E-DataSync: Distributed E-Commerce Inventory Management

Team Members:

Christopher Apton (apton@usc.edu)

Daniel Jang (hyundong@usc.edu)

Shahzaib Saqib Warraich (warraich@usc.edu)

Background and Skills:

Christopher Apton:

I am experienced with time-series analysis along with coding experience. I've done a lot of work with data analysis, classical machine learning methods, along with more advanced methods. I've worked with a Public Editor in Berkeley for a year at a startup where I learned how to contribute to front and back-end development. My undergraduate degree is in Data Theory with a minor in Bioinformatics where I worked with large-scale genomic data. I'm proficient in Python, R, SQL, C++, Java, HTML, Javascript, Typescript, Matlab, and Tableau.

Daniel Jang:

I have extensive experience in data analysis and am proficient in Python, including its libraries essential for exploratory data analysis and machine learning. During my Data Science undergrad at UC Berkeley, I dove into collaborative projects year-round. These experiences not only boosted my technical skills but also polished my teamwork and communication in a collaborative setting. Currently, I am employed as a data analyst at Gori Company, where I specialize in analyzing vast sets of logistics data to contribute to the company's overall growth. In this role, I focus on diverse tasks such as fraud detection, predicting client volumes and trends, and ensuring the generation of accurate and cleaned data for the accounting team.

Shahzaib Saqib Warraich:

Experienced professional in CV and NLP with a 3-year background in academia and industry. Authored publications in Deep Learning, collaborated with startups, and served as an AI Research Engineer for impactful projects with clients like Siemens, Honda, Bosch, and Schneider Electric. Proficient in Python and SQL, with expertise in AI, cloud computing, and structured/unstructured databases.

Project Requirements:

- Develop a hash function that uniquely identifies data and partitions the data into distributed databases.
- Develop a user interface which expert database managers can use to process the data in the distributed database by using features such as insertion, modification, and deletion of records.
- Develop a web application which allows end users to access the database and interact with it.

Planned implementation:

An E-commerce inventory management system that allows database managers and end users to communicate and interact with the distributed databases. We will have three relational tables in the distributed databases including *products*, *suppliers*, and *orders.* As of now, we have data for the *products* table and plan to generate synthetic datasets for *suppliers* and *orders* tables based on the original *product* data. We will be using the **product names** in the *product* tables to split (based on hash values) the data into multiple distributed databases. The purpose of this web application is to manage inventory and its audience is mainly targeted for inventory managers who are managing large e-commerce databases.

Relations:

Table #1: *Products* (Real data), link: Kaggle

Columns: *product, category, sub-category, brand name, sale price, market price, type, rating, description*

Primary key: product

Table #2: *Suppliers* (Synthetic data)

Columns: *brand name, address, description, founding year, number of products*

Primary key: brand name

Table #3: *Orders* (Synthetic data)

Columns: *date, product, quantity sold.*

Primary key: (date, product)

User Interface:

- The Database Manager selects the relevant database table (*products*, *suppliers*, or *orders*) they want to interact with.
- Command Line Interface (CLI) which allows DB Managers to either add, modify, or delete record(s) from the chosen database table.
- An additional option to get record(s) from the chosen database table.

Web Application:

- Create a homepage that allows users to select the relevant database table (*products*, *suppliers*, or *orders*) they want to interact with .
- Based on the selected database table, direct the user to another page that has four main buttons: *insert*, *modify*, *delete*, and *search*.
- Each of the four buttons will direct to a new page and prompt the user to interact with the interface.
    - Text fields for users to enter data.
    - Additional small buttons will be available: "*Go back*", "*Save*" and "*Empty*".
        - First one will go back to the homepage page without saving anything.

- Second one will save all the entered records and send the request to the database table.
- Third one will empty all fields if the user has either saved or entered some data fields.

Functionality of buttons:

- *Insert*
    - 1xN blank data fields. All of them are mandatory. Additional small buttons will be available: "*Go back*", "*Save*" and "*Empty*".
- *Modify*
    - The *unique product name* will be entered based on which the current record will be retrieved and displayed from the relevant database table. Additional small buttons will be available: "Go back", "Save" and "Empty".
    - Afterwards, 1xN blank data fields will be displayed and none of them will be mandatory. Additional small buttons will be available: "*Go back*", "*Save*" and "*Empty*".
- *Delete*
    - The *unique product name* will be entered and the corresponding record will be deleted. Additional small buttons will be available: "*Go back*", "*Save*" and "*Empty*".
- *Search*

- The *unique product name* will be entered and the corresponding record will be retrieved from the databases. Additional small buttons will be available: "*Go back*", "*Save*" and "*Empty*".

Frameworks/Libraries/Tools/Tech Stack: Python, JavaScript, MySQL/MongoDB, MERN/MEAN stack, Streamlit

Team Responsibilities:

Christopher: Backend

Daniel: Frontend

Shahzaib: Distributed Database Management

Timeline:

Event	Due Date
In-person group meeting. Pitch in ideas. Discuss how we should tackle the project and work on proposal as a team	2/2
Clean dataset and generate synthetic data for our relational tables. Create required features, test and make sure the features work.	3/1
Make sure both of the deliverables are working to demonstrate in class. Show that we are able to use the main features on our web applications. Everything must be fully implemented at this time.	4/17
Finish up with the last details. Add up any additional features that we might be interested in. Make sure everything is working as intended and create a final report.	5/3