REPORT:

SAMPLING DATA:

Basically here I created my own code to generate a file of more than 15 GB as a sample:

```python
import json

def copy_json_batchwise(input_file, output_file, batch_size=1000):
    with open(input_file, 'r') as infile:
        with open(output_file, 'w') as outfile:
            for _ in range(2200):  # Run for 10 iterations
                # Read a batch of entries from the input file
                batch = []
                for _ in range(batch_size):
                    line = infile.readline()
                    if not line:
                        break
                    batch.append(line.strip())

                # If no more entries, break the loop
                if not batch:
                    break

                # Write the batch to the output file
                for entry in batch:
                    outfile.write(entry)
                    outfile.write('\n')  # Add newline after each entry

            # Ensure the output file ends with proper JSON format
            outfile.write("]\n")  # Close the JSON array

if __name__ == "__main__":
    input_file = 'data.json'  # Replace with the path to your input JSON file
    output_file = 'i211377_sample.json'  # Replace with the path to the output JSON file
    copy_json_batchwise(input_file, output_file)
```

BATCH PREPROCESSING DATA:

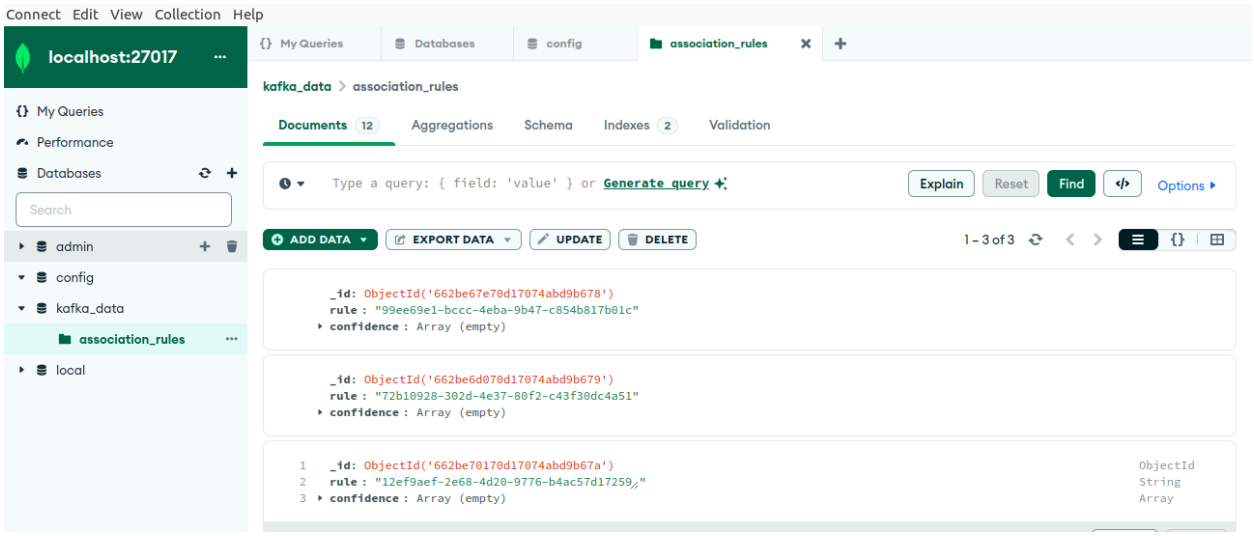
Here we removed extra columns just considered some of them especially also_buy, removed those items which do not have also buy values means which was null.

```python
import json

def preprocess_batch(data):
    preprocessed_data = []
    for entry in data:
        # Check if the "also_buy" field is not empty
        if entry.get("also_buy"):
            # Clean and format each entry
            cleaned_entry = {
                "title": entry.get("title", ""),
                "brand": entry.get("brand", ""),
                "also_buy": entry.get("also_buy", []),
                "asin": entry.get("asin", ""),
                # Add more keys as needed
            }
            preprocessed_data.append(cleaned_entry)
    return preprocessed_data

def main():
    chunk_size = 100  # Number of entries to read at a time
    with open('i211377_sample.json', 'r') as file:
        with open('preprocessed_data.json', 'w') as outfile:
            outfile.write("[\n")  # Write opening bracket of the JSON array
            first_entry = True  # Flag to track if it's the first entry being written
            # Read the file in chunks
            for i in range(5):  # For testing purposes, consider removing this limit for the full file
                chunk = []
                for _ in range(chunk_size):
```

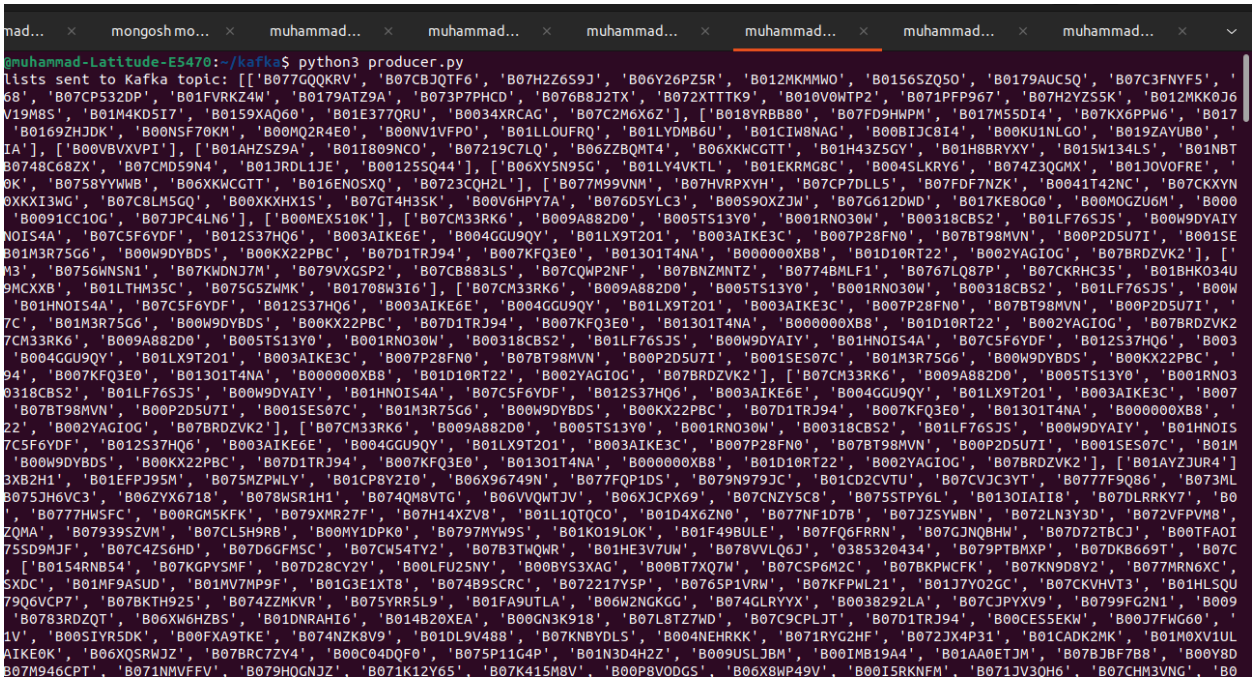
```python
                chunk = []
                for _ in range(chunk_size):
                    line = file.readline()
                    if not line:
                        break
                    entry = json.loads(line)
                    preprocessed_entry = preprocess_batch([entry])
                    if preprocessed_entry:  # Check if the entry is not empty after preprocessing
                        if not first_entry:
                            outfile.write(",\n")  # Add comma and newline if it's not the first entry
                        else:
                            first_entry = False
                        json.dump(preprocessed_entry[0], outfile, indent=4)
            outfile.write("\n]")  # Write closing bracket of the JSON array

if __name__ == "__main__":
    main()
```

APPLYING APRIORI ALGORITHM:

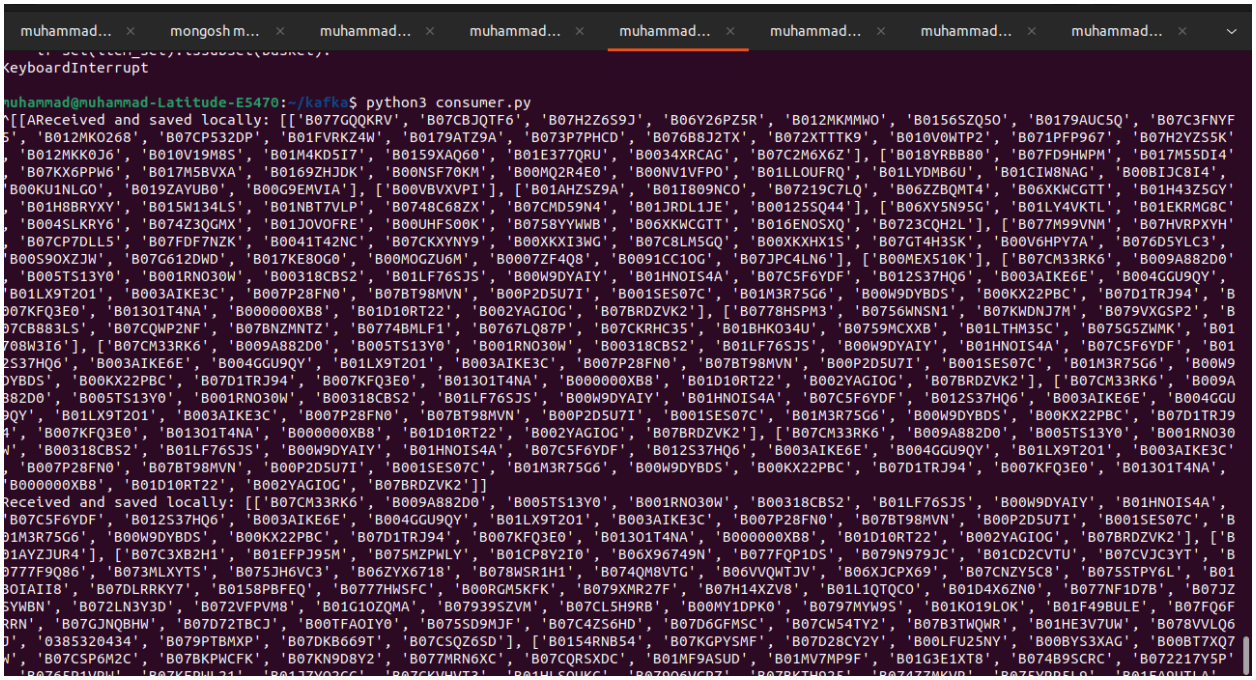
Started and run mongoDB and mongoDB compass as CUI 🙂



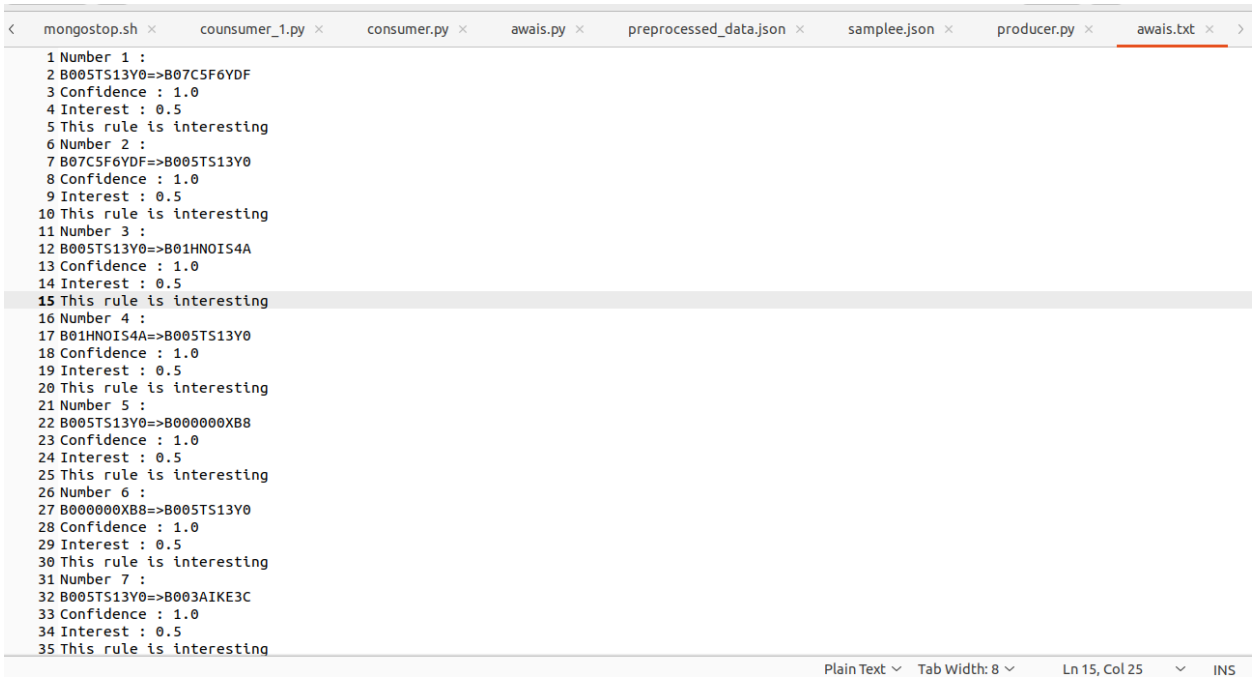
After the I created a producer which produces the values 🙂



And a consumer too which consumes these values which I printed on each iteration:



This consumer is checking for interesting and not interesting association rules for items and storing them in the file



And then storing it on Mongo DB as well:

After all I created bash commands to run my code for a bit bonus part JazakAllah!