Report - template

Assignment 3 - MongoDB

The report does not have to be longer than 1-2 pages, excluding screenshots/images.

Group: 51
Students: Tore Fossland

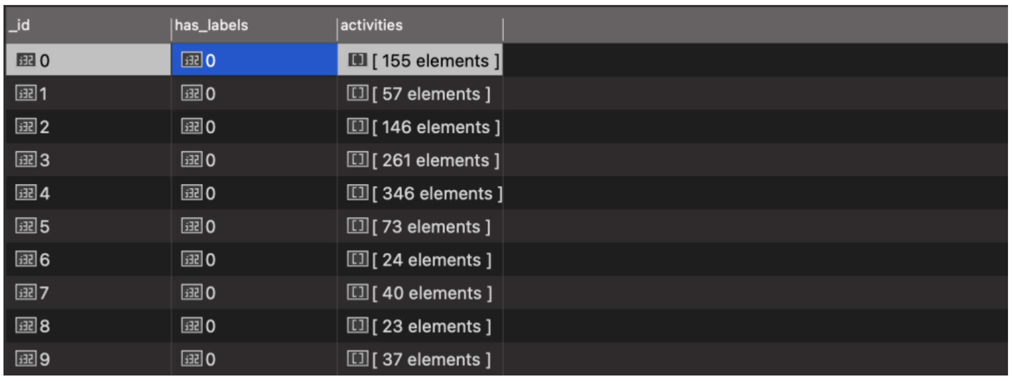
Introduction

Assignment 3 in TDT4225 consists of solving some database tasks using a MongoDB and coding in python. The open source Geolife GPS Trajectory dataset is my data source, which is a recording of a broad range of user's outdoor activities. The first part of the assignment was setting up the database itself using docker a container. Then I defined and created a suitable schema according to the mongoDB documentation, and lastly I cleaned and inserted the data from CSV files. Secondly I made queries to answer a set of given questions. I was however not able to complete all of them in time. Some code was also provided to serve as a starting point.

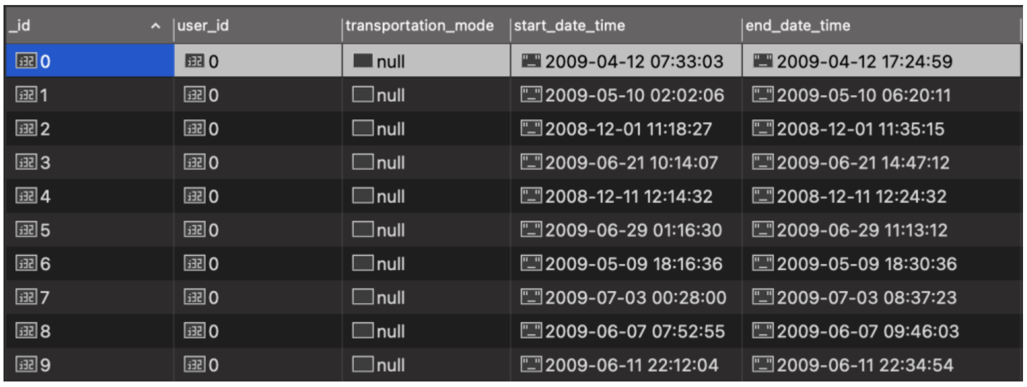
GitHub repo: https://github.com/ToreFossland/TDT4225

Results

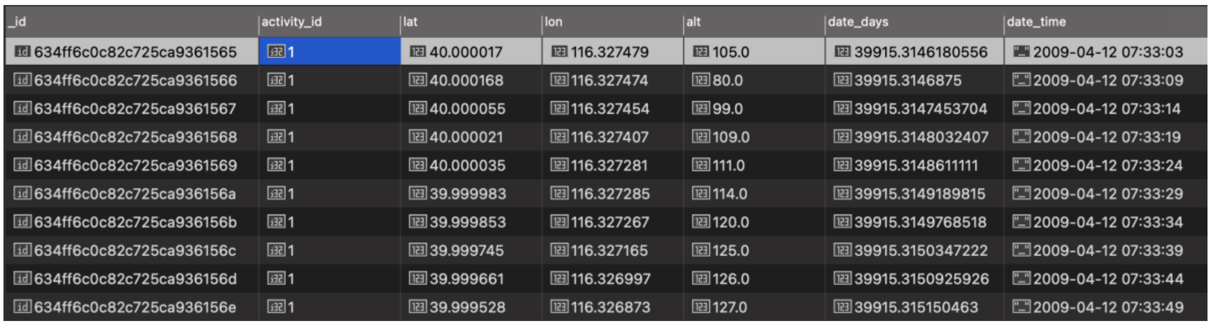
First ten entries in the 3 tables users, activities and trackpoints.



_id	has_labels	activities
0	0	[155 elements]
1	0	[57 elements]
2	0	[146 elements]
3	0	[261 elements]
4	0	[346 elements]
5	0	[73 elements]
6	0	[24 elements]
7	0	[40 elements]
8	0	[23 elements]
9	0	[37 elements]



_id	user_id	transportation_mode	start_date_time	end_date_time
0	0	null	2009-04-12 07:33:03	2009-04-12 17:24:59
1	0	null	2009-05-10 02:02:06	2009-05-10 06:20:11
2	0	null	2008-12-01 11:18:27	2008-12-01 11:35:15
3	0	null	2009-06-21 10:14:07	2009-06-21 14:47:12
4	0	null	2008-12-11 12:14:32	2008-12-11 12:24:32
5	0	null	2009-06-29 01:16:30	2009-06-29 11:13:12
6	0	null	2009-05-09 18:16:36	2009-05-09 18:30:36
7	0	null	2009-07-03 00:28:00	2009-07-03 08:37:23
8	0	null	2009-06-07 07:52:55	2009-06-07 09:46:03
9	0	null	2009-06-11 22:12:04	2009-06-11 22:34:54

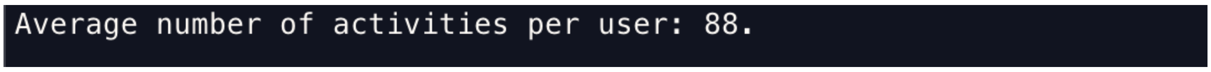


_id	activity_id	lat	lon	alt	date_days	date_time
634ff6c0c82c725ca9361565	1	40.000017	116.327479	105.0	39915.3146180556	2009-04-12 07:33:03
634ff6c0c82c725ca9361566	1	40.000168	116.327474	80.0	39915.3146875	2009-04-12 07:33:09
634ff6c0c82c725ca9361567	1	40.000055	116.327454	99.0	39915.3147453704	2009-04-12 07:33:14
634ff6c0c82c725ca9361568	1	40.000021	116.327407	109.0	39915.3148032407	2009-04-12 07:33:19
634ff6c0c82c725ca9361569	1	40.000035	116.327281	111.0	39915.3148611111	2009-04-12 07:33:24
634ff6c0c82c725ca936156a	1	39.999983	116.327285	114.0	39915.3149189815	2009-04-12 07:33:29
634ff6c0c82c725ca936156b	1	39.999853	116.327267	120.0	39915.3149768518	2009-04-12 07:33:34
634ff6c0c82c725ca936156c	1	39.999745	116.327165	125.0	39915.3150347222	2009-04-12 07:33:39
634ff6c0c82c725ca936156d	1	39.999661	116.326997	126.0	39915.3150925926	2009-04-12 07:33:44
634ff6c0c82c725ca936156e	1	39.999528	116.326873	127.0	39915.315150463	2009-04-12 07:33:49

Counts of users, activities and trackpoints in the dataset after filtering.

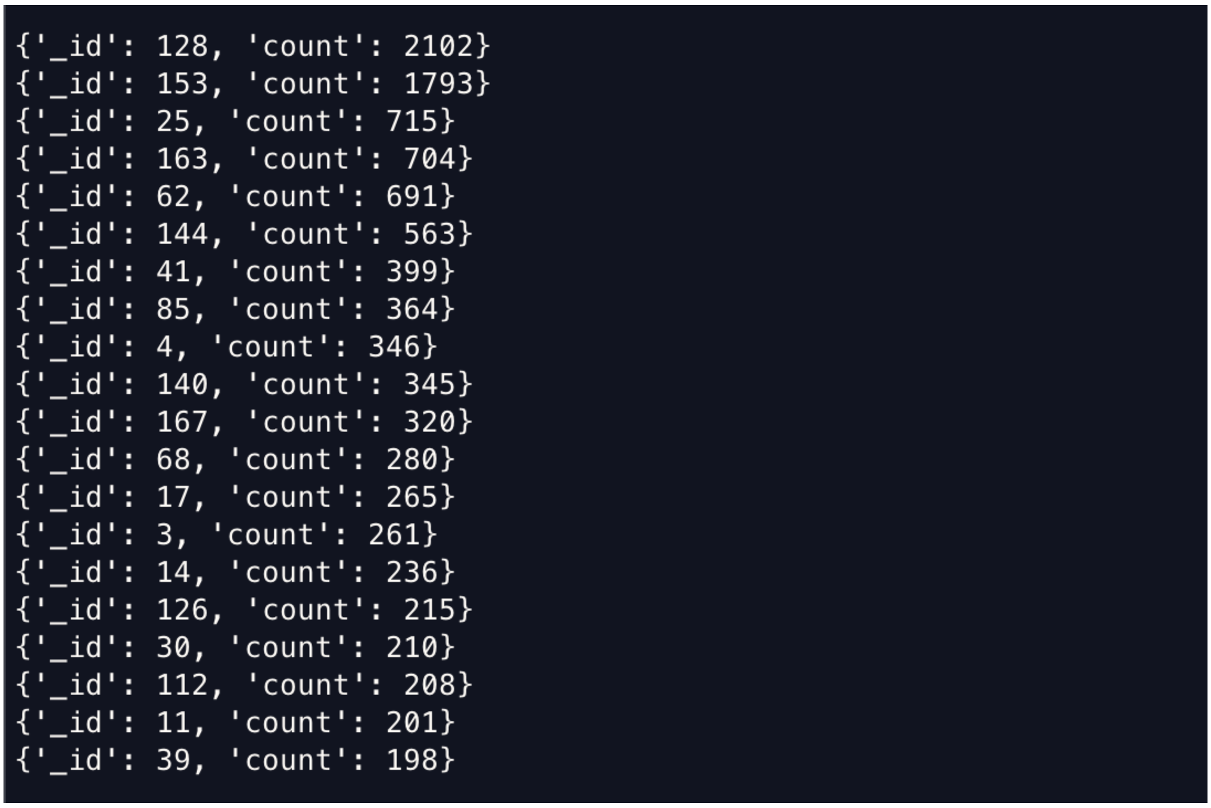
```
User count: 182, activity count: 16048, trackpoint count: 9681756.
```

Average number of activities per user.



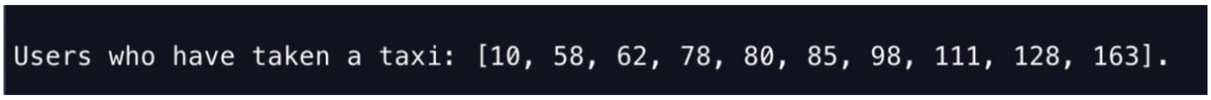
```
Average number of activities per user: 88.
```

The top 20 users with the highest number of activities by user id.



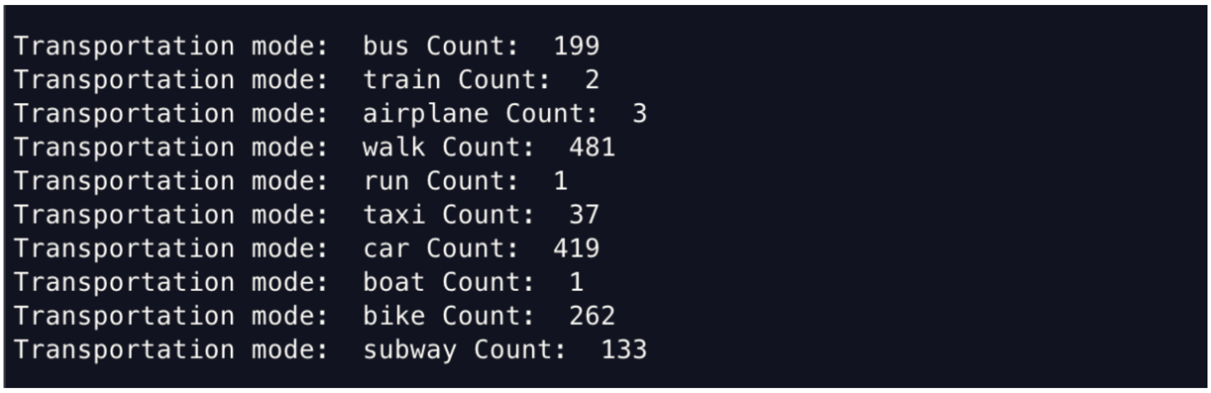
```
{'_id': 128, 'count': 2102}
{'_id': 153, 'count': 1793}
{'_id': 25, 'count': 715}
{'_id': 163, 'count': 704}
{'_id': 62, 'count': 691}
{'_id': 144, 'count': 563}
{'_id': 41, 'count': 399}
{'_id': 85, 'count': 364}
{'_id': 4, 'count': 346}
{'_id': 140, 'count': 345}
{'_id': 167, 'count': 320}
{'_id': 68, 'count': 280}
{'_id': 17, 'count': 265}
{'_id': 3, 'count': 261}
{'_id': 14, 'count': 236}
{'_id': 126, 'count': 215}
{'_id': 30, 'count': 210}
{'_id': 112, 'count': 208}
{'_id': 11, 'count': 201}
{'_id': 39, 'count': 198}
```

The highest number of activities by user id.



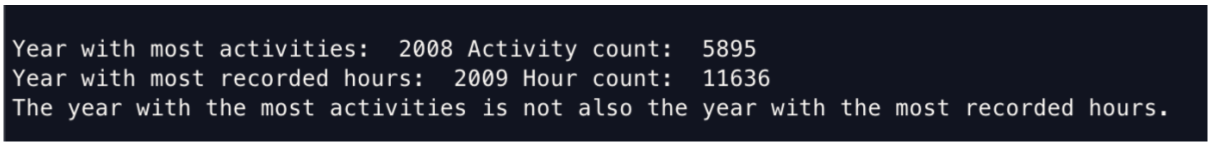
```
Users who have taken a taxi: [10, 58, 62, 78, 80, 85, 98, 111, 128, 163].
```

All users who have taken a taxi.



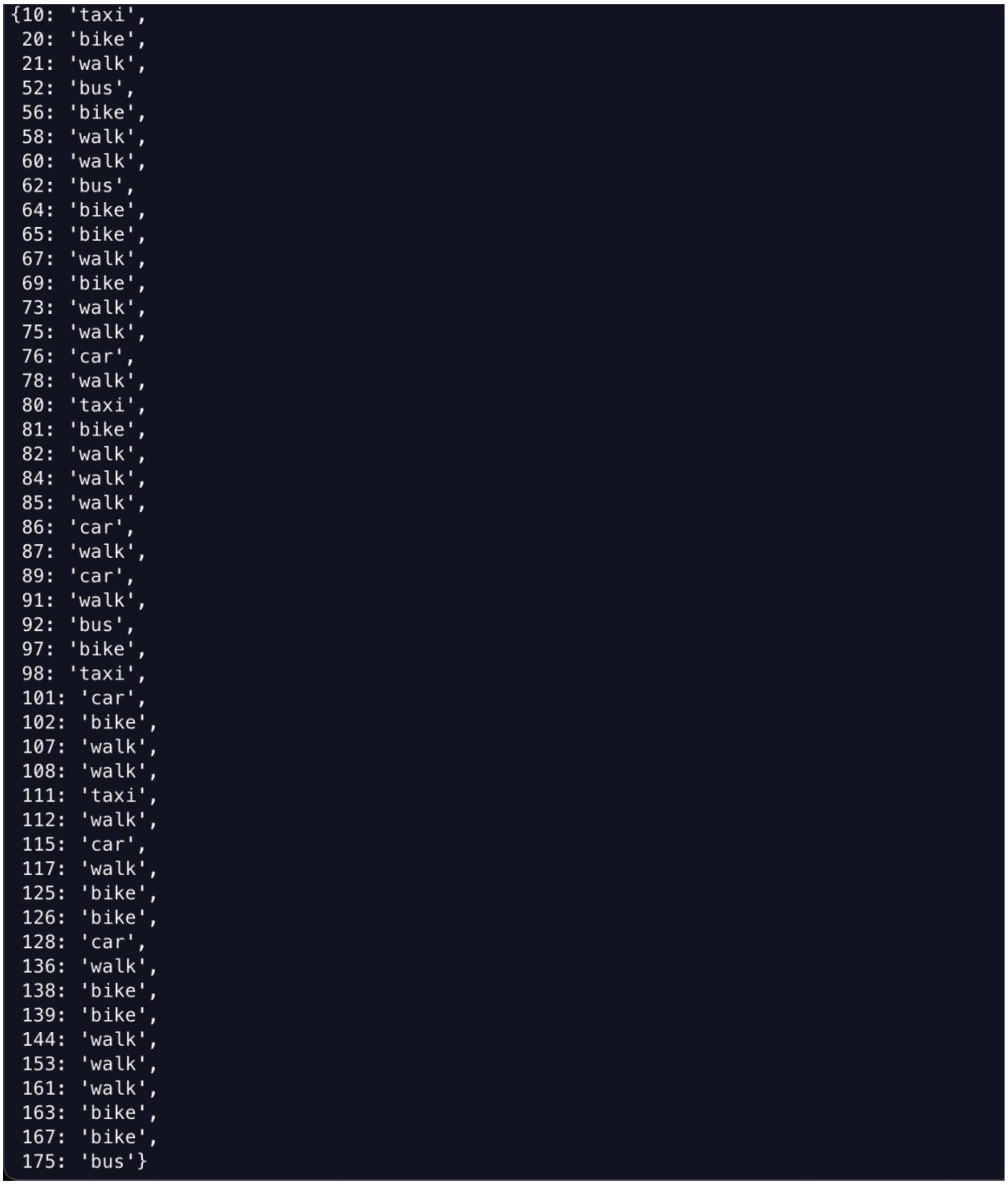
```
Transportation mode:  bus Count:  199
Transportation mode:  train Count:  2
Transportation mode:  airplane Count:  3
Transportation mode:  walk Count:  481
Transportation mode:  run Count:  1
Transportation mode:  taxi Count:  37
Transportation mode:  car Count:  419
Transportation mode:  boat Count:  1
Transportation mode:  bike Count:  262
Transportation mode:  subway Count:  133
```

We first have the year with the most activities, and then the year with the most recorded hours. As you can see from the screenshot they are not the same.



```
Year with most activities:  2008 Activity count:  5895
Year with most recorded hours:  2009 Hour count:  11636
The year with the most activities is not also the year with the most recorded hours.
```

All users who have registered transportation_mode and their most used transportation_mode.



```
{10: 'taxi',
 20: 'bike',
 21: 'walk',
 52: 'bus',
 56: 'bike',
 58: 'walk',
 60: 'walk',
 62: 'bus',
 64: 'bike',
 65: 'bike',
 67: 'walk',
 69: 'bike',
 73: 'walk',
 75: 'walk',
 76: 'car',
 78: 'walk',
 80: 'taxi',
 81: 'bike',
 82: 'walk',
 84: 'walk',
 85: 'walk',
 86: 'car',
 87: 'walk',
 89: 'car',
 91: 'walk',
 92: 'bus',
 97: 'bike',
 98: 'taxi',
 101: 'car',
 102: 'bike',
 107: 'walk',
 108: 'walk',
 111: 'taxi',
 112: 'walk',
 115: 'car',
 117: 'walk',
 125: 'bike',
 126: 'bike',
 128: 'car',
 136: 'walk',
 138: 'bike',
 139: 'bike',
 144: 'walk',
 153: 'walk',
 161: 'walk',
 163: 'bike',
 167: 'bike',
 175: 'bus'}
```

Discussion

After reading up on MongoDB schema design from the assignment description, I decided to use an array of references to the “many” objects in the one-to-many relation between user and activity in the user collection. This made some queries a lot easier. I did however not do this for the activity or trackpoints collections because of time and feasibility constraints. Having an array of trackpoints references for each activity would become too big.

I learnt how to set up my MongoDB database locally from the instructions in the problem text, which was placed in a docker container. Cleaning and placing the data into CSV before insertion made the process a lot easier to manage, and gave me a good overview of my data.

During the initial part of the assignment I struggled with knowing which data the database contained at any given time, like in assignment 2. Since DBeaver is not compatible with MongoDB I used Studio 3T, which provides the same functionality as DBeaver. This gave me a full overview of which data was inserted into the database and made troubleshooting much easier.

I did not end up using a lot of git since I completed the assignment by myself. I also did not have much use for issues etc. since the scope of the assignment was limited.

MySQL vs MongoDB

Discuss the differences between MySQL and MongoDB (relational databases vs NoSQL databases). Which database did you prefer to solve these tasks? What are the pros and cons using one versus the other?

One of the biggest differences between relational databases vs NoSQL databases lies in the name. Where relational databases have different objects being related to each other, NoSQL objects are not. One can achieve similar functionality with NoSQL by adding extra arrays and references but it comes at a cost.

Another major difference is that relational databases require a predefined schema to decide the database structure before using it, while NoSQL can handle unstructured data using flexible schemas.

When deciding between the two choices one has to that scalability into consideration. Relational databases are vertically scalable, which means one has to increase the power of the one server our database is running on. This can get very expensive. NoSQL databases on the other hand, scale horizontally. This means you can upgrade your database by adding new servers instead of having just one. This makes NoSQL a better choice when you want to store very large and rapidly evolving data sets.

How the data is stored is also different between them. NoSQL databases store data in key-value pairs, like JSON, while SQL databases are table based organized into rows and columns as defined in the schema.

Which database did you prefer to solve these tasks?
I preferred MongoDB for setting up the database and inserting the big volumes of data. It was very intuitive and easy to work with in the beginning. I did however struggle quite a bit with the querying, because I have used this type of database very little earlier. So for this part of the assignment I preferred standard SQL querying, just based on previous experience. I do however think that MongoDB is superior for querying on big volumes of data since one can select only the data one needs.

What are the pros and cons using one versus the other?
MySQL:

- Pros:
  - More people have experience with it.
  - ACID compliance.
  - Does not require much code.

- Cons:
  - Requires the use of predefined schemas, changing this can be very annoying after the database has been filled with data.
  - Can be harder to scale when dealing with big datasets.

NoSQL:

- Pros:
  - Flexible data models: One can create documents without a predefined schema.
  - Easier to scale databases by adding servers for storing large sets of data.
  - Faster queries because of not needing to perform joins etc.
- Cons:
  - Can be prone to data duplication.
  - Query language still less known than SQL.

Feedback

Optional - give us feedback on the task if you have any. The assignment is new this semester and we would love to improve if there were any problems.