QA Consulting.

MongoDB Exercise Book

QAC BIG DATA

Prepared by
Devdatta Gonsai

devdatta.gonsai@qa.com

Contents

Task 1 – Primer Dataset

Do some preliminary commands to understand what you can do to create databases and collections within MongoDB.

1. Create a new database called 'testdatabase'

2. Check which database you are currently using

3. Check what databases are available on your MongoDB server

4. You'll notice your database isn't listed there yet, it needs at least one document in it to appear. Insert a record that just includes a field called 'name' with your name as the value.

5. Show all databases available again.

6. MongoDB has a useful database for getting to grips with it, but you'll need to import it. It's the 'primer-dataset.json' you have in your ExampleData folder. Follow the steps below to import it.
    a. Open a new command prompt in the location where MongoDB is installed.
    b. Run the following command. This will use the mongoimport executable to import our data. Replace the path file to represent where you have saved your data.

        --db specifies which database you're going to insert
        --collection can specify the collection name, else it will take it from the file name
        --drop target will drop the collection before importing the data, in case there was anything there already
        --file specifies the location of the file containing the data

        mongoimport --db test --collection restaurants --drop --file /path/to/ExampleData/primer-dataset.json

7. Now you have imported the sample data, change the current database to use the test database.

8 Return all documents in the restaurants collection to get a feel for what the data looks like

9 Insert a new record into this dataset of a restaurant that you like
db.testCollection.update({ "name":"Sal'S Deli" }, { $set:{"owner":"Javier"} }, {multi:true})

10 Return all documents in the restaurant collection where the borough is "Queens"
db.testCollection.find({"borough":"Queens"}).pretty()

11 Return all documents in the restaurant collection where the grade is A
db.testCollection.find({"grades.grade":"A"}).pretty()

12 Return all documents in the restaurant collection where the grade score is greater than 30
db.testCollection.aggregate([{$unwind:{path:"$grades"}},{$match:{"grades.score":{$gt:30}}}]).pretty() Return all documents in the restaurant collection where the borough is "Queens" and the cuisine is "Italian".
db.testCollection.find({"grades.grade":"A"}).pretty()

13 Return the top 2 documents in the collection that have the highest score value in the grades embedded document,
db.testCollection.find().sort({"grades.score":-1}).limit(2).pretty()

Task 2 – Movie Lens

1. Create a new database called movielens

2. In your ExampleData folder you will have two .json files for the Movielens data. Use the mongoimport command to import these into the movielens database, into collections for users and movies. Be careful to change the mongoimport command to change the specified database.

3. Return all documents in the movies collection. Read the output carefully to understand what kind of data the movies collection holds.

4. Return all documents in the users collection

5. You'll notice that the users collection holds a lot of data that is difficult to parse. This is because the embedded document 'movies' contains all the reviews that a user has made. Return only one document of user and limit the number of 'movie' values to be returned to only 3. Read this carefully to understand what data the user documents hold.
   db.users.findOne({},{"movies":{$slice:3}})
6. How many movies are there in this collection?
   db.movies.count()
7. Return the name and occupation of the user called Barry Erin
   db.users.find({"name":"Barry Erin"}, {_id:0,"name":1,"occupation":1})
8. Return all unique occupations in the database
   db.users.distinct("occupation")
9. How many users have the 'scientist' occupation?
   db.users.find({"occupation":"scientist"},{}).count()
10. Who are the 10 oldest writers?
    db.users.find({"occupation":"writer"},{"_id":0,"name":1,"age":1,"occupation":1}).sort({"age":-1}).limit(10)
11. Update all 'writer' occupations to 'author'.
    db.users.update({"occupation":"writer"},{$set:{occupation:"author"}},{multi:true})
12. Insert a user record for yourself without a movies field
    db.users.insert({"name":"Javier Alcazar","gender":"M","age":20,"occupation":"Student"})
13. Update your record to add a movies field rating a film of your choice
    db.users.update({"name":"Javier

Alcazar"},{$set:{"movies":[{"movieid":666,"rating":6,"timestamp":666666666}]}})

14 Remove yourself from the database.
db.users.remove({"name":"Javier Alcazar"})

Task 3 – Zips

1. Import the zips.json file into the zipsDB database with the collection name as zips. You will need to use mongoimport for this.
   mongoimport --db zipsdb --collection zips --file C:\Users\Admin\Documents\MongoDB\zips.json
2. List the information about the first city in the collection.
   db.zips.findOne()
3. Insert a document of your own home town/city, based on the same format as other documents into the collection.
   db.zips.insert({"city":"Cadiz","loc":[123,321],"pop":12345,"state":"DN"})
4. Find all of the cities in the state of Tennessee (TN).
   db.zips.find({"state":"TX"}).pretty()
5. List all of the distinct states.
   db.zips.distinct("state")
6. Remove the document you inserted previously from the zips collection.
   db.zips.remove({"city":"Cadiz"})
7. Count the amount of documents in the collection.
   db.zips.count()
8. How many zip codes appear in the state of Texas (TX)?
   db.zips.find({"state":"TX"}).count()
9. Find the most populated city in the collection?
   db.zips.find().sort({"pop":-1}).limit(1)
10. Find the average population of New York (NY).
    db.zips.aggregate([{$group:{_id:"$state",avgPop:{$avg:"$pop"}}},{$match:{_id:"NY"}}])
11. Find the total population of Illinois (IL).
    db.zips.aggregate([{$group:{_id:"$state",sumPop:{$sum:"$pop"}}},{$match:{_id:"IL"}}])
12. Calculate the average population of cities in Illinois (IL) and Florida (FL) taken together with a population greater than 10,000.
    db.zips.aggregate([{$match:{"pop":{"$gt":10000}}},{$group:{_id:"$state",avgPop:{$avg:"$pop"}}},{$match:{$or:[{_id:"IL"},{_id:"FL"}]}}])
13. Find the top 3 cities with the highest population.
    db.zips.find().sort({"pop":-1}).limit(3)
14. How many cities does the state of WY have?
    db.zips.find({"state":"WY"}).count()

15 List all of the states with the amount of cities they have in them.
db.zips.aggregate([{$group:{_id:"$state",count:{$sum:1}}}])

16 List all of the states that have less than 80 cities.
db.zips.aggregate([{$group:{_id:"$state",count:{$sum:1}}},{$match:{count:{"$lt":80}}}])

17 You'll notice that there are a number of cities listed more than once. Which city appears the most?
db.zips.aggregate([{$group:{_id:"$city",count:{$sum:1}}},{$sort:{count:-1}},{$limit:1}])

18 Building on the previous question, list the top 3 cities that appear the most in this collection.
db.zips.aggregate([{$group:{_id:"$city",count:{$sum:1}}},{$sort:{count:-1}},{$limit:3}])

19 List the cities that appear in more than 10 states.
db.zips.aggregate([{$group:{_id:"$city",states:{$addToSet:"$state"}}},{$project:{count:{$size:"$states"}}},{$match:{count:{"$gt":10}}}])

Scenarios

Below there are two tasks, each of which describes a scenario that requires a MongoDB database. Each scenario includes a small handful of user stories to get you started but you must expand on this much further for each scenario you undertake with a minimum of 8 user stories in total. Before you begin creating anything in MongoDB you should have full documentation on what the system needs to have.

For each scenario:

- Write appropriate user stories
- Create a diagram showing a rough structure of how the database will work
- Create the database in MongoDB
- Populate collections with some data
- Create queries necessary to answer the questions in your user stories

Library Rentals – this library contains many different items such as books, audio CDs, DVDs, vinyl records, video tapes, cassettes, magazines, and academic papers. Ensure your database stores different information on each of these types.

1. As a Librarian I want to query if any rentals are overdue so I can chase this up
   db.Library.aggregate([{$match:{"takeInDue":{$exists:true}}},{$project:{title:1,compareDate:{$strcasecmp:["$takeInDue","2004-01-31"]}}},{$match:{compareDate:{"$lt":0}}}])
2. As a Customer I want to query by a particular author or artist in order to find items by someone I like
   db.Library.find({"author":"Will Smith"}).pretty()
3. As a Social Media Consultant I want to know which items get rented out the most so I can create effective campaigns for them
   db.Library.aggregate([{$group:{_id:"$title",rented:{$sum:"$timesRented"}}},{$sort:{rented:-1}}])

4 As a Receptionist I want to know how many copies of a particular item are in the library

db.Library.aggregate([{$match:{"takeInDue":null}},{$group:{_id:"$title",total:{$sum:1}}}])

5 As a Manager I want to know how many items of each type are in the library (including items taken out by customers)

db.Library.aggregate([{$group:{_id:"$type",count:{$sum:1}}}])

Emergency Services Logs – this should be documentation on calls made to the emergency services line. This should include information on what a customer is calling about, who handled the call, the type of resolution, and any additional comments at a minimum.

1 As a Managing Director I want to query what the peak times for particular service requests are so I can appropriately assign staff

2 As a Calls Manager I want to query which members of staff have taken the most calls so I know who is performing well

3 As an Managing Director I want to query what the maximum call duration has been the past month so I can track statistics on this