**Report**

Assignment 2 - MySQL

**Group**: 49

**Students**: Andreas Amundsen

**Introduction**

Briefly explain the task and the problems you have solved. How did you work as a group? If you used Git, a link to the repository would be nice.

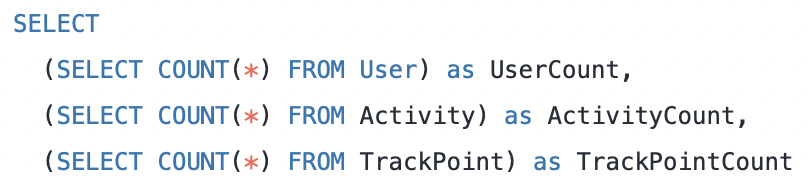
I have solved the problem of iterating trough a large dataset using a programming language, cleaning and parsing that data and inserting it to a database.

I have a lot of experience from group projects before, but this time I chose to work alone.

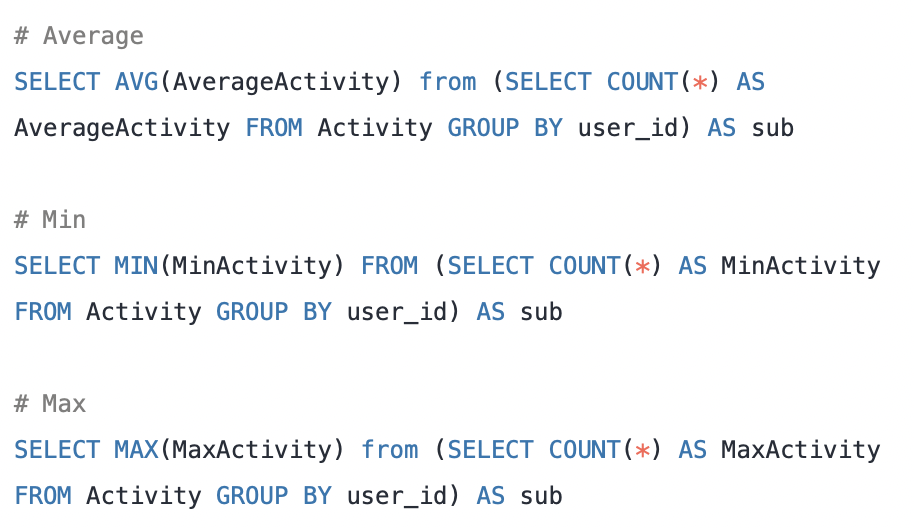
This was partly due to that I didn’t want to work with strangers and I felt that I could learn a lot more by working alone.

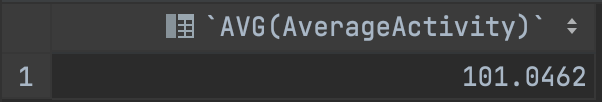
**Results**

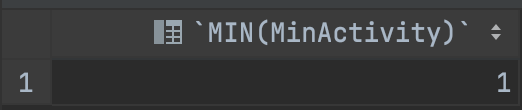


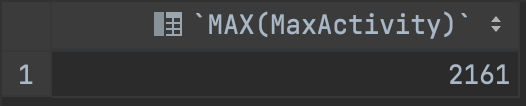


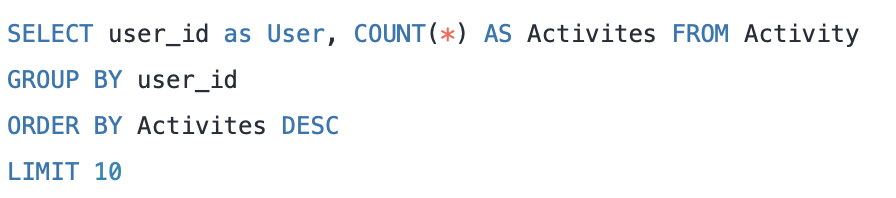


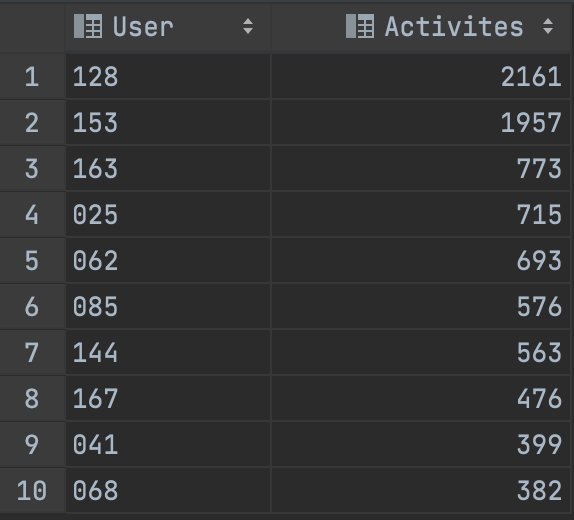












*Done with Python. See four() in task2.py*

I assume the question asks how many users have started an activity one day and ended it the next.

Users that have started an activity one day and ended it the next: 93



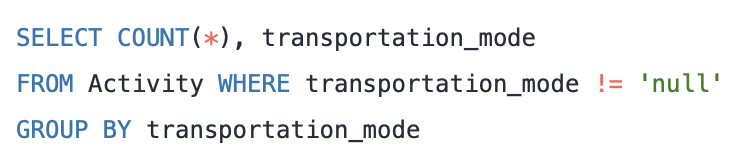
*Done with Python. See five() in task2.py*

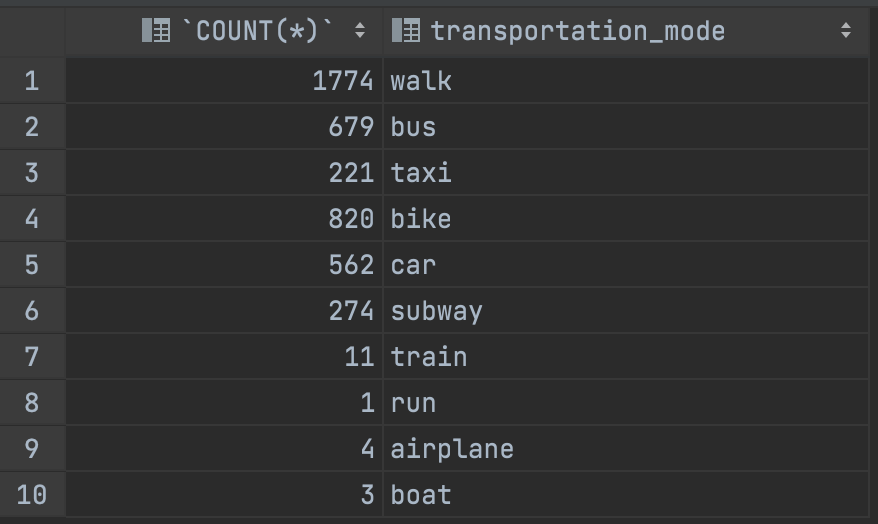
Number of duplicate rows: 0

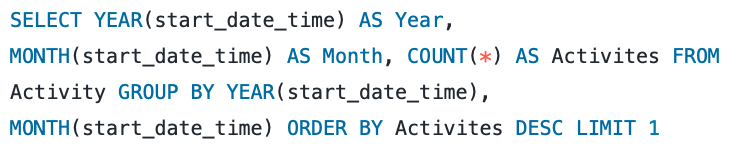
*Done with Python. See seven() in task2.py*

List of users who have never taken a taxi (output from seven() in alphabetical order)

['000','001','002','003','004','005','006','007','008','009','011','012','013','014','015','016','017','018','019','020','022','023','024','025','026','027','028','029','030','031','032','033','034','035','036','037','038','039','040','041','042','043','044','045','046','047','048','049','050','051','053','054','055','057','059','060','061','063','064','066','067','068','069','070','071','072','073','074','075','076','077','079','081','082','083','086','087','088','089','090','091','092','093','094','095','096','097','099','100','101','103','104','106','107','108','109','110','112','113','115','116','117','118','119','120','121','122','123','124','125','127','129','130','131','132','133','134','135','136','137','138','140','141','142','143','144','145','146','147','148','149','150','151','152','154','155','156','157','158','159','160','162','164','165','166','168','169','170','171'],'172','173','174','176','177','178','179','180','181']









November of 2008 is the month with the most activities, totalling 1068

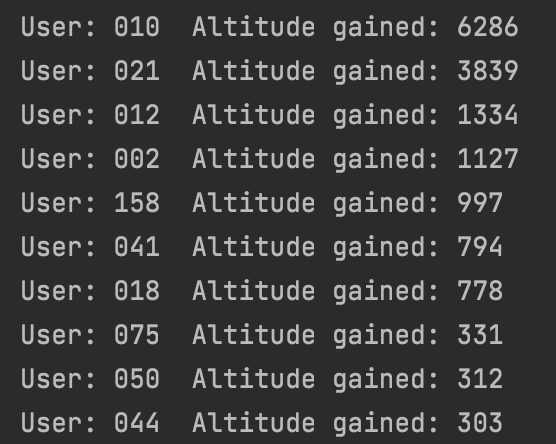
*Done with Python. See nine() in task2.py*

User 62 had the highest number of activities in November of 2008, spending a total of 402 hours. User 128 which had the second most activities that month, spent a total of 1866 hours.

*Done with Python. See ten() in task2.py*

In 2008 user with id 112 walked a total of 12677.87 kilometres

*Done with Python. See eleven() in task2.py*

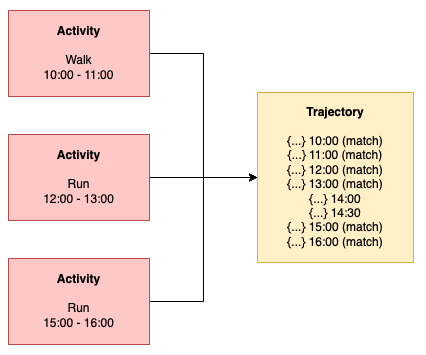


**Discussion**

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?

Implementation decisions

I chose to index the data and then insert it to the database. For instance, I would go through a trajectory file and create a dictionary of each trajectory, and add this to an array. The alternative is to iterate trough the data and add to the database on the fly. My method is more time consuming, but far more scalable and flexible. If the dataset for instance is re-released with new parameters, I could simply add more parameters to the dictionary and easily handle the change. This is because the codebase is more structured and easier to read, rather than code where both parsing and SQL calls is done all at once. On the other hand, I am not sure if this method is feasible if the trajectory files becomes very large.

Either way, an important factor to consider when creating complicated programs like these is that someone might work on the project later. One could make a very effective program, but that does not help if a developer later can’t read the code, because it is too messy and complex.

I found out that a trajectory file could have multiple matching activities. This can be handled in different ways:

1. A TrackPoint instance could be created containing only trajectories corresponding to an activity
2. The activity could point to the whole trajectory file

I chose the latter (see the image on the right). This leaves one to do some more complicated SQL, if we want to get the only trajectories corresponding to an activity later. On the other hand, the first solution would have a far more complex implementation, as one would need to split the trajectories. In addition, with the first solution it would be more complicated to find the trajectories before and after an activity.

The program took a very long time to complete (2.5 hours), and I think that is because of my algorithm when handling users with labels. When matching labels, I take one label and compare it to an activity, which means parsing and iterating trough the contents of a whole trajectory file. This essentially means that I parse and iterate trough a trajectory files for each entry in the labels file.

Learning outcome

1. I learned that Python is a breeze to work with when you want to quickly implement solutions. Methods like os.walk(), split() etc. makes life as a developer very easy. On the other hand, MySQL is strongly typed, while Python is not. This can lead to errors when inserting data. Next time I would probably use [Python type annotations](https://florimond.dev/en/posts/2018/07/why-i-started-using-python-type-annotations-and-why-you-should-too/) to avoid such issues.
2. I learned that how you design the algorithm can have huge effects on the time complexity. My solution ended up being ineffective.
3. I learned that when working with analysing data and deciding between solutions, there often are no right or wrong approaches. This is just the nature of machine learning and analysing data. You need to make an implementation choice and deal with the potential consequences. Here I am referring to matching activities to trajectory files.

What I would do differently the next time

**In general**

1. Plan and research better beforehand how the algorithms should be implemented to avoid pain spots down the line.

**For improving performance**

1. Insert multiple values in one SELECT query: <https://stackoverflow.com/questions/5526917/how-to-do-a-batch-insert-in-mysql>
2. When handling user with labels, I think it would be more effective to take every trajectory file, and compare it to the contents of the labels file. This means that I only must parse a trajectory file once, rather than multiple times.

**Working as a group vs by oneself**

1. Although I certainly learned a lot by working alone I think there are multiple benefits with working in a group. Others might have clever algorithmic solutions I would not have thought of and collaboration in general will often lead to better solutions, because you take different viewpoints into account.

**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.

What I liked:

* The assignment was fun and exciting to do, and I learnt a lot. I really liked the heavy focus on programming.
* It was great that mysql-connector code was supplied, so one could get up and running quickly.

What I didn’t like

* I think the problem text could be improved as I found it sometimes to be messy and hard to read. Take for instance the bullet point in Tasks/Part 1, ”When matching transport\_mode...”. I found this point to be very confusing. I get the fact that students are left with how to do the implementation, but I would have at least liked to see some examples on different approaches, to make the task itself clearer.
* I feel like restricting the report to only 1-2 pages, especially when the headings and title takes so much space and the font size is large, limits the opportunity for the student to reflect and show their competence.