MongoDB practical session

Ecole Centrale de Lyon, January 2026, Előd Egyed-Zsigmond, Nathan Nowakowski

Consignes

Renommez le fichier : TP_Mongo_GrX_NOM1_NOM2.txt qui se trouve dans l'archive contenant le sujet du TP en remplaçant NOM1 et NOM2 par vos noms. Insérez vos réponses après l'énoncé correspondant dans le fichier. Pour les requêtes, insérez les premières lignes (ne pas dépasser une trentaine) du résultat.

Vous devez déposer vos comptes rendus sur Moodle avant le mercredi 7 janvier 2026, 18h00.

Part 2 : Simple Queries

Express simple queries for the following searches (provide the query and a few lines of the results for each question):

1. List all stations located in the city of **VILLEURBANNE**.
2. List all stations **not** located in **VILLEURBANNE**.
3. List all stations that do not have a second address (**address2** is empty).
4. Count the number of stations where the second address has a value.
5. List stations with more than 2 bikes available (**available_bikes**).
6. List all distinct cities (**communes**).
7. Find stations that are in **VILLEURBANNE** or **VENISSIEUX** and have more than 2 available bikes
8. List all distinct cities, sorted in alphabetical order.
9. List stations in the 9th arrondissement (**Lyon 9ème**) in ascending order of available bikes.
10. Project results of the previous question onto the station name, address, and number of available bikes.
11. Display cities and the number of Vélo'V stations in each commune.
12. Sort the results by commune.
13. Sort the results by the number of stations in descending order.
14. Group the Vélo'V stations in **VILLEURBANNE** by the number of available bikes, sorted by this number.
15. Add to the previous result the names of stations for each number of available bikes.
16. Calculate the average number of available bikes per city.
17. Create a new collection called "simple_velov" that contains the content of the "properties" field directly at the root level of each document. (look for other $aggregate stages)

Geographical Queries

Create a 2D Sphere Index

Specify the Coordinates as Geographical Data

- Use the following commands to create a 2D sphere index:
- `use mongoLab;`

- `db.velov_geo.createIndex({ "geometry.coordinates": "2dsphere" });`

Verify the Index Creation

- Confirm that the index has been created with the command:
- `db.velov_geo.getIndexes();`

Geographical Queries

18. **Find Vélo’V Stations Within 500m of a Point**

- Look for Vélo’V stations within a radius of 500 meters from the point:

```
$geometry: {
  type: "Point",
  coordinates: [ 4.863132722360224, 45.77022676914935 ]
}
```

(Use the `$near` operator.)

19. **List the 5 Closest Stations to Specific Coordinates**

- List the five nearest stations to the coordinates:

```
{
  type: "Point",
  coordinates: [ 4.863132722360224, 45.77022676914935 ]
}
```

(Use the `$geoNear` operator.)

Textual Querie

Textual Query

20. **Find velov stations with the address containing: "Europe".**

First solution with a regular expression:

```
db.velov_geo.find({
  "properties.address": { "$regex": "Europe", "$options": "i" }
});
```

- `db.velov_geo.find(...)`: Specifies the collection to search.

- `"properties.address"`: The field you are targeting.

- `$regex`: This operator allows for pattern matching. By passing "Europe", it looks for that sequence of characters anywhere within the field.

- **`$options: "i"`**: This is an optional flag for **case-insensitivity**. It ensures that "EUROPE" or "europe" are also returned. If you want a strict match for exactly "Europe", you can remove the `$options` line.

or a shorter form :

```
db.velov_geo.find({ "properties.address": /Europe/i });
```

Performance Tip

Using a regex search with a leading wildcard (essentially what happens when you search for a substring) can be slow on very large collections because it cannot efficiently use standard indexes. If this collection grows significantly, you might consider using a Text Index for better performance.
To optimize your search, you can create a **Text Index** on the `address` field. Unlike a standard regex search, a text index tokenizes the string, allowing MongoDB to find words much faster, especially if the `velov_geo` collection grows.
To create a text index on the address field use the following command:

```
db.velov_geo.createIndex({ "properties.address": "text" });
```

Now you can use the operators $text and $search to search :

```
db.velov_geo.find({
  $text: { $search: "Europe" }
});
```

Note that you find no documents. Indeed the text index is language dependant. By setting the language to **French**, MongoDB will use a French-specific "stop word" list (ignoring words like *le*, *la*, *de*) and a French **stemmer**, which understands word variations (e.g., searching "vélo" could match "vélos").

The Query to Create a French Text Index

To create the index on the `address` field with French as the default language, use the following command:

```
db.velov_geo.createIndex(
  { address: "text" },
  { default_language: "french" }
);
```

When you execute this query, you will probably get an error message. Indeed in MongoDB you can have only one text index for a collection.

In order to have one with the French language, you have to delete the existing one.
You can check the existing indexes to get the names and use the following command to drop the text index:

```
db.velov_geo.dropIndex("properties.address_text");
```

Now you can create the new text index with french as default language and try the search query.

21 Add Two New Vélo'V Stations to the Database

Steps:

a) Find Information About Two Stations
   - Visit the [Velov data page at the grandlyon website](#).
   - Locate two stations and note down their details, such as:
     - Station name
     - Location (latitude and longitude)
     - Total number of stands
     - Number of available bikes
     - Number of free stands

b) Extract/create JSON Files for the Two Stations
   - Using a text editor (e.g., Notepad++, Sublime Text, or VS Code), create a JSON file with the structure matching the existing `velov_geo.json` format.
   - Describe your procedure and provide the 2 json elements

c) Add the two stations to the collection. Describe your procedure

22 Data enrichment

Look for downloadable data in JSON format on the website https://data.grandlyon.com/ (or elsewhere) that can complement the information of Vélo'V stations.

a) Provide the URL(s) that allows you to download the data and explain how you think it can complement the MongoDB collection of Vélo'V stations.

b) Download the data and import it into MongoDB, simplify it to have more than one document. Describe your procedure.

c) Provide at least 3 MongoDB queries that combine the new information with the old data: 1 that involves matching addresses (properties.address, properties.nom, … check the fields), another that implies geographic coordinate comparison. For the additional queries use your imagination.

   Describe your method. Provide: query, results and explanation.