Task 1

```
C:\Program Files\MongoDB\Server\4.0\bin>mongoimport --db services --collection zipcodes --file C:\Users\Gediminas\Desktop\zips.json
2018-12-03T12:04:21.698+0000    connected to: localhost
2018-12-03T12:04:22.049+0000    imported 29353 documents

C:\Program Files\MongoDB\Server\4.0\bin>mongoimport --db services --collection restaurants --file C:\Users\Gediminas\Desktop\restaurants.json
2018-12-03T12:05:32.958+0000    connected to: localhost
2018-12-03T12:05:33.292+0000    imported 25359 documents
```

Task 2:
Q1:

var cursor = db.restaurants.find({name: /Pizza/i})

```
> var cursor = db.restaurants.find({name: /Pizza/i})
> cursor.pretty()
{
        "_id" : ObjectId("55cba2476c522cafdb053b06"),
        "location" : {
                "coordinates" : [
                        -73.9791458,
                        40.744328
                ],
                "type" : "Point"
        },
        "name" : "Domino'S Pizza"
}
{
        "_id" : ObjectId("55cba2476c522cafdb053b0c"),
        "location" : {
                "coordinates" : [
                        -73.9806854,
                        40.7778589
                ],
                "type" : "Point"
        },
        "name" : "Domino'S Pizza"
}
```

Q2:

```
db.zipcodes.createIndex({"loc": "2dsphere"})
 db.restaurants.createIndex({"loc": "2dsphere"})

cursor.forEach(function(one_ns){ //Task 2 part 2
        var some_zs=db.zipcodes.find({loc: {$nearSphere :one_ns.location}}).limit(1).toArray();
        print("Restaurant: "); printjson(one_ns.location);
         printjson(some_zs[0].city)
});
```

```
> cursor.forEach(function(one_ns){
... var some_zs=db.zipcodes.find({loc: {$nearSphere :one_ns.location}}).limit(1).toArray();
... print("Restaurant: "); printjson(one_ns.location);
... printjson(some_zs[0].city)
... });
Restaurant:
{ "coordinates" : [ -73.9254891, 40.59284479999999 ], "type" : "Point" }
"BROOKLYN"
Restaurant:
{ "coordinates" : [ -73.8520417, 40.6666793 ], "type" : "Point" }
"KEW GARDENS"
Restaurant:
{ "coordinates" : [ -73.74878939999999, 40.7271425 ], "type" : "Point" }
"QUEENS VILLAGE"
Restaurant:
{ "coordinates" : [ -73.782866, 40.757529 ], "type" : "Point" }
"BAYSIDE"
Restaurant:
{ "coordinates" : [ -73.8047246, 40.7328495 ], "type" : "Point" }
"FRESH MEADOWS"
Restaurant:
{ "coordinates" : [ -74.027017, 40.632575 ], "type" : "Point" }
"BROOKLYN"
```

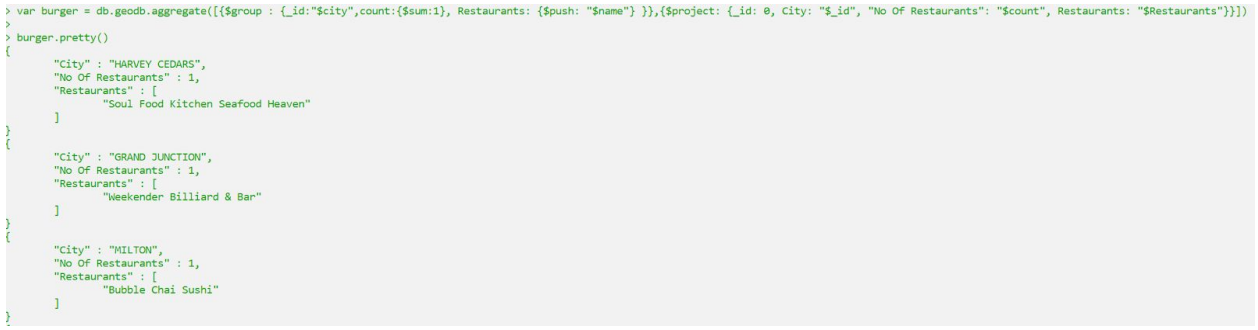
Q3:

```
var cursor = db.restaurants.find()

cursor.forEach(function(one_ns){
        var some_zs=db.zipcodes.find({loc: {$nearSphere :one_ns.location}}).limit(1).toArray();

        one_ns.city=some_zs[0].city
        db.geodb.insertOne(one_ns)
});
```

```
 var cursor = db.restaurants.find()
 cursor.forEach(function(one_ns){
.. var some_zs=db.zipcodes.find({loc: {$nearSphere :one_ns.location}}).limit(1).toArray();
.. one_ns.city=some_zs[0].city
.. db.geodb.insertOne(one_ns)
.. });
```

Q4:

var burger = db.geodb.aggregate([{$group : {_id:"$city",count:{$sum:1}, Restaurants: {$push: "$name"} }},{$project: {_id: 0, City: "$_id", "No Of Restaurants": "$count", Restaurants: "$Restaurants"}}])



```
> var burger = db.geodb.aggregate([{$group : {_id:"$city",count:{$sum:1}, Restaurants: {$push: "$name"} }},{$project: {_id: 0, City: "$_id", "No Of Restaurants": "$count", Restaurants: "$Restaurants"}}])
>
> burger.pretty()
{
        "City" : "HARVEY CEDARS",
        "No Of Restaurants" : 1,
        "Restaurants" : [
                "Soul Food Kitchen Seafood Heaven"
        ]
}
{
        "City" : "GRAND JUNCTION",
        "No Of Restaurants" : 1,
        "Restaurants" : [
                "Weekender Billiard & Bar"
        ]
}
{
        "City" : "MILTON",
        "No Of Restaurants" : 1,
        "Restaurants" : [
                "Bubble Chai Sushi"
        ]
}
```

Q5:

It is not sensible in either storing or retrieving. For example, if we were looking for a Dominos pizza restaurant, with the current modelling, we know what kind of store we are looking for, thanks to the coordinates and then we match the coordinates with the zip code and find the city it's in or its location. If we were to put the restaurantsInCity with the zip codes collection, every time we would search for a restaurant we will get extra locations and extra restaurants that we weren't looking for, and that takes time to compile so we are slowing down the program.