Assignment 2

TDT4225 - Very Large, Distributed Data Volumes

Deadline: Oct 10 at 14:00

Introduction

In this assignment you'll be working individually or in groups of up to 3 students to solve some database tasks using the MySQL database **(version 8.0.39)** and coding in Python.

This exercise will be graded and counts 25% of your final grade. The grading will be finished within 3 weeks of the delivery deadline.

Groups 1-80 have access to a virtual machine at IDI's cluster, running Ubuntu. This machine will have MySQL already installed and set up for you to use. The other groups may run MySQL themselves, e.g. using docker. The virtual machines have limited power, so a regular PC is more powerful.

This assignment will look at an open dataset of trajectories, and your task is to perform exploratory data analysis (EDA), design tables, clean and insert the data, and then make queries to answer a set of questions. Your Python program will handle these steps.

Along with this assignment sheet, you have been given several files:

- Dataset - Porto Taxi Trajectory dataset.
- `DbConnector.py` - a Python class that connects to MySQL on the virtual machine.
- `example.py` - a Python class with examples on how to create tables, insert, fetch and delete data in MySQL.
- `requirements.txt` - a file containing some pip packages that should be used in this assignment.

It is strongly recommended that you read through and understand the whole assignment sheet before you start solving the tasks. There are also some tips at the bottom of this assignment, which could be very handy!

Dataset – Porto Taxi Trajectory dataset

The Porto Taxi Trajectory dataset contains detailed records of taxi trips collected from the city of Porto, Portugal, over a period of one year (July 2013 – June 2014). Each trip includes a sequence of GPS points (trajectories) sampled every 15 seconds while the taxi was active.

In the provided dataset you will find a single CSV file containing all trajectories. Each row represents one taxi trip with an identifier, metadata, and the corresponding list of GPS points.

Columns:

- **TRIP_ID:** Unique identifier for each trip.
- **CALL_TYPE:** Code indicating how the trip was initiated.
  - *A*: Trip dispatched from the central.
  - *B*: Trip requested directly from a taxi stand.
  - *C*: Trip hailed randomly on the street.
- **ORIGIN_CALL:** ID of the client who initiated the call (only set when `CALL_TYPE = 'A'`). Otherwise `NULL`.
- **ORIGIN_STAND:** Taxi stand ID where the trip started (only set when `CALL_TYPE = 'B'`). Otherwise `NULL`.
- **TAXI_ID:** Unique identifier of the taxi performing the trip. The same taxi may appear in many trips.
- **TIMESTAMP:** Start time of the trip, expressed as Unix time (seconds since 1970-01-01). Should be converted to a standard `DATETIME` format for queries.
- **DAYTYPE:** Indicates whether the trip occurred on a normal day, a holiday, or the day before a holiday.
  - *A*: Normal day (workday or weekend).
  - *B*: Holiday or special day.
  - *C*: Day before a holiday.
- **MISSING_DATA:** Boolean flag indicating whether there are missing GPS points in the `POLYLINE`. `FALSE` means the data is complete, `TRUE` means some points

are missing.

- **POLYLINE**: A list of GPS points forming the trajectory. Each point is represented as `[longitude, latitude]`, sampled at 15-second intervals.

Example (POLYLINE):

[[-8.618643, 41.141412], [-8.618499, 41.141376], [-8.618346, 41.141353]]

Notes on Data Quality and Analysis
The dataset contains trips with incomplete trajectories, indicated by the `MISSING_DATA` field. This makes it particularly suitable for exploratory data analysis (EDA). The Porto dataset is also interesting because the nested trajectory structure (`POLYLINE`) requires careful transformation and storage to support efficient analysis.

Setup database

These steps will guide you through how to connect to the virtual machine from your computer and create a MySQL-database user with privileges.

1. First, you need to use NTNU's VPN to access the virtual machines. Log onto NTNU-VPN with your Feide user (check out this link if you have not done this before).

2. Open your terminal and enter the following command: `ssh your_username@tdt4225-xx.idi.ntnu.no` where **"your_username"** is the Feide-username and **"xx"** is your group number for this project (01, 02, 03… etc).
   (If this message pops up, enter **yes**:
   ```
   The authenticity of host 'tdt4225-xx.idi.ntnu.no
   (xxx.xxx.xxx.xxx)' can't be established.
   ECDSA key fingerprint is …
   Are you sure you want to continue connecting (yes/no)?
   yes
   ```
)
   You will then be asked to enter your password, use the Feide-password here. If the password is correct, you have now successfully logged in!

3. Now we want to create a MySQL-user that the group will use during this assignment. First, enter `sudo mysql` in your terminal windows (which is now inside the virtual machine) and type in your Feide password. If successful, you are now on the MySQL server and can write MySQL-queries directly in this window: `mysql> "some query"`.

4. Run the following commands in MySQL to create a new user and give the user admin rights.
   ```
   mysql> CREATE USER 'YOUR_USERNAME_HERE'@'%' IDENTIFIED
   WITH mysql_native_password BY
   'YOUR_PASSWORD_IN_PLAIN_TEXT_HERE';

   mysql> GRANT ALL PRIVILEGES ON *.* TO
   'YOUR_USERNAME_HERE'@'%'WITH GRANT OPTION;
   ```
   Yes, write the username using quotas ''. Here is an example for username=testuser and password=test123:
   ```
   CREATE USER 'testuser'@'%.ntnu.no' IDENTIFIED WITH
   mysql_native_password BY 'test123';
   GRANT ALL PRIVILEGES ON *.* TO 'testuser'@'%.ntnu.no'
   WITH GRANT OPTION;
   ```

Now your new user has been granted admin rights. The % sign signifies a wildcard, meaning that your new user will have access to all schemas in MySQL. You will want to flush the privileges so that MySQL will load your user with its new rights. Type `mysql> FLUSH PRIVILEGES;`

To check if the user is created, type `mysql> SELECT User FROM mysql.user;`. If you see your newly created user in the list, you have created the user successfully.

5. Let's create a database. Type `mysql> SHOW DATABASES;`, there should be four default databases, ignore these.
   Create a new database by typing `mysql> CREATE DATABASE db_name;`, where **"db_name"** is your chosen name of the database, example `mysql> CREATE DATABASE test_db;`. Try typing `mysql> SHOW DATABASES;` again and check if the database is there. If it is there, congratulations!

To exit the MySQL program, simply type `exit` and you are back at the virtual machine. Now you have to type `sudo service mysql restart` in the terminal. Once this is complete, you should be able to use the database from your Python program (or the MySQL terminal).

Setup Docker

If your group does *not* have access to a virtual machine, you can instead run MySQL locally using Docker. Docker allows you to start a container with MySQL that isolates database setup, dependencies, and configuration, making it easier to reproduce the environment on your own laptop. A typical command might look like:

```
docker run --name=mysql-local -e MYSQL_ROOT_PASSWORD=secret -p
3306:3306 -d mysql:8.0
```

You can then connect with the root user, create your own database, and proceed as with the VM setup. Using Docker is allowed and encouraged if you prefer to work locally. For more detailed instructions, see How to Set Up and Configure MySQL in Docker (DataCamp tutorial).

Setup Python

We will be using a MySQL connector for Python in this assignment. If you for some reason would like to complete the assignment in another language, you are allowed to do so. ***(NB! We will not provide support or documentation for other languages, so we strongly encourage you to use Python.)***

With the files provided in this assignment, you will find a `requirements.txt`, `DbConnector.py` and `example.py`, among others.

1. To set up the required pip-packages for this assignment, simply run `pip install -r requirements.txt` while you are in the directory with the requirements-file. This will install the connector, along with tabulate (used to print pretty tables in python) and haversine (used to calculate distance between two coordinates).

2. Now, open `DbConnector.py` and look at the code. There will be a TODO there to set up the database correctly with the settings from the database setup.
   Update the values accordingly:
   Host = tdt4225-xx.idi.ntnu.no, where xx is your group number
   Database = the name you provided for your database in step 5 of the database setup (test_db from the example)
   User = the username provided in step 4, (testuser from the example)
   Password = the password provided in step 4 (test123 from the example)
   Port = this is optional, and default is 3306
   - Passwords should be stored as an environmental variable if your code is public. See here, or search Google to find out how to do it.
3. Open `example.py`, the file has a main method that creates a connection to the database from DbConnector, creates a table named "Person", inserts some names in the database, displays the data, and then drops the table.
   Try to run the code. If everything is set up correctly, you will establish a connection to the database and insert/delete data. You can use this file for inspiration when solving the tasks in this assignment.

Tasks

The tasks are divided into three parts. Part 1 will focus on exploratory data analysis (EDA), where you analyze and clean the Porto dataset before inserting it into the database. Part 2 will then involve writing queries to the database in order to gain insights from the data. Finally, Part 3 will focus on writing a report where you discuss your answers and reflect on your findings.

We recommend looking at the documentation for the MySQL-Python connector before you start coding.

Part 1

In this task you will begin by performing exploratory data analysis (EDA) on the Porto Taxi Trajectory dataset. The goal is to understand the structure and quality of the data, identify potential issues such as missing values, and prepare the dataset for insertion into your own MySQL database.

1. Through EDA you should examine the main attributes (e.g., trip IDs, call types, day types, and the trajectories stored in the POLYLINE field), gain an overview of the data distribution, and investigate the presence and impact of missing trajectories. The insights you gather here will help guide how you clean and organize the data for later analysis.
2. Once you have a clear understanding of the dataset, you should design and create the necessary tables in your MySQL database and insert the cleaned data. You are free to adapt the structure as you see fit, but do explain your reasoning in the report. You should also document your chosen schema explicitly, for example by including the `CREATE TABLE` statements you used.

Note: This part is deliberately kept open-ended to encourage exploration. You are expected to include all findings you consider relevant in your final submission.

Part 2

Some of the tasks can be answered using MySQL-queries only, while some might require both queries and Python code to manipulate the data correctly.

Answer the following questions by writing a Python program using MySQL-queries (like in `example.py`):

1. How many taxis, trips, and total GPS points are there?
2. What is the average number of trips per taxi?
3. List the top 20 taxis with the most trips.
4. a) What is the most used call type per taxi?
   b) For each call type, compute the average trip duration and distance, and also report the share of trips starting in four time bands: 00–06, 06–12, 12–18, and 18–24.
5. Find the taxis with the most total hours driven as well as total distance driven. List them in order of total hours.
6. Find the trips that passed within 100 m of Porto City Hall. (longitude, latitude) = (-8.62911, 41.15794)
7. Identify the number of invalid trips. An invalid trip is defined as a trip with fewer than 3 GPS points.
8. Find pairs of different taxis that were within 5m and within 5 seconds of each other at least once.
9. Find the trips that started on one calendar day and ended on the next (midnight crossers).
10. Find the trips whose start and end points are within 50 m of each other (circular trips).
11. For each taxi, compute the average idle time between consecutive trips. List the top 20 taxis with the highest average idle time.

Part 3

Write a short report (see `report-template.docx` in Blackboard exercise 2 where you will display and discuss your results from the tasks. Include screenshots from both part 1 (EDA) and part 2 (all the results to each task).

Hand in both the report (as PDF) and your code to BlackBoard within Oct 10, at 14:00.

Tips

- Using the MySQL terminal on your Ubuntu machine could be useful for checking simple queries without having to use Python and the connector.
  - E.g. just checking if the data is correct "SELECT * FROM User LIMIT 5"
  - Open the MySQL terminal by typing `sudo mysql`, log in with Feide-password and type `use name_of_db` to do this.
- Using dictionaries/hashmaps are faster for lookups than lists/arrays in your Python code
- Remember foreign key cascade rules
- How to use and take advantage of the datetime data type in your queries can be found here.
- Using variables in MySQL may come in handy
- As stated, using Haversine for calculating distance is recommended
- As stated, using Tabulate for printing tables in your Python program is recommended
- If you want to visualize the data in the dataset, you can get inspired here.