Term: Fall 2024 **Subject:** Computer Science & Engineering (CSE) **Number:** 512

Course Title: Distributed **D**atabase **S**ystems (CSE 512)

Assignment 2

Due on Monday, October 28th at 11:59 pm

This assignment is designed to help you understand data processing in the database layer by using the aggregation pipeline on a distributed No-SQL database management system, MongoDB. The objective is that you can perform special operations like group and match to the data using ***aggregation pipeline***, run the **MongoDB** image in a **Docker** container, and answer the questions by writing the necessary set of **Python** functions using the **Pymongo** connection.

Please submit a zip folder of your codes and a pdf document with the results.

If you need more information about how to implement Docker with Python and MongoDB, please check the Hands-on Session under the folder name - Implementation of docker with python and mongodb. (starting from 16:00th minute)

As data is generated randomly, the results must be different for each student.

ASU Academic Integrity:
Students in this class must adhere to ASU's academic integrity policy, which can be found at https://provost.asu.edu/academic-integrity/policy. If you are caught cheating, you will be reported to the Fulton Schools of Engineering Academic Integrity Office (AIO). The AIO maintains a record of all violations and has access to academic integrity violations committed in all other ASU colleges/schools.

Below are the steps you need to follow to set the environment and fulfill this assignment:

1. Install Docker by following the link and test with the "**hello-world**" image.
https://docs.docker.com/get-docker

For Windows, WSL (Windows Subsystem for Linux) is required. Run the installer as administrator. You can use this link for WSL installation: https://learn.microsoft.com/en-us/windows/wsl/install

For Ubuntu/Linux, you need to install and run with "sudo".

After installation, run these commands below. This will print a message that Docker is installed and running. **Make sure you have a running docker**.

```
docker pull hello-world
docker run hello-world
```

2. Create a Python project. Open a shell and run as administrator. Go to your project directory in the shell and install **pymongo** and virtualenv. Define your virtual environment and activate it. Activation commands are various for operating systems. Please write the correct command and run your python file. Note - Setting up a Virtual Environment is **Optional**.

```
cd your-project-directory
pip install virtualenv
pip install pymongo
virtualenv venv -p python3
source ./venv/bin/activate for Linux and Mac
.\venv\Scripts\Activate for Windows
python your-file-name.py
```

3. Generate 100 Mock data in JSON format.

4.Download the assignment2.py file and implement the required Python functions explained below. Run your python file by using the instructions above.

5. Write a report and add results and screenshots from the Python project.

Part 1 (5 points): Run MongoDb in a Docker container.
Pull the MongoDb image from the DockerHub and run it using specific port configurations. Run **docker ps** command take a **screenshot of the console** to add it to the report.

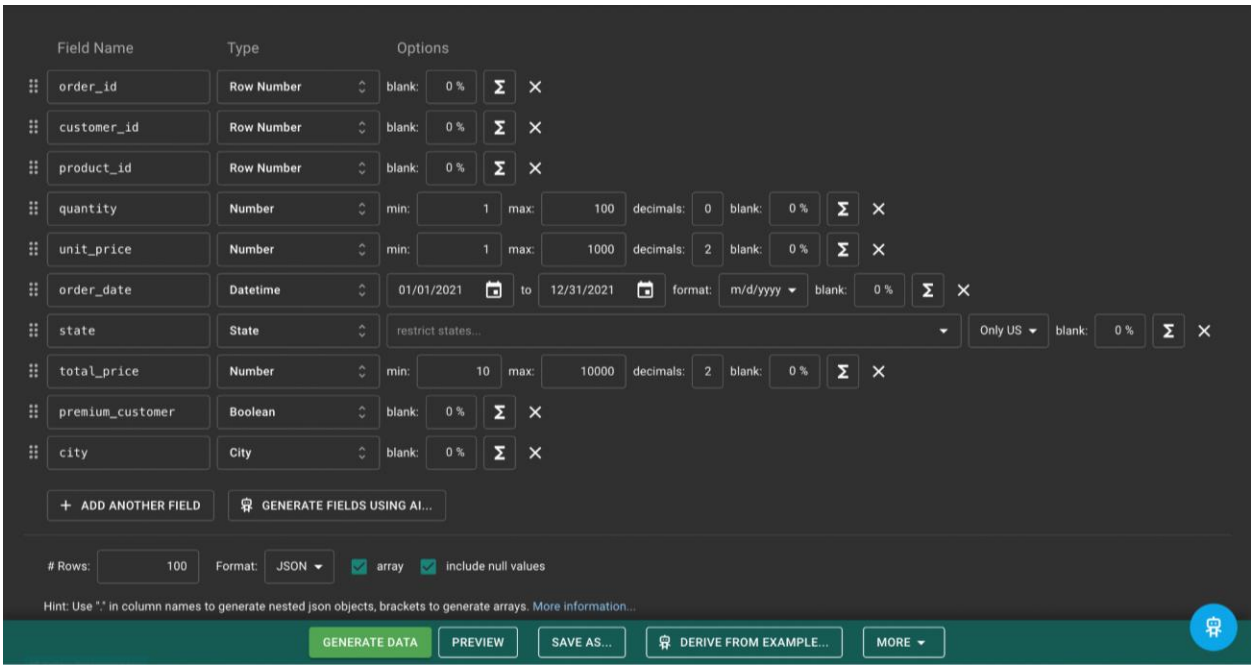
Part 2 (5 points): Generate 100 Mock Data in JSON format.

Use the website: https://www.mockaroo.com/

Click “**Generate Fields Using AI**” button.
Copy the sample JSON data below and paste it to the example data part on the website


```
{
  "order_id": 1,
  "customer_id": 1,
  "product_id": 1,
  "quantity": 95,
  "unit_price": 276.7,
  "order_date": "10/21/2021",
  "state": "California",
  "total_price": 3357.43,
  "premium_customer": true,
  "city": "Northridge"
}
```


Then click the “**Replace Existing Data Fields”** button.
Change the **All Countries** field with **Only US** at state row.
Change the number of rows from **1000 to 100** and format style from **CSV to JSON** at the bottom of the page like the image below.
Click the Generate **Data** button to create a mock data file and download it.



Part 3 (5 points): insert_mock_data()
Implement a Python function **insert_mock_data()** that **inserts** the generated Mock data in a **JSON file** into the **MongoDB**. **Print** the ObjectId of the data. Please take a **screenshot of the console** and add it to the report.

Database name = **ecommerce**
Collection name = **orders**

```
Data inserted successfully.
[('66fdbb3ea786b8d4d8d363d5',), ('66fdbb3ea786b8d4d8d363d6',), ('66fdbb3ea786b8d4d8d363d7',), ('66fdbb3ea786b8d4d8
d363d8',), ('66fdbb3ea786b8d4d8d363d9',), ('66fdbb3ea786b8d4d8d363da',), ('66fdbb3ea786b8d4d8d363db',), ('66fdbb3e
a786b8d4d8d363dc',), ('66fdbb3ea786b8d4d8d363dd',), ('66fdbb3ea786b8d4d8d363de',), ('66fdbb3ea786b8d4d8d363df',),
('66fdbb3ea786b8d4d8d363e0',), ('66fdbb3ea786b8d4d8d363e1',), ('66fdbb3ea786b8d4d8d363e2',), ('66fdbb3ea786b8d4d8d
363e3',), ('66fdbb3ea786b8d4d8d363e4',), ('66fdbb3ea786b8d4d8d363e5',), ('66fdbb3ea786b8d4d8d363e6',), ('66fdbb3ea
```

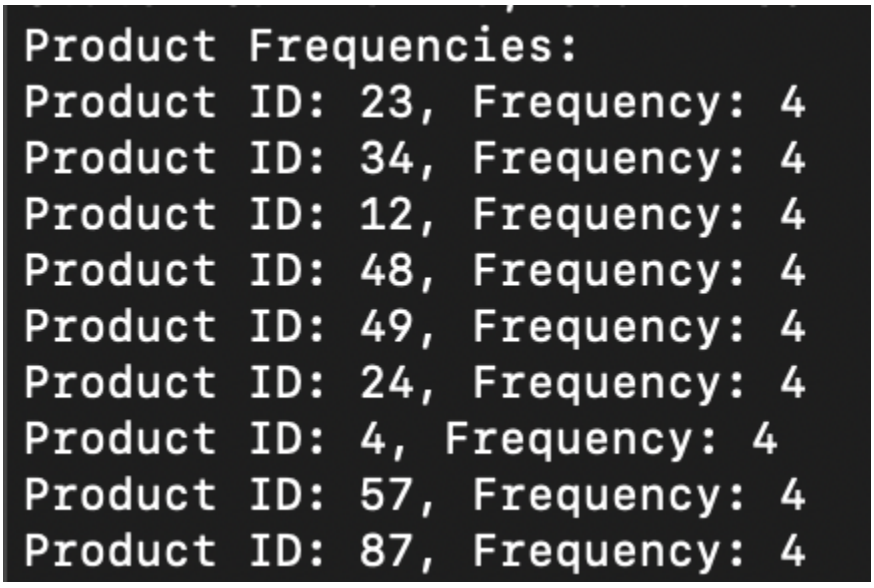
Part 4 (15 points): `find_order_totals()`

Using the database generated from the previous step, implement a Python function `find_order_totals()` that finds the total number of orders and the number of orders per state, sorted by count in ascending order using the aggregation pipeline of MongoDB. Print the result to the console. Take a screenshot of the console and add it to the report.

```
Total number of orders: 300
Number of orders per state:
State: Utah, Count: 3
State: Nebraska, Count: 3
State: Mississippi, Count: 3
State: Connecticut, Count: 3
State: North Dakota, Count: 3
State: Iowa, Count: 3
State: Pennsylvania, Count: 3
State: New Jersey, Count: 3
State: Minnesota, Count: 3
State: Michigan, Count: 3
State: Arkansas, Count: 3
State: Maryland, Count: 3
State: Washington, Count: 3
State: West Virginia, Count: 6
State: Oklahoma, Count: 6
```

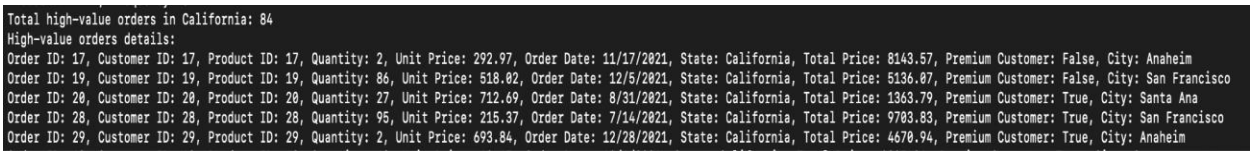
Part 5 (15 points): `find_product_frequencies()`

Implement a Python function `find_product_frequencies()` that finds the products and their frequencies sorted by frequency in descending order. Print the result to the console. Take a screenshot of the console and add it to the report.



Part 6 (10 points): `ca_highvalue_orders()`

Implement a Python function `ca_highvalue_orders()` that counts and finds the orders in California where the order amount exceeds $1,000. Print the count and orders' information. Take a screenshot of the console and add it to the report.



Part 7 (15 points): `top_states_highvalue()`

Implement a Python function `top_states_highvalue()` that finds the top ten states with the most orders where the order amount exceeds $500 using the aggregation framework. Print the rank, state, and order count. Take a screenshot of the console and add it to the report.

```
Top 10 States with High-Value Orders (>$500):
Rank 1: State: California, Order Count: 98
Rank 2: State: Texas, Order Count: 84
Rank 3: State: North Carolina, Order Count: 49
Rank 4: State: Florida, Order Count: 49
Rank 5: State: Louisiana, Order Count: 35
Rank 6: State: Arizona, Order Count: 35
Rank 7: State: Missouri, Order Count: 28
Rank 8: State: New York, Order Count: 28
Rank 9: State: South Carolina, Order Count: 21
Rank 10: State: Nevada, Order Count: 21
```

Part 8 (10 points): `find_customer_premium()`

Implement a Python function `find_customer_premium()` that counts and finds the customers who have placed premium orders (order amount exceeds $2,000) in Texas. Print the count of customers and their information. Take a screenshot of the console and add it to the report.

```
Total premium customers in Texas: 5
Premium customers' details:
Customer ID: 31, Order ID: 31, Product ID: 31, Quantity: 4, Total Price: 9509.41, Order Date: 11/3/2021,
Premium Customer: True, City: Tyler, State: Texas
Customer ID: 45, Order ID: 45, Product ID: 45, Quantity: 55, Total Price: 4362.59, Order Date: 5/8/2021,
Premium Customer: True, City: Dallas, State: Texas
Customer ID: 78, Order ID: 78, Product ID: 78, Quantity: 44, Total Price: 7431.74, Order Date: 8/1/2021,
Premium Customer: True, City: San Antonio, State: Texas
```

Part 9 (10 points): `find_orders_by_date()`

Implement a Python function `find_orders_by_date()` that counts and finds the orders placed in New York City on a specific date of your choice.. Print the count of orders and their details. Take a screenshot of the console and add it to the report.

```
Total orders placed in New York City on 1/9/2021: 13
Order details:
Order ID: 36, Customer ID: 36, Product ID: 36, Quantity: 43, Total Price: 8898.32, Order Date: 1/9/2021,
Premium Customer: False, City: New York City, State: New York
Order ID: 36, Customer ID: 36, Product ID: 36, Quantity: 43, Total Price: 8898.32, Order Date: 1/9/2021,
Premium Customer: False, City: New York City, State: New York
Order ID: 36, Customer ID: 36, Product ID: 36, Quantity: 43, Total Price: 8898.32, Order Date: 1/9/2021,
Premium Customer: False, City: New York City, State: New York
Order ID: 36, Customer ID: 36, Product ID: 36, Quantity: 43, Total Price: 8898.32, Order Date: 1/9/2021,
Premium Customer: False, City: New York City, State: New York
Order ID: 36, Customer ID: 36, Product ID: 36, Quantity: 43, Total Price: 8898.32, Order Date: 1/9/2021,
Premium Customer: False, City: New York City, State: New York
```

Part 10 (10 points): Writing a report.

Write a report to explain what the aggregation framework is and how it is used. Please add results and screenshots from the Python project and MongoDB program.

---The END---