

Report

Assignment 2 - MySQL

Group: 69

Students: Hauk Aleksander Olaussen, Vidar Michaelsen and Noran Baskaran.

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. The insertion of all the data into the database from the provided folder took around 7 minutes on the desktop that hosts the database.

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 optimized his solutions.

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](#)

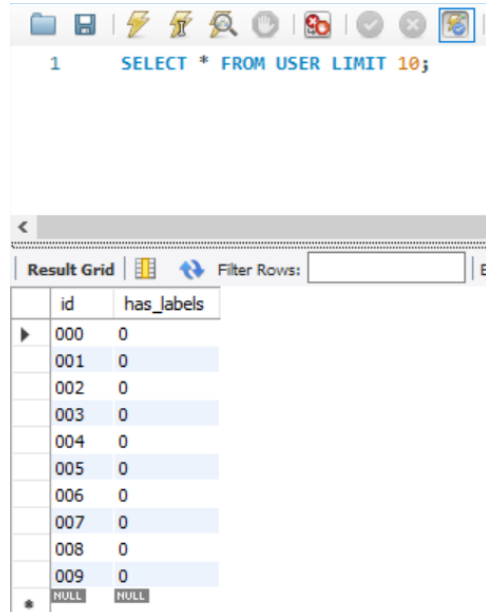
To connect to the database, fill a `.env` file with the following data:

```
DBUSER=group69_user
DBPASS=69eretmorsomttall
DBHOST=84.202.106.55
DBNAME=tdt4225_group69
```

Results

Results from part 1

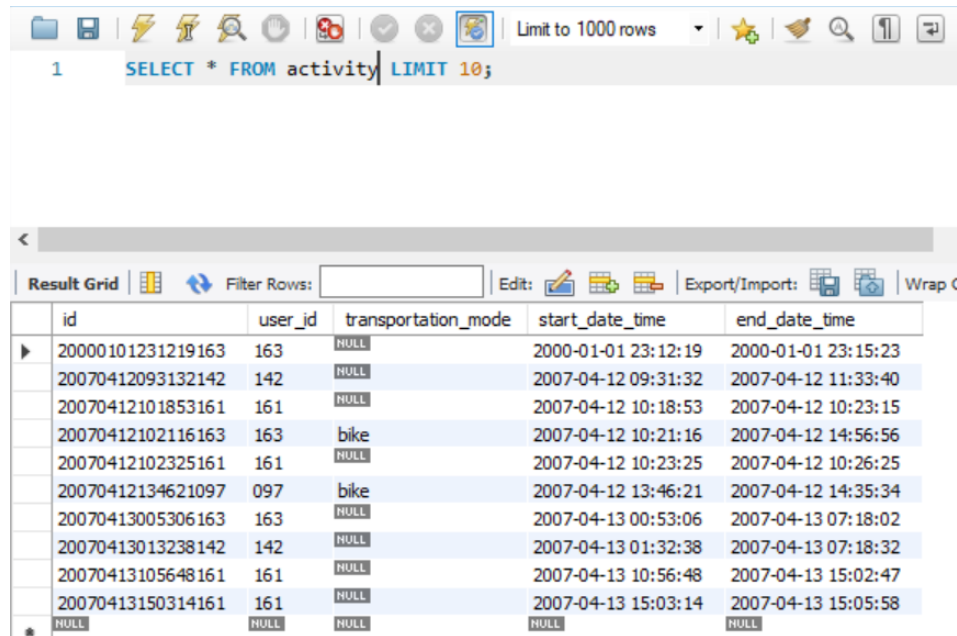
The table containing the users.



1 `SELECT * FROM USER LIMIT 10;`

id	has_labels
000	0
001	0
002	0
003	0
004	0
005	0
006	0
007	0
008	0
009	0
NULL	NULL

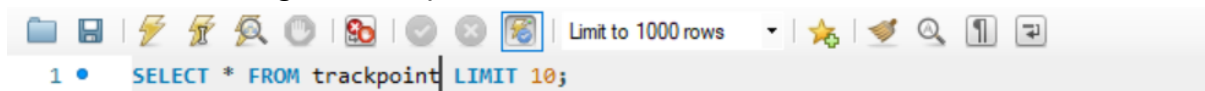
The table containing the activities.



1 `SELECT * FROM activity LIMIT 10;`

id	user_id	transportation_mode	start_date_time	end_date_time
20000101231219163	163	NULL	2000-01-01 23:12:19	2000-01-01 23:15:23
20070412093132142	142	NULL	2007-04-12 09:31:32	2007-04-12 11:33:40
20070412101853161	161	NULL	2007-04-12 10:18:53	2007-04-12 10:23:15
20070412102116163	163	bike	2007-04-12 10:21:16	2007-04-12 14:56:56
20070412102325161	161	NULL	2007-04-12 10:23:25	2007-04-12 10:26:25
20070412134621097	097	bike	2007-04-12 13:46:21	2007-04-12 14:35:34
20070413005306163	163	NULL	2007-04-13 00:53:06	2007-04-13 07:18:02
20070413013238142	142	NULL	2007-04-13 01:32:38	2007-04-13 07:18:32
20070413105648161	161	NULL	2007-04-13 10:56:48	2007-04-13 15:02:47
20070413150314161	161	NULL	2007-04-13 15:03:14	2007-04-13 15:05:58
NULL	NULL	NULL	NULL	NULL

The table containing the trackpoints.



id	activity_id	lat	lon	altitude	date_days	date_time
9704049	20081024020959000	40.008304	116.319876	492	39745.0902662037	2008-10-24 02:09:59
9704050	20081024020959000	40.008413	116.319962	491	39745.0903240741	2008-10-24 02:10:04
9704051	20081024020959000	40.007171	116.319458	-46	39745.0903819444	2008-10-24 02:10:09
9704052	20081024020959000	40.007209	116.319484	-48	39745.0904398148	2008-10-24 02:10:14
9704053	20081024020959000	40.007287	116.31959	-41	39745.0904976852	2008-10-24 02:10:19
9704054	20081024020959000	40.007366	116.319727	-40	39745.0905555556	2008-10-24 02:10:24
9704055	20081024020959000	40.007429	116.319873	-39	39745.0906134259	2008-10-24 02:10:29
9704056	20081024020959000	40.007499	116.320013	-39	39745.0906712963	2008-10-24 02:10:34
9704057	20081024020959000	40.007514	116.320165	-39	39745.0907291667	2008-10-24 02:10:39
9704058	20081024020959000	40.007578	116.320176	-40	39745.090787037	2008-10-24 02:10:44
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Results from part 2

Task 1:

Screenshot of terminal showing the result of a function `total_amount_of_entries()`.

```
TASK 1

Amount of users: 182
Amount of activities: 7477
Amount of trackpoints: 1195994
Total amount of entries in database: 1203653
```

Runtime: ca. 19.5 seconds

Task 2:

Screenshot of terminal showing result of function `min_max_avg_activites()`.

```
TASK 2

The minimal number of activities per user is: 1
The maximal number of activities per user is: 1007
The average number of activities per user is: 46.1543
```

Runtime: ca. 60ms

Task 3:

Screenshot of terminal showing result of function

`top_ten_users_by_activites()`.

```
TASK 3

1 User: 153 has 1007 activities
2 User: 128 has 999 activities
3 User: 062 has 453 activities
4 User: 144 has 391 activities
5 User: 163 has 380 activities
6 User: 025 has 311 activities
7 User: 140 has 174 activities
8 User: 112 has 139 activities
9 User: 004 has 136 activities
10 User: 092 has 119 activities
```

Runtime: ca. 36ms

Task 4:

Screenshot of terminal showing result of function

`users_start_on_one_day_end_the_next_day()`.

```
TASK 4

Number of users who started an activity one day, and ended it the next: 59
```

Runtime: ca. 32ms

Task 5:

Screenshot of terminal showing result of function

`find_duplicate_activities()`. What this does is check if two activities has the same `transportation_mode`, `start_date_time` and `end_date_time`, making them “equal”.

```
TASK 5

Returned a list of lists containing id of activities with matching 'transportation_mode', 'start_date_time' and 'end_date_time'
It contains 280 elements
```

Runtime: ca. 45ms

Task 6:

Screenshot of terminal showing result of function `covid_19_tracking()`.

TASK 6

Fetching overlapping activities in time

Calculating times and distance

There are 68 users who needs to be contacted, as they have been close to other users

Runtime: ca. 3 minutes

Task 7:

Screenshot of terminal showing result of function `never_taken_taxi()`.

TASK 7

Resturned a list of ids for all users having not taken a taxi

The list contains the ids of 175 users

Runtime: ca. 280ms

Task 8:

Screenshot of terminal showing result of function `transportation_mode_count()`.

TASK 8

Transportation mode bike has been used by 17 discinct users

Transportation mode bus has been used by 8 discinct users

Transportation mode car has been used by 8 discinct users

Transportation mode run has been used by 1 discinct users

Transportation mode subway has been used by 4 discinct users

Transportation mode taxi has been used by 7 discinct users

Transportation mode walk has been used by 31 discinct users

Runtime: ca. 31ms

Task 9:

Screenshot of terminal showing result of function `most_active_year()` and the result of function `user_with_most_activities_from_9a()`.
The user with the most activities (025) have also the most total hours this month.

```
TASK 9a

Year and month with most activities:
Year: 2008 Month: 12 Amount: 397

TASK 9b

Top users with most activities in year 2008 and month 12
1 | User 025 had 66 activities | A total of 8.8881 hours were used
2 | User 062 had 48 activities | A total of 8.2286 hours were used
```

Runtime: ca. 0.31s for 9a
ca. 0.32s for 9b

Task 10:

Screenshot of terminal show result of function `distance_walked_in_2008()`.

```
TASK 10

User 112 walked 70.1750264107954km in 2008
```

Runtime: ca. 535ms

Task 11:

Screenshot of terminal show result of function `most_altitude_gained()`.

```
TASK 11

1 | User 153 has gained 216360.1464 total meters
2 | User 128 has gained 184539.6360 total meters
3 | User 062 has gained 106757.4192 total meters
4 | User 144 has gained 80812.5384 total meters
5 | User 163 has gained 77858.1120 total meters
6 | User 140 has gained 67554.0432 total meters
7 | User 034 has gained 45877.5816 total meters
8 | User 041 has gained 41774.6688 total meters
9 | User 004 has gained 39008.9136 total meters
10 | User 092 has gained 36825.9360 total meters
11 | User 085 has gained 36100.8168 total meters
12 | User 025 has gained 32054.5968 total meters
13 | User 011 has gained 26978.1528 total meters
14 | User 003 has gained 23985.6264 total meters
15 | User 039 has gained 23744.2248 total meters
16 | User 000 has gained 23491.2408 total meters
17 | User 167 has gained 22751.7960 total meters
18 | User 147 has gained 22199.1936 total meters
19 | User 101 has gained 20247.8640 total meters
20 | User 174 has gained 19449.8976 total meters
```

Runtime: ca. 3 seconds

Task 12: Screenshot of terminal show result of function `invalid_activities()`.

```
TASK 12

Top 3 users with most invalid activities
1 | User 128 has 256 invalid activities
2 | User 153 has 156 invalid activities
3 | User 062 has 128 invalid activities
The function will return all the users, and how many invalid activities they have - if the have one
```

Runtime: ca. 3.5 seconds

Discussion

For part one of the project, we learned a lot. At first we did not use the `cursor.executemany()` function and only used `cursor.execute()`, where we inserted by executing and committing one record at a time, which took an ungodly amount of time. But after optimizing our preprocessor (`Preprocessor.py`) and our database handler (`DbHandler.py`) the process of cleaning and dumping the data into different files then inserting it, got sped up from 20 hours to basically 5 minutes. This is because when we tried to commit entries one at a time it would take one trip to the database and back each insertion, but with `executemany()`, we do a batch

insertion where we only do one round trip total – saving a lot a time. We could not get the `executemany()` function to work with the whole list of trackpoints (over 1 million entries). This might be because of how MySQL is set up at the home desktop – and we were not able to alter this from our laptops from campus. Because of this, we needed to split this list into separate chunks and insert these separately. We batch insert 10 000 at a time, for a total of 120 chunks/commits. All the code for inserting data can be found in `DbHandler.py`.

For part two, almost all the tasks went smoothly except for task 5 and 6.

For task 5 we assume the task as finding the ids of the task where the fields `transportation_mode`, `start_date_time` and `end_date_time` are equal – as this is the only data in the activity not being a unique id or foreign key. 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 hard to solve, but heavy computational. We did find a solution in pure SQL but dropped it. This is because the solution we found by finding activities with overlapping times in SQL and distance calculation in python performed 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. The id for an activity is the name of the file without the extension, concatenated with the id of the user having the ownership of it. This will remove any problems around the uniqueness of keys for different activities.