Take Home Test Data Engineer

Project Scenario

A company, XYZ, is a large retailer that wants to develop a new online platform to sell grocery products. As a data engineer, your role will involve designing and constructing a comprehensive system for price elasticity analysis. This includes tasks such as crawling competitor product price data, designing the data system, and building machine learning systems.

It's important to note that the websites from which the data will be crawled may change without prior notice. Therefore, the system should be designed to handle these changes and continue to function without any interruptions. This could involve implementing a monitoring system that alerts the team of any changes to the website structure or using a web scraping tool that can adapt to website changes automatically.

The system should be designed to handle a large volume of data and be scalable to accommodate future growth. The data should be cleaned and pre-processed to ensure accuracy and consistency. The machine learning component of the system should be able to analyze the price data and provide pricing recommendations.

The system should expose an API that provides access to the processed data and pricing recommendations. The API should be designed to be fast, secure, and protect sensitive data. It should be deployable on cloud infrastructure such as AWS, AliCloud, or other popular cloud platforms, to make it easily scalable, cost-effective, and reliable.

You will need to collaborate closely with the data science and business teams to ensure that the system meets their needs and aligns with the company's goals.

Implementation Guidelines

As part of the interview process, we would like you to build a mini-project that covers all the requirements mentioned in the scenario. The project should demonstrate your ability to design and build an end-to-end system for price elasticity analysis, including data crawling, data processing, machine learning, and API development. Here are the details and requirements for the mini-project:

1. **Data Crawling**
    - Use a Python web scraping library such as BeautifulSoup, Scrapy, Selenium, etc. to crawl product price data from websites.
    - Write a Python script that performs the crawling and saves the data to a database.
    - Implement the logic to handle website structure changes, ensuring the system can adapt to changes without interruptions.

- Implement logging and error handling mechanisms to facilitate monitoring and troubleshooting.

2. **Data Processing**
   - Choose a suitable database such as PostgreSQL, MongoDB, ElasticSearch, etc., to store and process the crawled data.
   - Create a Docker container for the chosen database and configure it in the Docker Compose file.
   - Write Python code to clean and preprocess the data, ensuring accuracy and consistency.
   - Develop a Python script to connect to the database, store the processed data, and perform any necessary queries.
3. **Machine Learning**
   - No need to develop the actual machine learning model. Instead, you can develop a dummy model that simulates the behavior of analyzing price data and generating pricing recommendations.
   - The prediction will be performed batch-wise. This means that the system should be able to process and analyze a large volume of price data in batches and generate pricing recommendations periodically or at specified intervals.
4. **API Development**
   - Utilize a web framework like Flask or FastAPI to develop the API for accessing the processed data and pricing recommendations.
   - Define appropriate API endpoints and routes to handle data retrieval and pricing recommendation requests.
   - Implement authentication and authorization mechanisms to secure the API endpoints.

Data Requirements

The system will utilize three tables to store the necessary data: productmaster, product, and pricerecommendation. The data requirements for each table are as follows:

- **productmaster**:
  - `id`: Unique identifier for the product master.
  - `type`: Type or category of the product.
  - `name`: Name of the product.
  - `detail`: Additional details or description of the product.

- **product**:
  - `id`: Unique identifier for the product entry.
  - `name`: Name of the product.
  - `price`: Current price of the product.
  - `originalprice`: Original price of the product.
  - `discountpercentage`: Discount percentage applied to the product.
  - `detail`: Additional details or description of the product.
  - `platform`: Platform from which the product data was crawled.
  - `productmasterid`: Foreign key referencing the product master associated with this entry.
  - `createdat`: Timestamp indicating the date and time when the data was crawled.
- **pricerecommendation**:
  - `productmasterid`: Foreign key referencing the product master associated with this recommendation.
  - `price`: Predicted price recommendation for the corresponding product master.
  - `date`: Date for which the price recommendation is generated.

You can crawl the product data from the following URLs:

- `https://www.klikindomaret.com/page/unilever-officialstore`
- `https://www.blibli.com/cari/unilever%20indonesia%20official?seller=Official+Store&category=53400`
- `https://www.tokopedia.com/unilever/product`

You can find the sample data in the following link: here

Note that these data requirements serve as a starting point, and you can adjust the data format as needed to align with your design.

You can use Docker Compose to build and run the entire system locally. The containers will ensure that the necessary dependencies are included and provide an isolated environment for the project. Although deploying the system on a cloud platform is not required for this test, it would be beneficial if you could prepare the project in a way that makes it easier to deploy in the cloud with minimal effort.

Once you have implemented the project, please upload the code to a GitHub repository and provide proper documentation on how to run the system locally.

During your interview, be prepared to explain your implementation choices, demonstrate the functionality of the system, and discuss how the project can be further scaled and improved.