OCC2 MongoDB Report

Klink, Carl Lefebvre, Romain Matthews, Louis-Marie
Muller, Julie

March the 10th, 2025

Setup

Cloning the repository

We start our journey by simpling cloning our repository, which contains the original version of the dataset we are working on: `earthquakes_big.geojson.json`.

```
git clone git@github.com:Jlemlr/MongoDB.git
```

Preparing the dataset

First, as instructed, we will display one row of the original dataset. (We only formatted the first row for readability, no transformation of any kind was performed.)


```
{
    "type": "Feature",
    "properties": {
        "mag": 0.8,
        "place": "6km W of Cobb, California",
        "time": 1370259968000,
        "updated": 1370260630761,
        "tz": -420,
        "url": "http://earthquake.usgs.gov/earthquakes/eventpage/nc72001620",
        "detail": "http://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/
          nc72001620.geojson",
        "felt": null,
        "cdi": null,
        "mmi": null,
        "alert": null,
        "status": "AUTOMATIC",
        "tsunami": null,
        "sig": 10,
        "net": "nc",
```

```
        "code": "72001620",
        "ids": ",nc72001620,",
        "sources": ",nc,",
        "types": ",general-link,geoserve,nearby-cities,origin,phase-data,
        scitech-link,",
        "nst": null,
        "dmin": 0.00898315,
        "rms": 0.06,
        "gap": 82.8,
        "magType": "Md",
        "type": "earthquake"
    },
    "geometry": {
        "type": "Point",
        "coordinates": [
            -122.7955,
            38.8232,
            3
        ]
    },
    "id": "nc72001620"
}
```


We will also make sure to keep a note of the number of lines of the original dataset.

```
root@6d7fb3f3b522:/workspaces/MongoDB# wc -l earthquakes_big.geojson.json
7668 earthquakes_big.geojson.json
```

There are are 7668 newline characters. In other words, there are 766**9** lines. :)

Next, we load our original dataset `earthquakes_big.geojson.json` into a Pandas DataFrame and transform it into another DataFrame called `merged_df`, the same we used for the Cassandra assignment. For this, we will use `prepare_dataset` function defined in `utils.py`. This is the same preparation code than we used in the Cassandra assignment, with a few improvements. (Which are documented in the file `utils.py`.) **The code that was used is included in the appendices of this report (at the very end of this document).**

Before anything, we install the Python librairies needed by our code:

```
pip install pandas
```

Then, we create the `merged_df` DataFrame.

```
merged_df = utils.prepare_dataset(utils.DATASET_PATH)
```

The resulting `merged_df` dataset has the correct types applied. Unneeded (*i.e.* static) columns are removed. Timestamps are converted to the right format and

are made timezone-independent. Nested properties are flattened, etc.

We can see that there are duplicates of the same magnitude: `mb_Lg` and `MbLg` stand for the same thing, as for `ML` and `Ml` and `ml`, and finally, `mb` and `Mb`. Those were fixed too.

```
df['magtype'] = df['magtype'].str.lower().str.replace('_', '')
df['magtype'].unique()
```

We then perform one additional transformation on it to make use of the `location` type specific to MongoDB, as specified in the documentation.

```
def transform_to_geojson(df):
    """Transform the location for MongoDB."""

    df['location'] = df['coordinates'].apply(lambda coords: {
      'type': 'Point',
      'coordinates': coords[:2]
    })

    # Delete `coordinates` column
    df.drop('coordinates', axis=1, inplace=True)

    return df

df = transform_to_geojson(merged_df)
```

We do the same for our dates to be correctly picked up as `ISODate` by `mongoimport`. They are required to be in a specific format.

```
def transform_to_mongodbdate(datetime):
    if datetime is not None:
        return {
            "$date": datetime,
        }
    else:
        return None

df['time'] = df['time'].apply(transform_to_mongodbdate)

df['updated'] = df['updated'].apply(transform_to_mongodbdate)
```

We then save our file:

```
df.to_json('earthquakes_transformed.json', orient='records', lines=True)
```

We now have a nicely formatted JSON file that is correctly formatted for importing into MongoDB!

```
{
    "id": "nc72001620",
```

```
"mag": 0.8,
"place": "6km W of Cobb, California",
"time": {
    "$date": 1370259968000
},
"updated": {
    "$date": 1370260630761
},
"url": "http:\/\/earthquake.usgs.gov\/earthquakes\/eventpage\/nc72001620",
"detail": "http:\/\/earthquake.usgs.gov\/earthquakes\/feed\/v1.0\/detail\/
  nc72001620.geojson",
"felt": null,
"cdi": null,
"mmi": null,
"alert": null,
"status": "AUTOMATIC",
"tsunami": false,
"sig": 10,
"net": "nc",
"code": "72001620",
"ids": [
    "nc72001620"
],
"sources": [
    "nc"
],
"types": [
    "general-link",
    "geoserve",
    "nearby-cities",
    "origin",
    "phase-data",
    "scitech-link"
],
"nst": null,
"dmin": 0.00898315,
"rms": 0.06,
"gap": 82.8,
"magtype": "md",
"type": "earthquake",
"location": {
    "type": "Point",
    "coordinates": [
        -122.7955,
        38.8232
    ]
```

```
    },
    "depth": 3.0
}
```

Loading the JSON into MongoDB

Creating the server

We will use Docker to create and run a local MongoDB server, as well as a shell to access and interact with the database.

For this, we create and run a container with that image.

```
docker run --name mongodb -d -p 27017:27017 mongo
```

We now have a MongoDB container (which is like a virtual MongoDB server) running!

MongoDB Compass

Compass is a tool that provides a GUI for interacting with a MongoDB database. It provides interesting features such as a graphical interface (as previously stated), as well as access to the Mongo shell, index optimisation features, natural language queries, and many more.

If MongoDB is installed, here are the steps to install it:

1. Start Compass
2. Create a connection (`mongodb://localhost:27017/`). If this doesn't work make sure your container was created with the name `mongodb` withh the port `27017` open.
3. Create a new database and open the `earthquakes_transformed.json.json` file from the github repository you downloaded.

You can now directly write your queries in the UI of `MongoDb` Compass.

Importing the JSON file into MongoDB

We now need to connect to our MongoDB standalone cluster, *i.e.* server, to create the database and import the transformed `earthquakes_transformed.json` dataset.

To do that, from the MongoDB container, we run: `mongosh` to run the shell.

```
root@279d3593702c:/# mongosh
Current Mongosh Log ID: 67d0d9e85ba9a55c1051e943
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true
  &serverSelectionTimeoutMS=2000&appName=mongosh+2.4.0
Using MongoDB:          8.0.5
Using Mongosh:          2.4.0
```

```
For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/
```

This is useful to us, as we get the parameters necessary to connect to our MongoDB server. This output is followed by non-critical warnings that relate to performance (because of the filesystem) and security considerations, details into which we won't delve as this is only a local database used for testing.

Now that we know how to connect to our server, we can exit the shell by typing `exit`.

We will now run `mongoimport` to import the our JSON into a collection of a new database in our server.

```
root@279d3593702c:/root# mongoimport --uri mongodb://127.0.0.1:27017/
  ?directConnection=true --db=dibi --collection earthquakes --type json
  --file earthquakes_transformed.json
2025-03-15T18:23:54.027+0000    connected to: mongodb://127.0.0.1:27017/
  ?directConnection=true
2025-03-15T18:23:54.398+0000    7669 document(s) imported successfully.
  0 document(s) failed to import.
```

Notice we used `mongodb://127.0.0.1:27017/?directConnection=true` because this is what appeared in the log when we ran `mongosh`.

So, from reading the logs, all 7669 earthquakes were imported.

Let's now connect back to our server and our database, and check the number of documents in our `earthquakes` collection. (And also check that the `dibi` database and `earthquakes` collection were both successfully created by the `mongoimport` command!)

```
test> show dbs
TP      22.21 MiB
admin   40.00 KiB
config  72.00 KiB
dibi     1.37 MiB
local   72.00 KiB
test     1.37 MiB
test> use dibi
switched to db dibi
dibi> show collections
earthquakes
dibi> db.earthquakes.countDocuments()
7669
```

Okay, so our `dibi` database was successfully created. Our collection `earthquakes` was also created and does contain our 7669 earthquakes as expected!

Let's now remove the `test` database for proper housekeeping.

```
dibi> use test
switched to db test
test> db.dropDatabase()
{ ok: 1, dropped: 'test' }
```

This was because, at first, we imported the documents in the `test` collection.

We're all set, the setup is completed!

Queries

Easy queries

0 Counting the number of earthquakes To make sure we did not loose any records in the whole "preparation and loading" process, we will count, one last time, the number of records in the collection.

```
dibi> db.earthquakes.countDocuments()
7669
```

All good, we still have 7669 records!

1 Retrieve the first earthquake stored in the collection

```
dibi> db.earthquakes.find().limit(1)
[
  {
    _id: ObjectId('67d15c3057df22624788ad13'),
    id: 'ak10729211',
    mag: 1.2,
    place: '97km WSW of Cantwell, Alaska',
    time: ISODate('2013-06-03T11:41:03.000Z'),
    updated: ISODate('2013-06-03T11:48:50.916Z'),
    url: 'http://earthquake.usgs.gov/earthquakes/eventpage/ak10729211',
    detail: 'http://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/
      ak10729211.geojson',
    felt: null,
    cdi: null,
    mmi: null,
    alert: null,
    status: 'AUTOMATIC',
    tsunami: false,
    sig: 22,
    net: 'ak',
    code: '10729211',
    ids: [ 'ak10729211' ],
    sources: [ 'ak' ],
    types: [
```

```
      'general-link',
      'geoserve',
      'nearby-cities',
      'origin',
      'tectonic-summary'
    ],
    nst: null,
    dmin: null,
    rms: 0.83,
    gap: null,
    magtype: 'Ml',
    type: 'earthquake',
    location: { type: 'Point', coordinates: [ -150.8246, 63.1584 ] }
  }
]
```

2 Get the place of earthquakes with magnitude greater than 4.0 We imposed a limit of 2 for the document to remain readable.

```
dibi> db.earthquakes.find({ mag: { $gt: 4.0 } }, { place: 1, _id: 0 }).limit(2)
[
  { place: '155km WSW of Panguna, Papua New Guinea' },
  { place: "251km E of Kuril'sk, Russia" }
]
```

3 Find the magnitude of earthquakes that occurred in California The complete output being very long, only the first five rows are displayed here.

```
dibi> db.earthquakes.find({ place: /California/ }, { _id: 0, mag: 1 })
[
  { mag: 1.4 },
  { mag: 0.8 },
  { mag: 1 },
  { mag: 1.1 },
  { mag: 0.4 },
  ...
]
```

4 Find the top 5 strongest earthquakes

```
dibi> db.earthquakes.find({}, { mag: 1 }).sort({ mag: -1 }).limit(5)
...
[
  { _id: ObjectId('67d0dc0298be26e220e73c19'), mag: 8.3 },
  { _id: ObjectId('67d0dc0298be26e220e73ce9'), mag: 7.4 },
  { _id: ObjectId('67d0dc0298be26e220e7471f'), mag: 6.8 },
  { _id: ObjectId('67d0dc0298be26e220e73b55'), mag: 6.7 },
```

```
  { _id: ObjectId('67d0dc0298be26e220e74054'), mag: 6.4 }
]
```

5 Find the possible types of magnitudes

```
dibi> db.earthquakes.distinct("magtype")
[
  null,   'h',   'mb',
  'mblg', 'md',  'me',
  'ml',   'mt',  'mw',
  'mwb',  'mwp', 'mww'
]
```

For reference, here are their meanings:

Code	Meaning
h	Local magnitude (historical)
mb	Body-wave magnitude
mblg	Body-wave magnitude (long period)
md	Duration magnitude
me	Energy magnitude
ml	Local magnitude
mt	Tectonic magnitude
mw	Moment magnitude
mwb	Broadband moment magnitude
mwp	Moment magnitude (preferred)
mww	Moment magnitude (worldwide)

6 List and order all earthquakes that caused a tsunami The query:

```
db.earthquakes.find(
  {
    tsunami: true
  },
  {
    _id: 0,
    place: 1,
    mag: 1,
    magtype: true,
    _id: 0
  }
)
```

The result:

```
[
  { mag: 6.2, place: '22km SSE of Buli, Taiwan', magtype: 'mww' },
```

```
  {
    mag: 4.4,
    place: '87km NNE of Punta Cana, Dominican Republic',
    magtype: 'md'
  },
  {
    mag: 4.2,
    place: '32km SE of Boca de Yuma, Dominican Republic',
    magtype: 'md'
  },
  { mag: 4.8, place: '5km W of Isla Vista, California', magtype: 'mw' },
  { mag: 6.7, place: 'Sea of Okhotsk', magtype: 'mt' },
  { mag: 8.3, place: 'Sea of Okhotsk', magtype: 'mw' },
  {
    mag: 5.7,
    place: '11km WNW of Greenville, California',
    magtype: 'mw'
  },
  { mag: 6.3, place: '90km NW of Nuku`alofa, Tonga', magtype: 'mw' },
  { mag: 7.4, place: '282km SW of Vaini, Tonga', magtype: 'mw' },
  {
    mag: 4.7,
    place: '60km NNE of Stoney Ground, Anguilla',
    magtype: 'mb'
  },
  { mag: 6.4, place: 'Off the coast of Aisen, Chile', magtype: 'mw' },
  { mag: 4.1, place: '71km N of Hatillo, Puerto Rico', magtype: 'md' },
  { mag: 4.4, place: '44km SW of Homer, Alaska', magtype: 'ml' },
  {
    mag: 4,
    place: '10km S of Rancho Palos Verdes, California',
    magtype: 'mw'
  },
  {
    mag: 4.4,
    place: '124km SSE of Old Iliamna, Alaska',
    magtype: 'ml'
  }
]
```

7 Get earthquakes that have a "felt" value specified Because the `felt` value is often null, we will check the earthquakes for whom the `felt` value is provided.

```
dibi> db.earthquakes.find({ felt: { $ne: null } }, { _id: 0, felt: 1, mag: 1 })
[
```

```
  { mag: 2.6, felt: 6 },  { mag: 5.6, felt: 7 },
  { mag: 4.5, felt: 2 },  { mag: 2, felt: 2 },
  { mag: 4.5, felt: 6 },  { mag: 5.1, felt: 1 },
  { mag: 2.5, felt: 1 },  { mag: 2.7, felt: 3 },
  { mag: 1.7, felt: 6 },  { mag: 6.2, felt: 53 },
  { mag: 1.98, felt: 3 }, { mag: 2.5, felt: 21 },
  { mag: 1.1, felt: 1 },  { mag: 3.4, felt: 9 },
  { mag: 3.4, felt: 4 },  { mag: 4.8, felt: 1 },
  { mag: 1.8, felt: 2 },  { mag: 3.2, felt: 3 },
  { mag: 3.4, felt: 0 },  { mag: 5.6, felt: 35 }
]
```

Interesting to notice the correlation between the manitude and the `felt` value, although it is sometimes very far off!

Complex queries

1 Count the daily number of earthquakes between the 4th and the 15th of June, 2013 Let's start with the number of earthquakes per day, between the 4th and the 15th (not included):

```
db.earthquakes.aggregate([
  {
    $match: {
      time: {
        $gte: new ISODate('2013-05-04T00:00:00.000Z'),
        $lt: new ISODate('2013-05-15T00:00:00.000Z')
      }
    }
  },
  {
    $project: {
      day: {
        $dayOfMonth: "$time"
      },
    }
  },
  {
    $group: {
      _id: "$day",
      count: {
        $sum: 1
      }
    }
  },
  {
    $sort: {
```

```
      _id: 1
    }
  },
  {
    $project: {
      _id: 0,
      date: {
        $concat: [
          {
            $toString: "$_id"
          },
          " of May"
        ]
      },
      count: 1
    }
  },
])
```

We get:

```
[
  { count: 86, date: '4 of May' },
  { count: 224, date: '5 of May' },
  { count: 262, date: '6 of May' },
  { count: 335, date: '7 of May' },
  { count: 289, date: '8 of May' },
  { count: 303, date: '9 of May' },
  { count: 306, date: '10 of May' },
  { count: 265, date: '11 of May' },
  { count: 275, date: '12 of May' },
  { count: 241, date: '13 of May' },
  { count: 246, date: '14 of May' }
]
```

So, it seems like the number stays roughly the same. The very small number for the 4th of May is because this is when our dataset begins.

2 Get the average magnitude of earthquakes in California Let's find the average magnitude of earthquakes located in California.

```
db.earthquakes.aggregate([
  { $match: { place: /California/ } },
  { $group: { _id: null, avg_magnitude: { $avg: "$mag" } } }
])
```

3 Find the closest earthquake to a given location (latitude: 38.8232, longitude: -122.7955) To filter queries based on the location, we first need to create an index on it:

```
dibi> db.earthquakes.createIndex({location: "2dsphere"})
location_2dsphere
```

Only then can we make our query:

```
db.earthquakes.find(
  {
    location: {
      $near: {
        $geometry: {
          type: "Point",
          coordinates: [48.86, 2.35]
        }
      }
    }
  },
  { _id: 0, id: 1, place: 1, location: 1, url: 1 }
).limit(1)
```

```
[
  {
    id: 'usc000gntf',
    place: '135km SSE of Burum, Yemen',
    url: 'http://earthquake.usgs.gov/earthquakes/eventpage/usc000gntf',
    location: { type: 'Point', coordinates: [ 49.561, 13.279 ] }
  }
]
```

4 Number of earthquakes per hour Do earthquakes happen more at a certain time of the day? Is there a relationship between the hour and the likelihood of having an earthquake?

```
db.earthquakes.aggregate([
  {
    $match: {
      time: {
        $ne: null
      }
    }
  },
  {
    $project: {
```

```
      hour: { $hour: "$time" }
    }
  },
  {
    $group: {
      Total: { $sum: 1 },
      _id: "$hour"
    }
  },
  {
    $sort: { _id: 1 }  // Sort by hour
  },
  {
    $project: {
      _id: 0,
      Hour: {
        $concat: [
          {
            $toString: "$_id"
          },
          "H"
        ]
      },
      Total: 1
    }
  }
])
```

This allows us to get the total number of earthquakes per hour:

```
[
  { Total: 312, Hour: '0H' },
  { Total: 297, Hour: '1H' },
  { Total: 286, Hour: '2H' },
  { Total: 320, Hour: '3H' },
  { Total: 365, Hour: '4H' },
  { Total: 338, Hour: '5H' },
  { Total: 339, Hour: '6H' },
  { Total: 330, Hour: '7H' },
  { Total: 335, Hour: '8H' },
  { Total: 335, Hour: '9H' },
  { Total: 337, Hour: '10H' },
  { Total: 341, Hour: '11H' },
  { Total: 351, Hour: '12H' },
  { Total: 312, Hour: '13H' },
  { Total: 275, Hour: '14H' },
  { Total: 299, Hour: '15H' },
```

```
  { Total: 309, Hour: '16H' },
  { Total: 309, Hour: '17H' },
  { Total: 329, Hour: '18H' },
  { Total: 331, Hour: '19H' },
  { Total: 268, Hour: '20H' },
  { Total: 320, Hour: '21H' },
  { Total: 274, Hour: '22H' },
  { Total: 356, Hour: '23H' }
]
```

As expected, we see little to no correlation between the hour and the total number of earthquakes, which confirms existing studies.

Hard Queries

1 Localisation of earthquakes From Pandas, we can quickly get a visualisation of earthquakes.

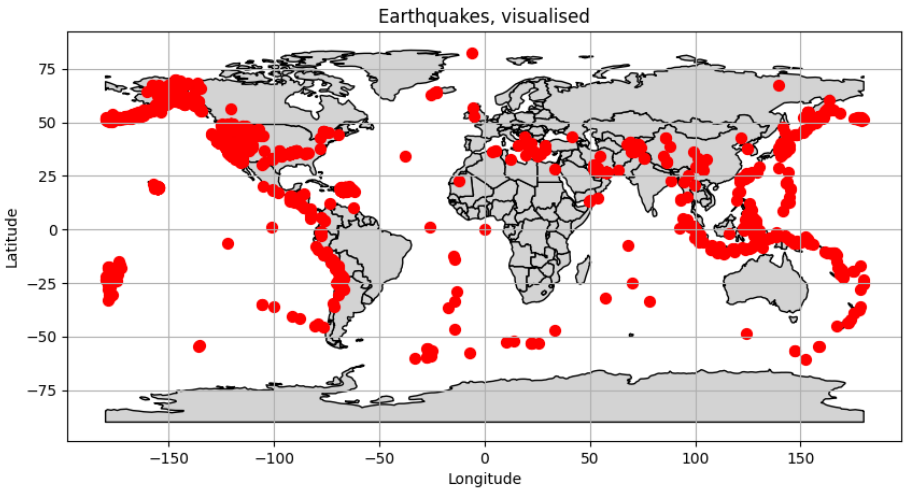

Figure 1: Earthquakes locations

So the majority of earthquakes seem to emanate from the west coast of the US, or rather, the majority of recorded earthquakes present in our original dataset.

This might be because the original dataset only reports earthquakes that were detected in the US. In other words, this might mean that foreign earthquakes need to have a higher magnitude to appear in this dataset. To verify this theory, we will compute the average magnitude of earthquakes depending on how far they are from the Menlo Park, the location of CSIN, California Integrated Seismic Network.

Menlo Park coordinates are: 37.4530° N, 122.1817° W.

```
db.earthquakes.aggregate([
  {
    $geoNear: {
      near: {
        type: "Point",
        coordinates: [122.1817, 37.4530]
      },
      distanceField: "distance",
      spherical: true
    }
  },
  {
    $project: {
      _id: 0,
      mag: 1,
      distance: {
        $round: [
          {
            $divide: [
              "$distance",
              1000000
            ]
          }
        ]
      }
    }
  },
  {
    $group: {
      _id: "$distance",
      mean: { $avg: "$mag" },
      count: { $sum: 1 }
    }
  },
  {
    $project: {
      _id: 0,
      distance: "$_id",
      mean: 1,
      count: 1
    }
  },
  {
    $sort: {
      distance: 1
    }
```

```
  }
])
```

We also included the count of earthquakes to make sure we had a significant number to compute the average.

```
[
  { mean: 4.8, count: 2, distance: 0 },
  { mean: 4.6, count: 10, distance: 1 },
  { mean: 4.658823529411765, count: 51, distance: 2 },
  { mean: 4.831297709923664, count: 131, distance: 3 },
  { mean: 4.512765957446808, count: 47, distance: 4 },
  { mean: 2.8312925170068026, count: 294, distance: 5 },
  { mean: 1.6407451212300415, count: 1691, distance: 6 },
  { mean: 1.747196261682243, count: 214, distance: 7 },
  { mean: 2.5025316455696203, count: 158, distance: 8 },
  { mean: 1.330056967572305, count: 2282, distance: 9 },
  { mean: 1.0683776824034334, count: 2330, distance: 10 },
  { mean: 2.2737288135593223, count: 118, distance: 11 },
  { mean: 4.5, count: 5, distance: 12 },
  { mean: 4.469565217391304, count: 24, distance: 13 },
  { mean: 2.8826446280991735, count: 242, distance: 14 },
  { mean: 4.627272727272727, count: 11, distance: 15 },
  { mean: 4.907692307692307, count: 13, distance: 16 },
  { mean: 4.79047619047619, count: 21, distance: 17 },
  { mean: 4.705, count: 20, distance: 18 },
  { mean: 4.32, count: 5, distance: 19 }
]
```

What we observe is puzzling at first, it seems like the average magnitude is at its highest when it is recorded very close to California and when it is recorded very far (19,000+ km), and we might assume this is because there is an area very prone to powerful earthquakes located 19,000km away from California.

However, when we look at the count, we understand the relationship better: there is a visible correlation between the number of recorded earthquakes and the average magnitude. The higher the former, the lower the latter. This is becasue there are probably areas outside California that are better equipped to record earthquakes. This explains why we have many earthquakes between 5,000km away and and 11,000km. Since these places record many more earthquakes, they probably record earthquakes with a lower magnitude too, which explains why the average magnitude is lower for those places.

Appendices

utils.py

This is the code that was used to prepare the original `earthquakes_big.geojson.json` dataset. It is very similar to the code we used in the Cassandra assignment, except it is more efficient and performs additional transformations.

What it does is:

- It loads the original dataset into a Python object (more precisely, a Pandas DataFrame object);
- It removes columns that do not bring any information (*i.e.* static columns);
- It applies the correct types.
- It transforms timezone-dependent timestamps into proper timestamps;
- It formats values correctly and applies the corresponding types.

```
"""Utility module for preparing the original JSON earthquakes dataset.

This module was taken from the previous Cassandra report and provides functions
to prepare the `earthquakes_big.geojson.json` dataset in several ways.
"""

import pandas as pd
import numpy as np

def prepare_dataset(df_path) -> pd.DataFrame:
    """Prepare the original dataset file named 'merged_df'.

    This method performs several improvements on the original JSON earthquakes
    dataset, such as applying the correct types, removing unneeded (*i.e.*
    static) columns, fixing the timestamps incorrectly encoded, etc."""

    df = pd.read_json(df_path, lines=True)

    df.drop('type', axis=1, inplace=True) # Contains one unique value

    df_properties = pd.json_normalize(df['properties'])

    # Format types, sources and ids using correct array notation
    df_properties['types'] = df_properties['types'].apply(lambda x: x[1:-1].split(','))
    df_properties['sources'] = df_properties['sources'].apply(lambda x: x[1:-1].split(','))
    df_properties['ids'] = df_properties['ids'].apply(lambda x: x[1:-1].split(','))

    # Convert TZ-aware timestamps
    df_properties['time'] = df_properties.apply(lambda t: pd.Timestamp(
      t['time'], unit='ms', tz=t['tz']) if not np.isnan(t['time']) else None,
```

```
  axis=1
)
df_properties['updated'] = df_properties.apply(lambda t: pd.Timestamp(
  t['updated'], unit='ms', tz=t['tz']) if not np.isnan(t['updated']) else None,
  axis=1
)
df_properties.drop('tz', axis=1, inplace=True)

# There are some duplicate values in `magType` column but formatted differently.
# So let's remove the duplicate values from the `magType` column.
df_properties['magType'] = df_properties['magType'].str.lower().str.replace('_', '')

# If it caused a tsunami (convert to Boolean correct format)
df_properties['tsunami'] = df_properties['tsunami'] == 1

df_geometry = pd.json_normalize(df['geometry'])
df_geometry.drop('type', axis=1, inplace=True) # Contains one unique value

# Concatenate properties and geometry
merged_df = pd.concat(
  [
    df.drop(['properties', 'geometry'], axis=1),
    df_properties,
    df_geometry
  ],
  axis=1
)

merged_df.rename(str.lower, axis='columns', inplace=True)

str_columns = [
  'alert',
  'code',
  'detail',
  'id',
  'magtype',
  'place',
  'net',
  'url',
  'status',
  'type'
]
merged_df[str_columns] = merged_df[str_columns].astype(pd.StringDtype())

return merged_df
```

```
DATASET_PATH = 'earthquakes_big.geojson.json'
```