



Assignment 1

Hand in this assignment until **Thursday, 04 May 2023, 12:00** at the latest.

Exam-style Exercises

Exercises marked with **E** are similar in style to those you will find in the exam. You can use these to hone your expectations and gauge your skills.

Running out of ideas?

Are you hitting a roadblock? Are some of the exercises unclear? Do you just need that one hint to get the ball rolling? Refer to the [#forum](#) channel on our Discord server and check the tag for this assignment—maybe you'll find just the help you need.

Information about this assignment's data files.

In this assignment, you will work with JSON and CSV data sets obtained from the [Twitter API](#). If you are not familiar with basic Twitter concepts, please consult [Wikipedia](#). Along with this assignment, we provided you with two JSON files `tweets.json` and `users.json` as well as two CSV files `tweets.csv` and `users.csv`, and our Python module `DB1.py`.

tweets.json An array of tweet objects related to *Tübingen*.

users.json An array of user objects related to the tweets in `tweets.json`.

tweets.csv and users.csv These contain the same data as the beforementioned JSON files as CSVs.

DB1.py The Python module you need to load the CSV files.

Task 0: Introduction To The Infrastructure

(0 credits)

- Before we are able to provide comments on your solutions**, you have to complete one initial task first. In your GitHub Classroom repository, you should find a file called `README.md`. Add your name, surname, student number, subject and field of study to the file and commit+push the changes. Please make sure the file always exists and is populated with the correct information.
- Each assignment submission in DB1 must be placed into a separate folder in the **root** directory of your GitHub Classroom repository. The name of the folder has to be in the format **solution<number>**. For this assignment, for example, your submission folder would be called **solution01**.
- In general, the only accepted file formats are PDF of appropriate size (< 2MB), plain text files (.txt) and source files (`sql`, `.py`, ...). Other formats may not be commented on unless stated otherwise. Your submitted code has to compile and work. If compiling, running and/or understanding your source code is particularly complex, please write documentation accordingly.

Task 1: Twitter Mini World

(1 credit)

Explore the JSON data sets. Documentation on the semantics of particular fields can be found in the [Twitter API documentation](#). Then, characterize the data set in terms of the “Mini World” vocabulary (objects, attributes, relationships, constraints) discussed in the lecture. Do not exhaustively describe all attributes (there are many!), but select a few which seem important to you.

Task 2: Structured Information

(1 credit)


Describe informally and *briefly* how relationships between objects in the CSV files are encoded in the data. What are the differences in comparison to the JSON files?

Task 3: JSONiq/PyQL Queries

(1 credit)


We will now query the provided data sets using the query languages JSONiq and PyQL. Please consider the hints at the end of the assignment sheet **prior** to working out solutions.

Implement each of the following queries in (1) JSONiq and (2) PyQL.

- (a)  Return the text of all tweets having more than 5 retweets (based on `retweet_count`).
- (b) Return all tweets of the user with the screen name **Tagblatt**.
- (c) For each user, count the number of the tweets she published. (Not based on `statuses_count`!)

Hint

The result size for this task is 397 which exceeds the default result size limit of RumbleDB and triggers a warning. To work around this, simply use `--materialization-cap 1000` when running your JSONiq query in RumbleDB.

- (d) For every natural language ('de', 'en', ...) calculate the number of tweets that have been published.
- (e)  Return screen names of all users for which no tweet exists in the data set.

Hint

Remember the JSONiq `empty()` function and consider `len([]) == 0` in PyQL.

Additional information on JSONiq

Running JSONiq queries requires a JSONiq engine like RumbleDB. Refer to the [post on our Discord server](#) on how to install RumbleDB on your system.

Additional information on PyQL

The provided CSV files can be loaded using the `DB1` Python module and queried using the PyQL Python subset as discussed in the lecture.

- The Python [documentation](#) and [tutorial](#) may be helpful.
- Duplicates can be eliminated by converting a list to a set and then back to a list i.e.: `list(set(xs))`.
- PyQL (i.e. Python) comprehensions can have multiple generators:

[<e> for <x> in <xs> for <y> in <ys>]