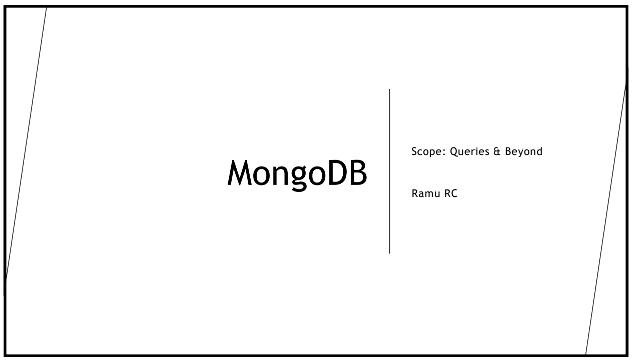
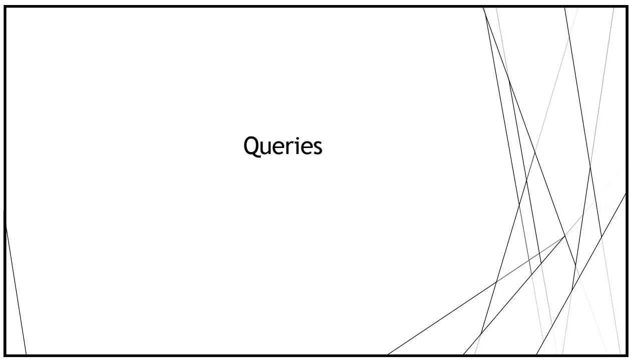


MongoDB

Scope: Queries & Beyond

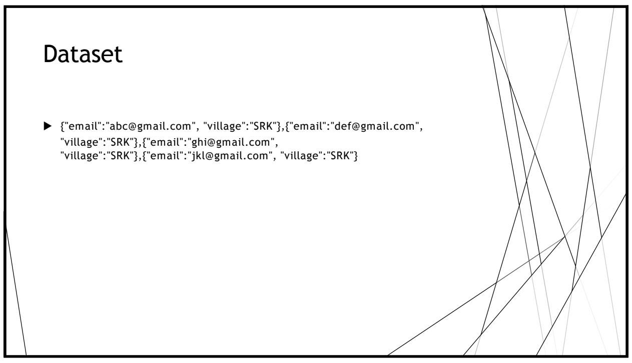
Ramu RC

1



Queries

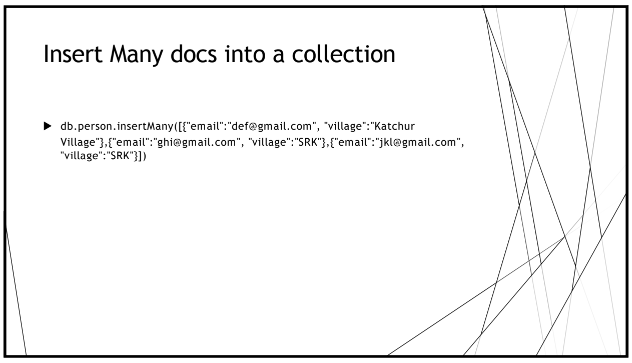
2



Dataset

- {"email":"abc@gmail.com", "village":"SRK"},{"email":"def@gmail.com", "village":"SRK"},{"email":"ghi@gmail.com", "village":"SRK"},{"email":"jkl@gmail.com", "village":"SRK"}

3



Insert Many docs into a collection

- db.person.insertMany([{"email":"def@gmail.com", "village":"Katchur Village"},{"email":"ghi@gmail.com", "village":"SRK"},{"email":"jkl@gmail.com", "village":"SRK"}])

4

And the result is

- { "_id" : ObjectId("5e27a1b4342ecb499af53b41"), "email" : "abc@gmail.com", "village" : "SRK" }
- { "_id" : ObjectId("5e27a292342ecb499af53b42"), "email" : "def@gmail.com", "village" : "SRK" }
- { "_id" : ObjectId("5e27a2cd342ecb499af53b43"), "email" : "def@gmail.com", "village" : "SRK" }
- { "_id" : ObjectId("5e27a2cd342ecb499af53b44"), "email" : "ghi@gmail.com", "village" : "SRK" }
- { "_id" : ObjectId("5e27a2cd342ecb499af53b45"), "email" : "jkl@gmail.com", "village" : "SRK" }
- { "_id" : ObjectId("5e27a306342ecb499af53b46"), "email" : "def@gmail.com", "village" : "SRK" }
- { "_id" : ObjectId("5e27a306342ecb499af53b47"), "email" : "ghi@gmail.com", "village" : "SRK" }
- { "_id" : ObjectId("5e27a306342ecb499af53b48"), "email" : "jkl@gmail.com", "village" : "SRK" }

5

Regex

- > db.person.find({"village" : /R/})
- > db.person.find({"email" : /gmail/})
- > db.person.find({"email" : /yahoo/})

6

Update Query

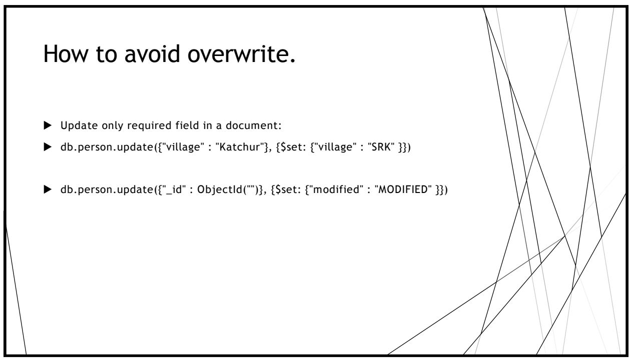
- searchCriteria = {"village" : "SRK"}
- updateDetails = {"village" : "Katchur"}
- options = {"multi": true}
- db.person.update(searchCriteria, updateDetails)

7

Update contd.,

- db.person.update({"village" : "Katchur"}, {"email" : "abc@gmail.com", "village" : "SRK" })
- db.person.update({"_id":"5e27a1b4342ecb499af53b41"}, {"village" : "Katchur"})
- db.person.update({"_id":ObjectId("5e27a1b4342ecb499af53b41")}, {"village" : "Katchur"})

8

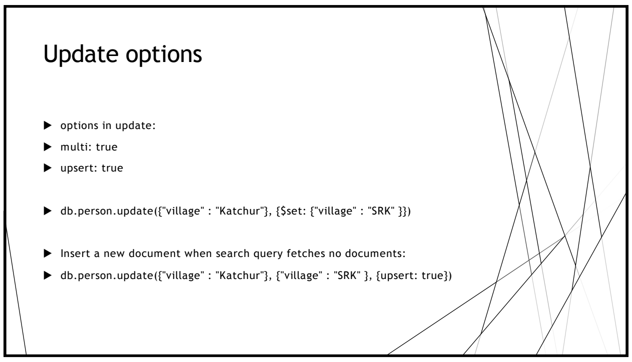


How to avoid overwrite.

- Update only required field in a document:
- db.person.update({"village" : "Katchur"}, {$set: {"village" : "SRK" }})

- db.person.update({"_id" : ObjectId("")}, {$set: {"modified" : "MODIFIED" }})

9



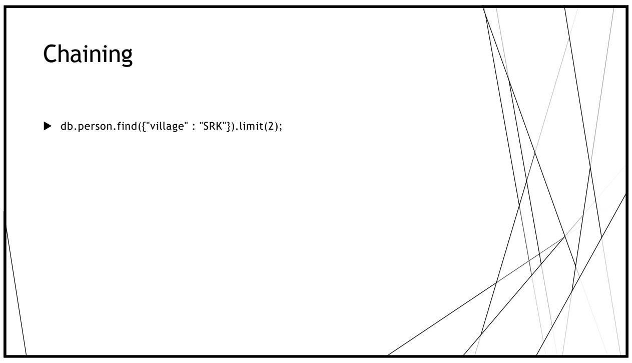
Update options

- options in update:
- multi: true
- upsert: true

- db.person.update({"village" : "Katchur"}, {$set: {"village" : "SRK" }})

- Insert a new document when search query fetches no documents:
- db.person.update({"village" : "Katchur"}, {"village" : "SRK" }, {upsert: true})

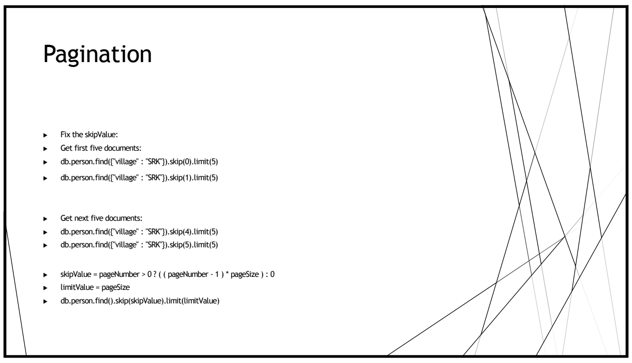
10



Chaining

- db.person.find({"village" : "SRK"}).limit(2);

11



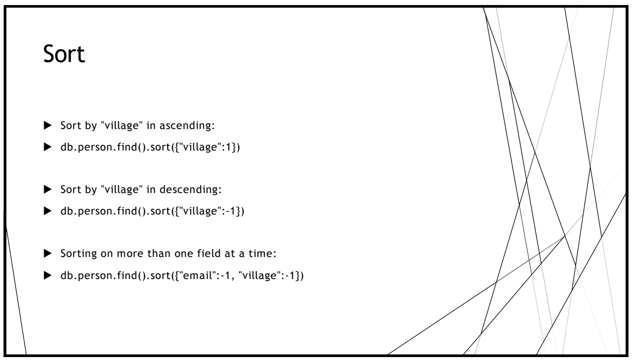
Pagination

- Fix the skipValue:
- Get first five documents:
- db.person.find({"village" : "SRK"}).skip(0).limit(5)
- db.person.find({"village" : "SRK"}).skip(1).limit(5)

- Get next five documents:
- db.person.find({"village" : "SRK"}).skip(4).limit(5)
- db.person.find({"village" : "SRK"}).skip(5).limit(5)

- skipValue = pageNumber > 0 ? ((pageNumber - 1) * pageSize) : 0
- limitValue = pageSize
- db.person.find().skip(skipValue).limit(limitValue)

12



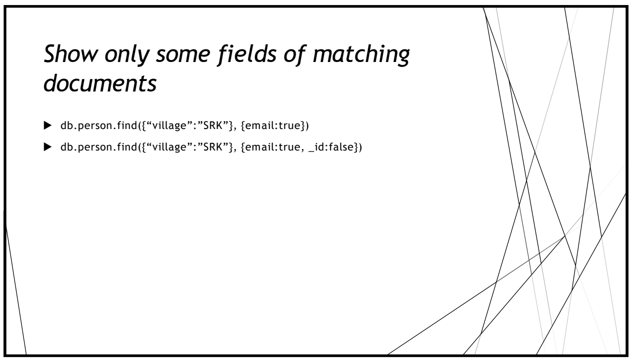
Sort

- Sort by "village" in ascending:
- db.person.find().sort({"village":1})

- Sort by "village" in descending:
- db.person.find().sort({"village":-1})

- Sorting on more than one field at a time:
- db.person.find().sort({"email":-1, "village":-1})

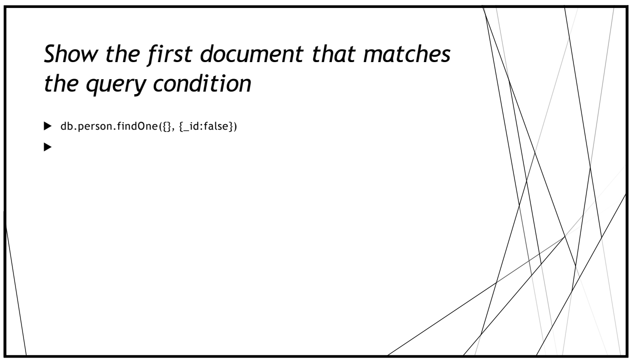
13



Show only some fields of matching documents

- db.person.find({“village”:”SRK”}, {email:true})
- db.person.find({“village”:”SRK”}, {email:true, _id:false})

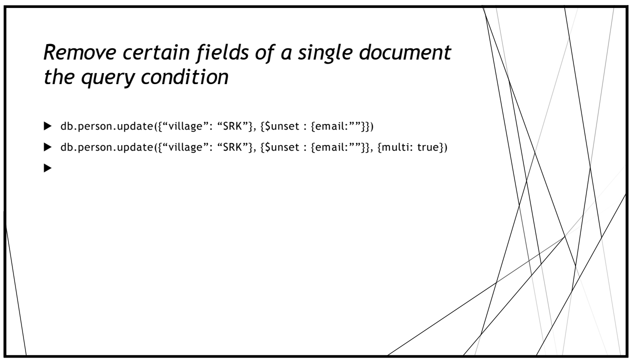
14



Show the first document that matches the query condition

- db.person.findOne({}, {_id:false})
-

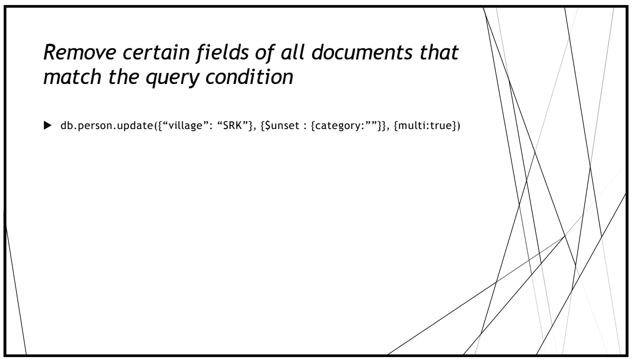
15



Remove certain fields of a single document the query condition

- db.person.update({“village”: “SRK”}, {$unset : {email:””}})
- db.person.update({“village”: “SRK”}, {$unset : {email:””}}, {multi: true})
-

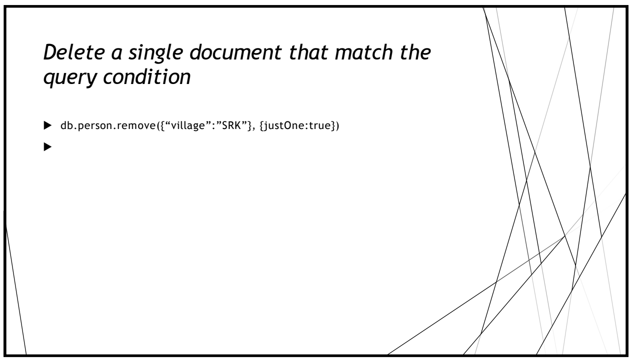
16



Remove certain fields of all documents that match the query condition

- db.person.update({“village”: “SRK”}, {$unset : {category:””}}, {multi:true})

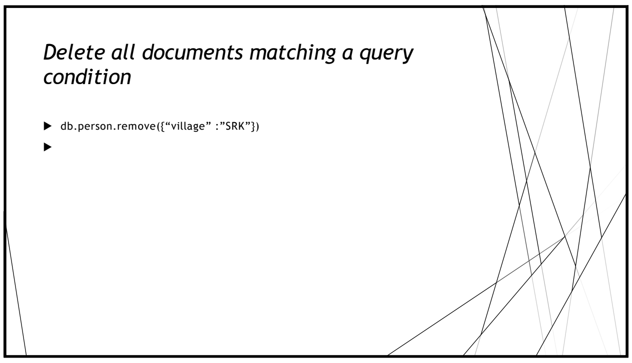
17



Delete a single document that match the query condition

- db.person.remove({“village”:”SRK”}, {justOne:true})
-

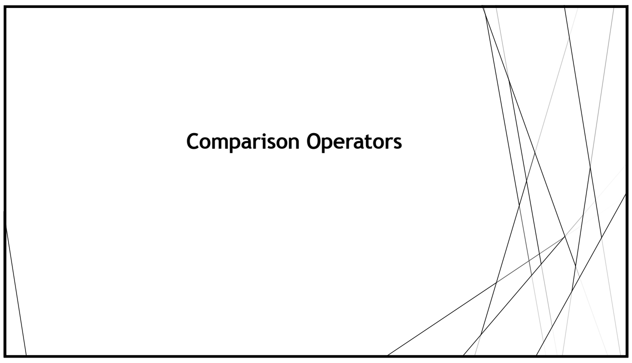
18



Delete all documents matching a query condition

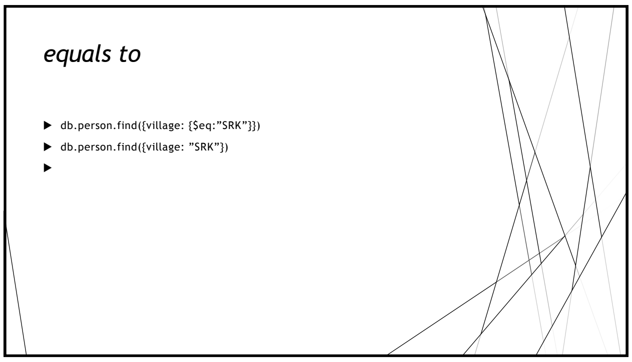
- db.person.remove({“village” :”SRK”})
-

19



Comparison Operators

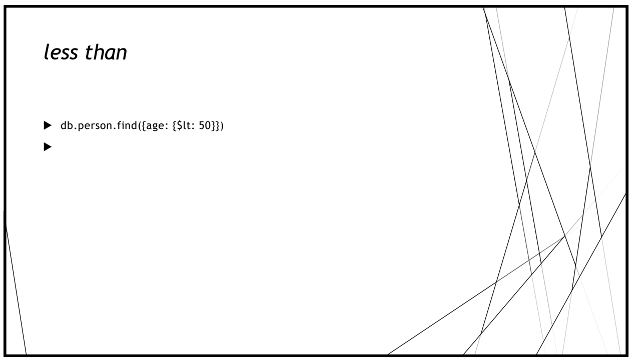
20



equals to

- db.person.find({village: {$eq:”SRK”}})
- db.person.find({village: ”SRK”})
-

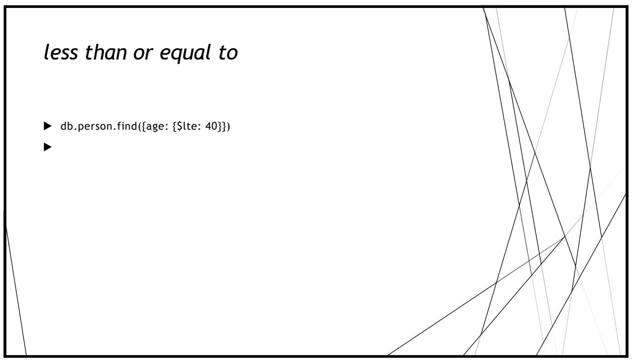
21



less than

- db.person.find({age: {$lt: 50}})
-

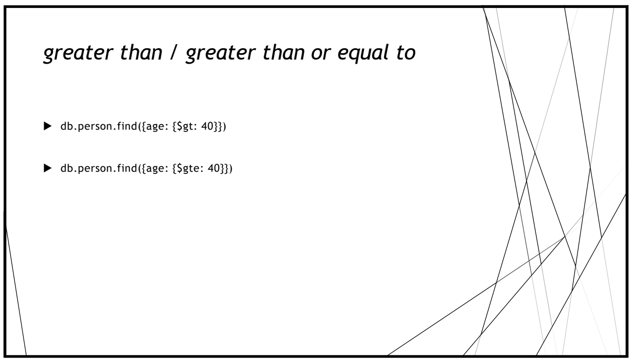
22



less than or equal to

- db.person.find({age: {$lte: 40}})
-

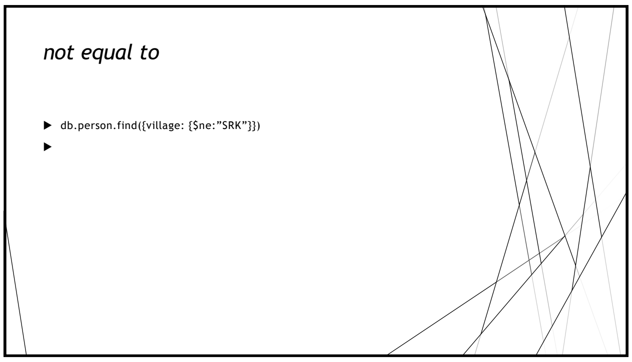
23



greater than / greater than or equal to

- db.person.find({age: {$gt: 40}})
- db.person.find({age: {$gte: 40}})

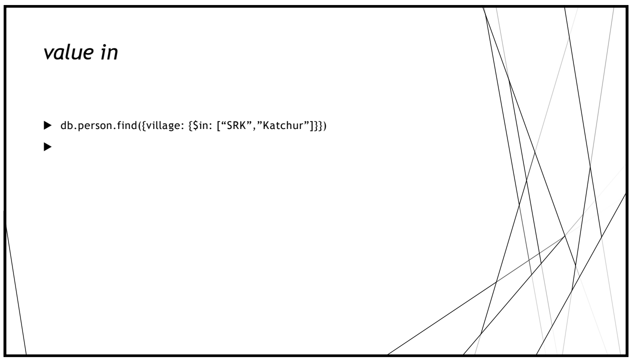
24



not equal to

- db.person.find({village: {$ne:”SRK”}})
-

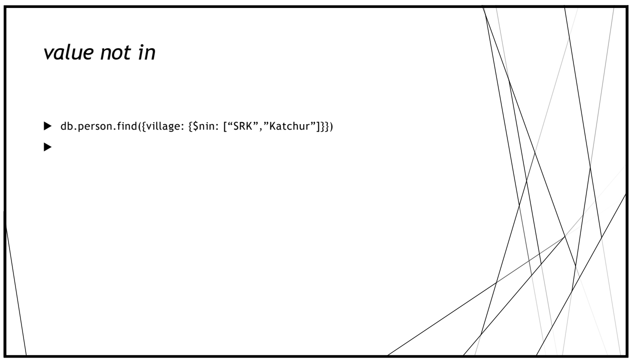
25



value in

- db.person.find({village: {$in: [“SRK”,”Katchur”]}})
-

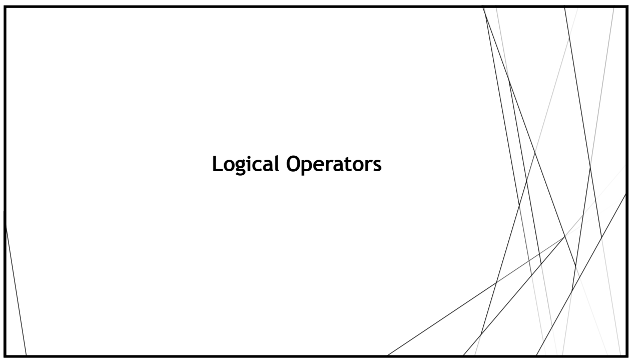
26



value not in

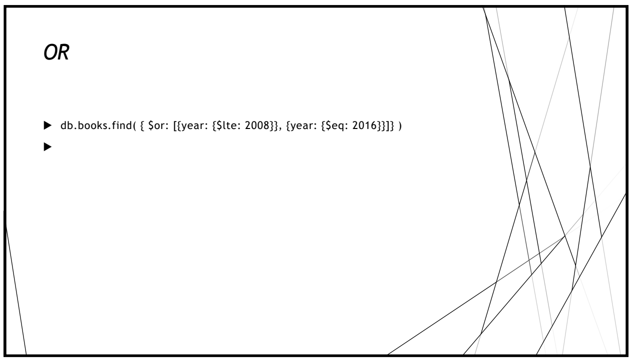
- db.person.find({village: {$nin: [“SRK”,”Katchur”]}})
-

27



Logical Operators

28



OR

- db.books.find({ $or: [{year: {$lte: 2008}}, {year: {$eq: 2016}}]})
-

29

AND

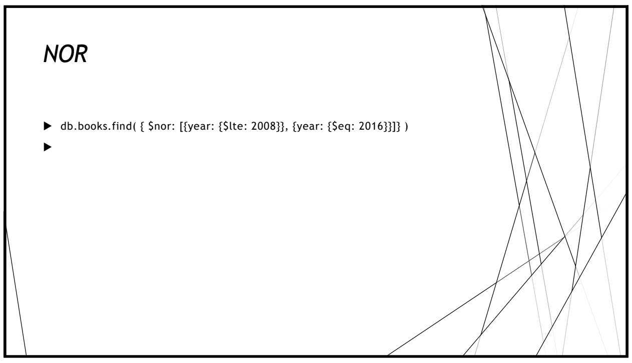
- db.books.find({ $and: [{year: {$eq: 2008}}, {category: {$eq: “Fiction”}}]})
-

30

NOT

- db.books.find({$not: {year: {$eq: 2016} }})
-

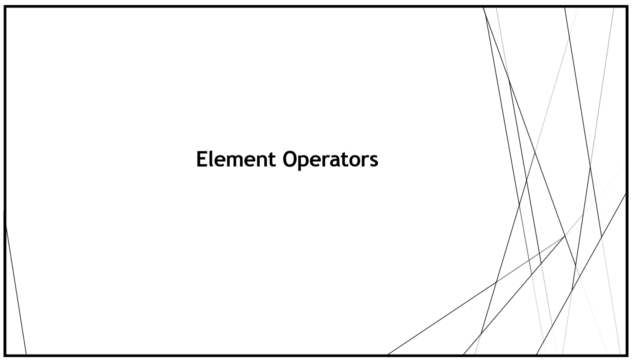
31



NOR

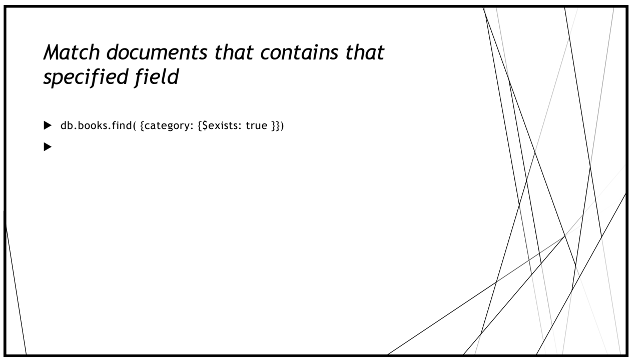
- db.books.find({ $nor: [{year: {$lte: 2008}}, {year: {$eq: 2016}}]})
-

32



Element Operators

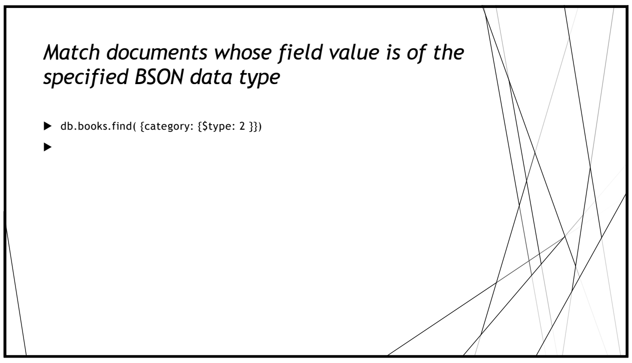
33



Match documents that contains that specified field

- db.books.find({category: {$exists: true }})
-

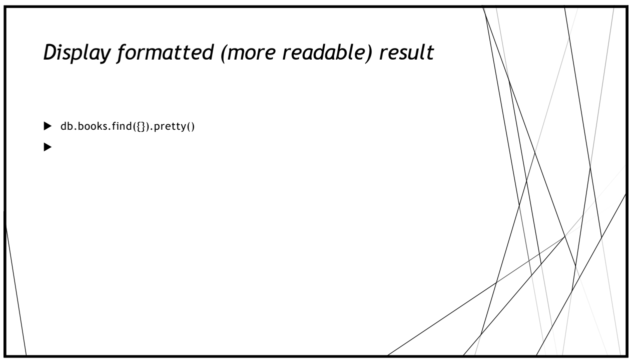
34



Match documents whose field value is of the specified BSON data type

- db.books.find({category: {$type: 2 }})
-

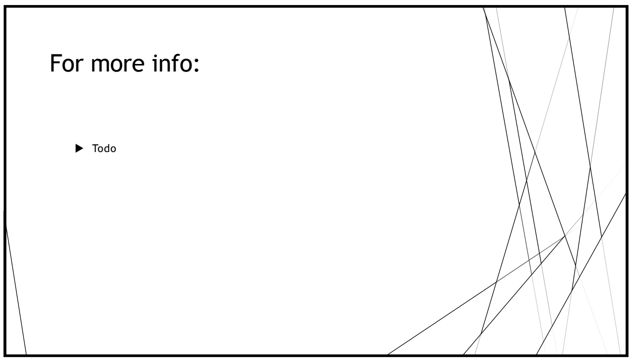
35



Display formatted (more readable) result

- db.books.find({}).pretty()
-

36



For more info:

- Todo

37