

Distributed Database Systems

FALL – 2025

Assignment -1

Ashish Raj Shekhar

1223100216

1. Docker screenshot

```
ashishrajshekar@Ashishs-MacBook-Pro Assign-1 % docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED    STATUS    PORTS                NAMES
16a18555e97a   mongo-assign1  "docker-entrypoint.s..." 6 hours ago Up 6 hours    0.0.0.0:27017->27017/tcp  mongo-assign1
```

The screenshot shows the Docker Desktop application window. The left sidebar contains navigation options: Containers (selected), Images, Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The main panel displays the 'Containers' section with a search bar and a toggle for 'Only show running containers'. Below this is a table of running containers. The table has columns for Name, Container ID, Image, Port(s), CPU (%), Last started, and Actions. Two containers are listed: 'fairtestai-db' and 'mongo-assign1'. The 'mongo-assign1' container is highlighted with a green dot, indicating it is running. Below the table, there are two walkthroughs: 'Multi-container applications' and 'Containerize your application'. The bottom status bar shows 'Engine running', RAM usage (1.91 GB / 16 GB), CPU usage (0.13%), and disk usage (19.31 GB / 58.37 GB).

Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
fairtestai-db	39717c8ba5e7	postgres:15	5432:5432	0%	11 days ago	[Play] [Stop] [Refresh] [Delete]
mongo-assign1	16a18555e97a	mongo-assign1	27017:27017	0.95%	6 hours ago	[Play] [Stop] [Refresh] [Delete]

2. Mockaroo json dataset creation screenshot

The screenshot displays the Mockaroo web interface for creating a JSON dataset. The top navigation bar includes links for SCHEMAS, DATASETS, APIS, PROJECTS, SCENARIOS, FUNCTIONS, DATABASES, and DE-IDENTIFY, along with settings, help, and user options (SIGN IN, UPGRADE NOW).

The main configuration area shows a table of fields with their names, types, and options:

Field Name	Type	Options
order_id	Row Number	blank: 0 %
customer_id	Row Number	blank: 0 %
product_id	Row Number	blank: 0 %
quantity	Number	min: 0 max: 100 decimals: 0 blank: 0 %
unit_price	Number	min: 0 max: 1000 decimals: 1 blank: 0 %
order_date	Datetime	10/21/2021 to 10/21/2021 format: m/d/yyyy blank: 0 %
state	State	restrict states... Only US blank: 0 %
total_price	Number	min: 0 max: 10000 decimals: 2 blank: 0 %
premium_customer	Boolean	blank: 0 %
city	City	blank: 0 %

Below the fields, there are options for the number of rows (100), format (JSON), and checkboxes for array and include null values. A hint suggests using "." in column names for nested objects and brackets for arrays.

The bottom section shows the generated JSON data, starting with a sample object:

```
{ "order_id": 1, "customer_id": 1, "product_id": 1, "quantity": 16, "unit_price": 399.4, "order_date": "10/21/2021", "state": "District of Columbia", "total_price": 6590.4 }
```

The data is displayed as a list of 41 objects, each representing a row in the dataset. The fields are: order_id, customer_id, product_id, quantity, unit_price, order_date, state, total_price, premium_customer, and city.

3. Insert the data using insert_many

```
(venv) ashishrajshekar@Ashishs-MacBook-Pro Assign-1 % python3 Assignment\ 1.py
Data inserted successfully:
[ObjectId('68de16013858a0ede4637f5e'), ObjectId('68de16013858a0ede4637f5f'), ObjectId('68de16013858a0ede4637f60'), ObjectId('68de16013858a0ede4637f61'), ObjectId('68de16013858a0ede4637f62'), ObjectId('68de16013858a0ede4637f63'), ObjectId('68de16013858a0ede4637f64'), ObjectId('68de16013858a0ede4637f65'), ObjectId('68de16013858a0ede4637f66'), ObjectId('68de16013858a0ede4637f67'), ObjectId('68de16013858a0ede4637f68'), ObjectId('68de16013858a0ede4637f69'), ObjectId('68de16013858a0ede4637f6a'), ObjectId('68de16013858a0ede4637f6b'), ObjectId('68de16013858a0ede4637f6c'), ObjectId('68de16013858a0ede4637f6d'), ObjectId('68de16013858a0ede4637f6e'), ObjectId('68de16013858a0ede4637f6f'), ObjectId('68de16013858a0ede4637f70'), ObjectId('68de16013858a0ede4637f71'), ObjectId('68de16013858a0ede4637f72'), ObjectId('68de16013858a0ede4637f73'), ObjectId('68de16013858a0ede4637f74'), ObjectId('68de16013858a0ede4637f75'), ObjectId('68de16013858a0ede4637f76'), ObjectId('68de16013858a0ede4637f77'), ObjectId('68de16013858a0ede4637f78'), ObjectId('68de16013858a0ede4637f79'), ObjectId('68de16013858a0ede4637f7a'), ObjectId('68de16013858a0ede4637f7b'), ObjectId('68de16013858a0ede4637f7c'), ObjectId('68de16013858a0ede4637f7d'), ObjectId('68de16013858a0ede4637f7e'), ObjectId('68de16013858a0ede4637f7f'), ObjectId('68de16013858a0ede4637f80'), ObjectId('68de16013858a0ede4637f81'), ObjectId('68de16013858a0ede4637f82'), ObjectId('68de16013858a0ede4637f83'), ObjectId('68de16013858a0ede4637f84'), ObjectId('68de16013858a0ede4637f85'), ObjectId('68de16013858a0ede4637f86'), ObjectId('68de16013858a0ede4637f87'), ObjectId('68de16013858a0ede4637f88'), ObjectId('68de16013858a0ede4637f89'), ObjectId('68de16013858a0ede4637f8a'), ObjectId('68de16013858a0ede4637f8b'), ObjectId('68de16013858a0ede4637f8c'), ObjectId('68de16013858a0ede4637f8d'), ObjectId('68de16013858a0ede4637f8e'), ObjectId('68de16013858a0ede4637f8f'), ObjectId('68de16013858a0ede4637f90'), ObjectId('68de16013858a0ede4637f91'), ObjectId('68de16013858a0ede4637f92'), ObjectId('68de16013858a0ede4637f93'), ObjectId('68de16013858a0ede4637f94'), ObjectId('68de16013858a0ede4637f95'), ObjectId('68de16013858a0ede4637f96'), ObjectId('68de16013858a0ede4637f97'), ObjectId('68de16013858a0ede4637f98'), ObjectId('68de16013858a0ede4637f99'), ObjectId('68de16013858a0ede4637f9a'), ObjectId('68de16013858a0ede4637f9b'), ObjectId('68de16013858a0ede4637f9c'), ObjectId('68de16013858a0ede4637f9d'), ObjectId('68de16013858a0ede4637f9e'), ObjectId('68de16013858a0ede4637f9f'), ObjectId('68de16013858a0ede4637fa0'), ObjectId('68de16013858a0ede4637fa1'), ObjectId('68de16013858a0ede4637fa2'), ObjectId('68de16013858a0ede4637fa3'), ObjectId('68de16013858a0ede4637fa4'), ObjectId('68de16013858a0ede4637fa5'), ObjectId('68de16013858a0ede4637fa6'), ObjectId('68de16013858a0ede4637fa7'), ObjectId('68de16013858a0ede4637fa8'), ObjectId('68de16013858a0ede4637fa9'), ObjectId('68de16013858a0ede4637faa'), ObjectId('68de16013858a0ede4637fab'), ObjectId('68de16013858a0ede4637fac'), ObjectId('68de16013858a0ede4637fad'), ObjectId('68de16013858a0ede4637fae'), ObjectId('68de16013858a0ede4637faf'), ObjectId('68de16013858a0ede4637fb0'), ObjectId('68de16013858a0ede4637fb1'), ObjectId('68de16013858a0ede4637fb2'), ObjectId('68de16013858a0ede4637fb3'), ObjectId('68de16013858a0ede4637fb4'), ObjectId('68de16013858a0ede4637fb5'), ObjectId('68de16013858a0ede4637fb6'), ObjectId('68de16013858a0ede4637fb7'), ObjectId('68de16013858a0ede4637fb8'), ObjectId('68de16013858a0ede4637fb9'), ObjectId('68de16013858a0ede4637fba'), ObjectId('68de16013858a0ede4637fbb'), ObjectId('68de16013858a0ede4637fbc'), ObjectId('68de16013858a0ede4637fbd'), ObjectId('68de16013858a0ede4637fbe'), ObjectId('68de16013858a0ede4637fbf'), ObjectId('68de16013858a0ede4637fc0'), ObjectId('68de16013858a0ede4637fc1')]
```

4. Total number of orders and total orders grouped by state

```
Total number of orders: 100
Number of orders per state:
State: Alaska, Count: 1
State: Arkansas, Count: 1
State: Georgia, Count: 1
State: Kentucky, Count: 1
State: Louisiana, Count: 1
State: Maryland, Count: 1
State: Minnesota, Count: 1
State: Mississippi, Count: 1
State: New Jersey, Count: 1
State: North Carolina, Count: 1
State: North Dakota, Count: 1
State: Oklahoma, Count: 1
State: Tennessee, Count: 1
State: Utah, Count: 1
State: Alabama, Count: 2
State: District of Columbia, Count: 2
State: Illinois, Count: 2
State: Indiana, Count: 2
State: Michigan, Count: 2
State: Missouri, Count: 2
State: Nevada, Count: 2
State: Arizona, Count: 3
State: Colorado, Count: 3
State: Connecticut, Count: 3
State: Nebraska, Count: 3
State: Ohio, Count: 3
State: Washington, Count: 3
State: Wisconsin, Count: 3
State: Virginia, Count: 4
State: Florida, Count: 6
State: Massachusetts, Count: 6
State: Pennsylvania, Count: 6
State: California, Count: 9
State: New York, Count: 9
State: Texas, Count: 11
```

5. Product and their frequencies

```
Product Frequencies:
Product ID: 1, Frequency: 1
Product ID: 2, Frequency: 1
Product ID: 3, Frequency: 1
Product ID: 4, Frequency: 1
Product ID: 5, Frequency: 1
Product ID: 6, Frequency: 1
Product ID: 7, Frequency: 1
Product ID: 8, Frequency: 1
Product ID: 9, Frequency: 1
Product ID: 10, Frequency: 1
Product ID: 11, Frequency: 1
Product ID: 12, Frequency: 1
Product ID: 13, Frequency: 1
Product ID: 14, Frequency: 1
Product ID: 15, Frequency: 1
Product ID: 16, Frequency: 1
Product ID: 17, Frequency: 1
Product ID: 18, Frequency: 1
Product ID: 19, Frequency: 1
Product ID: 20, Frequency: 1
Product ID: 21, Frequency: 1
Product ID: 22, Frequency: 1
Product ID: 23, Frequency: 1
Product ID: 24, Frequency: 1
Product ID: 25, Frequency: 1
Product ID: 26, Frequency: 1
Product ID: 27, Frequency: 1
Product ID: 28, Frequency: 1
Product ID: 29, Frequency: 1
Product ID: 30, Frequency: 1
Product ID: 31, Frequency: 1
Product ID: 32, Frequency: 1
Product ID: 33, Frequency: 1
Product ID: 34, Frequency: 1
Product ID: 35, Frequency: 1
Product ID: 36, Frequency: 1
Product ID: 37, Frequency: 1
Product ID: 38, Frequency: 1
Product ID: 39, Frequency: 1
Product ID: 40, Frequency: 1
Product ID: 41, Frequency: 1
Product ID: 42, Frequency: 1
Product ID: 43, Frequency: 1
Product ID: 44, Frequency: 1
Product ID: 45, Frequency: 1
Product ID: 46, Frequency: 1
Product ID: 47, Frequency: 1
Product ID: 48, Frequency: 1
Product ID: 49, Frequency: 1
```

```
Problems Output Debug Console Terminal Ports
Product ID: 48, Frequency: 1
Product ID: 49, Frequency: 1
Product ID: 50, Frequency: 1
Product ID: 51, Frequency: 1
Product ID: 52, Frequency: 1
Product ID: 53, Frequency: 1
Product ID: 54, Frequency: 1
Product ID: 55, Frequency: 1
Product ID: 56, Frequency: 1
Product ID: 57, Frequency: 1
Product ID: 58, Frequency: 1
Product ID: 59, Frequency: 1
Product ID: 60, Frequency: 1
Product ID: 61, Frequency: 1
Product ID: 62, Frequency: 1
Product ID: 63, Frequency: 1
Product ID: 64, Frequency: 1
Product ID: 65, Frequency: 1
Product ID: 66, Frequency: 1
Product ID: 67, Frequency: 1
Product ID: 68, Frequency: 1
Product ID: 69, Frequency: 1
Product ID: 70, Frequency: 1
Product ID: 71, Frequency: 1
Product ID: 72, Frequency: 1
Product ID: 73, Frequency: 1
Product ID: 74, Frequency: 1
Product ID: 75, Frequency: 1
Product ID: 76, Frequency: 1
Product ID: 77, Frequency: 1
Product ID: 78, Frequency: 1
Product ID: 79, Frequency: 1
Product ID: 80, Frequency: 1
Product ID: 81, Frequency: 1
Product ID: 82, Frequency: 1
Product ID: 83, Frequency: 1
Product ID: 84, Frequency: 1
Product ID: 85, Frequency: 1
Product ID: 86, Frequency: 1
Product ID: 87, Frequency: 1
Product ID: 88, Frequency: 1
Product ID: 89, Frequency: 1
Product ID: 90, Frequency: 1
Product ID: 91, Frequency: 1
Product ID: 92, Frequency: 1
Product ID: 93, Frequency: 1
Product ID: 94, Frequency: 1
Product ID: 95, Frequency: 1
Product ID: 96, Frequency: 1
Product ID: 97, Frequency: 1
Product ID: 98, Frequency: 1
Product ID: 99, Frequency: 1
Product ID: 100, Frequency: 1
Total high-value orders in California: 7
High-value order details:
```

6. Orders where amount is greater than 1000 and in California

```
Product ID: 100, Frequency: 1
Total high-value orders in California: 7
High-value order details:
Order ID: 3, Customer ID: 3, Product ID: 3, Quantity: 91, Unit Price: 43.9, Order Date: 10/21/2021, State: California, Total Price: 5978.49, Premium Customer: False, City: Berkeley
Order ID: 16, Customer ID: 16, Product ID: 16, Quantity: 98, Unit Price: 346.9, Order Date: 10/21/2021, State: California, Total Price: 6574.93, Premium Customer: False, City: Garden Grove
Order ID: 26, Customer ID: 26, Product ID: 26, Quantity: 3, Unit Price: 811.5, Order Date: 10/21/2021, State: California, Total Price: 8123.25, Premium Customer: True, City: Long Beach
Order ID: 37, Customer ID: 37, Product ID: 37, Quantity: 50, Unit Price: 105.1, Order Date: 10/21/2021, State: California, Total Price: 5922.43, Premium Customer: True, City: San Bernardino
Order ID: 87, Customer ID: 87, Product ID: 87, Quantity: 26, Unit Price: 687.0, Order Date: 10/21/2021, State: California, Total Price: 6943.72, Premium Customer: True, City: Glendale
Order ID: 88, Customer ID: 88, Product ID: 88, Quantity: 15, Unit Price: 339.5, Order Date: 10/21/2021, State: California, Total Price: 3173.13, Premium Customer: True, City: Santa Clara
Order ID: 100, Customer ID: 100, Product ID: 100, Quantity: 94, Unit Price: 978.9, Order Date: 10/21/2021, State: California, Total Price: 9167.86, Premium Customer: False, City: San Jose
```

7. 10 states with amount exceeding more than 500

```
Top states with high-value orders (> $500):
Rank 1: State: Texas, High-value orders: 11
Rank 2: State: New York, High-value orders: 9
Rank 3: State: California, High-value orders: 8
Rank 4: State: Florida, High-value orders: 6
Rank 5: State: Massachusetts, High-value orders: 6
Rank 6: State: Pennsylvania, High-value orders: 6
Rank 7: State: Virginia, High-value orders: 4
Rank 8: State: Arizona, High-value orders: 3
Rank 9: State: Colorado, High-value orders: 3
Rank 10: State: Connecticut, High-value orders: 3
```

8. Customers with order more than 2000

```
Total premium customers in Texas (> $2,000): 9
Premium customer order details:
Customer ID: 5, Order ID: 5, Product ID: 5, Quantity: 30, Unit Price: 370.2, Order Date: 10/21/2021, Total Price: 9077.11, Premium Customer: True, City: El Paso
Customer ID: 12, Order ID: 12, Product ID: 12, Quantity: 88, Unit Price: 365.7, Order Date: 10/21/2021, Total Price: 3334.07, Premium Customer: True, City: El Paso
Customer ID: 28, Order ID: 28, Product ID: 28, Quantity: 4, Unit Price: 596.8, Order Date: 10/21/2021, Total Price: 4895.16, Premium Customer: True, City: Houston
Customer ID: 31, Order ID: 31, Product ID: 31, Quantity: 46, Unit Price: 778.1, Order Date: 10/21/2021, Total Price: 3980.29, Premium Customer: False, City: El Paso
Customer ID: 35, Order ID: 35, Product ID: 35, Quantity: 25, Unit Price: 216.1, Order Date: 10/21/2021, Total Price: 8974.95, Premium Customer: True, City: El Paso
Customer ID: 43, Order ID: 43, Product ID: 43, Quantity: 68, Unit Price: 611.5, Order Date: 10/21/2021, Total Price: 5079.34, Premium Customer: False, City: El Paso
Customer ID: 61, Order ID: 61, Product ID: 61, Quantity: 64, Unit Price: 726.5, Order Date: 10/21/2021, Total Price: 8576.16, Premium Customer: False, City: San Antonio
Customer ID: 70, Order ID: 70, Product ID: 70, Quantity: 0, Unit Price: 384.6, Order Date: 10/21/2021, Total Price: 5428.0, Premium Customer: False, City: Midland
Customer ID: 93, Order ID: 93, Product ID: 93, Quantity: 87, Unit Price: 207.1, Order Date: 10/21/2021, Total Price: 7398.02, Premium Customer: True, City: San Antonio
```

9. New York orders on 10/21/2021

```
Customer: True, City: San Antonio
Total orders in New York City on 10/21/2021: 3
Order details:
Order ID: 39, Customer ID: 39, Product ID: 39, Quantity: 59, Unit Price: 387.0, Total Price: 3769.73, Premium Customer: True
Order ID: 84, Customer ID: 84, Product ID: 84, Quantity: 42, Unit Price: 890.0, Total Price: 2420.65, Premium Customer: False
Order ID: 90, Customer ID: 90, Product ID: 90, Quantity: 25, Unit Price: 301.9, Total Price: 6204.12, Premium Customer: False
ashishrajshekhara@Ashishs-MacBook-Pro Assign-1 %
```

10. The aggregation pipelines helps us perform operations with a sequential flow where in the first step we extract the data , then load it and then perform transformations like an etl pipeline.

- \$match : this is used to filter the documents
- \$sort: we use this to display the fetched result in the desired order(ASC/DESC)

And similarly we have others like sort, limit etc.

```
pipeline = [
  {'$match': {'state': 'Texas', 'total_price': {'$gt': 2000}}},
  {
    '$group': {
      '_id': '$customer_id',
      'orders': {'$push': '$$ROOT'},
    },
  },
  {'$sort': {'_id': 1}},
]
```

Here each state takes documents from the initial query as input , applies sorting based on greater than condition operations then provides the result back to the customer.

also find_order_totals function uses \$group to combine orders per state after counting the collection and then uses \$sort puts the states in ascending order.

```
def find_order_totals():
    """Finds the total number of orders and the number of orders per state, sorted by count in ascending order."""
    total_orders = collection.count_documents({})
    pipeline = [
        {'$group': {'_id': '$state', 'order_count': {'$sum': 1}}},
        {'$sort': {'order_count': 1, '_id': 1}},
    ]
    per_state_count = list(collection.aggregate(pipeline))

    print('Total number of orders:', total_orders)
    print('Number of orders per state:')
    for state in per_state_count:
        print(f"State: {state['_id']}, Count: {state['order_count']}")
```