Primi comandi:

show dbs
use restaurant (prova)
show collections

Codice Lettura:

```
db.restaurants.find()  /* digitare it per andare avanti */
db.restaurants.find({name:”Glorious Food”})

db.restaurants.find().sort( { restaurant_id: 1 } ).limit(1).pretty()


/* Count restaurants */
db.restaurants.count()   /*  25359   */

/* Count restaurants with American (Vegetarian, Chinese) cuisine
*/
db.restaurants.find({cuisine:”American”}).count()   /* 6183-2418-
102   */

/* How many different types of cooking  */
db.restaurants.distinct("cuisine").length


/* Count no Vegetarian restaurants  */
db.restaurants.find({cuisine:{$ne:"Vegetarian"}}).count()
```

Codice Inserimento:

```
r = {} /* crea nuovo oggetto  */

/* Create multiple fields for that object and assign a value to them */
r.restaurant_id = "101"
 r.name = "Da Mario"
 r.cuisine = "Italiano"

a = {}
a.building = "1-A"

/* Create an array and assign it to the field coord of object a */
a.coord = [ 41.8902102, 12.4922309 ]
a.street = "Piazza del Colosseo"
r.address = a

/* Create an empty array */
r.grades = []

/* This is a faster way to add data inside an object. ISODate is an object and handles dates and time */
g = { date: ISODate("2016-07-31T13:00:00Z"), grade: "A", score: 10 }

/* Add the object to the last position of the array r.grades */
r.grades.push(g)
g = { date: ISODate(), grade: "D", score: 4 }
r.grades.push(g)

/* Finally add the created object into restaurants */
db.restaurants.insert(r)
```

OPPURE

Codice Inserimento:

```
db.restaurants.insert({
	restaurant_id: "101",
	name: "Da Mario",
	cuisine: "Italiano",
	address: {
		building: "1-A",
		coord: [ 41.8902102, 12.4922309 ],
		street: "Piazza del Colosseo"
},
grades: [
	{date: ISODate("2016-07-31T13:00:00Z"), grade: "A", score:
	10},
	{date: ISODate(), grade: "D", score: 4}
	]
})

db.restaurants.find( { restaurant_id: "101"} ).pretty()
db.restaurants.find( { restaurant_id: "101"}, { grades: 1}).pretty()
```

Codice Lettura:

```
/* Find every restaurant with the value of borough equal to Brooklyn*/
db.restaurants.find({borough: "Brooklyn"}).pretty()

/* Show every distinct value for the field borough*/
db.restaurants.distinct("borough")
/* Count how many they are  */
db.restaurants.distinct("borough").length

/*projection*/
db.restaurants.find({borough: "Brooklyn"}, {name: 1, borough: 1, _id: 0}).pretty()

/* Trovare quelli che hanno almeno un grade A ordinandoli per nome o per quartiere  */

db.restaurants.find({“grades.grade”:”A”}).sort({name:1}).limit(10).pretty()
db.restaurants.find({“grades.grade”:”A”}).sort({name:-1}).limit(10).pretty()  /* con ordinamento inverso */


/* grades written after the date "2013-10-01" */
db.restaurants.find({ "grades.date" : { "$gte" : ISODate("2013-10-01")}})


/* Documents with all the grades but A */

db.restaurants.find({"grades.grade": {"$ne": "A"}}, {grades: 1, "grades.grade": 1}).pretty()

/* object_id, quartieri e nomi di tutti i ristoranti ordinati per cucina  */
```

```
db.restaurants.find({},{name:1,borough:1}).sort({cuisine:1}).limit(
10)

/* quartieri e nomi di tutti i ristoranti ordinati per cucina  */

db.restaurants.find({},{_id:0,name:1,borough:1}).sort({cuisine:1}).
limit(10)

/*name contain string coffee, ignoreCase when a i is added after
the last "/" */
db.restaurants.find({name: /coffee/i}, {name: 1, _id: 0}).pretty()

/*name starts with Starbucks*/
db.restaurants.find({name: /^Starbucks/}, {name: 1, _id:
0}).pretty()

/*name ends with shop*/
db.restaurants.find({name: /Shop$/}, {name: 1, _id: 0}).pretty();

/*ignoreCase with i*/
db.restaurants.find({name: /^starbucks/i}, {name: 1, _id:
0}).pretty()
```

Codice Lettura avanzato:

```
/* restaurants with all scores greater then 20 (REQUIRES
JAVASCRIPT) */

db.restaurants.find( { $where: function(){

        for(i = 0; i < this.grades.length; i++){
                if(this.grades[i].score <= 20){return false;}
        }
        return true;
} } )
```

```
/* oppure in alternativa (controllare anche come eliminare gli score vuoti)*/

db.restaurants.find({"grades.score":{$not:{$lte:20}}}).pretty()
```

Aggregate e group :

```
/* group by borough */

db.restaurants.aggregate([{"$group": {"_id": "$borough"}}])

/* count how many restaurant per borough and sort the result */

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


/* ristoranti raggruppati per cucina e ordinati dal più numeroso */

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

/*    Aggregate doesn't require $group so we can use it to sort, project, match, etc. */

db.restaurants.aggregate([
   {"$sort": {"grades.score": -1}},
   {"$project": {"name": 1, "grades.score": 1}}
])
```

```
/* The restaurant with the highest score value */

db.restaurants.aggregate([
   {"$project": {"name": 1, "grades.score": 1}},
   {"$sort": {"grades.score": -1}},
   {"$limit": 1}
]).pretty()

/* show only restaurants with grade A */
db.restaurants.aggregate([
   {"$unwind": "$grades"},
   {"$match": {"grades.grade": "A"}},
   {"$project": {"restaurant_id": 1, "name": 1, "grades": 1}}
]).pretty()

/* count how many grades A every restaurant has */
db.restaurants.aggregate([
   {"$unwind": "$grades"},
   {"$match": {"grades.grade": "A"}},
   {"$group": {
         "_id": {"_id": "$_id",
           "name": "$name",
           "restaurant_id": "$restaurant_id"},
         "count": {"$sum": 1}
      }}]).pretty()

/* contare i ristoranti, raggruppandoli per cucina, e calcolandone
la media    */
/* (o max, min, sum) degli score */
/* e ordinandoli per quest'ultimo in ordine decrescente      */
```

db.restaurants.aggregate([{$unwind:'$grades'},{$group:
{_id:'$cuisine', total:{$sum:1}, avgSco:{$avg:'$grades.score'}}},
{$sort:{avgSco:-1}}])

/* contare, sommare, ecc. gli score per il ristorante con _id =
ObjectId("5e7249c0918aca3c0bbb7aad"). */

```
db.restaurants.aggregate([{"$match":
{"_id":ObjectId("5e7249c0918aca3c0bbb7aad")}},{"$unwind":"$grades"},{"$g
roup":{"_id":{"_id":"$_id","name":"$name",
"restaurant_id":"$restaurant_id"},"score_count":{"$sum":1},"sum_score":{
"$sum":"$grades.score"},"max_score":{"$max":"$grades.score"},"min_score"
:{"$min":"$grades.score"},"avg_score":{"$avg":"$grades.score"},
}}]).pretty()
```