Create and query a MongoDB collection – Practice 5

Data Management and Visualization *- Politecnico di Torino*

Goal

The objective of the practice is to create and successfully populate a collection of documents. Then, query the database exploiting different MongoDB functionalities and patterns. Optionally, configure a replica set in a Docker environment and connect to the MongoDB database.

Database Connection

Create a free cluster following the instructions provided in the previous Lab. **You do not need to repeat these steps if you already did them once**.

Get string connection from MongoDB Atlas server, under Database **Deployment > Connect**, selecting "**Drivers**" for the connection with pymongo (notebook), or "**Compass**" to connect with MongoDB Compass.

To proceed, connect to the Lab notebook at the following link and make a **Copy to your Drive**.

https://colab.research.google.com/drive/1l1ymlexh1JOju4RqDKzKhc8MN125MFk4

Running queries of interest

Each document of the collection has a structure with the following fields:

```
{
   _id: <ObjectId>,name: <string>, // name of the restaurant
   tag: <list[string]>, // tags assigned by the users
   orderNeeded: <boolean>, // if the user should reserve
   maxPeople:<int>, // maximum number of customers
   review:<float>, // average vote
   cost:<string>, // classification of the menu price between low, medium and high
   location:{
         type:"Point",coordinates:[<lat>,<long>]
   }, // geographical point
   contact:{
         phone:<string>, // telephone of the restaurant
         facebook:<string> // link to the facebook page
   }
}
```

Run the following queries of interest:

1. Find all restaurants whose cost is medium. Show the result in the pretty format.
2. Select the name and the number of seats (maxPeople) available of all the restaurants whose review is bigger than 4 and cost is medium or low
3. Select the name, the phone of the restaurants that can contain more than 5 people and:
   a. whose tag contains "italian" or "japanese" and cost is medium or high OR
   b. whose tag does not contain neither "italian" nor "japanese", and whose review is higher than 4.5

   Remove from the output the field _id.
4. Calculate the average review of all restaurants
5. Count the number of restaurants whose review is higher than 4.5 and can contain morethan 5 people
6. Find the restaurant in the collection which is nearest to the point [45.0644, 7.6598]
   Hint: remember to create the geospatial index.
7. Find how many restaurants in the collection are within 500 meters from the point[45.0623, 7.6627]
8. Add the tag “pizza” to all the restaurants that contain the tag “italian”. If the tag “pizza” isalready present, you should not insert it.
9. Decrease the review score of 0.2 for all the restaurants with the tag ‘fastfood’
10. For only the restaurants with a review higher than 3, find the tags which appear more than 1 time. For each tag, show how many documents include it.
11. For each cost category, compute the minimum review rate, the maximum review rate, theaverage review rate and the number of restaurants. Sort the result in descending orderaccording to the number of restaurants in each cost category.
12. Find the median value of maxPeople attribute

Replica set (Bonus)

Initial Setup

To create a replica set we will use the Docker Platform.

- Linux/MacOS requirements: Docker, Docker Compose
- Windows requirements: Docker Desktop for windows

Place the Docker Compose file (`docker-compose.yml` , provided with practice) in a folder (e.g., `$HOME/MongoDB`).

Open a terminal and run the following command (if the Docker Containers do not start, tryrenaming the file docker-compose.yml into docker-compose.yaml):

```
docker-compose up -d
```

The docker services will run in detached mode. If you want to see the output of each container, remove the option `-d` form the previous command.

Configure the replica set

The docker-compose.yml file defines 3 instances of MongoDB. The name of each instance can be retrieved using the command:

```
docker ps
```

All the following commands should be launched inside the folder of the project (e.g., `$HOME/MongoDB`).

1. Connect to the Mongo shell of one instance using the following command

   ```
   docker-compose exec name_docker_mongo mongo
   ```

   where `name_docker_mongo` is the name of one mongo instance.

2. Configure the replica set with the following requirements (copy and paste the following):

   a. replica set name should be equal to `rspoli`

   b. the priority of `polimongodb1` instance should the highest one

   c. the priority of `polimongodb3` instance should the lower one

   ```
   rsconf = {
     _id : "rspoli",
     members: [
       {
         "_id": 0,
         "host": "polimongodb1:27017",
         "priority": 4
       },
       {
         "_id": 1,
         "host": "polimongodb2:27017",
         "priority": 2
       },
       {
         "_id": 2,
         "host": "polimongodb3:27017",
         "priority": 1
       }
     ]
   }

   rs.initiate(rsconf);
   rs.conf();
   ```

3. Check the configuration with the following command:

```
rs.status()
```

4. Shut down from a new shell the primary node. To shut down a node use the following command

   ```
   docker-compose stop name_primary_mongo
   ```

   where `name_docker_mongo` is the name of the mongo container running the primary node.
5. Identify the new primary node
6. Restart the shut down node

   ```
   docker-compose restart name_primary_mongo
   ```

7. Re-check the status of replica set