Tweet Data Analysis System

# System Overview and Instructions for Use

This system is designed to help you analyze tweet data and query it for specific terms. It provides insights such as the number of tweets per day, the number of unique users, average likes, place IDs, times of day when tweets were posted, and the most active users for any given term. Below, I’ll explain how you can set up, run, and query this system on your own computer, along with the rationale behind the design choices.

# Step 1: Setting Up the System

1. Environment Setup:

- Ensure you have Python 3.x installed on your computer.  
- Install the necessary libraries using the following command:  
 ```bash  
 pip install pandas matplotlib  
 ```  
- These libraries are essential for data manipulation (pandas) and visualization (matplotlib).

2. Loading the Data:

- Place your dataset (in this case, the `assesment.csv` file) in the working directory of your Python environment.  
- Load the data into a pandas DataFrame using:  
 ```python  
 import pandas as pd  
 df = pd.read\_csv('assesment.csv')  
 ```

# Step 2: Data Preparation and Cleaning

1. Data Type Conversion:

- Convert columns to the appropriate data types (e.g., datetime for dates, integers for IDs).  
- This is crucial because correct data types ensure accurate calculations and efficient data processing.  
 ```python  
 df['created\_at'] = pd.to\_datetime(df['created\_at'], errors='coerce')  
 ```

2. Handling Missing Values:

- Drop or fill in missing values to avoid errors during analysis.  
- This step ensures that the analysis functions run smoothly without encountering NaN values.  
 ```python  
 df = df.dropna(subset=['created\_at'])  
 ```

# Step 3: Running the Query System

1. Filtering Data by Term:

- Use the `filter\_data\_by\_term` function to filter the dataset for tweets containing a specific term.  
 ```python  
 def filter\_data\_by\_term(df, term):  
 filtered\_df = df[df['text'].str.contains(term, case=False, na=False)]  
 return filtered\_df  
  
 term = 'music'  
 filtered\_df = filter\_data\_by\_term(df, term)  
 ```

2. Answering Specific Queries:

- Tweets per Day:  
 ```python  
 tweets\_per\_day\_count = tweets\_per\_day(filtered\_df)  
 print(tweets\_per\_day\_count)  
 ```  
- Unique Users:  
 ```python  
 unique\_users = unique\_users\_count(filtered\_df)  
 print(f'Number of unique users: {unique\_users}')  
 ```  
- Average Likes:  
 ```python  
 avg\_likes = average\_likes(filtered\_df)  
 print(f'Average number of likes: {avg\_likes:.2f}')  
 ```  
- Place IDs:  
 ```python  
 place\_ids = place\_ids\_list(filtered\_df)  
 print(f'Place IDs: {place\_ids}')  
 ```  
- Tweets by Hour:  
 ```python  
 plot\_tweets\_by\_hour(tweets\_by\_hour\_count)  
 ```  
- Most Active User:  
 ```python  
 most\_active\_user\_id, most\_active\_user\_count = most\_active\_user(filtered\_df)  
 print(f'Most active user ID: {most\_active\_user\_id} with {most\_active\_user\_count} tweets')  
 ```

# Step 4: Justification of Design Choices

1. Modular Design:

- Each function (e.g., `filter\_data\_by\_term`, `tweets\_per\_day`, `most\_active\_user`) is designed to perform a specific task. This modular approach allows for easier testing, debugging, and reuse of code.

2. Data Type Handling:

- Ensuring that each column is in the correct data type is critical for accurate and efficient analysis. For example, treating dates as `datetime` objects allows for easy extraction of day, month, or hour components.

3. Use of `pandas` and `matplotlib`:

- `pandas` is chosen for data manipulation due to its flexibility and powerful functions for handling structured data.  
- `matplotlib` is used for visualization because it provides a simple yet versatile way to create various plots and charts, making the data more interpretable.

4. Interactive and Reusable:

- The system is designed to be interactive, allowing users to easily change the term they are querying. It is also reusable; you can use the same functions for different datasets with minimal adjustments.

# Step 5: Running the System on Your Computer

To run this system on your own computer:  
1. Follow the setup instructions to ensure your environment is ready.  
2. Load your dataset and convert data types as necessary.  
3. Use the provided functions to filter the data and answer your specific queries.  
4. Visualize the results using `matplotlib` to gain insights.

# Conclusion

This system is designed to be straightforward and user-friendly, enabling you to perform complex data queries and analysis on tweet data. The design choices made were centered around flexibility, ease of use, and the ability to provide meaningful insights from your data.