Problem Statement: Automated Data Query and Retrieval System Using Offline(free & open source) Large Language Models With CSV, MongoDB, LlamaIndex, and LangChain

Overview

You are required to develop an automated data query and retrieval system using Large Language models (LLM with open source & freely available) . The goal of this assignment is to demonstrate your ability to work with CSV data, interact with a MongoDB database, and utilize a language model (LLM) to generate MongoDB queries dynamically based on user inputs.

Requirements or Steps to Follow

1. **CSV Data Management:**
   - You will be provided with a CSV file containing various columns of data.
   - Your first task is to write a Python script to load this data into a MongoDB collection.
   - Each row of the CSV should be stored as a separate document in the MongoDB database.

2. **Dynamic Query Generation using LLM:**
   - The next step involves building a Python-based interface where the user can input the name of a CSV column header.
   - Based on the user's input, you will use an LLM to generate a MongoDB query that can retrieve relevant data from the database.
   - Ensure that the generated query is both syntactically correct and logically sound for the given input.

3. **Data Retrieval and Presentation:**
   - Execute the MongoDB query generated by the LLM to fetch the required data from the database.
   - Once the data is retrieved, you have two options for presenting it:

        - **Display the Data:** Present the data to the user in a human-readable format (e.g., a table or printed output).
        - **Save the Data:** Save the retrieved data back into a new CSV file that the user can download or view.Give names to files as per test cases.(ex. test_case1.csv etc)

4. **User Interaction:**
    - The system should be user-friendly, allowing the user to input column names, ask questions about the data, and choose whether to display or save the results.

5. **Error Handling:**
    - Implement robust error handling to manage cases where:
        - The user inputs an invalid or non-existent column name.
        - The LLM generates an incorrect or incomplete query.
        - There are issues with MongoDB connectivity or data retrieval.

Additional Considerations

- **Security:** Ensure that the system is secure, particularly when interacting with the LLM and MongoDB.
- **Efficiency:** The system should be optimized for performance, especially when handling large CSV files or complex queries.
- **Scalability:** Consider how the system might be scaled to handle multiple CSV files, larger datasets, or more complex user queries.
- **Documentation:** Provide clear documentation explaining how to use the system, including any setup or installation instructions.

Deliverables

1. **Python Scripts:**
   - You have to provide the end to end python script which covers all steps mentioned above in a single script only.
   - A script to load CSV data into MongoDB.
   - A script or module to generate and execute MongoDB queries using an LLM based on user inputs.
   - A script to display or save the retrieved data.
2. **Documentation:**
   - A README file with detailed instructions on how to set up and use the system.
   - Documentation of the code, including comments and explanations for key functions.
3. **Test Case Output:**
   - Provide test cases demonstrating the system's functionality, including edge cases and error scenarios.
4. **Output Data:**
   - Include a sample CSV file that can be used to test the system.
   - You have to save the Query generated by the model for each test case and put it in one file name as Quries_generated.txt and send it to us.

   For Ex. What are the products with a price greater than $50?

   Query generated by Model - db.collection.find({ "Price": { "$gt": 50 } })

Example Use Case

- A user uploads a CSV file containing information about products in a store (e.g., Product ID, Name, Price, Category).
- The user then inputs a column name, such as "Price", and asks, "What are the products with a price greater than $50?"
- The system generates the appropriate MongoDB query, retrieves the data, and either displays it or saves it as a new CSV file.

You have to consider the following Test Cases and send the output for 3 test cases in csv format along with code files and a Query generated by offline model for respective data :

1. Find all products with a rating below 4.5 that have more than 200 reviews and are offered by the brand 'Nike' or 'Sony'.
2. Which products in the Electronics category have a rating of 4.5 or higher and are in stock?
3. List products launched after January 1, 2022, in the Home & Kitchen or Sports categories with a discount of 10% or more, sorted by price in descending order.