Objective

The purpose of this assignment is to evaluate your ability to work with real-world data, perform preprocessing, analysis, and prepare results that could be further integrated into APIs or downstream applications.

This is not an optimisation task – the goal is to demonstrate your **end-to-end data science workflow**: from data ingestion, cleaning, analysis, through text embeddings, clustering, and ranking, up to designing an API endpoint with a basic model/service integration.

You are expected to spend max **7 days** on this assignment.

Provided Dataset

You will receive a CSV file (`articles_dataset.csv`) containing **50 web articles** from 3 domains:

- **Technology**
- **Health**
- **Sports**

Each record contains:

- `id` (unique identifier)
- `date` (publication date)
- `url` (link to article)
- `title` (headline)
- `summary` (short description)
- `category` (label for ground truth analysis)

Tasks

1. **Data Preparation & Exploration**
   - Load the dataset and explore the basic statistics (number of articles per category, word counts, etc.).
   - Handle missing or inconsistent values.
2. **Text Preprocessing**
   - Tokenize, clean (stop words, punctuation), and normalize the text.
   - Extract basic entities (e.g., named entities for people, organizations, places).
   - Compute text embeddings (e.g., using `sentence-transformers` or OpenAI embeddings).
3. **Clustering & Ranking**
   - Perform clustering of articles based on embeddings (e.g., KMeans or hierarchical clustering).
   - Rank articles within each cluster based on a scoring method (e.g., cosine similarity to cluster centroid, or frequency of keywords).
4. **Basic Analysis**
   - Show how well the unsupervised clusters align with the provided `category` field.
   - Comment on where clustering succeeds or fails.
5. **LLM Integration (Optional, Simplified) - some free option if available**
   - Demonstrate a simple connection to an LLM (e.g., via OpenAI API, Hugging Face, or a mock service).
   - Example: Summarize the 3 most representative clusters using the LLM.
6. **API Endpoint**
   - Implement a **FastAPI** (or similar) endpoint that takes an article text as input and:

        - Cleans and preprocesses it,
        - Generates embeddings,
        - Assigns it to the closest cluster,
        - Returns a JSON response with top 3 clusters.

7. **Performance Considerations**
    - Use **multithreading** or asynchronous processing for expensive operations (e.g., embedding calculation).
    - Ensure memory efficiency in your implementation.
8. **Final Deliverables**
    - A Jupyter Notebook (or `.py` scripts) containing all steps.
    - A short written **commentary** (1–2 pages) describing:
        - Initial results,
        - Observations and limitations,
        - Suggestions on how results could be improved with more time and resources.
    - The FastAPI app (example Github code).

Evaluation Criteria

- Correctness and clarity of code.
- Ability to handle the dataset end-to-end.
- Understanding of text preprocessing and embeddings.

- Reasonable clustering and ranking approach.
- Clear commentary on results and tuning opportunities.
- Basic FastAPI endpoint functionality.