Cloud Computing and Big Data

LIS 4102 SECTION 1

Dr. Genoveva Vargas-Solar

Assignment 1: Hands-On MongoDB Querying

Melina Escobedo Zárate

ID 164094

February 13th ‘22

Index

3.1 Querying the Database 3
3.2 Updating the Database 6
3.3 Complex querying (aggregation) 7
3.4 References between collections and join 8
3.4 Indexing 10

3.1 Querying the Database

db.Restaurants.find ({ “name": { $ne:”" } }, { "name": 1 }).sort({ "name": 1 })

```
> db.Restaurants.find({"name":{$ne:""}},{"name": 1}).sort({"name": 1})
< { _id: ObjectId("61f4c27e84ead6511f7b2d84"),
    name: '#1 Garden Chinese' }
  { _id: ObjectId("61f4c28384ead6511f7b55d8"),
    name: '#1 Me. Nick\'S' }
  { _id: ObjectId("61f4c27b84ead6511f7b1b60"),
    name: '#1 Sabor Latino Restaurant' }
  { _id: ObjectId("61f4c28084ead6511f7b3b9d"),
    name: '$1.25 Pizza' }
  { _id: ObjectId("61f4c27f84ead6511f7b33be"),
    name: '\'\'U\'\' Like Chinese Restaurant' }
  { _id: ObjectId("61f4c28384ead6511f7b5c3d"),
    name: '\'\'W\'\' Cafe' }
  { _id: ObjectId("61f4c27d84ead6511f7b28fb"),
    name: '\'Wichcraft' }
  { _id: ObjectId("61f4c28184ead6511f7b4191"),
    name: '\'Wichcraft' }
  { _id: ObjectId("61f4c27884ead6511f7b06a5"),
    name: '(Lewis Drug Store) Locanda Vini E Olii' }
  { _id: ObjectId("61f4c28284ead6511f7b4f31"),
    name: '(Library)  Four & Twenty Blackbirds' }
  { _id: ObjectId("61f4c27c84ead6511f7b2125"),
    name: '(Public Fare) 81St Street And Central Park West (Delacorte Theatre)' }
  { _id: ObjectId("61f4c27b84ead6511f7b1c3f"),
    name: '/ L\'Ecole' }
  { _id: ObjectId("61f4c28384ead6511f7b5916"),
    name: '002 Mercury Tacos Llc' }
  { _id: ObjectId("61f4c27b84ead6511f7b1a9a"),
    name: '1 2 3 Burger Shot Beer' }
  { _id: ObjectId("61f4c27b84ead6511f7b1a43"),
    name: '1 Banana Queen' }
  { _id: ObjectId("61f4c27f84ead6511f7b35fb"),
    name: '1 Buen Sabor' }
  { _id: ObjectId("61f4c27884ead6511f7b0d98"), name: '1 Darbar' }
  { _id: ObjectId("61f4c27584ead6511f7afa22"),
    name: '1 East 66Th Street Kitchen' }
  { _id: ObjectId("61f4c27a84ead6511f7b1908"), name: '1 Oak' }
  { _id: ObjectId("61f4c27c84ead6511f7b2419"), name: '1 Or 8' }
```

db.Restaurants.find ({ “cuisine”: ”Italian” }, { “name”: 1, “address.zipcode”: 1, “address.coordinates”: 1 }.sort({ “address.zipcode”: 1, “name”: -1 }))

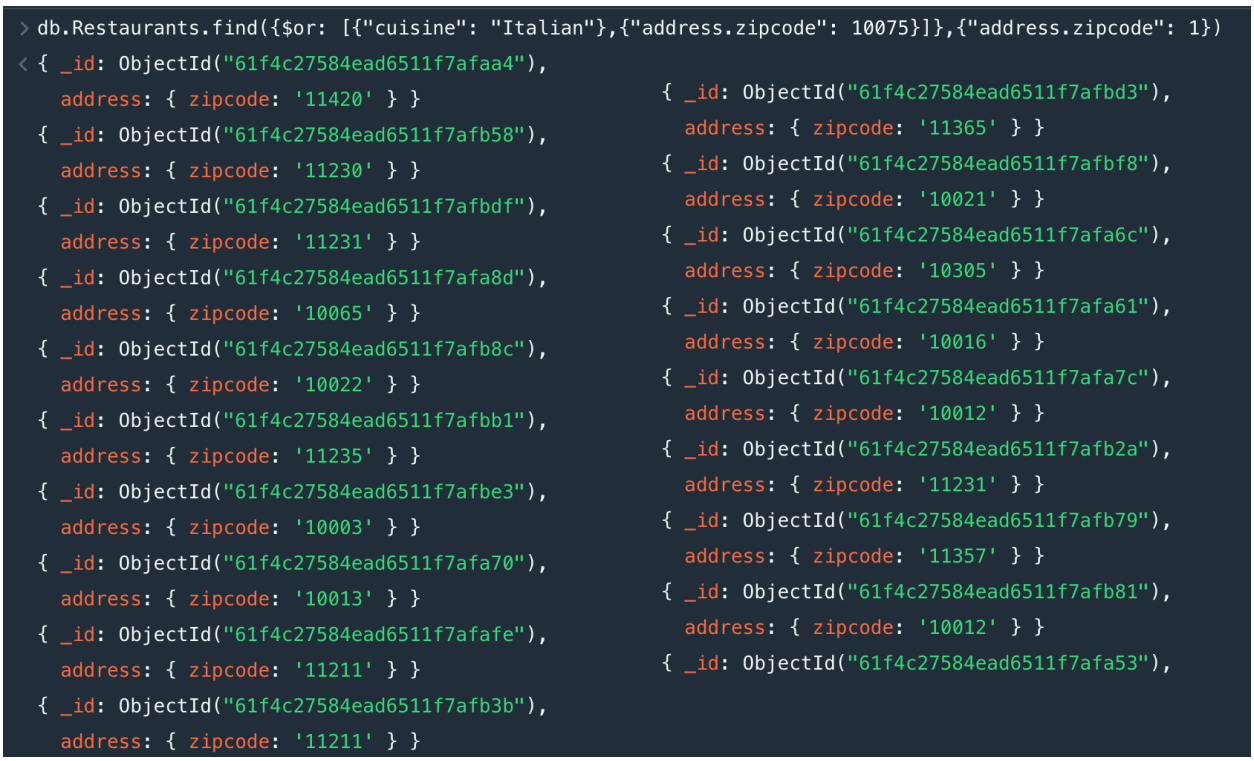
```
> db.Restaurants.find({"cuisine": "Italian"}, {"name": 1, "address.zipcode": 1, "address.coordinates": 1}).sort({"address.zipcode": 1, "name": -1})
< { _id: ObjectId("61f4c27984ead6511f7b0f6a"),
    address: { zipcode: '10001' },
    name: 'Tre Dici' }
  { _id: ObjectId("61f4c28084ead6511f7b403a"),
    address: { zipcode: '10001' },
    name: 'Stella 34' }
  { _id: ObjectId("61f4c27d84ead6511f7b29e8"),
    address: { zipcode: '10001' },
    name: 'Spinelli\'S Pizza/Gyro Ii' }
  { _id: ObjectId("61f4c27584ead6511f7afc70"),
    address: { zipcode: '10001' },
    name: 'Salumeria Beillese/ Biricchino Rest' }
  { _id: ObjectId("61f4c27784ead6511f7b03f2"),
    address: { zipcode: '10001' },
    name: 'Pepe Giallo' }
  { _id: ObjectId("61f4c28084ead6511f7b3cf5"),
    address: { zipcode: '10001' },
    name: 'Fb8020 Pizza Concession' }
  { _id: ObjectId("61f4c27e84ead6511f7b2d4f"),
    address: { zipcode: '10001' },
    name: 'Famous Famiglia' }
  { _id: ObjectId("61f4c27c84ead6511f7b1e12"),
    address: { zipcode: '10001' },
    name: 'Company' }
  { _id: ObjectId("61f4c27784ead6511f7b02e9"),
    address: { zipcode: '10001' },
    name: 'Bottino' }
  { _id: ObjectId("61f4c27f84ead6511f7b3576"),
    address: { zipcode: '10002' },
    name: 'Via Tribunali' }
  { _id: ObjectId("61f4c27a84ead6511f7b1566"),
    address: { zipcode: '10002' },
    name: 'Tre' }
  { _id: ObjectId("61f4c27d84ead6511f7b251a"),
    address: { zipcode: '10002' },
    name: 'The Meatball Shop' }
  { _id: ObjectId("61f4c27e84ead6511f7b3075"),
```

db.Restaurants.find ({ $and: [{ "address.zipcode": “10075" }, { "telephoneNumber": { $exists : true } }] }, { "name": 1, "address.zipcode": 1, "telephoneNumber" : 1 })

```
> db.Restaurants.find({$and: [{"address.zipcode" : "10075"}, {"telephoneNumber" : {$exists : true}}]}, {"name" : 1, "address.zipcode" : 1, "telephoneNumber" : 1})
< { _id: ObjectId("61f4c27584ead6511f7afc9b"),
    address: { zipcode: '10075' },
    name: 'Due',
    telephoneNumber: '24680356' }
  { _id: ObjectId("61f4c28284ead6511f7b4d1b"),
    address: { zipcode: '10075' },
    name: 'Spigolo',
    telephoneNumber: '67895432' }
```

db.Restaurants.find ({ "grades.score": { $gte: 50 } }, { "name": 1, "grades.score": 1 })

db.Restaurants.find ({ $or: [{ "cuisine": “Italian" }, { "address.zipcode": 10075 }] }, { "address.zipcode": 1 })



db.Restaurants.find ({ $and: [{ $and: [{$or: [{ "address.zipcode": ‘10075' }, { "address.zipcode": ‘10098' }] }, { $or: [{ cuisine: ‘Italian' }, { cuisine: ‘American' }] }] }, { "grades.score" : { $gte : 50 } }] }, { _id: 0, name: 1, cuisine: 1, "address.zipcode": 1, "grades.score": 1 })

```
> db.Restaurants.find({$and: [{$or: [{"address.zipcode": "10075"}, {"address.zipcode": "10098"}]},
  {$or: [{"cuisine": "Italian"},{"cuisine": "American"}]},{"grades.score": {$gte: 50}}]},
  {"name": 1, "cuisine": 1, "address.zipcode": 1})
< { _id: ObjectId("61f4c27984ead6511f7b0f5b"),
    address: { zipcode: '10075' },
    cuisine: 'American',
    name: 'Three Star Diner' }
Atlas atlas-xiwb9m-shard-0 [primary] MongoDB>
```

db.Restaurants.find ({ $or: [{ "grades.grade": “C" }, { "grades.grade": “P" }, { "grades.grade": “Q" }] }, { "name": 1, "cuisine": 1, "address.zipcode": 1 })

```
> db.Restaurants.find({$or: [{"grades.grade": "C"},{"grades.grade": "P"},{"grades.grade": "Q"}]},{"name": 1, "cuisine": 1, "address.zipcode": 1})
< { _id: ObjectId("61f4c27584ead6511f7afaa4"),
    address: { zipcode: '11420' },
    cuisine: 'Italian',
    name: 'Don Peppe' }
  { _id: ObjectId("61f4c27584ead6511f7afb58"),
    address: { zipcode: '11230' },
    cuisine: 'Italian',
    name: 'Mama Lucia' }
  { _id: ObjectId("61f4c27584ead6511f7afa48"),
    address: { zipcode: '11371' },
    cuisine: 'American',
    name: 'Terminal Cafe/Yankee Clipper' }
  { _id: ObjectId("61f4c27584ead6511f7afa87"),
    address: { zipcode: '11366' },
    cuisine: 'Chinese',
    name: 'King Yum Restaurant' }
  { _id: ObjectId("61f4c27584ead6511f7afaa8"),
    address: { zipcode: '10065' },
    cuisine: 'American',
    name: 'Dangerfield\'S Night Club' }
  { _id: ObjectId("61f4c27584ead6511f7afab1"),
    address: { zipcode: '10006' },
    cuisine: 'Irish',
    name: 'Blarney Stone' }
  { _id: ObjectId("61f4c27584ead6511f7afb22"),
    address: { zipcode: '10462' },
    cuisine: 'American',
    name: 'Bronx Grill' }
  { _id: ObjectId("61f4c27584ead6511f7afb53"),
    address: { zipcode: '11004' },
    cuisine: 'Hamburgers',
    name: 'Burger King' }
  { _id: ObjectId("61f4c27584ead6511f7afb72"),
    address: { zipcode: '11106' },
    cuisine: 'Greek',
    name: 'Omonia Cafe' }
  { _id: ObjectId("61f4c27584ead6511f7afb7a"),
```

3.2 Updating the Database

db.Restaurants.updateOne ({ "name": “Juni” }, { $set: {"cuisine": "American (New)” }, $currentDate: { "lastModified": true } })

```
> db.Restaurants.updateOne({"name": "Juni"},{$set: {"cuisine": "American (New)"},$currentDate: {"lastModified": true}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
> db.restaurants.find( { "cuisine": "American (New)" } )
<
> db.Restaurants.find( { "cuisine": "American (New)" } )
< { _id: ObjectId("61f4c27984ead6511f7b119e"),
    address:
     { building: '12',
       coord: [ -73.9852329, 40.745971 ],
       street: 'East 31 Street',
       zipcode: '10016' },
    cuisine: 'American (New)',
    grades:
     [ { date: 2014-09-19T00:00:00.000Z, grade: 'A', score: 12 },
       { date: 2013-08-05T00:00:00.000Z, grade: 'A', score: 5 },
       { date: 2012-06-07T00:00:00.000Z, grade: 'A', score: 0 } ],
    name: 'Juni',
    restaurant_id: '41156888',
    lastModified: 2022-02-05T20:59:40.012Z }
```

db.Restaurants.update ({ "restaurant_id": “41156888" }, { $set: { "address.street": "East 31st Street” } })

```
> db.Restaurants.update({"restaurant_id": "41156888"},{$set: {"address.street": "East 31st Street"}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
```

db.Restaurants.update ({ "address.zipcode": "10016", cuisine: “Other" }, { $set: { cuisine: "Category to be determined” }, $currentDate: { "lastModified": true } }, { multi: true })

```
> db.Restaurants.update({"address.zipcode": "10016", cuisine: "Other"},{$set: {cuisine: "Category to be determined"},$currentDate: {"lastModified": true }},{multi: true})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 20,
    modifiedCount: 20,
    upsertedCount: 0 }
```

db.Restaurants.replaceOne/updateOne ({ “restaurant_id": “41154403" }, { "name": "Vella 2","address": {"city": "1480","street": "2 Avenue","zipcode": “10075" } })

```
> db.Restaurants.replaceOne({"restaurant_id": "41154403"},{"name": "Vella 2","address": {"city": "1480","street": "2 Avenue","zipcode": "10075"}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
```

3.3 Complex querying (aggregation)

db.Restaurant.aggregate ([{ $group: { _id: “$cuisine", restaurants: { $sum: 1 } } }, { $sort: { restaurants: -1 } }])

```
> db.Restaurant.aggregate([{$group: {_id: "$cuisine",restaurants: {$sum: 1}}},{$sort: {restaurants: -1}}])
<
Atlas atlas-xiwb9m-shard-0 [primary] MongoDB>
```

db.Restaurants.aggregate ([{ $match: { "cuisine": “Italian" } }, { $group: { _id: “$address.zipcode", cuisines: { $sum : 1 } } }, { $sort: { cuisines: -1 } }])

```
> db.Restaurants.aggregate([{$match: {"cuisine" : "Italian"}},{$group: {_id: "$address.zipcode",cuisines: { $sum : 1}}},{$sort: {cuisines: -1}}])
< { _id: '10013', cuisines: 51 }
  { _id: '10012', cuisines: 45 }
  { _id: '10014', cuisines: 43 }
  { _id: '10019', cuisines: 42 }
  { _id: '10003', cuisines: 40 }
  { _id: '10011', cuisines: 37 }
  { _id: '10036', cuisines: 35 }
  { _id: '10022', cuisines: 32 }
  { _id: '10028', cuisines: 22 }
  { _id: '11215', cuisines: 22 }
  { _id: '10016', cuisines: 21 }
  { _id: '10017', cuisines: 20 }
  { _id: '10021', cuisines: 20 }
  { _id: '10024', cuisines: 19 }
  { _id: '11201', cuisines: 19 }
  { _id: '10023', cuisines: 18 }
  { _id: '10010', cuisines: 17 }
  { _id: '10065', cuisines: 17 }
  { _id: '10458', cuisines: 17 }
  { _id: '10009', cuisines: 17 }
  Type "it" for more
Atlas atlas-xiwb9m-shard-0 [primary] MongoDB>
```

db.Restaurants.aggregate([{$match: {$and: [{cuisine : 'Italian'},{restaurant_id: {$gte: '41205309'}}, {averagePrice: {$exists: true}}]},},{$group: {_id : null,averagePrices:{$avg: "$averagePrice"},averageZips: {$avg:{'$toInt': '$address.zipcode'}}}},])

```
> db.Restaurants.aggregate([{$match: {$and: [{cuisine : 'Italian'},{restaurant_id: {$gte: '41205309'}},{averagePrice: {$exists: true}}
                          ]},},{$group: {_id : null,averagePrices:{$avg: "$averagePrice"},averageZips:{$avg:{'$toInt': '$address.zipcode'}}}},])
< { _id: null, averagePrices: 25, averageZips: 10023.5 }
Atlas atlas-xiwb9m-shard-0 [primary] MongoDB>
```

3.4 References between collections and join

db.createCollection (“comments")

```
> db.createCollection("comments")
< { ok: 1 }
Atlas atlas-xiwb9m-shard-0 [primary] MongoDB>
```

db.comments.insertMany ([{ restaurant_id: "30075445", client_id: "124781426", comment: "Excellent service!", date: new Date (“2014-12-03”), type: “positive" }, { restaurant_id: "30112340", client_id: "475732110", comment: "The best food I ever had!", date: new Date (“2015-01-21"), type: “positive" },{ restaurant_id: "30191841", client_id: "626141589", comment: "I hated it, never returning.", date: new Date (“2020-05-13"), type: “negative" }])

```
> db.comments.insertMany([
    {restaurant_id: "30075445", client_id: "124781426", comment: "Excellent service!", date: new Date("2014-12-03"), type: "positive"},
    {restaurant_id: "30112340", client_id: "475732110", comment: "The best food I ever had!", date: new Date("2015-01-21"), type: "positive"},
    {restaurant_id: "30191841", client_id: "626141589", comment: "I hated it, never returning.", date: new Date("2020-05-13"), type: "negative"}
])
< { acknowledged: true,
  insertedIds:
   { '0': ObjectId("6200aa1c6e54af2ec1ddd0b7"),
     '1': ObjectId("6200aa1c6e54af2ec1ddd0b8"),
     '2': ObjectId("6200aa1c6e54af2ec1ddd0b9") } }
Atlas atlas-xiwb9m-shard-0 [primary] MongoDB>
```

db.Restaurants.aggregate ([{ $lookup: { from: “comments", localField: “restaurant_id", foreignField: “restaurant_id", as: “restaurant_comments" } }, { $match: { "restaurant_comments": { $ne: [ ] } } }, { $project: { restaurant_comments: { comment: 1 }, name: 1, cuisine: 1, restaurant_id: 1, address: 1} },])

```
> db.Restaurants.aggregate([{$lookup:{from: "comments",localField: "restaurant_id",foreignField: "restaurant_id",as: "restaurant_comments"}},
        {$match: {"restaurant_comments": {$ne: []}}},{$project: { restaurant_comments:{comment: 1}, name: 1, cuisine: 1, restaurant_id: 1, address: 1}},])
< { _id: ObjectId("61f4c27584ead6511f7afa15"),
  address:
   { building: '1007',
     coord: [ -73.856077, 40.848447 ],
     street: 'Morris Park Ave',
     zipcode: '10462' },
  cuisine: 'Bakery',
  name: 'Morris Park Bake Shop',
  restaurant_id: '30075445',
  restaurant_comments:
   [ { comment: 'Excellent service!' },
     { comment: 'Excellent service!' } ] }
{ _id: ObjectId("61f4c27584ead6511f7afa16"),
  address:
   { building: '469',
     coord: [ -73.961704, 40.662942 ],
     street: 'Flatbush Avenue',
     zipcode: '11225' },
  cuisine: 'Hamburgers',
  name: 'Wendy\'S',
  restaurant_id: '30112340',
  restaurant_comments:
   [ { comment: 'I hated it, never returning.' },
     { comment: 'I hated it, never returning.' } ] }
{ _id: ObjectId("61f4c27584ead6511f7afa17"),
  address:
   { building: '351',
     coord: [ -73.98513559999999, 40.7676919 ],
     street: 'West   57 Street',
     zipcode: '10019' },
  cuisine: 'Irish',
  name: 'Dj Reynolds Pub And Restaurant',
  restaurant_id: '30191841',
  restaurant_comments:
   [ { comment: 'The best food I ever had!' },
     { comment: 'The best food I ever had!' } ] }
```

db.comments.insertMany ([{ "restaurant_id": "40356649", "client_id": "00001", "comment": "", "date": ISODate("2022-01-30T01:00:00-06:00"), "type": “positive” }, { "restaurant_id": "40359480", "client_id": "00002", "comment": "", "date": ISODate(“2022-01-31T01:00:00-06:00"), "type": “negative” }, { "restaurant_id" : "40360076", "client_id" : "00003", "comment": "", "date": ISODate (“2022-02-01T01:00:00-06:00"), "type": “positive"}, { "restaurant_id" : "40358429", "client_id": "00004", "comment": "", "date": ISODate (“2022-02-02T01:00:00-06:00"), "type": “positive” }, { "restaurant_id" : "40360045", "client_id": "00005", "comment": "", "date": ISODate (“2022-02-03T01:00:00-06:00"), "type": “negative” },])

```
> db.comments.insertMany( [
    { "restaurant_id" : "40356649", "client_id" : "00001", "comment": "", "date": ISODate("2022-01-30T01:00:00-06:00"), "type": "positive"},
    { "restaurant_id" : "40359480", "client_id" : "00002", "comment": "", "date": ISODate("2022-01-31T01:00:00-06:00"), "type": "negative"},
    { "restaurant_id" : "40360076", "client_id" : "00003", "comment": "", "date": ISODate("2022-02-01T01:00:00-06:00"), "type": "positive"},
    { "restaurant_id" : "40358429", "client_id" : "00004", "comment": "", "date": ISODate("2022-02-02T01:00:00-06:00"), "type": "positive"},
    { "restaurant_id" : "40360045", "client_id" : "00005", "comment": "", "date": ISODate("2022-02-03T01:00:00-06:00"), "type": "negative"},
  ] )
< { acknowledged: true,
    insertedIds:
     { '0': ObjectId("62015881236c5b1c04a084b6"),
       '1': ObjectId("62015881236c5b1c04a084b7"),
       '2': ObjectId("62015881236c5b1c04a084b8"),
       '3': ObjectId("62015881236c5b1c04a084b9"),
       '4': ObjectId("62015881236c5b1c04a084ba") } }
Atlas atlas-xiwb9m-shard-0 [primary] MongoDB>
```

db.Restaurants.aggregate ([{ $lookup:{from: "comments",localField: “restaurant_id",foreignField: "restaurant_id",as: “restaurants_n_comments" } }, { $unwind: "$restaurants_n_comments" }, { $match: { “restaurants_n_comments.comment": { $exists: true, $ne: “" } } }, { $project: { "name": 1,"comments": “$restaurants_n_comments.comment" } }])

```
> db.Restaurants.aggregate([{$lookup:{from: "comments",localField: "restaurant_id",foreignField: "restaurant_id",as: "restaurants_n_comments"}},
  {$unwind: "$restaurants_n_comments" },{$match: {"restaurants_n_comments.comment": { $exists: true, $ne: ""}}},{$project:
  {"name": 1,"comments": "$restaurants_n_comments.comment"}}])
< { _id: ObjectId("61f4c27584ead6511f7afa15"),
    name: 'Morris Park Bake Shop',
    comments: 'Excellent service!' }
  { _id: ObjectId("61f4c27584ead6511f7afa15"),
    name: 'Morris Park Bake Shop',
    comments: 'Excellent service!' }
  { _id: ObjectId("61f4c27584ead6511f7afa16"),
    name: 'Wendy\'S',
    comments: 'The best food I ever had!' }
  { _id: ObjectId("61f4c27584ead6511f7afa16"),
    name: 'Wendy\'S',
    comments: 'The best food I ever had!' }
  { _id: ObjectId("61f4c27584ead6511f7afa17"),
    name: 'Dj Reynolds Pub And Restaurant',
    comments: 'I hated it, never returning.' }
  { _id: ObjectId("61f4c27584ead6511f7afa17"),
    name: 'Dj Reynolds Pub And Restaurant',
    comments: 'I hated it, never returning.' }
```

3.4 Indexing

db.Restaurants.createIndex({"cuisine": 1})

```
> db.Restaurants.createIndex({"cuisine": 1})
< 'cuisine_1'
Atlas atlas-xiwb9m-shard-0 [primary] MongoDB>
```

db.Restaurants.createIndex ({ "cuisine": 1, "address.zipcode": -1 })

```
> db.Restaurants.createIndex({"cuisine": 1, "address.zipcode": -1})
< 'cuisine_1_address.zipcode_-1'
Atlas atlas-xiwb9m-shard-0 [primary] MongoDB>
```

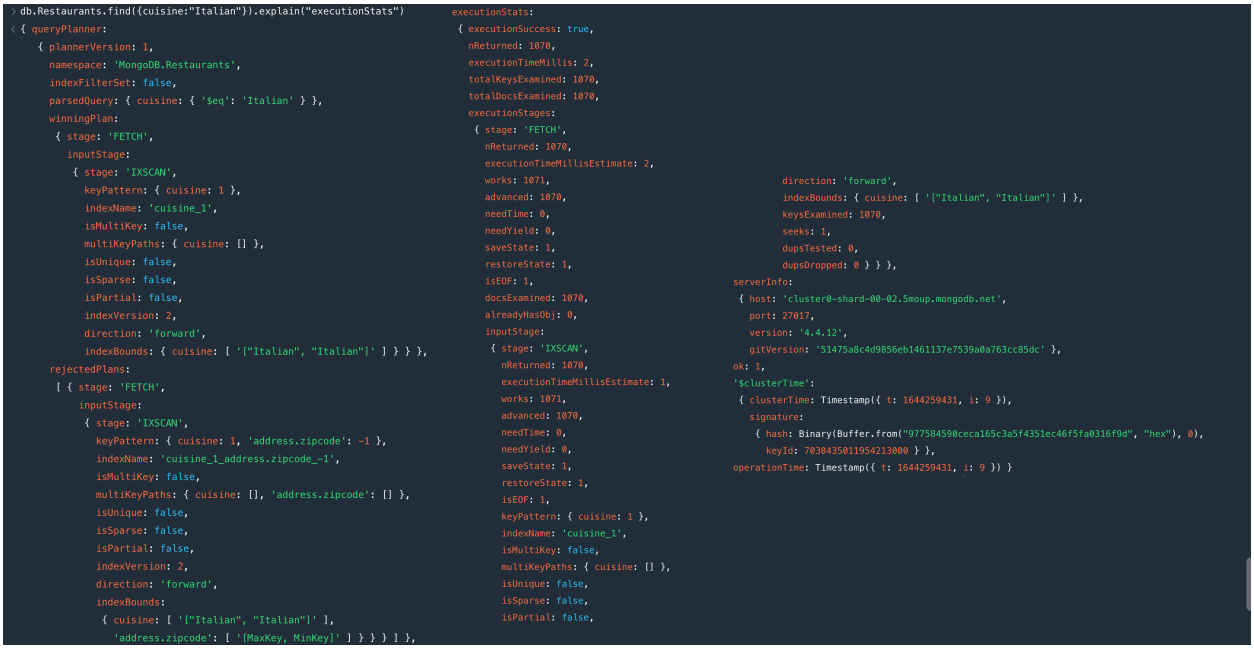
db.Restaurants.getIndexes ( )

```
> db.Restaurants.getIndexes()
< [
    { v: 2, key: { _id: 1 }, name: '_id_' },
    { v: 2, key: { cuisine: 1 }, name: 'cuisine_1' },
    {
      v: 2,
      key: { cuisine: 1, 'address.zipcode': -1 },
      name: 'cuisine_1_address.zipcode_-1'
    }
  ]
Atlas atlas-xiwb9m-shard-0 [primary] MongoDB>
```

db.Restaurants.find ({ cuisine: "Italian" }).explain( )

What information is provided by the system?
With the **explain( )** method MongoDB execute a query plan information, this means that it returns a document with the query plan and, optionally, the execution statistics. The amount of information would be affected by the *verbosity* mode (by default is the queryPlanner verbosity mode)db.

Restaurants.find ({ cuisine: "Italian" }).explain("executionStats")



With **explain(“executionStats”)** the statistics are contained, they describe the completed query execution (refers to the modifications that would be performed, but they doesn't apply) for the winning plan. However, this method doesn't provide query execution information for the rejected plans.

db.Restaurants.dropIndexes/dropIndex ("cuisine_1", “cuisine_1_address.zipcode_-1")

```
> db.Restaurants.dropIndex("cuisine_1")
< {
    nIndexesWas: 3,
    ok: 1,
    '$clusterTime': {
      clusterTime: Timestamp({ t: 1644259970, i: 12 }),
      signature: {
        hash: Binary(Buffer.from("c504e7082fdabdcdeb61863d2fb98fcc69379f6b", "hex"), 0),
        keyId: Long("7030435011954212874")
      }
    },
    operationTime: Timestamp({ t: 1644259970, i: 12 })
  }
```

```
> db.Restaurants.dropIndex("cuisine_1_address.zipcode_-1")
< {
    nIndexesWas: 2,
    ok: 1,
    '$clusterTime': {
      clusterTime: Timestamp({ t: 1644260851, i: 1 }),
      signature: {
        hash: Binary(Buffer.from("304fa31961bc888c0483a1cf402507da9bf0f815", "hex"), 0),
        keyId: Long("7030435011954212874")
      }
    },
    operationTime: Timestamp({ t: 1644260851, i: 1 })
  }
```

Without the Indexes, the information provide by the system has some changes, for example the rejectedPlans are not there any more, as well as the Indexes that where drop before.

```
> db.Restaurants.find({cuisine: "Italian"}).explain()
< { queryPlanner:
    { plannerVersion: 1,
      namespace: 'MongoDB.Restaurants',
      indexFilterSet: false,
      parsedQuery: { cuisine: { '$eq': 'Italian' } },
      winningPlan:
       { stage: 'COLLSCAN',
         filter: { cuisine: { '$eq': 'Italian' } },
         direction: 'forward' },
      rejectedPlans: [] },
   serverInfo:
    { host: 'cluster0-shard-00-02.5moup.mongodb.net',
      port: 27017,
      version: '4.4.12',
      gitVersion: '51475a8c4d9856eb1461137e7539a0a763cc85dc' },
   ok: 1,
   '$clusterTime':
    { clusterTime: Timestamp({ t: 1644260925, i: 13 }),
      signature:
       { hash: Binary(Buffer.from("489185c5bec79de1a599cbd2d8c0fb100e2de2a1", "hex"), 0),
         keyId: 7030435011954213000 } },
   operationTime: Timestamp({ t: 1644260925, i: 13 }) }
```