Distributed Database Systems

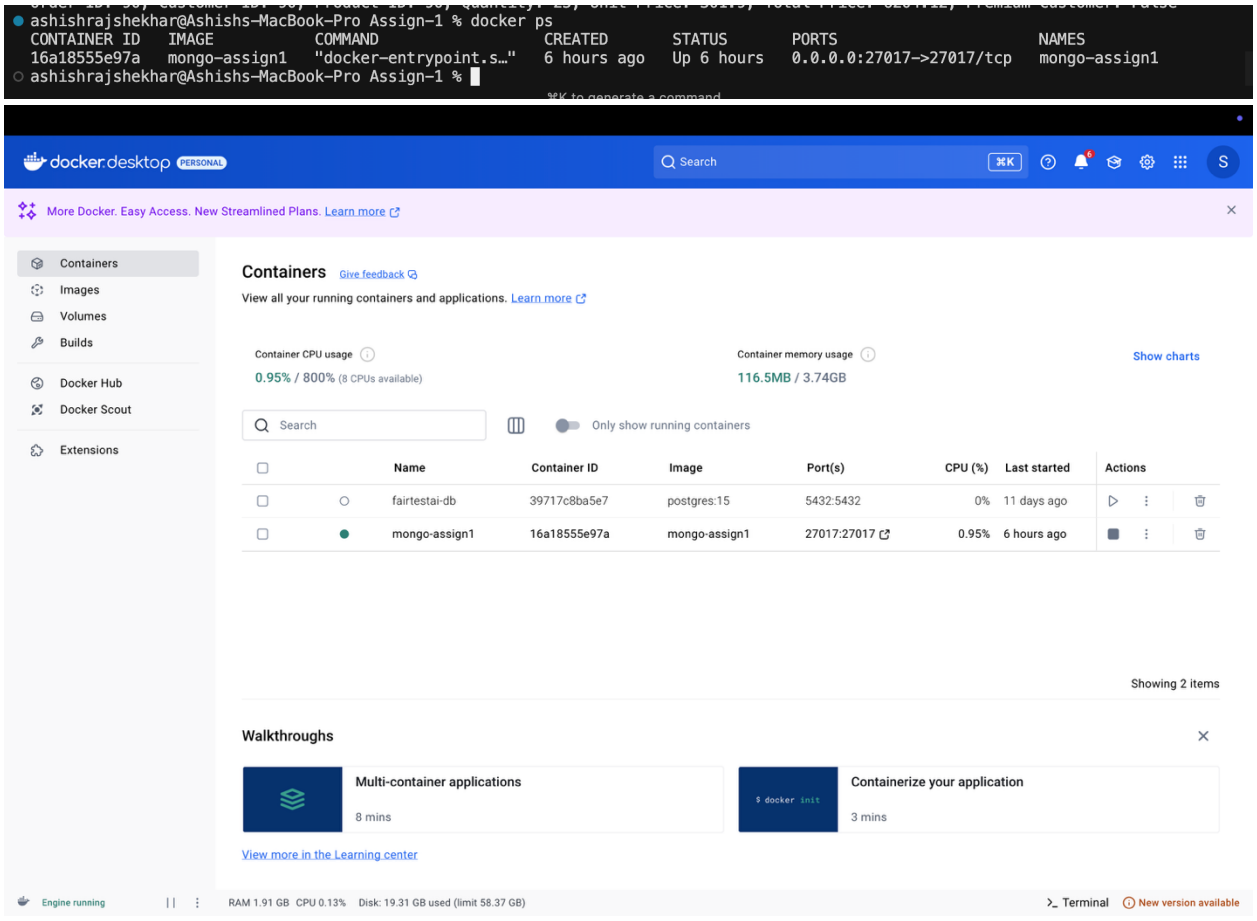
FALL – 2025

Assignment -1

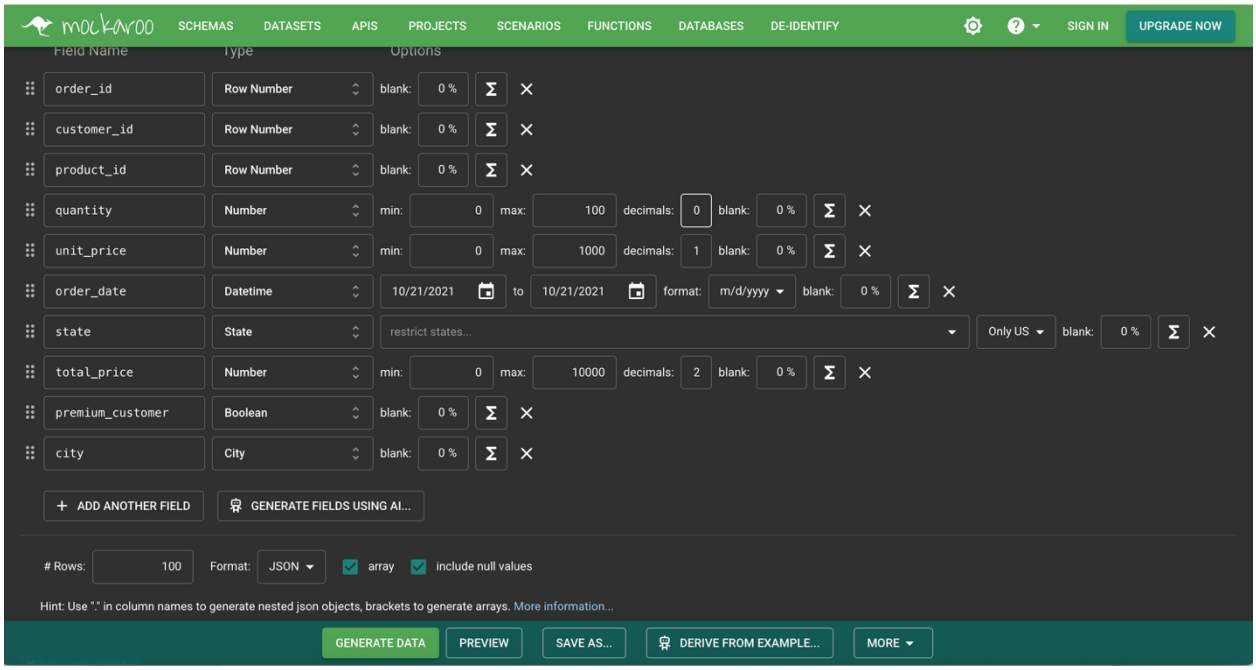
Ashish Raj Shekhar

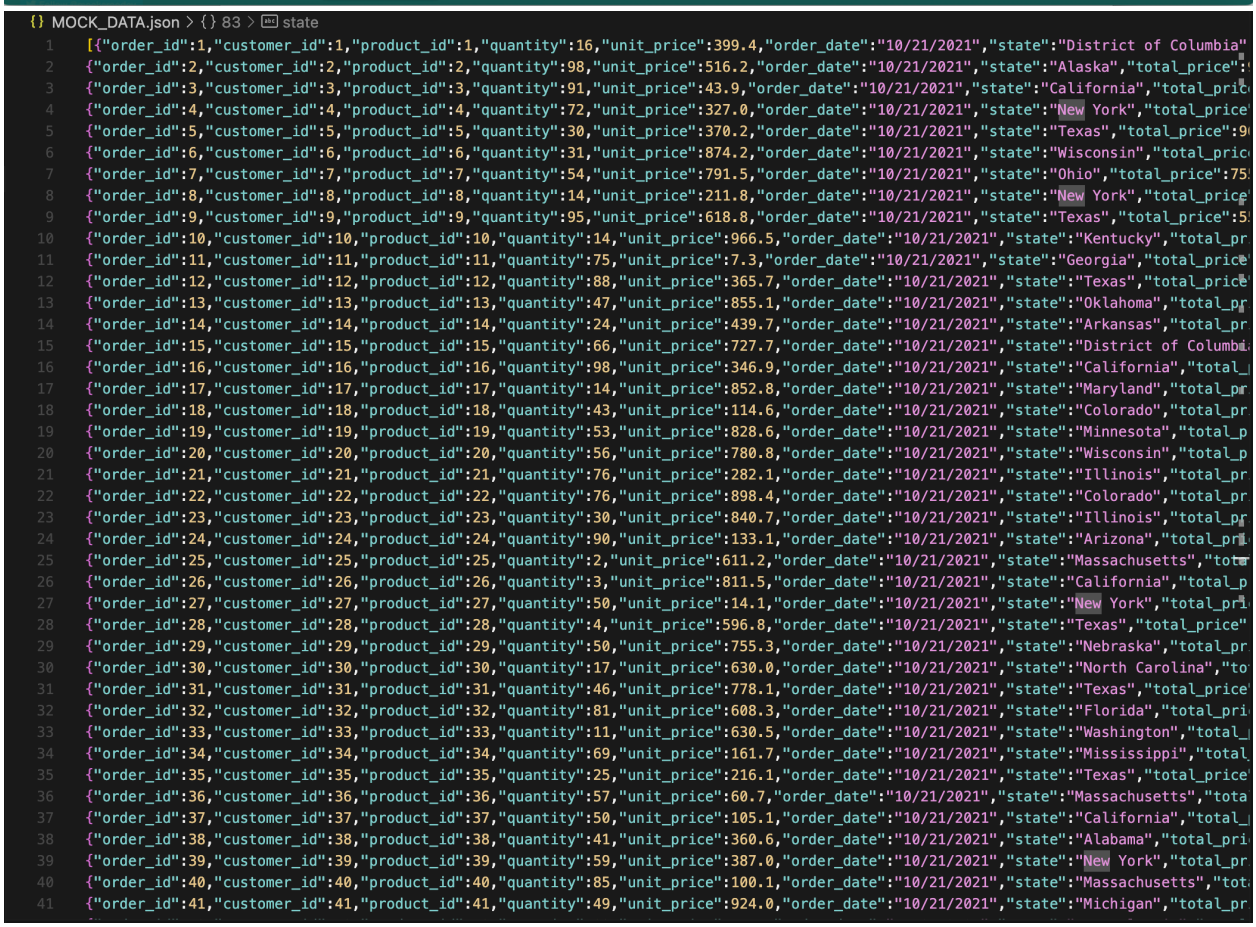
1223100216

1. Docker screenshot



2. Mockaroo json dataset creation screenshot



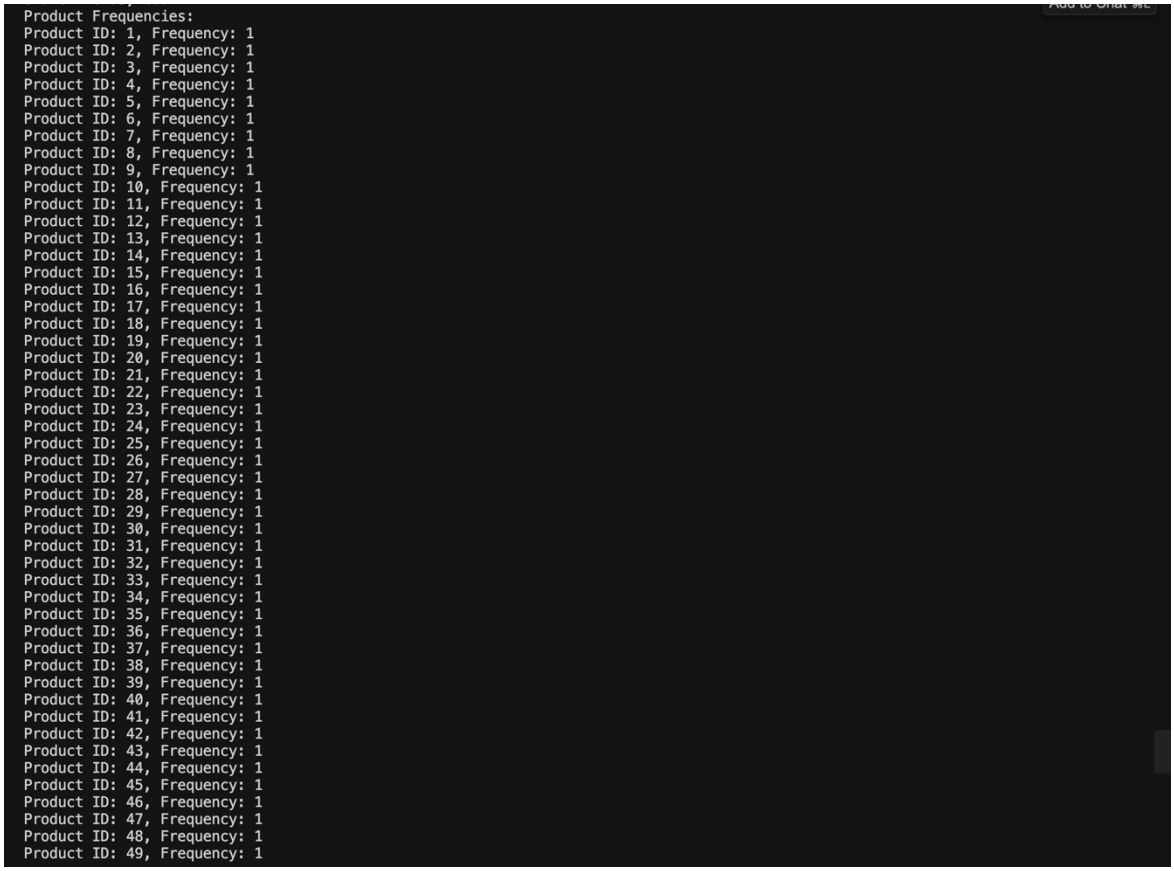
3. Insert the data using insert_many

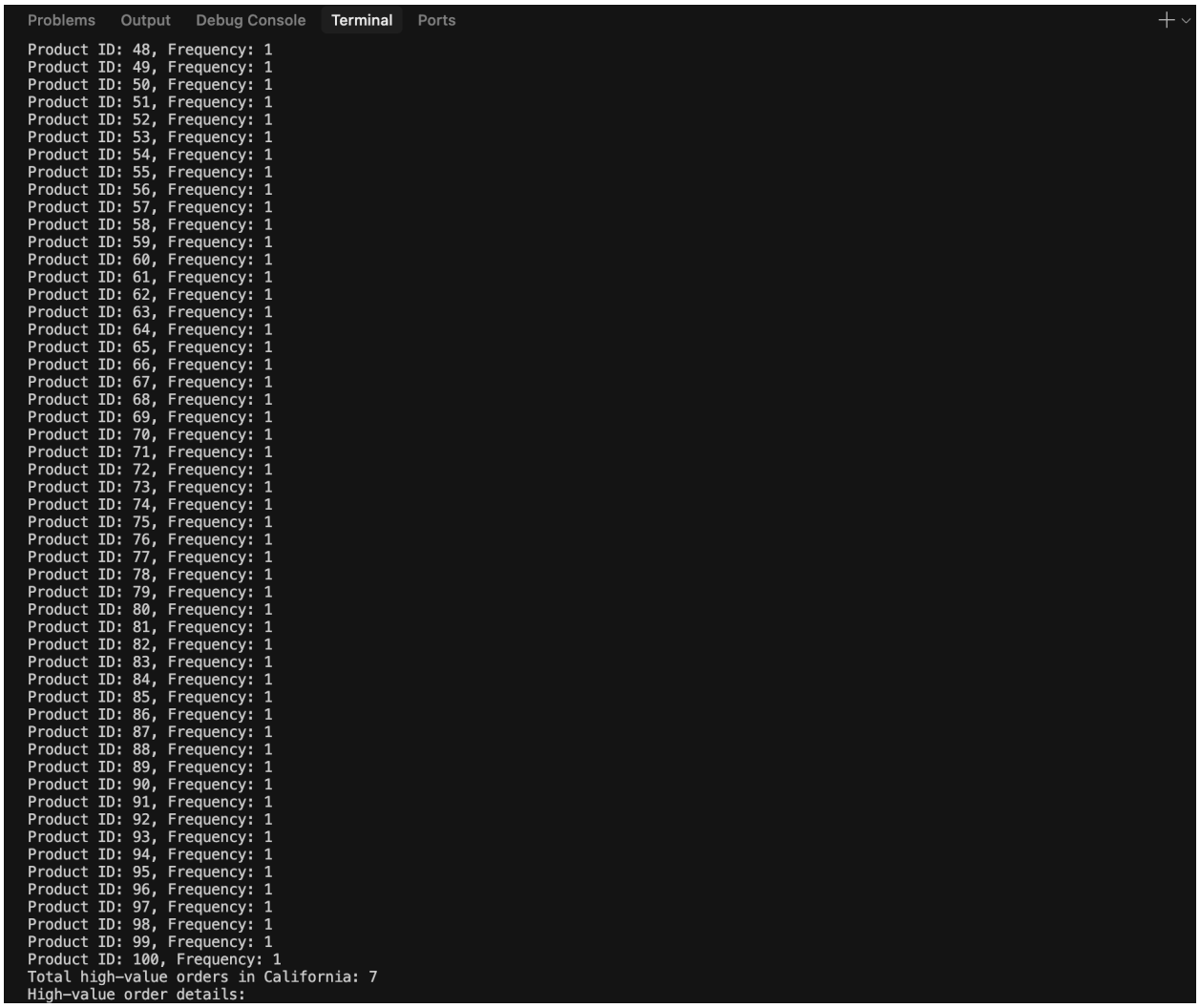
```
(venv) ashishrajshekhar@Ashishs-MacBook-Pro Assign-1 % python3 Assignment\ 1.py
Data inserted successfully:
[ObjectId('68de16013858a0ede4637f5e'), ObjectId('68de16013858a0ede4637f5f'), ObjectId('68de16013858a0ede4637f60'), ObjectId('68de16013858a0ede4637f61'), ObjectId('68de16013858a0ede4637f62'), 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

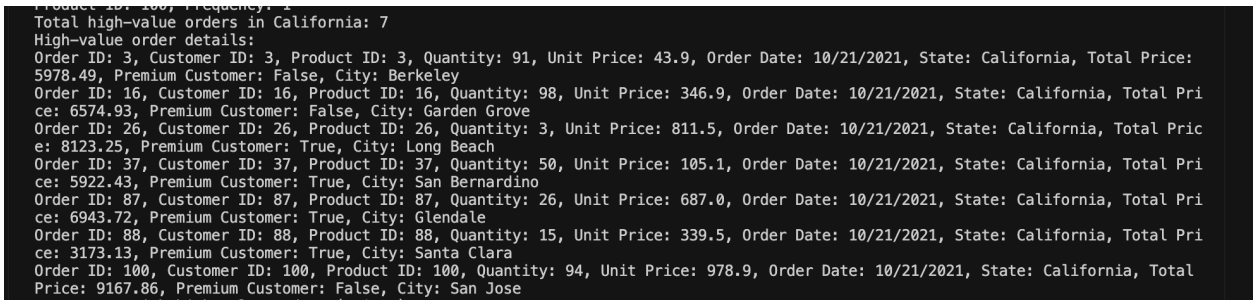
```
Total number of orders: 100
Number of orders per state:
State: Alaska, Count: 1
State: Arkansas, Count: 1
State: Georgia, Count: 1
State: Kentucky, Count: 1
State: Louisiana, Count: 1
State: Maryland, Count: 1
State: Minnesota, Count: 1
State: Mississippi, Count: 1
State: New Jersey, Count: 1
State: 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






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



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 C
ustomer: 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, Premiu
m 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, Premiu
m 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, Premiu
m 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, Premiu
m 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, Premiu
m 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, Premiu
m Customer: True, City: San Antonio
```

9. New York orders on 10/21/2021

```
 tomer: 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
ashishrajshekhar@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 tranformations like an etl piepline.
    a. $match : this is used to filter the documents
    b. $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']}")
```