**Question 1**

1. The address in user collection is changed into an array to solve the problem of multiple address. And it will be easy to get all data of address to be created as a nested collection.
2. Publisher is built as a separate collection, because it will be helpful if we want to add more publishers in the future. It is possible we currently don’t have any books in our library published by this new publisher, but we still want to have information about this publisher for statistic use. Also, MongoDB has 16MB size limitation for each collection (BSON document). A library will has so much data, and will keep increasing. Moreover, when adding books with same publisher into database, we only need to enter the information of publishers once.
3. The publisher in book collection is changed into an array to solve the problem that a book may have multiple publishers.
4. The address in publish collection is changed into an array to solve the problem of multiple address.
5. The subjects in book collection is changed into an array to solve the problem of multiple subjects.
6. The note in book collection is changed into an array to solve the problem of multiple note
7. The availability in book collection and active in user collection are changed into Boolean type to make more sense and convince in use
8. All dates are changed from String type into Date type to make more sense and convince in use
9. Zip codes are changed from int type into string to make more sense and avoid missing leading zeros.

**Question 2**

Inserted by load() function, json files is provided.

Imported js file is provided as well

**Question 3**

db.book.find().pretty()

.pretty() is optional

**Question 4**

db.book.find({author:{ $ref : "author", $id : db.author.findOne({"first\_name" : "Danielle","last\_name" : "Steel"}).\_id}}).pretty()

**Question 5**

db.user.find({dateOfCreation : {$gt: new Date("2014-12-15")}, "addresses.city" : "Boston"}).pretty()

**Question 6**

db.book.find({$where : "this.publisher.length > 1 "}).pretty()

**Question 7**

db.book.find({$where : "this.note.length > 0 "}).pretty()

**Question 8**

I agree with the decision to use MongoDB for Craigslist data storage. Comparing with MySQL, MongoDB shows great advantages on big data storage and simple/hierarchical data storage. As the article mentioned about MySQL, a simple schema change on their vast archive took months to complete (because of too much data). MySQL is definitely not suitable for Craigslist. Although I don’t agree that NoSQL is easy to use, NoSQL is really good at simple/hierarchical data storage. And it has flexible data models. Craigslist just need a database to store information about job and classifieds, NoSQL will be a better choice (for big/simple data).

**Question 9**

a) Craigslist

NoSQL

As stated in question 9. NoSQL is better at handling big or simple data. Although it may not 100% accurate, Craigslist has tons of data. (accuracy is not so important as bank). Also it elastic scaling, because of its distribute data.

b) Amazon.com’s product recommendations database

NoSQL

Similar to Craigslist, product recommendations won’t need so much managements, so NoSQL can automatic repair and simpler data models. Also, product recommendation can be a big data.

c) Twitter posts

NoSQL

Again, Twitter posts is a lot of data. NoSQL can be a better tool to manage these data. It is cheap and elastic in scaling. Twitter posts is not so risk averse and in simple design, so NoSQL will be a better choice.

d) A traditional banking application

Relational database

Banking is very risk averse and conservative, a relational database with relational data model is more suitable in this situation. Ralational database is better at ACID (Atomic, Consistent, Isolated and Durable). NoSQL databases typically do not have strict ACID requirements.