$gt : 70}},{"address.coord.1" : {$lt : -65.754168}}]}).pretty()

or

db.addresses.find({"cuisine" : {$ne :"American "},"grades.score" : {$gt : 70},"address.coord.1" : {$lt : -65.754168}}).pretty()

1. db.addresses.find({"cuisine" : {$ne :"American "},"grades.grade" : 'A', "borough": {$ne: 'Brooklyn'}}).sort({cuisine: -1}).pretty()
2. db.addresses.find({name: {$regex: /^Wil/}}, {"restaurant\_id": 1, "name": 1, "borough": 1, "cuisine": 1, "\_id": 0}).pretty()
3. db.addresses.find({name: {$regex: /ces$/}}, {"restaurant\_id": 1, "name": 1, "borough": 1, "cuisine": 1, "\_id": 0}).pretty()
4. db.addresses.find({name: {$regex: /Reg/}}, {"restaurant\_id": 1, "name": 1, "borough": 1, "cuisine": 1, "\_id": 0}).pretty()
5. db.addresses.aggregate([{$or: [{cuisine: 'American'},{cuisine: 'Chinese'}], borough: 'Bronx'}, {$group: {..}}])
6. db.addresses.find({borough: {$in: ["Staten Island","Queens","Bronx","Brooklyn"]}}, {"restaurant\_id": 1, "name": 1, "borough": 1, "cuisine": 1}).pretty()
7. db.addresses.find({borough: {$in: ["Staten Island","Queens","Bronx","Brooklyn"]}}, {"restaurant\_id": 1, "name": 1, "borough": 1, "cuisine": 1, “\_id”: 0}).count()
8. db.addresses.find({"grades.score": {$not : {$gt: 10}}},{"restaurant\_id": 1, "name": 1, "borough": 1, "cuisine": 1,"\_id": 0}).pretty() or

We can also use $elemMatch

1. db.addresses.find({$or: [{name: {$regex: /^Wil/}},{"$and": [{"cuisine" : {$ne :"American "}},{"cuisine": {$ne :"Chinese"}}]}]} ,{"restaurant\_id" : 1,"name":1,"borough":1,"cuisine" :1, "\_id": 0 });
2. db.addresses.find({grades : {$elemMatch:{ "score": 11, "date": ISODate("2014-08-11T00:00:00Z"),"grade": 'A'}}},{"restaurant\_id" : 1,"name":1,"grades":1).pretty()
3. .
4. .
5. db.addresses.find({},{"\_id": 0,"name": 1}).sort({"name":1}).limit(5).pretty()
6. db.addresses.find({},{"\_id": 0,"name": 1}).sort({"name": -1}).limit(5).pretty()
7. db.addresses.find({},{"\_id": 0,"borough": 1, "cuisine": 1, "name": 1}).sort({"cuisine": 1, "borough": -1}).limit(5).pretty()
8. db.addresses.find({"address.street": {$exists: true}}, {"\_id": 0, "address.street": 1, "name": 1}).limit(5).pretty()
9. db.addresses.find({"address.coord": {$type: 1}},{"\_id": 0, "name": 1, "address.coord": 1}).limit(2).pretty()
10. db.addresses.find({"grades.score": {$mod: [7, 0]}}, {"\_id": 0, "name": 1, "grades": 1}).limit(2).pretty()
11. db.addresses.find({name: {$regex: /mon/}}, {"\_id": 0, "name": 1, "borough": 1, "address.coord": 1, "cuisine": 1}).limit(2).pretty()
12. db.addresses.find({name: {$regex: /^Mad/}}, {"\_id": 0, "name": 1, "borough": 1, "address.coord": 1}).limit(2).pretty()