# Assignment 3 – MongoDB

**Group**: 23

**Students**: Dries Costermans, Tibo Van den Eede, Marco Zoso

**Introduction**

In this assignment, we created collections, cleaned data, inserted data and fetched the data to answer some questions. We set up the MongoDB database which required cleaning and inserting the data into defined tables. Next, we wrote queries to the database to gain knowledge of the dataset. Tibo did the data insertion and query 9. Marco and Dries did the other queries.

**Discussion**

We tried to follow the instructions as it was explained on the assignment sheet. We learned to work with a NoSQL database and the difference between working with a MySQL database.

**Structure collections and documents**

In short, the relation between User and Activity and the relation between Activity and TrackPoint where both modelled as "one-to-many" (as opposed to "one-to-few" and “one-to-squillions”) with two-way referencing. This means that a User document contains a list of the ids of its activities and each activity has an id of its corresponding user. Likewise, an Activity document contains a list of the ids of its trackpoints and each trackpoint contains an id of its corresponding activity.

One-to-many was chosen in both cases since it matches the cardinality of the two relations the best. For example, a user rougly has around 100 acitivities and usually not just a few and an activity can have no more than 2500 trackpoints and usually doesn't have just a few. So, therefore one-to-many was chosen, which means that the one side of the relation (e.g. User) contains a list of references to the ids of the many side of the relation (e.g. Activity).

On top of that, two-way referencing was also implemented. This means that the many side of the relation also contains a reference to the one side of the relation. Despite it introducing some redundancy, it gives the benefit of accessing the one side of the relation via the many side, which was useful in the queries. Besides that, two-way referencing has the disadvantage of having to update both sides of the relation when making changes, but in our use case there is only need to insert it once and never change anything. So, this is not a big concern.

**MySQL vs MongoDB**

The main difference is that MySQL supports the relational database model and MongoDB supports the document based database model. In this way MySQL stores data as unordered sets of tuples in a table, while MongoDB stores its data in documents. These documents have a more flexible and potentially complex structure (such as nested values), while MySQL just has sets of tuples which all have to satisfy to the same conditions.

On the other hand, MySQL has the advantage of making joins between tables more easy to perform. In MongoDB you have to think about which data to store in which collection to make joins relatively easy to perform for future queries. This last point is a disadvantage of MongoDB that we experienced: in MySQL you don’t have to worry much about how to model one-to-n relationships, while in MongoDB there are many different ways to model these one-to-n relationships for which you have to deliberate on your use case which might be a good fit.

Besides that, MySQL is better for smaller amounts of data, while MongoDB allows the user to handle bigger databases.

Overall, we preferred to use SQL, but this opinion is a bit biased since we all had good knowledge of it and, for this reason, we found the queries easier in this language. At the same time, we think that the challenge established by the MongoDB assignment allowed us to learn how to use this query language properly; a desirable skill in the era of big data.

# Results of queries:

1. How many users, activities and trackpoints are there in the dataset (after it is inserted into the database).

Afbeelding met tekst

Automatisch gegenereerde beschrijving

2. Find the average number of activities per user.



3. Find the top 20 users with the highest number of activities.

Afbeelding met tafel

Automatisch gegenereerde beschrijving

4. Find all users who have taken a taxi.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

5. Find all types of transportation modes and count how many activities that are tagged with these transportation mode labels. Do not count the rows where the mode is null.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

6. a) Find the year with the most activities.



b) Is this also the year with the most recorded hours?



Just like we found out in the previous assignment, 2008, which is the year with the most activities, is not the year with the most recorded hours because 2009 has more. With regard to query 6b, it is worth specifying that the difference between data times in MongoDB returns the result in milliseconds; for this reason, we had to convert it into hours by dividing it by 3600000.

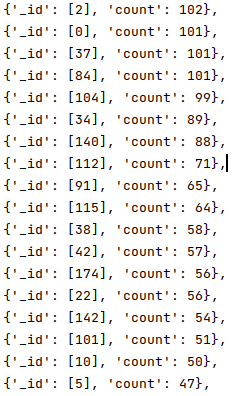
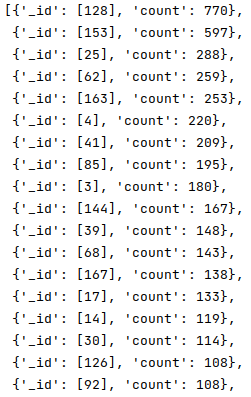
7. Find the total distance (in km) walked in 2008, by user with id=112.

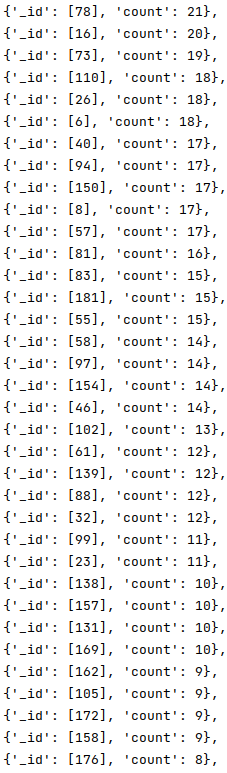
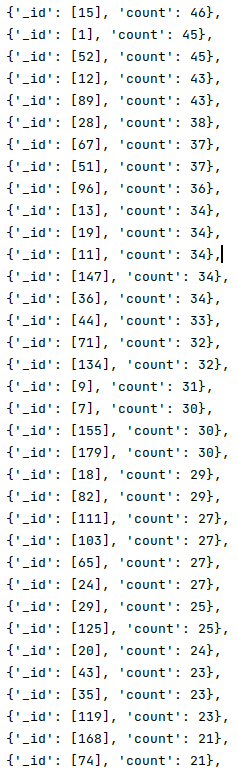


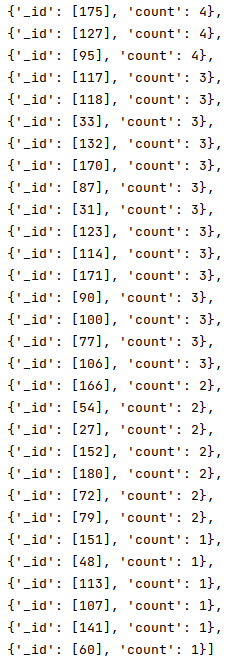
8. Find the top 20 users who have gained the most altitude meters.

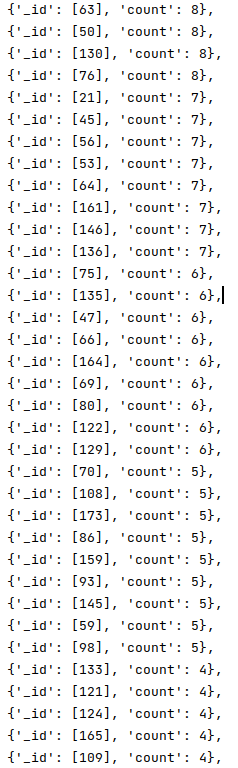
9. Find all users who have invalid activities, and the number of invalid activities per user

The ’count’ represents the number of invalid activities of a certain user id.





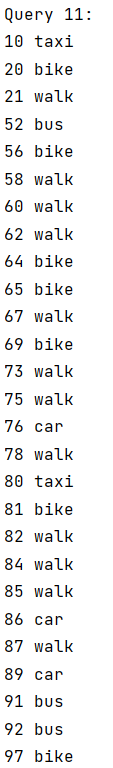




10. Find the users who have tracked an activity in the Forbidden City of Beijing.

11. Find all users who have registered transportation\_mode and their most used transportation\_mode.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

**Feedback**

Last assignment we didn’t see a big overlap with the course topic, this assignment was better in our opinion because it was more related to the course topic. We understand it is useful to make the switch from a SQL database to a NoSQL database and the consequences of this. It is also a good thing to have basically the same assignment for SQL as for MongoDB since this let’s you experience the differences firsthand between the two models.