**correct\_twitter2019-Project-and-Analysis**

Step-by-Step Guide to Set Up the System:

**1. Install Python and Dependencies:**

First, you need to make sure that you have Python and the necessary libraries installed on your machine.

Install Python:

Make sure to check the box to add Python to your system PATH during installation.

Install Required Libraries:

Open a terminal or command prompt and install the required Python libraries by running:

pip install pandas matplotlib

**2. Download the Dataset:**

You need to have a dataset file named correct\_twitter\_201904.tsv. This file should contain Twitter data with columns like text, created\_at, author\_id, like\_count, place\_id, etc.

**3. Python Code Implementation:**

You will need to write the Python code that I provided in one of your files. You can create a new Python file (twitter\_analysis.py) and copy-paste the code snippets below.

**Full Code Implementation:**

import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read\_csv('correct\_twitter\_201904.tsv', sep='\t')

df['created\_at'] = pd.to\_datetime(df['created\_at'], errors='coerce', utc=True)

def tweets\_per\_day(term):

filtered\_df = df[df['text'].str.contains(term, case=False, na=False)]

daily\_tweet\_count = filtered\_df.groupby(filtered\_df['created\_at'].dt.date)['id'].count()

return daily\_tweet\_count.reset\_index(name='tweet\_count')

def unique\_users\_posting\_term(term):

filtered\_df = df[df['text'].str.contains(term, case=False, na=False)]

unique\_user\_count = filtered\_df['author\_id'].nunique()

return unique\_user\_count

def average\_likes\_for\_term(term):

filtered\_df = df[df['text'].str.contains(term, case=False, na=False)]

average\_likes = filtered\_df['like\_count'].mean()

return average\_likes

def place\_ids\_for\_term(term):

filtered\_df = df[df['text'].str.contains(term, case=False, na=False)]

unique\_place\_ids = filtered\_df['place\_id'].dropna().unique()

return unique\_place\_ids

def times\_of\_day\_for\_term(term):

filtered\_df = df[df['text'].str.contains(term, case=False, na=False)]

tweet\_hours = filtered\_df['created\_at'].dt.hour

hour\_distribution = tweet\_hours.value\_counts().sort\_index()

return hour\_distribution

def most\_frequent\_user\_for\_term(term):

filtered\_df = df[df['text'].str.contains(term, case=False, na=False)]

tweet\_counts\_by\_user = filtered\_df['author\_id'].value\_counts()

top\_user = tweet\_counts\_by\_user.idxmax()

top\_user\_tweet\_count = tweet\_counts\_by\_user.max()

return top\_user, top\_user\_tweet\_count

term = "music"

print("Tweets per day containing the term:")

print(tweets\_per\_day(term))

print("\nNumber of unique users posting tweets containing the term:")

print(unique\_users\_posting\_term(term))

print("\nAverage number of likes for tweets containing the term:")

print(f"{average\_likes\_for\_term(term):.2f}")

print("\nPlace IDs where tweets containing the term came from:")

print(place\_ids\_for\_term(term))

print("\nNumber of tweets posted at each hour of the day:")

print(times\_of\_day\_for\_term(term))

print("\nUser who posted the most tweets containing the term:")

user, count = most\_frequent\_user\_for\_term(term)

print(f"User ID: {user} with {count} tweets.")

**4. Run the Python Script:**

Open your terminal or command prompt and navigate to the folder where the Python file (twitter\_analysis.py) and dataset are located.

Run the Python file using:

python twitter\_analysis.py

**Design Justifications:**

**Pandas Library:**

Pandas is ideal for loading, manipulating, and querying structured data like CSV or TSV files. It has powerful functions like .str.contains(), .groupby(), and .value\_counts() that make querying and data analysis efficient.

Alternative: SQL databases could be used for larger datasets, but for ease of use and quick analysis, Pandas provides more flexibility for smaller datasets.

**Datetime Handling:**

It's important to convert the created\_at column to a datetime format for time-based analysis, like grouping by day or extracting the hour of the tweet. Without this conversion, extracting date and time information would be error-prone.

**Value Counts and Groupby:**

This function quickly summarises the distribution of a column's values, such as the number of tweets per hour or tweets per user. It's highly efficient for frequency analysis.

Grouping allows us to aggregate data over a specific period, such as tweets posted per day. This helps in time-based trend analysis.

**Matplotlib for Visualization:**

If you want to visualize trends (e.g., tweet distribution over the day), Matplotlib is an excellent choice for basic plots like bar charts. It integrates seamlessly with Pandas.

**Modular Functions:**

Each query is encapsulated in its own function to make the code reusable and clean. You can call each function independently to answer specific questions.