Homework 5
Alex Ojemann

Section 1:

1:

```
> use new_mongo_db
switched to db new_mongo_db
```

2:

```
> db.dropDatabase()
{ "ok" : 1 }
```

3:

```
> use new_mongo_db
switched to db new_mongo_db
> db.createCollection("test_collection")
{ "ok" : 1 }
```

```
> db.createCollection("mycol", { capped : true, autoIndexId : true, size : 6142800, max : 10000 })
{
        "note" : "The autoIndexId option is deprecated and will be removed in a future release",
        "ok" : 1
}
```

4:

```
> use new_mongo_db
switched to db new_mongo_db
> db.test_collection.drop()
true
```

5:

```
> db.createCollection("test_collection")
{ "ok" : 1 }
> db.test_collection.insert({
...                             title: "Mongo Db practice",
...                             description: "this is my first MongoDB document"
...                           })
WriteResult({ "nInserted" : 1 })
```

6:

```
> db.test_collection.find().pretty()
{
        "_id" : ObjectId("62660cceb2699c824ad9bd8a"),
        "title" : "Mongo Db practice",
        "description" : "this is my first MongoDB document"
}
> db.test_collection.find()
{ "_id" : ObjectId("62660cceb2699c824ad9bd8a"), "title" : "Mongo Db practice", "description" : "this is my first MongoDB document" }
```

7:

```
> db.test_collection.update({'title':'Mongo Db practice'},{$set:{ 'title':'Updated MongoDB practice'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

8:

```
> db.test_collection.remove({ status : "P" }, 1)
WriteResult({ "nRemoved" : 0 })
> db.test_collection.remove({ status : "P" })
WriteResult({ "nRemoved" : 0 })
```

Section 2:

1:

```
> db.test_collection.count()
25359
```

2:

```
> db.test_collection.distinct("cuisine")
[
        "Afghan",
        "African",
        "American",
        "Armenian",
        "Asian",
        "Australian",
        "Bagels/Pretzels",
        "Bakery",
        "Bangladeshi",
        "Barbecue",
        "Bottled beverages, including water, sodas, juices, etc.",
        "Brazilian",
        "CafÃ©/Coffee/Tea",
        "Café/Coffee/Tea",
        "Cajun",
        "Californian",
        "Caribbean",
        "Chicken",
        "Chilean",
        "Chinese",
        "Chinese/Cuban",
        "Chinese/Japanese",
        "Continental",
        "Creole",
```

The first 24 results are shown

3:

```
> db.test_collection.find({"address.zipcode":"10023",cuisine:"Italian"},{name:1, _id:0})
{ "name" : "Fiorellos" }
{ "name" : "Gabriel'S Bar & Grill" }
{ "name" : "Il Violino" }
{ "name" : "Pomodoro Ristorante" }
{ "name" : "Arte Cafe" }
{ "name" : "Nick And Toni'S Cafe" }
{ "name" : "Bello Giardino" }
{ "name" : "Cesca" }
{ "name" : "Arte Around The Corner" }
{ "name" : "Riposo 72" }
{ "name" : "Salumeria Rossi Parmacotto" }
{ "name" : "Luce Restaurant & Enoteca" }
{ "name" : "Gina La Fornarina" }
{ "name" : "Lincoln Ristorante" }
{ "name" : "The Leopard At Des Artistes" }
{ "name" : "Pappardella" }
{ "name" : "Joanne Trattoria" }
```

4:

```
> db.test_collection.aggregate([{$match:{cuisine:"Greek"}},{$group:{_id:"$borough",num:{$sum:1}}}])
{ "_id" : "Queens", "num" : 58 }
{ "_id" : "Brooklyn", "num" : 14 }
{ "_id" : "Bronx", "num" : 4 }
{ "_id" : "Manhattan", "num" : 35 }
```

```
> db.createCollection("GreekBoroughs")
{ "ok" : 1 }
> db.GreekBoroughs.insert({ "_id" : "Queens", "num" : 58 })
WriteResult({ "nInserted" : 1 })
> db.GreekBoroughs.insert({ "_id" : "Brooklyn", "num" : 14 })
WriteResult({ "nInserted" : 1 })
> db.GreekBoroughs.insert({ "_id" : "Manhattan", "num" : 35 })
WriteResult({ "nInserted" : 1 })
> db.GreekBoroughs.insert({ "_id" : "Bronx", "num" : 4 })
WriteResult({ "nInserted" : 1 })
```

```
> db.GreekBoroughs.find().sort({num:-1}).limit(1)
{ "_id" : "Queens", "num" : 58 }
```

Here, I found all the boroughs, and their number of greek restaurants with the aggregate function, manually inserted them into a new collection and used the find and sort functions on that new collection to find the max value.

5:

```
> db.test_collection.find({name:{$regex: /Pho /}},{name:1,_id:0})
{ "name" : "Pho Bac Vietnamese Seafood Cuisine" }
{ "name" : "Pho Hoai Bay Ridge" }
{ "name" : "Pho Bang Restaurant" }
{ "name" : "Pho 32 & Shabu" }
{ "name" : "Pho Bang Restaurant" }
{ "name" : "Pho Viet-Nam Restaurant" }
{ "name" : "Pho Bang Restaurant" }
{ "name" : "Pho Grand" }
{ "name" : "Pho Vietnamese Restaurant" }
{ "name" : "Baoguette Pho Sure" }
{ "name" : "Pho Hoai Rest" }
{ "name" : "Pho Mac Vietnamese Cuisine" }
{ "name" : "Pho Hoang" }
{ "name" : "Pho Viet" }
{ "name" : "Pho Tay Ho 86 Vietnamese Restaurant" }
{ "name" : "Pho Seng" }
{ "name" : "Pho 32" }
{ "name" : "Pho Vietnam 87 Corporation" }
{ "name" : "Pho Rainbow Inc" }
{ "name" : "Pho Thanh Hoai 1" }
```

6:

```
> db.test_collection.aggregate([{$match:{cuisine:"Mexican"}},{$group:{_id:"$borough",num:{$sum:1}}},{$sort:{num:-1}}])
{ "_id" : "Manhattan", "num" : 273 }
{ "_id" : "Brooklyn", "num" : 212 }
{ "_id" : "Queens", "num" : 146 }
{ "_id" : "Bronx", "num" : 89 }
{ "_id" : "Staten Island", "num" : 33 }
{ "_id" : "Missing", "num" : 1 }
```

7:

```
> db.test_collection.aggregate([{$match:{cuisine:"Italian",borough:"Brooklyn"}},{$unwind:"$grades"},{$group:{_id:{restaurant_id:"$restaurant_id",name:"$name"},totalScore:{$sum:"
$grades.score"}}},{$sort:{totalScore:-1}},{$limit:5}])
{ "_id" : { "restaurant_id" : "41540476", "name" : "Joe'S Pizza" }, "totalScore" : 173 }
{ "_id" : { "restaurant_id" : "41410058", "name" : "Anella" }, "totalScore" : 153 }
{ "_id" : { "restaurant_id" : "41297604", "name" : "Tutta Pasta" }, "totalScore" : 124 }
{ "_id" : { "restaurant_id" : "40884999", "name" : "Doc Wine Bar" }, "totalScore" : 103 }
{ "_id" : { "restaurant_id" : "41525094", "name" : "Da Nonna Rosa" }, "totalScore" : 98 }
```

I understand that this problem didn't ask for the restaurant ID. The reason I provided it was so that my _id in the group function didn't only depend on the name of the restaurant because there could be multiple restaurants with the same name.