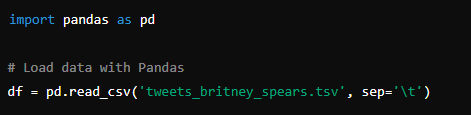
**Part 1: Data Ingestion**

Since the data is provided in TSV format and it contains tweets, a good approach would be to ingest it into a format that allows easy querying. However as we are working in large data set, for simplicity and speed, we could process the data using Pandas in Python.

**Steps for Data Ingestion:**

1. **Read the Data**:
   * Load the TSV files into a Pandas DataFrame for easy manipulation.
   * As the dataset is too large, consider using chunksize to load it in chunks.

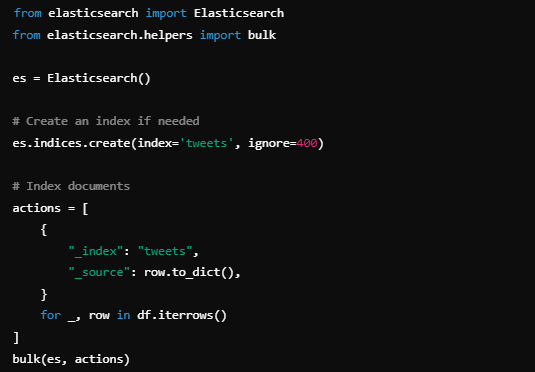


2. **Explore the Data**:

* Examine the structure of the data to identify columns such as the tweet text, user information, likes, time, and location (place IDs).
* Ensure to handle missing data or columns that might not be relevant.

3. **Index Data for Search**:

* If Elasticsearch is required, we would need to index the data into it. We can use the elasticsearch-py library to do this.



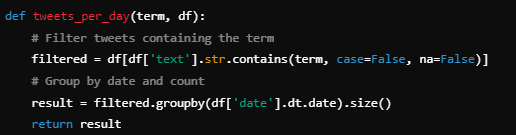
**4. Efficient Querying**:

* Ensure the data structure is optimized for fast queries. If using Pandas, create a column for the text, user, likes, date, and location, which are the most relevant for queries.
* If using Elasticsearch, ensure the appropriate fields are indexed and searchable.

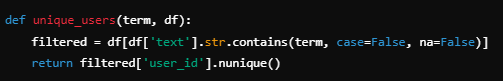
**Part 2: Query the Data**

Now, you need to create functions that can query the data based on the conditions provided. Below are examples of how you might approach each query.

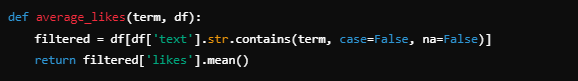
**1. How many tweets were posted containing the term on each day?**



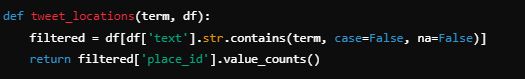
**2. How many unique users posted a tweet containing the term?**



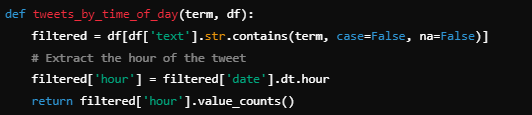
**3.How many likes did tweets containing the term get on average?**

****

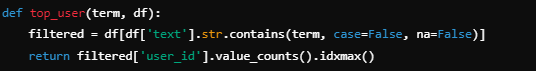
**4.** **Where (in terms of place IDs) did the tweets come from?**

****

**5.** **What times of day were the tweets posted?**

****

**6.** **Which user posted the most tweets containing the term?**

****

**Bonus: Privacy Protections**

**For privacy protection, there are a few techniques to explore:**

1. **Data Thresholding:** Ensure demographic groups with fewer than 10 users are hidden.
2. **Differential Privacy:** We could add random noise to the results using differential privacy libraries like PySyft or Google's Differential Privacy project to obscure sensitive data while maintaining statistical validity.

**Tools & Technologies**

* **Pandas**: For basic data ingestion and processing.
* **Elasticsearch**: If needed for scalable search queries.
* **Streamlit**: For creating a user-friendly dashboard to display and query the data.
* **Docker**: To containerize the app, ensuring it runs consistently across different environments.