MESIIN480323 : No SQL 19/01

PROJECT REPORT

On

MongoDB

reuters.json

GABRIEL CHAIX & MALO LEROY & CHARLES TERRIER

ÉCOLE SUPERIEURE D'INGENIEURS LEONARD DE VINCI

ANNEE 2023 / 2024

INDEX

INTRODUCTION AND PROBLEM STATEMENT

Reuters is a news agency founded in 1851 in London. It is one of the global and generalist news agencies, an activity that represents a part of its revenue, mainly devoted to financial information. It is one of the greatest news agencies in the world. We want to create a MongoDB database with reuters articles and query it in different levels of complexity.

I. Data Importation

In a first manner, we'll setup the docker container with mongo db image and make sure everything is working properly.

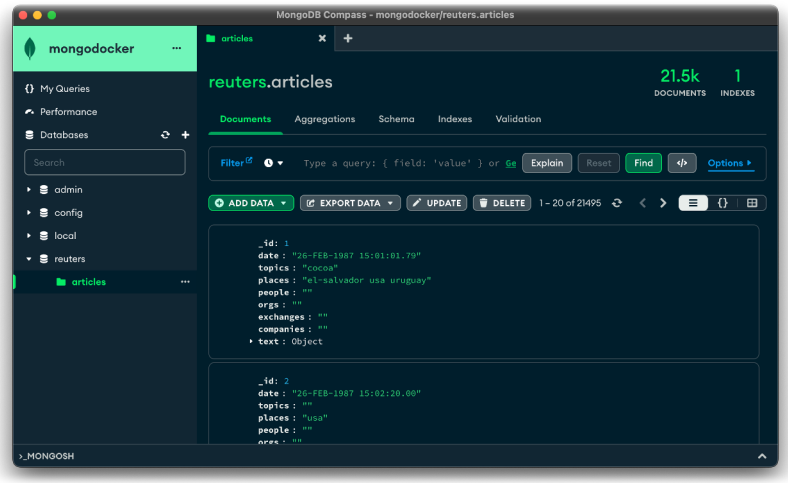
Now we must create our database using python and the pymongo package. To do this, we run the following script to build the database schema and fill the tables with our dataset content. After downloading our json file put it in the same directory as our code.

```
import json
from pymongo import MongoClient, InsertOne

client = MongoClient("mongodb://localhost:27017")
db = client.reuters
collection = db.articles
requesting = []

with open("MongoDB/reuters.json") as f:
    for jsonObj in f:
        myDict = json.loads(jsonObj)
        requesting.append(InsertOne(myDict))

result = collection.bulk_write(requesting)
client.close()
```

Then here is the importation script we've wrote

←

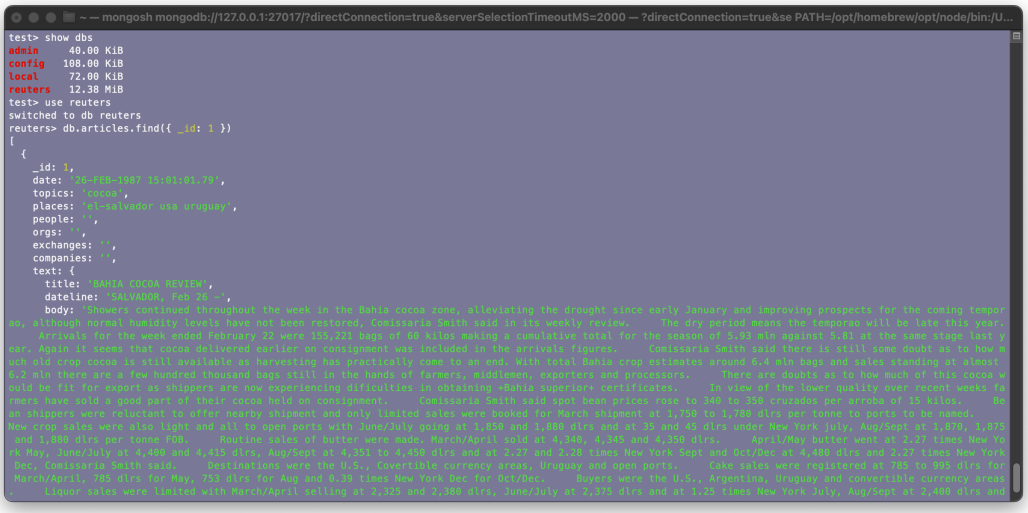
After importing all the rows, we make sure that each row is correct.



Using mongo compass, we can see everything looks alright.

II. Data cleaning

The first step we've taken into cleaning has been to update all date entry to the valid date type.



This aggregation pipeline transforms documents in a collection by converting the format of the date field to a date object and selectively projects specific fields as well as nested fields within text (such as title, dateline, and body), and the people field, essentially filtering and formatting the document structure for the output.

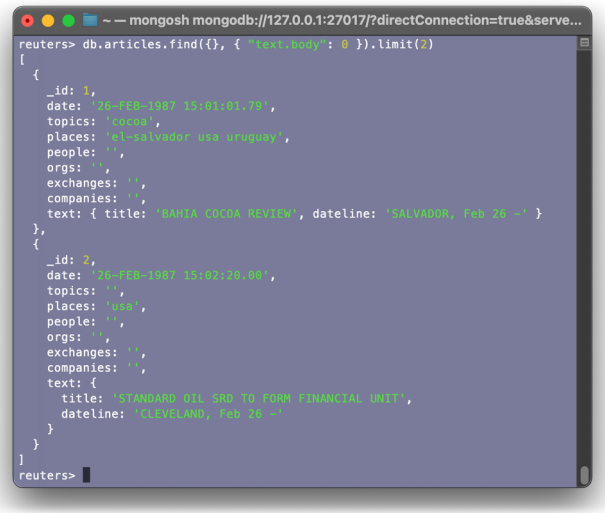
```
db.articles.aggregate([
  {
    $project: {
      date: {
        $toDate: "$date"
      },
      places: 1,
      companies: 1,
      topics: 1,
      exchanges: 1,
      _id: 1,
      orgs: 1,
      "text.title": 1,
      "text.dateline": 1,
      "text.body": 1,
      people: 1
    }
  },
  {
    $merge: {
      into: "articles",
      on: "_id",
      whenMatched: "replace",
      whenNotMatched: "insert",
    },
  },
])
```

This conversion requires to deal with few strings exception that we've modified using $set and $regexMatch statements.

III. Queries

A. Simple queries (6)

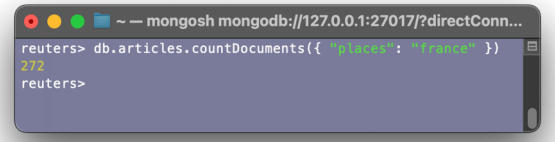
1. List of two first articles



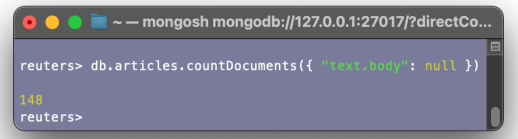
2. List of texts titles sorted by date

3. Titles, id and dateline of texts where id “120”

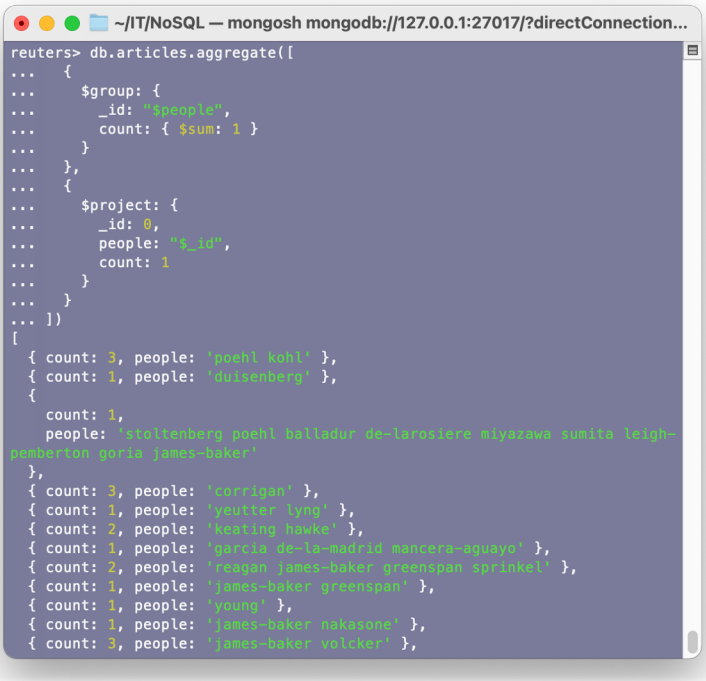
4. Count of articles whose place is France.



5. Count of publications whose “texts” is null.



6. Give the number articles written by each author ?



In this part, we performed a series of queries to get obvious and trivial information about the content of our dataset. Now, we will go further across the following complex queries.

B. Complex queries (2)

1. How much articles have been published per month?

Since we converted the dates to date type, we can simplify the format of dates.

2. For each topic, give the average number of people that wrote on. Sort the result decreasingly.

```
db.articles.aggregate([
  {
    $match: {
      "people": { $exists: true, $ne: "" }
    }
  },
  {
    $addFields: {
      "peopleArray": { $split: ["$people", " "] }
    }
  },
  {
    $group: {
      _id: "$topics",
      averageNumberOfPeople: {
        $avg: { $size: "$peopleArray" }
      }
    }
  },
  {
    $sort: {
      "averageNumberOfPeople": -1
    }
  }
])
```

C.Hard query (1)

1. Between 1984 and 2024, how long have the articles been existing for a certain topic for each country? Sort by average body length.

To answer this question correctly, we make an aggregation pipeline that performs the following operations:

Filters the documents between certain date having not null text. →

Turns people topics and places to array of strings using split. →

Groups documents by topics, then subgroups by places. →

Calculates the average length of text.body for each subgroup. →

Sorts results by average body length, in descending order. →

```
db.articles.aggregate([
  {
    $match: {
date: {
        $gte: ISODate("1984-01-01T00:00:00Z"),
        $lt: ISODate("2025-01-01T00:00:00Z")
      },
      "text.body": { $ne: null }
    }
  },
  {
    $project: {
      topics: { $split: ["$topics", " "] },
      places: { $split: ["$places", " "] },
      people: { $split: ["$people", " "] },
      textLength: { $strLenCP: "$text.body" }
    }
  },
  {
    $unwind: "$topics"
  },
  {
    $unwind: "$places"
  },
  {
    $group: {
      _id: {
        topic: "$topics",
        place: "$places"
      },
      averageLength: { $avg: "$textLength" }
    }
  },
  {
    $sort: {
      "averageLength": -1
    }
  }
]);
```

According to the results, we see that the
longest articles between 1984 and 2024 was talking about cotton-oil in the United States.