Project Overview:

In this project, I built a scalable data pipeline that processes user interaction data in real-time and stores it in a NoSQL database while performing aggregations. I then visualized the data using a dashboard. The project was divided into three core problems with an optional bonus task.

Problem 1: Random Data Generator and Kafka Producer

Objective:
Create a random data generator that simulates user interaction logs at scale and sends the generated data to a Kafka topic.

Approach:

- I implemented a **random data generator** that creates the following fields for each interaction:
  - `user_id`: A unique identifier for each user.
  - `item_id`: Identifier for the item interacted with.
  - `interaction_type`: Type of interaction (e.g., click, view, purchase).
  - `timestamp`: The time when the interaction occurred.
- The **Kafka producer** continuously sends this simulated data to a Kafka topic in real-time, ensuring scalability. The rate of data generation can be controlled through parameters such as batch size, interval time, and speed multiplier.

Libraries Used:

- `kafka-python`: To connect and produce messages to a Kafka topic.

Problem 2: Kafka Consumer and Real-Time Aggregations

Objective:
Consume the user interaction data from Kafka in real-time and perform aggregations like calculating averages, minimum/maximum interactions, and storing them in a NoSQL database.

Approach:

- I created a **Kafka consumer** that reads the messages from the Kafka topic created in Problem 1.
- For real-time aggregations, the consumer calculates:

    - Average number of interactions per user.
    - Maximum and minimum interactions per item.
- The aggregated data is then ingested into a **NoSQL database** (MongoDB was used in this case) for further use in visualization. The database schema was designed to store the aggregated results efficiently.

Libraries Used:

- pymongo: To interact with MongoDB for storing the aggregated results.
- kafka-python: For consuming data from Kafka.

Problem 3: Data Visualization and Dashboarding

Objective:
Visualize the real-time aggregation results in a simple, user-friendly dashboard.

Approach:

- I built a **dashboard** that pulls the aggregated data from the NoSQL database (MongoDB) and displays it in real-time.
- The dashboard includes visualizations of:
    - Average interactions per user.
    - Maximum and minimum interactions per item.
    - Any other relevant metrics (e.g., total interactions per user).
- I used **Plotly** for interactive plots and **Dash** to create a web-based dashboard that updates as new data is aggregated.

Libraries Used:

- dash: To create the web-based dashboard.
- plotly: To generate interactive plots for real-time visualization.

Bonus Task (Optional): Implementing Alerts

Objective:
Implement a mechanism to send alerts when specific thresholds are exceeded, such as when interaction counts surpass a set limit.

Approach:

- I added a basic alerting system where, when the number of interactions for an item exceeds a threshold, an alert is triggered. This could be integrated with an email or messaging service for real-time notifications.

Libraries Used:

- This feature was implemented using basic Python logic. (If you used any libraries for the alerting, list them here).

Design Choices

1. **Data Generation and Kafka Producer**:
    - The random data generator was designed to simulate user interactions at scale with configurable parameters to control the data rate. Kafka was chosen for real-time data streaming due to its scalability and performance in handling high-throughput data streams.
2. **Kafka Consumer and Aggregations**:
    - MongoDB was chosen as the NoSQL database due to its scalability and ease of integration with Python. I performed real-time aggregations on the consumer side to minimize latency and ensure that the data was processed as soon as it was generated.
3. **Dashboarding and Visualization**:
    - Plotly and Dash were selected because of their ease of use for creating interactive, web-based dashboards in Python. This allowed me to integrate the real-time data retrieval with visualizations to monitor key metrics.

Tools and Libraries Used:

- **Kafka**: For real-time streaming of interaction data.
- **MongoDB**: NoSQL database for storing aggregated results.
- **Dash/Plotly**: For building interactive dashboards.
- **Python**: Primary language used for implementing the entire pipeline.
