Project Mongo

CS336

1. Return the bar if either its phone or address is an empty string.

```
use db;
db.Bar.find({"$or":[{"addr":""},{"phone": ""}]})
```

2. Find the city that has more than 4 bars. Return the city name and the number of bars it has.

```
use db;
db.Bars.aggregate([
        {
                "$group": {
                        "_id": "$city",
                        "num_bars": {
                                "$sum": 1
                        }
                },
        },
        { "$match": { "num_bars": { "$gt": 4 } } }
]);
```

3. Return how many bars sell more than 5 kinds of beers.

```
use db;
db.Bars.aggregate([
    {
      "$project": {
        "_id": 1,
        "num_sold": {
                "$size": "$beers"
            },
      },
    },
    {"$match" : {"num_sold": {"$gt": 5}}},
        {"$count":"bars_selling_more_than_5"}
]);
```

4. Find the drinkers that have visited any bars either on Saturday or Sunday (or both) [hint: go check out "$elemMatch" function]

```
use db;
db.Drinkers.find(
  {
    "history": {
      "$elemMatch": {
        "day": { "$in": ["Saturday", "Sunday"] }
      }
    }
  }
);
```

5. Find the drinker who has ordered "Blue Tattoo" beer more than once

```
use db;
db.Drinkers.aggregate(
        {"$unwind": "$history"},
        {
                "$match": {
                        "history.set_of_beers": {
                                "$in": ["Blue Tattoo"]
                        }
                }
        },
        {
                "$group": {
                        "_id": "$name",
                        "numBlueTattoo": {"$count": {}}
                }
        },
        {
                "$match": {"numBlueTattoo": {"$gt" : 1}}
        },
        {"$project": {"_id": 1}}
)
```

6. Insert Lucy to Drinker collection. Lucy is from Edison, lives at "433 River Road" with phone number 732-571-9871, she is 23 years old and her list of favorite bar foods consists of: French fries, onion rings, nachos, and wings.

```
use db;
db.Drinkers.insertOne(
```

```
        {
                "name":"Lucy",
                "city":"Edison",
                "addr":"433 River Road",
                "phone":"732-571-9871",
                "age":23,
                "favorite_bar_foods":["French fries","Onion
rings","Nachos","Wings"]
        }
)
```

7. **Value count**: Given a relation R, represent R as JSON object J with the *smallest value count*
   We can represent R as the JSON Object J as

```
{“BD”: 11, “AC”: [01]},{“BD”: 10, “AC”:[00, 01, 10, 11]}]$
```

with 14 values.
8. For each timestamp T, define TotIncrement as sum of totalvote increments over all precincts (totalvote increment, as defined in 2.1 of Election project newPenna). Finds timestamp(s) with largest value of TotIncremenet along with this largest value. Submit CODE and result.

```
from pymongo import MongoClient
import json
client = MongoClient('localhost', 27017)

db = client.db
docs = db.Penna.find().sort("Timestamp",-1)
timestamps = docs.distinct("Timestamp")

mapped_timestamps = {}
mapped_increments = {}
max_tot_increment = 0

last_t = None
for t in timestamps:
    records = db.Penna.find({"Timestamp":t})
    print(f"Processing timestamp: {t}")
    mapped_timestamps[t] = {}
    mapped_increments[t] = 0
    tot_increment = 0
    for i in records:
        mapped_timestamps[t][i['precinct']] = i['totalvotes']
```

```
        if last_t and mapped_timestamps[last_t].get(i['precinct']):
            mapped_increments[t] += (mapped_timestamps[t][i['precinct']] -
mapped_timestamps[last_t][i['precinct']])
    last_t = t
    if mapped_increments[t] > max_tot_increment:
        max_tot_increment = mapped_increments[t]

with open("incremented.json","w") as f:
    json.dump(mapped_increments,f)

print("The max increment is",max_tot_increment)
print("Timestamps with this increment are:",[t[0] for t in
mapped_increments.items() if t[1] == max_tot_increment])
```

Result:
The max total increment is 84730
Timestamps with this increment are: ['2020-11-04 05:27:31']