This is a practice project for anyone who wants to practice using MongoDB. The dataset used in this project is a json file containing information about different countries. The dataset is in correct format, however, all the data itself is not correct or updated. It does provide a really good sample dataset to practice using MongoDB and thus, one can ignore the false facts. What is here meant by some false information can be demonstrated with these results:

```
[ db.countryInfo.distinct("name", {"$and" : [{"region": "Europe" }, {"currency": "EUR"}]}).sort()
        "Aland Islands",
        "Andorra",
        "Austria",
        "Belgium",
        "Cyprus",
        "Estonia",
        "Finland",
        "France",
        "Germany",
        "Greece",
        "Ireland",
        "Italy",
        "Kosovo",
        "Latvia",
        "Lithuania",
        "Luxembourg",
        "Malta",
        "Monaco",
        "Montenegro",
        "Netherlands",
        "Portugal",
        "San Marino",
        "Slovakia",
        "Slovenia",
        "Spain",
        "Vatican City State (Holy See)"
]
>
```

Here we can see that Aland Islands are listed as an individual country even though it is part of Finland.

These small faults can be ignored, and the dataset used for practice purposes.

1. First, we need to import the json file to MongoDB. This can be done on the following way:

```
:\Program Files\MongoDB\Server\5.0\bin>mongoimport --db=WBProjects --collection=countryInfo --jsonArray --file "C:\Program Files\MongoDB\countries.json"
2021-11-09T08:54:53.477+0100    connected to: mongodb://localhost/
2021-11-09T08:54:53.535+0100    250 document(s) imported successfully. 0 document(s) failed to import.

:\Program Files\MongoDB\Server\5.0\bin>
```

2. Start the MongoDB shell to connect to the MongoDB server:

```
C:\Users\s[redacted]>cd "C:\Program Files\MongoDB\Server\5.0\bin"

C:\Program Files\MongoDB\Server\5.0\bin>mongo
```

3. Now we can start issuing commands to the server and switch to a specific database. In the following screenshot we switch to WBProject Database. Show collections in it and show all the information inside of countryInfo (where the practice file is uploaded).

4. To make more sense of data we can use pretty() method to change the output to easier to read format.

5. db.stats() method is used to return a document that reports on the state of the current database.

```
> db.stats()
{
        "db" : "WBProjects",
        "collections" : 3,
        "views" : 0,
        "objects" : 251,
        "avgObjSize" : 945.0438247011953,
        "dataSize" : 237206,
        "storageSize" : 135168,
        "freeStorageSize" : 0,
        "indexes" : 3,
        "indexSize" : 45056,
        "indexFreeStorageSize" : 0,
        "totalSize" : 180224,
        "totalFreeStorageSize" : 0,
        "scaleFactor" : 1,
        "fsUsedSize" : 80928571392,
        "fsTotalSize" : 510770802688,
        "ok" : 1
}
>
```

6. Here we want to count all the records in countryInfo collection.

```
> db.countryInfo.find().count()
250
>
```

7. With SQL one can retrieve a single record from a database with: SELECT* FROM countryInfo LIMIT 1; - With MongoDB we can write the same query by typing:

8. We can add an aggregate method count() in order to count the number of record that match a query. Here we want to know how many countries are listed in the region of Europe.

```
> db.countryInfo.find( {"region" : "Europe" } ).count()
53
>
```

9. MongoDB provides equivalents to the WHERE clause in SQL in the form of Query and Projection operators. Here we specify that a value of a certain key must be one of those in each list. Here we want to know how many countries there are in Europe and Africa regions together. Countries are also shown with pretty()

```
0
> db.countryInfo.find( {"region" : {"$in" : ["Europe", "Africa"]} }).count()
113
> db.countryInfo.find({"region" : {"$in" : ["Europe", "Africa"]}}).pretty()
{
        "_id" : ObjectId("618a294d73166f12b5705a63"),
        "id" : 2,
        "name" : "Aland Islands",
        "iso3" : "ALA",
        "iso2" : "AX",
        "numeric_code" : "248",
        "phone_code" : "+358-18",
        "capital" : "Mariehamn",
        "currency" : "EUR",
        "currency_symbol" : "€",
        "tld" : ".ax",
        "native" : "Åland",
        "region" : "Europe",
        "subregion" : "Northern Europe",
        "timezones" : [
                {
                        "zoneName" : "Europe/Mariehamn",
                        "gmtOffset" : 7200,
                        "gmtOffsetName" : "UTC+02:00",
                        "abbreviation" : "EET",
                        "tzName" : "Eastern European Time"
                }
        ],
        "translations" : {
                "kr" : "□ □ □  □ □ ",
                "br" : "Ilhas de Aland",
                "pt" : "Ilhas de Aland",
                "nl" : "Ålandeilanden",
                "hr" : "Ålandski otoci",
                "fa" : "□□□□□ □□□□",
                "de" : "Åland",
                "es" : "Alandia",
                "fr" : "Åland",
                "ja" : "□ □ □ □ □ □ □ ",
                "it" : "Isole Aland",
                "cn" : "□ □ □ □ "
```

10. We can combine two selection criteria with the operator $and:

```
> db.countryInfo.find({"$and" : [{"region" : "Europe"}, {"currency" : "EUR"}]}).count()
26
>
```

11. There are also methods for aggregation in MongoDB. Here distinct() is used to get all distinct countries in Europe with EUR currency:

```
[ db.countryInfo.distinct("name", {"$and" : [{"region": "Europe" }, {"currency": "EUR"}]}).sort()
        "Aland Islands",
        "Andorra",
        "Austria",
        "Belgium",
        "Cyprus",
        "Estonia",
        "Finland",
        "France",
        "Germany",
        "Greece",
        "Ireland",
        "Italy",
        "Kosovo",
        "Latvia",
        "Lithuania",
        "Luxembourg",
        "Malta",
        "Monaco",
        "Montenegro",
        "Netherlands",
        "Portugal",
        "San Marino",
        "Slovakia",
        "Slovenia",
        "Spain",
        "Vatican City State (Holy See)"
]
>
```

12. To show how many countries have latitude greater than 4.00000000, we can do following:

```
> db.countryInfo.find({"latitude" : {"$gt" : 4.00000000} }).count()
0
> db.countryInfo.find({"latitude" : {"$gt" : "4.00000000"}}).count()
78
>
```

Note the difference here when not using “” for the 4.0000000 – this is due to the data formatting on the source file.

13. Here we show the latitude in descending order and limit the results to one: