REPORT:

SAMPLING DATA:

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

```
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)
```

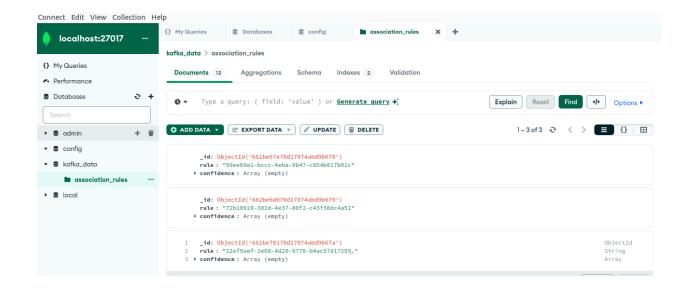
BATCH PREPROCESSING DATA:

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.

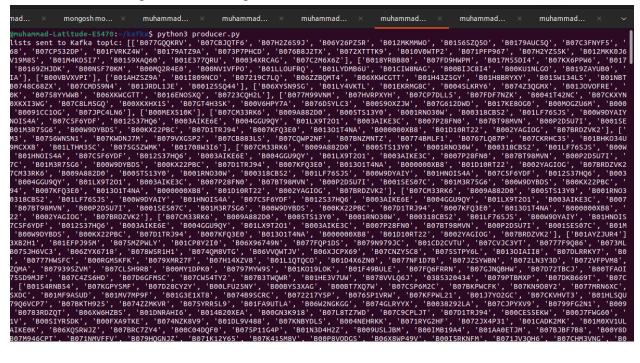
```
import json
def preprocess batch(data):
    preprocessed_data = []
    for entry in data:
        # Check if the "also buy" field is not empty
        if entry.get("also_buy"):
            # Clean and format each entry
            cleaned entry = {
                "title": entry.get("title", ""),
                "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):
                chunk = []
                for _ in range(chunk_size):
                    line = file.readline()
                    if not line:
                    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
                             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 $\ensuremath{ \ensuremath{ \ \ \ \ \ \ } }$



After the I created a producer which produces the values \bigcirc



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

```
muhammad... x mu
```

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

```
mongostop.sh ×
                                                                                                                                                                      awais.txt ×
                     counsumer 1.py × consumer.py ×
                                                                       awais.py ×
                                                                                         preprocessed data.json ×
                                                                                                                         samplee.json ×
                                                                                                                                                producer.py ×
  2 B005TS13Y0=>B07C5F6YDF
  3 Confidence : 1.0
  4 Interest : 0.5
5 This rule is interesting
  6 Number 2 :
   B07C5F6YDF=>B005TS13Y0
  8 Confidence : 1.0
9 Interest : 0.5
10 This rule is interesting
11 Number 3 :
12 B005TS13Y0=>B01HNOIS4A
13 Confidence : 1.0
14 Interest : 0.5
15 This rule is interesting
16 Number 4:
17 B01HN0IS4A=>B005TS13Y0
18 Confidence : 1.0
19 Interest : 0.5
20 This rule is interesting
21 Number 5 : 
22 B005TS13Y0=>B000000XB8
23 Confidence : 1.0
24 Interest : 0.5
25 This rule is interesting
26 Number 6 :
27 B000000XB8=>B005TS13Y0
28 Confidence : 1.0
29 Interest : 0.5
30 This rule is interesting
31 Number 7 :
 32 B005TS13Y0=>B003AIKE3C
 33 Confidence: 1.0
34 Interest: 0.5
35 This rule is interesting
```

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!