Assignment 2 - MySQL

Group: 64
Students: Nikolai L. Børseth

Introduction

The task was to define and create MySQL tables, then clean and insert data into them. After that task 2 gave a list of queries to perform on the database using MySQL and Python. The data used was from the *Microsoft Geolife GPS Trajectory dataset*, which contains data about the users outdoor movement.

I worked alone, but used Git as I work on two computers: GitHub. Please don't mind my history, since I get sloppy when only I am working in the repo…

All code is in the src folder, while the dataset is in the resources folder. For a closer explanation on the structure please read the provided README. The README also contains run instructions on how to set up a local MySQL db and what python files to run.

Results

Add your results from the tasks, both as text and screenshots. Short sentences are sufficient.

Task 1:

First 10 rows of tables.

User table:

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

Activities table:

	id	user_id	transportation_mode	start_date_time	end_date_time
▶	1	000	NULL	2008-10-23 02:53:04	2008-10-23 11:11:12
	2	000	NULL	2008-10-24 02:09:59	2008-10-24 02:47:06
	3	000	NULL	2008-10-26 13:44:07	2008-10-26 15:04:07
	4	000	NULL	2008-10-27 11:54:49	2008-10-27 12:05:54
	5	000	NULL	2008-10-28 00:38:26	2008-10-28 05:03:42
	6	000	NULL	2008-10-29 09:21:38	2008-10-29 09:30:28
	7	000	NULL	2008-10-29 09:30:38	2008-10-29 09:46:43
	8	000	NULL	2008-11-03 10:13:36	2008-11-03 10:16:01
	9	000	NULL	2008-11-03 23:21:53	2008-11-04 03:31:08
	10	000	NULL	2008-11-10 01:36:37	2008-11-10 03:46:12

Track points:

	id	activity_id	lat	lon	altitude	date_days	date_time
▶	1	1	39.984702	116.318417	492	39744.1201851852	2008-10-23 02:53:04
	2	1	39.984683	116.31845	492	39744.1202546296	2008-10-23 02:53:10
	3	1	39.984686	116.318417	492	39744.1203125	2008-10-23 02:53:15
	4	1	39.984688	116.318385	492	39744.1203703704	2008-10-23 02:53:20
	5	1	39.984655	116.318263	492	39744.1204282407	2008-10-23 02:53:25
	6	1	39.984611	116.318026	493	39744.1204861111	2008-10-23 02:53:30
	7	1	39.984608	116.317761	493	39744.1205439815	2008-10-23 02:53:35
	8	1	39.984563	116.317517	496	39744.1206018519	2008-10-23 02:53:40
	9	1	39.984539	116.317294	500	39744.1206597222	2008-10-23 02:53:45
	10	1	39.984606	116.317065	505	39744.1207175926	2008-10-23 02:53:50

Task 2:

Question 1:

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

```
Question 1
There are (182,) users, (16048,) activities and (9681756,) track points!
```

Question 2:

Find the average number of activities per user.

```
Question 2
The average user has 92.7630 activities
```

Question 3:

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

```
Question 3
Top 20 user by number of activities:
+----+-----------+------------------+
|    |   user_id |   num_activities |
|----+-----------+------------------|
|  1 |       128 |             2102 |
|  2 |       153 |             1793 |
|  3 |       025 |              715 |
|  4 |       163 |              704 |
|  5 |       062 |              691 |
|  6 |       144 |              563 |
|  7 |       041 |              399 |
|  8 |       085 |              364 |
|  9 |       004 |              346 |
| 10 |       140 |              345 |
| 11 |       167 |              320 |
| 12 |       068 |              280 |
| 13 |       017 |              265 |
| 14 |       003 |              261 |
| 15 |       014 |              236 |
| 16 |       126 |              215 |
| 17 |       030 |              210 |
| 18 |       112 |              208 |
| 19 |       011 |              201 |
| 20 |       039 |              198 |
+----+-----------+------------------+
```

Question 4:

Find all users who have taken a taxi.

```
Question 4
Users who have used a taxi: ['010' '058' '062' '078' '080' '085' '098' '111' '128' '163']
```

Question 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.

```
Question 5
Instaces of each type of transportation:
+----------------------+------------------+
| transportation_mode  |   activity_count |
|----------------------+------------------|
| walk                 |              481 |
| car                  |              419 |
| bike                 |              262 |
| bus                  |              199 |
| subway               |              133 |
| taxi                 |               37 |
| airplane             |                3 |
| train                |                2 |
| run                  |                1 |
| boat                 |                1 |
+----------------------+------------------+
```

Question 6:

a) Find the year with the most activities.

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

```
Question 6
The year with the most activities is: 2008 with a total of 5894 activities!
The year with the most hours: 2009 with a total of 9165 hours!
```

Question 7:

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

```
Question 7
User '112' walked 115.475km in 2008
```

Question 8:

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

Altitude gained per user, top 20:

	user_id	altitude_gained
1	128	633106.176
2	153	612079.8528
3	004	318204.1896
4	163	225811.08
5	003	218944.8504
6	144	213111.2832
7	085	205992.984
8	041	191967.9168
9	062	162850.068
10	030	156522.1152
11	039	137926.572
12	084	127535.0256
13	167	121324.116
14	025	118562.3232
15	000	111481.2096
16	002	111216.0336
17	140	92692.1184
18	126	89808.7104
19	034	84698.4336
20	022	58090.9176

Question 9:

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

Nr of invalid activities per user:

user_id	invalid_activities
128	720
153	557
025	263
062	249
163	233
004	219
041	201
085	184
003	179
144	157
039	147
068	139
167	134
017	129
014	118
030	112
126	105
000	101
092	101
037	100
084	99
002	98
104	97
034	88
140	86
112	67
091	63
038	58
115	58
022	55
042	55
174	54
142	52
010	50
015	46
101	46
001	45
005	45
052	44
012	43
089	40
028	36
051	36
096	35
036	34
067	33
011	32
044	32
009	31
019	31
134	31
007	30
147	30
155	30
013	29
071	29
179	28
018	27

024	27
082	27
065	26
111	26
029	25
125	25
103	24
035	23
119	22
043	21
016	20
020	20
074	19
078	19
168	19
026	18
073	18
006	17
040	17
110	17
008	16
057	16
081	16
094	16
150	16
055	15
083	15
097	14
154	14
181	14
046	13
058	13
102	13
032	12

061	12
139	12
023	11
088	11
099	11
131	10
138	10
105	9
157	9
158	9
162	9
169	9
172	9
050	8
063	8
076	8
130	8
176	8
021	7
045	7
053	7
056	7
064	7
146	7
161	7
047	6
066	6
069	6
075	6
080	6
122	6
129	6
136	6
164	6
059	5

070	5
086	5
098	5
108	5
135	5
145	5
159	5
173	5
093	4
095	4
031	3
077	3
087	3
090	3
100	3
106	3
109	3
114	3
117	3
118	3
123	3
132	3
171	3
027	2

033	2
054	2
072	2
079	2
152	2
165	2
166	2
170	2
180	2
048	1
060	1
107	1
113	1
141	1
151	1

Question 10:

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

```
Question 10
These users have tracked an activity in the forbidden city: ['004' '018' '019' '131']
```

Question 11:

Find all users who have registered transportation_mode and their most used transportation_mode.

```
Question 11
Most used transportation mode per user
+-----------+----------------------+
|   user_id | transportation_mode  |
|-----------+----------------------|
|       010 | taxi                 |
|       020 | bike                 |
|       021 | walk                 |
|       052 | bus                  |
|       056 | bike                 |
|       058 | car                  |
|       060 | walk                 |
|       062 | bus                  |
|       064 | bike                 |
|       065 | bike                 |
|       067 | walk                 |
|       069 | bike                 |
|       073 | walk                 |
|       075 | walk                 |
|       076 | car                  |
|       078 | walk                 |
|       080 | bike                 |
|       081 | bike                 |
|       082 | walk                 |
|       084 | walk                 |
|       085 | walk                 |
|       086 | car                  |
|       087 | walk                 |
|       125 | bike                 |
|       126 | bike                 |
|       128 | car                  |
|       136 | walk                 |
|       138 | bike                 |
|       139 | bike                 |
|       144 | walk                 |
|       153 | walk                 |
|       161 | walk                 |
|       163 | bike                 |
|       167 | bike                 |
|       175 | bus                  |
+-----------+----------------------+
```

Discussion

Task 1:

The biggest difficulty of task 1 was the insertion time. The users were fine, activities were kinda slow, but the track points took hours. At the time I was using *executemany()* on users and track points, since I had to update the track points with the id from the inserted activities. I then swapped to setting the activity ids during data preparation instead of using MySQL autoincrement, since we would be inserting data into an empty table anyway (so no danger of collision). This marginally increased the performance. It seemed that simply running *executemany()* on the track point when stored as a list of tuples [(tp1), (tp2), …] was simply slow. I then decided to switch to pandas with sqlalchemy, as that allows for directly inserting entire tables. Now inserting the track

points takes 5-10 minutes on my personal computer. Keep in mind that the MySQL db is also running in a docker container locally. The change to pandas also makes manipulating the data with python much simpler, and I used this in some of the queries of task 2.

Task 2:

The first 7 queries were not a problem. In query 8 I learn about how you can use LAG or LEAD to do calculations over rows (e.g. row_n-row_{n-1}). I was also scratching my head a while before I realized the altitude was in feet, not meters, since I kept getting really large numbers. So I should probably note down the units somewhere for the next exercise.

There was also room to make some assumptions when creating the queries. E.g. in query 8, I skipped all altitudes below 0, even though only "-777" is invalid. While it is technically possible to go below 0, in most cases I saw in the dataset there was a large jump. Like going from 60 feet to -40 feet in one track point then back up to 64 in the next, which seems unlikely. I therefore judged that we can discard the few that go below sea level in order to weed out the tracking mistakes. Another assumption was for the forbidden city, where I did "lat BETWEEN 39.915 AND 39.917" instead of "lat = 39.916" to give a bit larger of an area.