**Report**

Assignment 2 - MySQL

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

**Group**: 37

**Students**: Vladislav Levitin and Kristian Tveiten

**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.

The assignment consisted of two parts, first part was cleaning and inserting the Geolife dataset into a MySQL database and the second part was answering a set of questions by writing a Python program using MySQL-queries. The code for part 1 is in part1\_create\_tables.py and part1\_insert\_data.py files and the code for part 2 is located in part2.py file.

The assignment is uploaded to a GitHub repository:

**Results**

**Part 1**

User Table - first 10 rows

Activity Table - first 10 rows

TrackPoint Table - first 10 rows

**Part 2**

2.1: How many users, activities and trackpoints are there in the dataset (after it is

inserted into the database).

A picture containing text

Description automatically generated

2.2: Find the average number of activities per user.

Text

Description automatically generated

2.3: Find the top 20 users with the highest number of activities.

A picture containing table

Description automatically generated

2.4: Find all users who have taken a taxi.

Table

Description automatically generated with medium confidence

2.5: 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

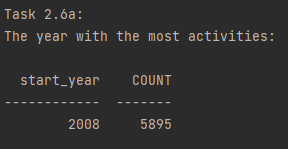
the mode is null

Text

Description automatically generated

2.6

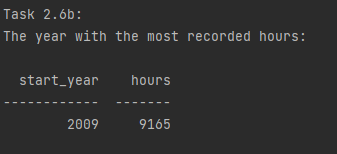
1. Find the year with the most activities.



1. Is this also the year with most recorded hours?

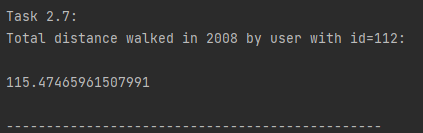
TIMESTAMPDIFF in MySQL does only consider full hours (does not round up).

2009 was the year with the most recorded full hours. 126% more minutes than 2008 and 132% more full hours.



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

In this task I shifted the lat and lon column so that each row had a start and end position. Then I calculated the distance for each set of points using haversine and then added up.



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

First off altitudes of -777 was set as NaN to avoid including those invalid values in the calculation. Further the user id’s were iterated through and all the activities per user were grouped together, then the difference between altitudes adjacent trackpoints were calculated. Then to only extract the altitude gained, the negative values were clipped to 0. The altitude gains were then summed up and converted to meters.

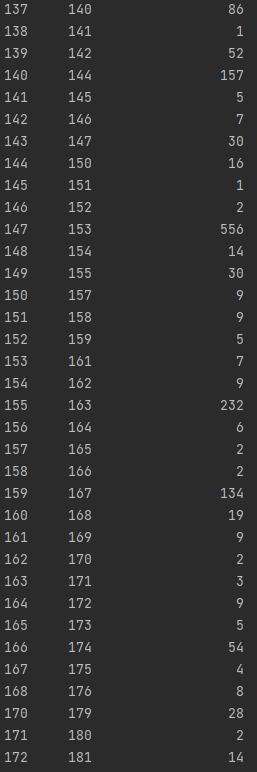
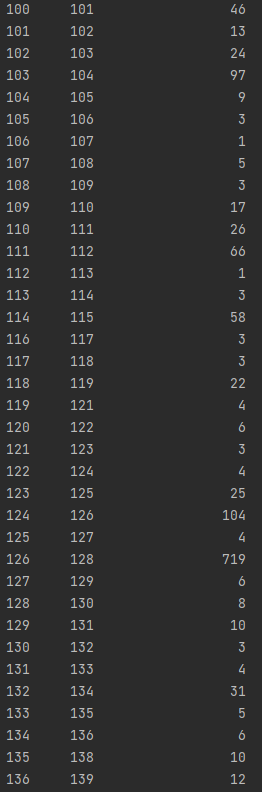
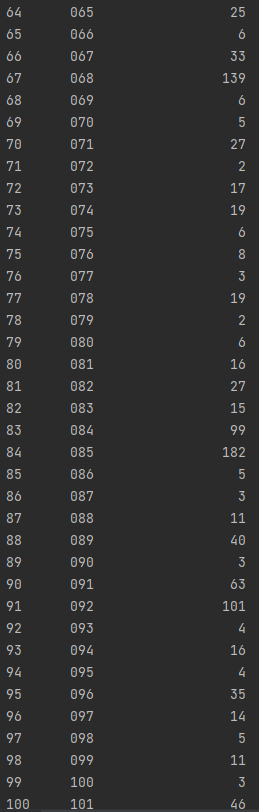
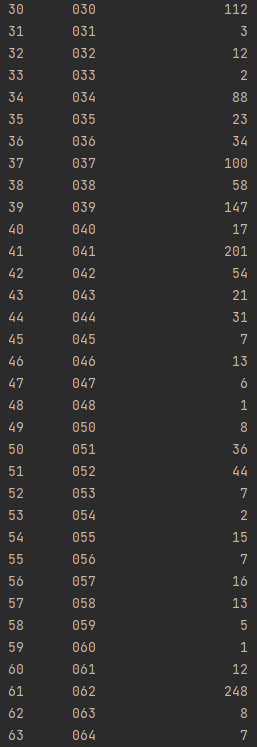
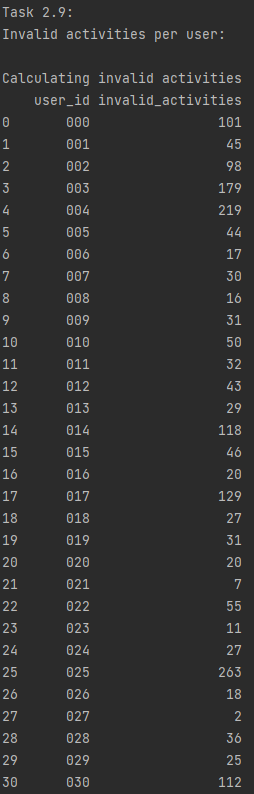
Et bilde som inneholder tekst

Automatisk generert beskrivelse

2.9: Find all users who have invalid activities, and the number of invalid activities

per user.

In this task again the activity\_id were grouped by, then time difference between trackpoints were calculated (in minutes). Then all trackpoints with a next trackpoint being logged 5 minutes or later were marked. Then all activities with at least one invalid trackpoint were counted. Apparently almost everyone had invalid activities.

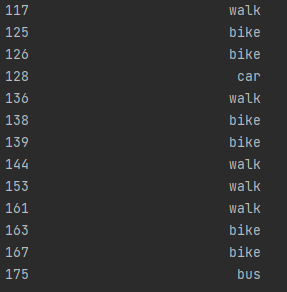
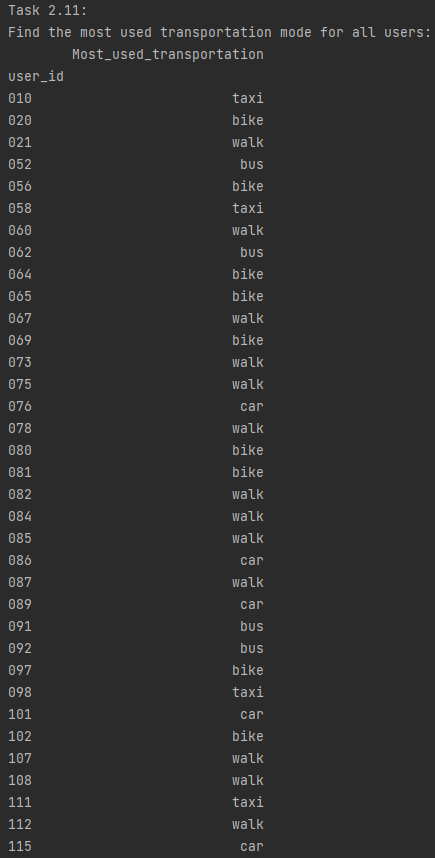


2.10: Find the users who have tracked an activity in the Forbidden City of Beijing.

2.11: Find all users who have registered transportation\_mode and their most used

transportation\_mode.

MySQL has no simple mode syntax, this is why it was implemented mostly using Python instead. When a user has multiple “modes” or most frequently used transport methods, one of them were picked at random.



**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?

My solution followed the assignment description.

Learnt to avoid for loops as much as possible when dealing with big datasets. Using pandas wisely does save a lot of processing time.

For the altitude task, the altitudes were clipped to 0 and the highest point of the earth being Mount Everest. This cuts out potential altitude readings while flying etc.