|  |
| --- |
| Mongo DB Assignment – 1 |
|  |  |
|  | * Query/find Documents: |
|  | * To get all the documents: |
|  | * db.movies.find() |
|  | * To get all the documents with writer set to “Quentin Tarantino”: |
|  | * db.movies.find({writer :"Quentin Tarantino"}) |
|  | * To get all the documents where actors include “Brad Pitt”: |
|  | * db.movies.find({actors:'Brad Pitt'}) |
|  | * To get all the documents with franchise set to “The Hobbit”: |
|  | * db.movies.find({Franchise:"The Hobbit"}) |
|  | * To get all movies released in the 90s: |
|  | * db.movies.find({$and:[{year:{$gt:1990}},{year:{$lt$lt:2000}}]}) |
|  | * To get movies released before the year 2000 or after 2010: |
|  | * db.movies.find({$or:[{year:{$lt:2000}},{year:{$gt:2010}}]}) |
|  |  |
|  | * Text Search: |
|  | * Find all movies that have a synopsis that contains the word “Bilbo”. |
|  | * db.movies.getIndexes() |
|  | * db.movies.createIndex({synopsis:"text"}) |
|  | * db.movies.getIndexes() |
|  | * db.movies.find({$text:{$search:"Bilbo"}}) |
|  | * or |
|  | * db.movies.find({synopsis:/Bilbo/g}) |
|  | * Find all movies that have a synopsis that contains the word “Gandalf”. |
|  | * db.movies.find({$text:{$search:"Gandalf"}}) |
|  | * all movies that have synopsis that contains the word “Bilbo” and not the word “Gandalf”. |
|  | * db.movies.find({$text:{$search:"bilbo -gandlf"}}) |
|  | * Find all movies that have a synopsis that contains the word “dwarves” or “hobbit”. |
|  | * db.movies.find({$text:{$search:"dwarves hobbit"}}) |
|  | * Find all movies that have a synopsis that contains the word “gold” and “dragon”. |
|  | * db.movies.find({$text:{$search:"gold +dragon"}}) |
|  |  |
|  | * Delete Documents: |
|  | * Delete the movie “Pee Wee Herman’s big Adventure” |
|  | * db.movies.remove({title:"Pee Wee Herman's Big Adventure"}) |
|  | * Delete the movie “Avatar” |
|  | * db.movies.remove({title:"Avatar"}) |
|  |  |
|  | * Relationships: |
|  | * Inserting documents into the users collection: |
|  | * db.users.insertOne({username:"GoodGuyGrey",first\_name:"GoodGuy",last\_name:"Grey"}) |
|  | * db.users.insertOne({username:"ScumbagSteve",full\_name:[{first:"Scumbag",last:"Steve"}]}) |
|  | * db.users.find() |
|  | * [ |
|  | * { |
|  | * \_id: ObjectId("617e5426c4057dd5ea3716d3"), |
|  | * username: 'GoodGuyGrey', |
|  | * first\_name: 'GoodGuy', |
|  | * last\_name: 'Grey' |
|  | * }, |
|  | * { |
|  | * \_id: ObjectId("617e54a8c4057dd5ea3716d4"), |
|  | * username: 'ScumbagSteve', |
|  | * full\_name: [ { first: 'Scumbag', last: 'Steve' } ] |
|  | * } |
|  | * ] |
|  | * Inserting documents into the posts collection: |
|  | * db.posts.insertMany([{username:"GoodGuyGrey",titletitle:"Passes out at party",body:"Wakes up early and cleans house"},{username:"GoodGuyGrey",title:"Steals your identity",body:"Raises your credit score"},{username:"GoodGuyGrey",title:"Reports a bug in your code",body:"Sends you a Pull Request"}]) |
|  | db.posts.insertMany([{username:"ScumbagSteve",titletitle:"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"}]) |
|  | * • db.posts.find() |
|  | * [ |
|  | * { |
|  | * \_id: ObjectId("617e58c5c4057dd5ea3716de"), |
|  | * username: 'GoodGuyGrey', |
|  | * titletitle: 'Passes out at party', |
|  | * body: 'Wakes up early and cleans house' |
|  | * }, |
|  | * { |
|  | * \_id: ObjectId("617e58c5c4057dd5ea3716df"), |
|  | * username: 'GoodGuyGrey', |
|  | * title: 'Steals your identity', |
|  | * body: 'Raises your credit score' |
|  | * }, |
|  | * { |
|  | * \_id: ObjectId("617e58c5c4057dd5ea3716e0"), |
|  | * username: 'GoodGuyGrey', |
|  | * title: 'Reports a bug in your code', |
|  | * body: 'Sends you a Pull Request' |
|  | * }, |
|  | * { |
|  | * \_id: ObjectId("617e58e0c4057dd5ea3716e1"), |
|  | * username: 'ScumbagSteve', |
|  | * titletitle: 'Borrows something', |
|  | * body: 'Sells it' |
|  | * }, |
|  | * { |
|  | * \_id: ObjectId("617e58e0c4057dd5ea3716e2"), |
|  | * username: 'ScumbagSteve', |
|  | * title: 'Borrows everything', |
|  | * body: 'The end' |
|  | * }, |
|  | * { |
|  | * \_id: ObjectId("617e58e0c4057dd5ea3716e3"), |
|  | * username: 'ScumbagSteve', |
|  | * title: 'Forks your repo on github', |
|  | * body: 'Sets to private' |
|  | * } |
|  | * ] |
|  | * Inserting documents into the comments collection: |
|  | * db.comments.insertOne({username:"GoodGuyGrey",comment:"Hope you got a good deal!",post: ObjectId("617e58e0c4057dd5ea3716e1")}) |
|  | * db.comments.insertOne({username:"GoodGuyGrey",comment:"What mine is yours",post: ObjectId("617e58e0c4057dd5ea3716e2")}) |
|  | * db.comments.insertOne({username:"GoodGuyGrey",comment:"Don't voilate the licensing agreement!",post: ObjectId("617e58e0c4057dd5ea3716e3")}) |
|  | * db.comments.insertOne({username:"ScumbagSteve",comment:"It still isn't clean",post: ObjectId("617e58c5c4057dd5ea3716de")}) |
|  | * db.comments.insertOne({username:"ScumbagSteve",comment:"Denied your PR cause I found a hack",post: ObjectId("617e58c5c4057dd5ea3716e0")}) |
|  | * db.comments.find() |
|  | * [ |
|  | * { |
|  | * \_id: ObjectId("617e59c0c4057dd5ea3716e5"), |
|  | * username: 'GoodGuyGrey', |
|  | * comment: 'Hope you got a good deal!', |
|  | * post: ObjectId("617e58e0c4057dd5ea3716e1") |
|  | * }, |
|  | * { |
|  | * \_id: ObjectId("617e5bcbc4057dd5ea3716e6"), |
|  | * username: 'GoodGuyGrey', |
|  | * comment: 'What mine is yours', |
|  | * post: ObjectId("617e58e0c4057dd5ea3716e2") |
|  | * }, |
|  | * { |
|  | * \_id: ObjectId("617e5e18c4057dd5ea3716e7"), |
|  | * username: 'GoodGuyGrey', |
|  | * comment: "Don't voilate the licensing agreement!", |
|  | * post: ObjectId("617e58e0c4057dd5ea3716e3") |
|  | * }, |
|  | * { |
|  | * \_id: ObjectId("617e5e68c4057dd5ea3716e8"), |
|  | * username: 'ScumbagSteve', |
|  | * comment: "It still isn't clean", |
|  | * post: ObjectId("617e58c5c4057dd5ea3716de") |
|  | * }, |
|  | * { |
|  | * \_id: ObjectId("617e5eb0c4057dd5ea3716e9"), |
|  | * username: 'ScumbagSteve', |
|  | * comment: 'Denied your PR cause I found a hack', |
|  | * post: ObjectId("617e58c5c4057dd5ea3716e0") |
|  | * } |
|  | * ] |
|  | * Querying related collections: |
|  | * Find all users |
|  | * db.users.find() |
|  | * Find all posts |
|  | * db.posts.find() |
|  | * Find all posts that was authored by “GoodGuyGreg” |
|  | * db.posts.find({username:"GoodGuyGrey"}) |
|  | * [ |
|  | * { |
|  | * \_id: ObjectId("617e58c5c4057dd5ea3716de"), |
|  | * username: 'GoodGuyGrey', |
|  | * titletitle: 'Passes out at party', |
|  | * body: 'Wakes up early and cleans house' |
|  | * }, |
|  | * { |
|  | * \_id: ObjectId("617e58c5c4057dd5ea3716df"), |
|  | * username: 'GoodGuyGrey', |
|  | * title: 'Steals your identity', |
|  | * body: 'Raises your credit score' |
|  | * }, |
|  | * { |
|  | * \_id: ObjectId("617e58c5c4057dd5ea3716e0"), |
|  | * username: 'GoodGuyGrey', |
|  | * title: 'Reports a bug in your code', |
|  | * body: 'Sends you a Pull Request' |
|  | * } |
|  | * ] |
|  | * Find all posts that was authored by “ScumbagSteve” |
|  | * db.posts.find({username:"ScumbagSteve"}) |
|  | * [ |
|  | * { |
|  | * \_id: ObjectId("617e58e0c4057dd5ea3716e1"), |
|  | * username: 'ScumbagSteve', |
|  | * titletitle: 'Borrows something', |
|  | * body: 'Sells it' |
|  | * }, |
|  | * { |
|  | * \_id: ObjectId("617e58e0c4057dd5ea3716e2"), |
|  | * username: 'ScumbagSteve', |
|  | * title: 'Borrows everything', |
|  | * body: 'The end' |
|  | * }, |
|  | * { |
|  | * \_id: ObjectId("617e58e0c4057dd5ea3716e3"), |
|  | * username: 'ScumbagSteve', |
|  | * title: 'Forks your repo on github', |
|  | * body: 'Sets to private' |
|  | * } |
|  | * ] |
|  | * Find all comments |
|  | * db.comments.find() |
|  | * Find all comments that was authored by “GoodGuyGreg” |
|  | * db.comments.find({username:"GoodGuyGrey"}) |
|  | * [ |
|  | * { |
|  | * \_id: ObjectId("617e59c0c4057dd5ea3716e5"), |
|  | * username: 'GoodGuyGrey', |
|  | * comment: 'Hope you got a good deal!', |
|  | * post: ObjectId("617e58e0c4057dd5ea3716e1") |
|  | * }, |
|  | * { |
|  | * \_id: ObjectId("617e5bcbc4057dd5ea3716e6"), |
|  | * username: 'GoodGuyGrey', |
|  | * comment: 'What mine is yours', |
|  | * post: ObjectId("617e58e0c4057dd5ea3716e2") |
|  | * }, |
|  | * { |
|  | * \_id: ObjectId("617e5e18c4057dd5ea3716e7"), |
|  | * username: 'GoodGuyGrey', |
|  | * comment: "Don't voilate the licensing agreement!", |
|  | * post: ObjectId("617e58e0c4057dd5ea3716e3") |
|  | * } |
|  | * ] |
|  | * Find all comments that was authored by “ScumbagSteve” |
|  | * db.comments.find({username:"ScumbagSteve"}) |
|  | * [ |
|  | * { |
|  | * \_id: ObjectId("617e5e68c4057dd5ea3716e8"), |
|  | * username: 'ScumbagSteve', |
|  | * comment: "It still isn't clean", |
|  | * post: ObjectId("617e58c5c4057dd5ea3716de") |
|  | * }, |
|  | * { |
|  | * \_id: ObjectId("617e5eb0c4057dd5ea3716e9"), |
|  | * username: 'ScumbagSteve', |
|  | * comment: 'Denied your PR cause I found a hack', |
|  | * post: ObjectId("617e58c5c4057dd5ea3716e0") |
|  | * } |
|  | * ] |
|  | * Find all comments belonging to the post “Reports a bug in your code” |
|  |  |
|  | * db.commit |

…………………………………….-------------------------------------------,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,

----------------------------------------------------

**Assignment no 2**

|  |
| --- |
|  |
|  | * Atlanta Population: |
|  | * Use db.zipcodes.fimd() to filter results to only the results where city is ATLANTA and state is GA. |
|  | * db.zipcodes.find({city:"ATLANTA",state:"GA"}) |
|  | * Use db.zipcodes.aggregate with $match to do the same as above. |
|  | * db.zipcodes.aggregate([{$match:{city:"ATLANTA",state:"GA"}}]) |
|  | * Use $group to count the number of zip codes in Atlanta. |
|  | * db.zipcodes.aggregate([{$match:{city:"ATLANTA"}},{$group:{\_id:"$city",count\_field:{$sum:1}}}]) |
|  | * [ { \_id: 'ATLANTA', count\_field: 41 } ] |
|  | * Use $group to find the total population in Atlanta. |
|  | * db.zipcodes.aggregate([{$match:{city:"ATLANTA"}},{$group:{\_id:"$city",total\_pop:{$sum:"$pop"}}}]) |
|  | * [ { \_id: 'ATLANTA', total\_pop: 630046 } ] |
|  |  |
|  | * Populations By State: |
|  | * Use aggregate to calculate the total population for each state. |
|  | * db.zipcodes.aggregate([{$group:{\_id:'$state',total\_pop:{$sum:"$pop"}}}]) |
|  | * [ |
|  | * { \_id: 'VT', totalpop: 562758 }, |
|  | * { \_id: 'DC', totalpop: 606900 }, |
|  | * { \_id: 'AL', totalpop: 4040587 }, |
|  | * { \_id: 'NC', totalpop: 6628637 }, |
|  | * { \_id: 'WI', totalpop: 4891769 }, |
|  | * { \_id: 'OK', totalpop: 3145585 }, |
|  | * { \_id: 'AZ', totalpop: 3665228 }, |
|  | * { \_id: 'NV', totalpop: 1201833 }, |
|  | * { \_id: 'HI', totalpop: 1108229 }, |
|  | * { \_id: 'UT', totalpop: 1722850 }, |
|  | * { \_id: 'TN', totalpop: 4876457 }, |
|  | * { \_id: 'DE', totalpop: 666168 }, |
|  | * { \_id: 'IN', totalpop: 5544136 }, |
|  | * { \_id: 'SD', totalpop: 695397 }, |
|  | * { \_id: 'MT', totalpop: 798948 }, |
|  | * { \_id: 'NJ', totalpop: 7730188 }, |
|  | * { \_id: 'KS', totalpop: 2475285 }, |
|  | * { \_id: 'WA', totalpop: 4866692 }, |
|  | * { \_id: 'MO', totalpop: 5110648 }, |
|  | * { \_id: 'WY', totalpop: 453528 } |
|  | * ] |
|  | * Sort the results by population, highest first. |
|  | * db.zipcodes.aggregate([{$group:{\_id:"$state",total\_pop:{$sum:"$pop"}}},{$sort:{total\_pop:-1}}]) |
|  | * [ |
|  | * { \_id: 'CA', total\_pop: 29754890 }, |
|  | * { \_id: 'NY', total\_pop: 17990402 }, |
|  | * { \_id: 'TX', total\_pop: 16984601 }, |
|  | * { \_id: 'FL', total\_pop: 12686644 }, |
|  | * { \_id: 'PA', total\_pop: 11881643 }, |
|  | * { \_id: 'IL', total\_pop: 11427576 }, |
|  | * { \_id: 'OH', total\_pop: 10846517 }, |
|  | * { \_id: 'MI', total\_pop: 9295297 }, |
|  | * { \_id: 'NJ', total\_pop: 7730188 }, |
|  | * { \_id: 'NC', total\_pop: 6628637 }, |
|  | * { \_id: 'GA', total\_pop: 6478216 }, |
|  | * { \_id: 'VA', total\_pop: 6181479 }, |
|  | * { \_id: 'MA', total\_pop: 6016425 }, |
|  | * { \_id: 'IN', total\_pop: 5544136 }, |
|  | * { \_id: 'MO', total\_pop: 5110648 }, |
|  | * { \_id: 'WI', total\_pop: 4891769 }, |
|  | * { \_id: 'TN', total\_pop: 4876457 }, |
|  | * { \_id: 'WA', total\_pop: 4866692 }, |
|  | * { \_id: 'MD', total\_pop: 4781379 }, |
|  | * { \_id: 'MN', total\_pop: 4372982 } |
|  | * ] |
|  | * Limit the result to just the first 3 results. What are the top 3 states in population? |
|  | * db.zipcodes.aggregate([{$group:{\_id:"$state",total\_pop:{$sum:"$pop"}}},{$sort:{total\_pop:-1}},{$limit:3}]) |
|  | * [ |
|  | * { \_id: 'CA', total\_pop: 29754890 }, |
|  | * { \_id: 'NY', total\_pop: 17990402 }, |
|  | * { \_id: 'TX', total\_pop: 16984601 } |
|  | * ] |
|  | * Populations By City: |
|  | * Use aggregation to calculate total population for each city. |
|  | * db.zipcodes.aggregate([{$group:{\_id:"$city",total\_pop:{$sum:"$pop"}}}]) |
|  | * [ |
|  | * { \_id: 'LARTO', total\_pop: 5507 }, |
|  | * { \_id: 'HORNELL', total\_pop: 14311 }, |
|  | * { \_id: 'CALIFORNIA', total\_pop: 20914 }, |
|  | * { \_id: 'WELLS BRIDGE', total\_pop: 232 }, |
|  | * { \_id: 'HARROD', total\_pop: 4897 }, |
|  | * { \_id: 'CAPUTA', total\_pop: 578 }, |
|  | * { \_id: 'ALDER CREEK', total\_pop: 74 }, |
|  | * { \_id: 'WILCOX', total\_pop: 2390 }, |
|  | * { \_id: 'KEELING', total\_pop: 2651 }, |
|  | * { \_id: 'HARRELLSVILLE', total\_pop: 1046 }, |
|  | * { \_id: 'WARREN', total\_pop: 276817 }, |
|  | * { \_id: 'RECTOR', total\_pop: 3891 }, |
|  | * { \_id: 'HAVILAND', total\_pop: 1936 }, |
|  | * { \_id: 'MAZA', total\_pop: 1800 }, |
|  | * { \_id: 'EROS', total\_pop: 2296 }, |
|  | * { \_id: 'SPRING GLEN', total\_pop: 288 }, |
|  | * { \_id: 'BIG CABIN', total\_pop: 2634 }, |
|  | * { \_id: 'SAN SIMEON', total\_pop: 500 }, |
|  | * { \_id: 'ANTIOCH', total\_pop: 123107 }, |
|  | * { \_id: 'GENTRY', total\_pop: 5401 } |
|  | * ] |
|  | * Sort the results by population, highest first. |
|  | * db.zipcodes.aggregate([{$group:{\_id:"$city",total\_pop:{$sum:"$pop"}}},{$sort:{total\_pop:-1}}]) |
|  | * [ |
|  | * { \_id: 'CHICAGO', total\_pop: 2452177 }, |
|  | * { \_id: 'BROOKLYN', total\_pop: 2341387 }, |
|  | * { \_id: 'HOUSTON', total\_pop: 2123053 }, |
|  | * { \_id: 'LOS ANGELES', total\_pop: 2102295 }, |
|  | * { \_id: 'PHILADELPHIA', total\_pop: 1639862 }, |
|  | * { \_id: 'NEW YORK', total\_pop: 1476790 }, |
|  | * { \_id: 'BRONX', total\_pop: 1209548 }, |
|  | * { \_id: 'SAN DIEGO', total\_pop: 1054316 }, |
|  | * { \_id: 'DALLAS', total\_pop: 999042 }, |
|  | * { \_id: 'DETROIT', total\_pop: 967468 }, |
|  | * { \_id: 'PHOENIX', total\_pop: 902249 }, |
|  | * { \_id: 'MIAMI', total\_pop: 848436 }, |
|  | * { \_id: 'COLUMBUS', total\_pop: 825448 }, |
|  | * { \_id: 'SAN JOSE', total\_pop: 817497 }, |
|  | * { \_id: 'SAN ANTONIO', total\_pop: 813188 }, |
|  | * { \_id: 'WASHINGTON', total\_pop: 780954 }, |
|  | * { \_id: 'BALTIMORE', total\_pop: 738846 }, |
|  | * { \_id: 'JACKSONVILLE', total\_pop: 735505 }, |
|  | * { \_id: 'SAN FRANCISCO', total\_pop: 723993 }, |
|  | * { \_id : 'CLEVELAND', total\_pop: 687451 } |
|  | * ] |
|  | * Limit the results to just the first 3 results. What are the top 3 cities in population? |
|  | * db.zipcodes.aggregate([{$group:{\_id:"$city",total\_pop:{$sum:"$pop"}}},{$sort:{total\_pop:-1}},{$limit:3}]) |
|  | * [ |
|  | * { \_id: 'CHICAGO', total\_pop: 2452177 }, |
|  | * { \_id: 'BROOKLYN', total\_pop: 2341387 }, |
|  | * { \_id: 'HOUSTON', total\_pop: 2123053 } |
|  | * ] |
|  | * What are the top 3 cities in population in Texas? |
|  | * db.zipcodes.aggregate([{$match:{state:"TX"}},{$group:{\_id:"$city",total\_pop:{$sum:"$pop"}}},{$sort:{total\_pop:-1}},{$limit:3}]) |
|  | * [ |
|  | * { \_id: 'HOUSTON', total\_pop: 2095918 }, |
|  | * { \_id: 'DALLAS', total\_pop: 940191 }, |
|  | * { \_id: 'SAN ANTONIO', total\_pop: 811792 } |
|  | * ] |

**Assignment no3**

|  |
| --- |
|  |
|  |
|  | * Mongo DB Exercise – With the Restaurants Data Set |
|  | * Write a MongoDb query to display all the documents in the collection restaurants. |
|  | * db.addresses.find() |
|  | * Write a MongoDb query to display the fields restaurants\_id, name, borough and cuisine for all the documents in the collection restaurant. |
|  | * db.addresses.find({},{"restaurant\_id":1,"name":1,"borough":1,"cuisine":1}) |
|  | * Write a MongoDb query to display the fields restaurants\_id, name, borough and cuisine, but exclude the field \_id for all the documents in the collection restaurant. |
|  | * db.addresses.find({},{"restaurant\_id":1,"name":1,"borough":1,"cuisine":1,"\_id":0}) |
|  | * Write a MongoDb query to display the fields restaurants\_id, name, borough and zip code, but exclude the field\_id for all the documents in the collection restaurant. |
|  | * db.addresses.find({},{"restaurant\_id" : 1,"name":1,"borough":1,"address.zipcode" :1,"\_id":0}); |
|  | * Write a MongoDb query to display the first 5 restaurants which is in the borough Bronx. |
|  | * db.addresses.find({"borough":"Bronx"}).limit(5) |
|  | * Write a MongoDb query to display all the restaurant which is in the borough Bronx. |
|  | * db.addresses.find({"borough":"Bronx"}) |
|  | * Write a MongoDb query to display the next 5 restaurants after skipping first 5 which are in the borough Bronx. |
|  | * db.addresses.find({"borough":"Bronx"}).skip(5).limit(5) |
|  | * Write a MongoDb query to find the restaurants who achieved a score more than 90. |
|  | * db.addresses.find({"grades.score":{$gt:90}}) |
|  | * Write a MongoDb query to find the restaurants that achieved a score, more than 80 but less than 100. |
|  |  |
|  | * Write a MongoDb query to find the restaurants which locate in latitude value less than -95.754168. |
|  | * db.addresses.find({"address.coord":{$lt:-95.754168}}) |
|  | * Write a MongoDB query to find the restaurants that do not prepare any cuisine of 'American' and their grade score more than 70 and latitude less than -65.754168. |
|  | * db.addresses.find({$and:[{"cuisine":{$ne:'American'}},{"grades.score":{$gt:70}},{"address.coord":{$lt:-65.754168}}]}) |
|  | * Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American' and achieved a score more than 70 and located in the longitude less than -65.754168. |
|  |  |
|  | * Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American ' and achieved a grade point 'A' not belongs to the borough Brooklyn. The document must be displayed according to the cuisine in descending order. |
|  | * db.addresses.find({"cuisine":{$ne:"American"},"grades.grade":'A',"borough":{$ne:"Brooklyn"}}).sort({"cuisine":-1}) |
|  | * Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Wil' as first three letters for its name. |
|  | * db.addresses.find({"name":/^Wil/},{"restaurant\_id":1,"name":1,"borough":1,"cuisine":1}) |
|  | * db.addresses.find({"name":/ces$/},{"restaurant\_id":1,"name":1,"borou gh":1,"cuisine":1}) |
|  | * db.addresses.find({"name":/.\*Reg.\*/},{"restaurant\_id":1,"name":1,"borough":1,"cuisine":1}) |
|  | * db.addresses.find({"borough":"Bronx",$or:[{"cuisine":"American"},{"cuisine":"Chinese"}]}) |
|  | * db .addresses.find({"borough":{$in:["Staten I Island","Queens","Bronx","Brooklyn"]}},{"restaurant\_id":1,"name":1,"borough":1,"cuisine":1}) |
|  | * db.addresses.find({"borough":{$nin:["Staten Island","Queens","Bronx","Brooklyn"]}},{"restaurant\_id":1,"name":1,"borough":1,"cuisine":1}) |
|  | * db.addresses.find({"grades.score":{$not:{$gt:10}}},{"restaurant\_id":1,"name":1,"cuisine":1,"borough":1}) |
|  | * db.addresses.find({$or:[{"name":/^Wil/},{$and:[{"cusine":{$ne:"American"}},{"cusine":{$ne:"Chinese"}}]}]},{"restaurant\_id":1,"name":1,"cuisine":1,"borough":1}) |
|  | * db.addresses.find({"grades.grade":'A',"grades.score":11, "grades.date":ISODate("2014-08- 11T00:00:00Z")},{"restaurant\_id":1,"name":1,"grades":1}) |
|  |  |
|  | db.addresses.find({"address.coord.1":{$gt:42,$lt:52}},{"restaurant\_id":1,"name":1,"address":1,"coord":1}) |
|  | * db.addresses.find().sort({"name":1}) |
|  | * db.addresses.find().sort({"name":-1}) |
|  | * db.addresses.find().sort({"cuisine":1,"borough":-1}) |
|  | * db.addresses.find({"address.street":{$exists:true}}) |