Task 1

Import people.json from the lab 3 into mongodb using mongoimport tool.

```
mongoimport --db dbName --collection collectionName --file people.json
```

What is the count using the below command?

```
db.people.count({ "email" : {"$exists": 1} })
```

Task 2

In this lab you're going to determine which queries are able to successfully use a given index for both filtering and sorting.
Given the following index:

```
{ "first_name": 1, "address.state": -1, "address.city": -1, "ssn": 1 }
```

Which of the following queries are able to use it for both filtering and sorting?

- db.people.find({ "first_name": "Jessica", "address.state": { $lt: "S"} }).sort({"address.state": 1 })
- db.people.find({ "first_name": "Jessica" }).sort({ "address.state": 1, "address.city": 1 })
- db.people.find({ "address.state": "South Dakota", "first_name": "Jessica" }).sort({"address.city": -1 })
- db.people.find({ "first_name": { $gt: "J" } }).sort({ "address.city": -1 })
- db.people.find({ "address.city": "West Cindy" }).sort({ "address.city": -1 })

Task 3

In this lab you're going to examine several example queries and determine which compound index will best service them.

```
> db.people.find({
   "address.state": "Nebraska",
   "last_name": /^G/,
   "job": "Police officer"
 })
```

```
> db.people.find({
   "job": /^P/,
   "first_name": /^C/,
   "address.state": "Indiana"
```

```
 }).sort({ "last_name": 1 })
```

```
> db.people.find({
   "address.state": "Connecticut",
   "birthday": {
     "$gte": ISODate("2010-01-01T00:00:00.000Z"),
     "$lt": ISODate("2011-01-01T00:00:00.000Z")
   }
 })
```

If you had to build one index on the people collection, which of the following indexes would best sevice all 3 queries?

- { "address.state": 1, "job": 1 }
- { "job": 1, "address.state": 1 }
- { "address.state": 1, "last_name": 1, "job": 1 }
- { "job": 1, "address.state": 1, "last_name": 1 }
- { "address.state": 1, "job": 1, "first_name": 1 }
- { "job": 1, "address.state": 1, "first_name": 1 }

Task 4

In this lab you're going to use the equality, sort, range rule to determine which index best supports a given query.

Given the following query:

```
db.accounts.find( { accountBalance : { $gte :
NumberDecimal(100000.00) }, city: "New York" } )
        .sort( { lastName: 1, firstName: 1 } )
```

Which of the following indexes best supports this query with regards to the equality, sort, range rule.

- { accountBalance: 1, city: 1, lastName: 1, firstName: 1 }
- { lastName: 1, firstName: 1, accountBalance: 1, city: 1 }
- { lastName: 1, firstName: 1, city: 1, accountBalance: 1 }
- { city: 1, lastName: 1, firstName: 1, accountBalance: 1 }

Task 5

For this lab, you're going to create an index so that the following aggregation query can be executed successfully.
After importing the restaurants dataset, without any indexes:

```
$ mongoimport -d m201 -c restaurants --drop restaurants.json
```

If you attempt to run the following query you'll receive an error.

```
db.restaurants.aggregate([
  { $match: { stars: { $gt: 2 } } },
  { $sort: { stars: 1 } },
  { $group: { _id: "$cuisine", count: { $sum: 1 } } }
])
```

```
{
  "ok": 0,
  "errmsg": "Sort exceeded memory limit of 104857600 bytes, but did not opt in to external sorting. Aborting operation. Pass allowDiskUse:true to opt in.",
  "code": 16819,
  "codeName": "Location16819"
}
```

Identify why this error is occuring, and build an index to resolve the issue.
Keep in mind that there might be several indexes that resolve this error, but we're looking for an index as small as possible that services this command.
Note: The index should be ascending in nature.