**Report**

Assignment 2 - MySQL

**Group**: 69

**Students**: Hauk Aleksander Olaussen, Noran Baskaran, Vidar MIC.

**Introduction**The code for preprocessing of the data can be found in the python file Preprocessor.py. This file will read the data from the provided dataset, and dump the cleaned and preprocessed data into three separate files – namely users.pickle, activities.pickle and trackpoints.pickle. As you might notice, we make use of the python library pickle. To run the preprocessing, run the Preprocessor.preprocess() function.

All the answers for part 2 can be found within the Queries.py. Many were done using only SQL, but some needed more work with python. One function exists for each task, clearly commented in the file to easily find the one you’re after. Other comments explain parts of the code not necessarily easily understood. Running the Queries.py file will run all the 12 queries and print their answers to the terminal. Images and runtimes for each function/query can be found in the next section.

For this assignment we have worked together physically at campus. Hauk did the whole part 1 of the assignment and sent the data to a database which is located at his home desktop, and the rest of the group did a code review to confirm and check if this process was done efficiently and correct.

For part 2 of the assignment, we created the queries together and Hauk put them in the queries script and modified them to print the results in a readable format.

Link to repo: [GitHub](https://github.com/Olaussen/TDT4225/tree/main/assignment2)

**Results**

**Results from part 1**

**Results from part 2**

**Task 1:** Screenshot of terminal showing the result of a function named total\_amount\_of\_entries.

**Et bilde som inneholder tekst

Automatisk generert beskrivelse**

**Task 2:** Screenshot of terminal showing result of function named min\_max\_avg\_activites.

**Et bilde som inneholder tekst

Automatisk generert beskrivelse**

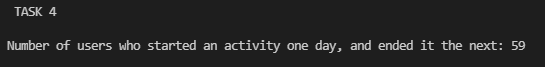
**Task 3:** Screenshot of terminal showing result of function named

top\_ten\_users\_by\_activites.

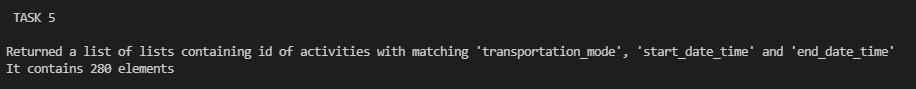
**Et bilde som inneholder tekst, elektronikk

Automatisk generert beskrivelse**

**Task 4:** Screenshot of terminal showing result of function named users\_start\_on\_one\_day\_end\_the\_next\_day.

****

**Task 5:** Screenshot of terminal showing result of function named find\_duplicate\_activities(). Looks messy, but what you see here is a list containing list which contains the id of activities that match other record in the activity table.

****

**Task 6:** Screenshot of terminal showing result of function named covid\_19\_tracking

**Et bilde som inneholder tekst

Automatisk generert beskrivelse**

**Task 7:** Screenshot of terminal showing result of function named never\_taken\_taxi.

**Et bilde som inneholder tekst

Automatisk generert beskrivelse**

**Task 8:** Screeenshot of terminal showing result of function named transportation\_mode\_count.

**Et bilde som inneholder tekst

Automatisk generert beskrivelse**

**Task 9:** Screenshot of terminal showing result of function named most\_active\_year

and the result of function named user\_with\_most\_activities\_from\_9a.

**Et bilde som inneholder tekst

Automatisk generert beskrivelse**

**Task 10:** Screenshot of terminal show result of function named distance\_walked\_in\_2008.

**Et bilde som inneholder tekst

Automatisk generert beskrivelse**

**Task 11:** Screenshot of terminal show result of function named most\_altitude\_gained.

**Et bilde som inneholder tekst

Automatisk generert beskrivelse**

**Task 12:** Screenshot of terminal show result of function named invalid\_activities.

**Et bilde som inneholder tekst

Automatisk generert beskrivelse**

**Discussion**

Discuss your solutions. Did you do anything differently than how it was explained in the assignment sheet, in that case why and how did that work? Were there any pain points or problems? What did you learn from this assignment?

**Må skrive noe om preprocessing og at vi dumpa i fil.**

For part one of the project, we learned a lot. At first we did not use the executemany function and only used cursor.execute, where we inserted by executing and committing one record and at a time, which took an ungodly amount of time. But after optimizing our preprocessor and our database handler (and this time we used the executemany function), the process of cleaning the data, dumping the cleaned data to different files then inserting it, got sped up from painful 20 hours to basically 5 minutes. This is because each execution does a trip to the database, but with executemany, we do a batch insertion where we only do one round trip total. But since we could not get the executemany function to work with the whole list of trackpoints, we batch insert 10 000 at a time.

For part two, almost all the tasks went smoothly except for task 5 and 6. For task 5 we understood the task as finding the ids of the task where the fields transportation\_mode, start\_date\_time and end\_date\_time match, because we would not get matches otherwise since they all have unique ids. Therefore, we think this was the most sensible interpretation. Task 6 was not heard but heavy computational. We did find a solution in pure SQL but dropped it since the solution we used with more work from python did a lot better.

Some of the data needed to be added to the database were not explicitly written to the files given in the assignment. An example of this is the start and end times for an activity. To find this data, we needed to read the date and time for the first and last trackpoint for each activity – and use these values for the fields in the activity table.