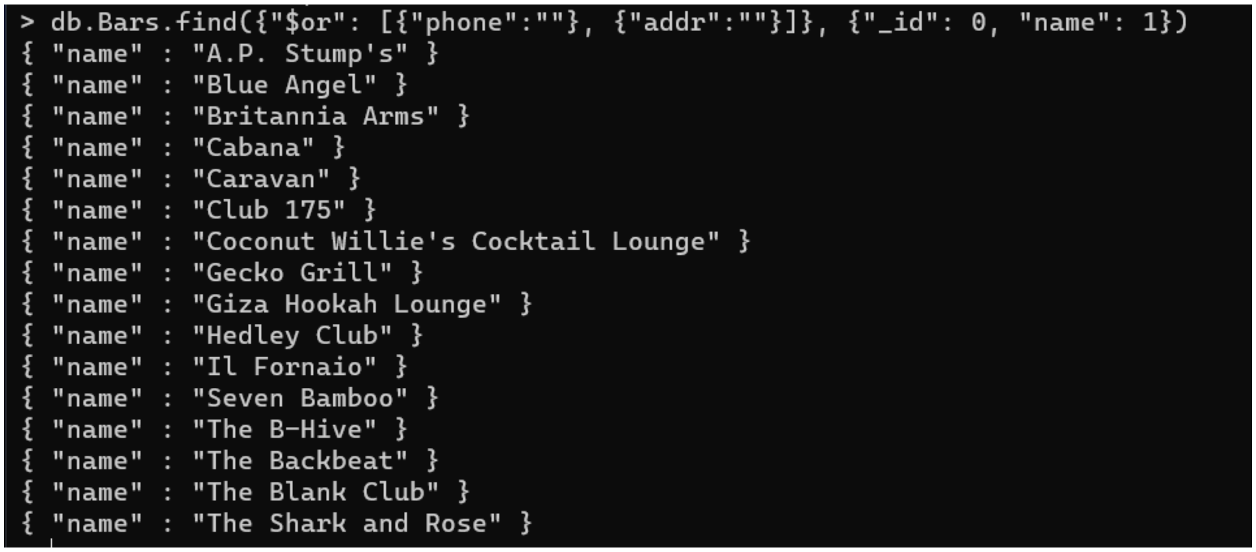
use db

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

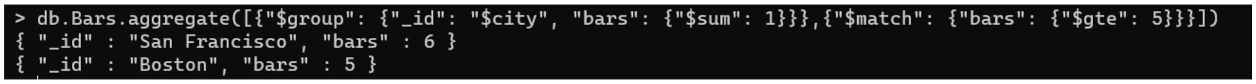
db.Bars.find({"$or": [{"phone":""}, {"addr":""}]}, {"_id": 0, "name": 1})



```
> db.Bars.find({"$or": [{"phone":""}, {"addr":""}]}, {"_id": 0, "name": 1})
{ "name" : "A.P. Stump's" }
{ "name" : "Blue Angel" }
{ "name" : "Britannia Arms" }
{ "name" : "Cabana" }
{ "name" : "Caravan" }
{ "name" : "Club 175" }
{ "name" : "Coconut Willie's Cocktail Lounge" }
{ "name" : "Gecko Grill" }
{ "name" : "Giza Hookah Lounge" }
{ "name" : "Hedley Club" }
{ "name" : "Il Fornaio" }
{ "name" : "Seven Bamboo" }
{ "name" : "The B-Hive" }
{ "name" : "The Backbeat" }
{ "name" : "The Blank Club" }
{ "name" : "The Shark and Rose" }
```

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

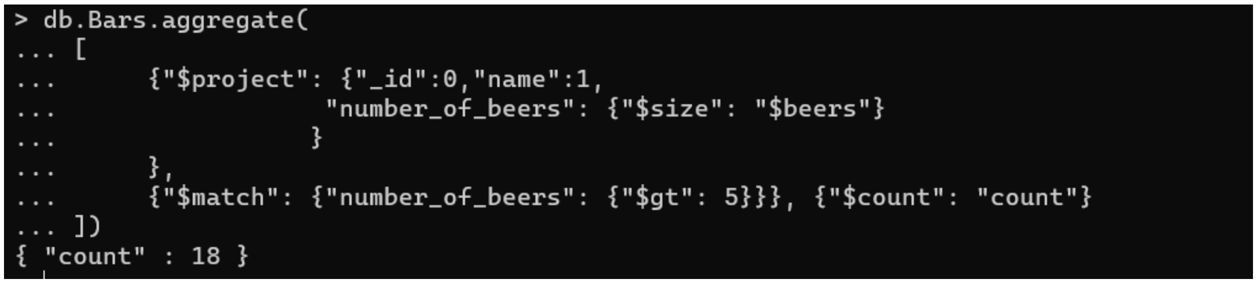
db.Bars.aggregate([{"$group": {"_id": "$city", "bars": {"$sum": 1}}},{"$match": {"bars": {"$gte": 5}}}])



```
> db.Bars.aggregate([{"$group": {"_id": "$city", "bars": {"$sum": 1}}},{"$match": {"bars": {"$gte": 5}}}])
{ "_id" : "San Francisco", "bars" : 6 }
{ "_id" : "Boston", "bars" : 5 }
```

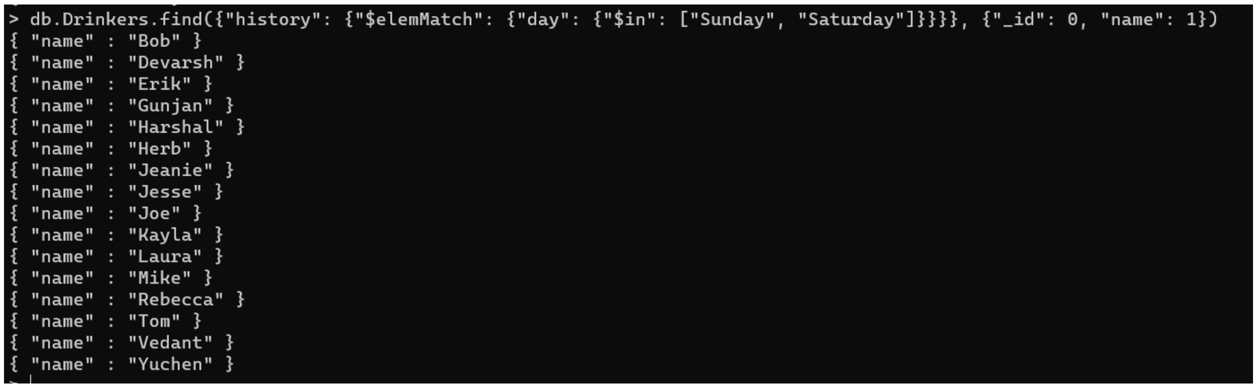
3. Return how many bars sell more than 5 kinds of beers.

```
db.Bars.aggregate(
[
   {"$project": {"_id":0,"name":1,
              "number_of_beers": {"$size": "$beers"}
             }
   },
   {"$match": {"number_of_beers": {"$gt": 5}}}, {"$count": "count"}
])
```

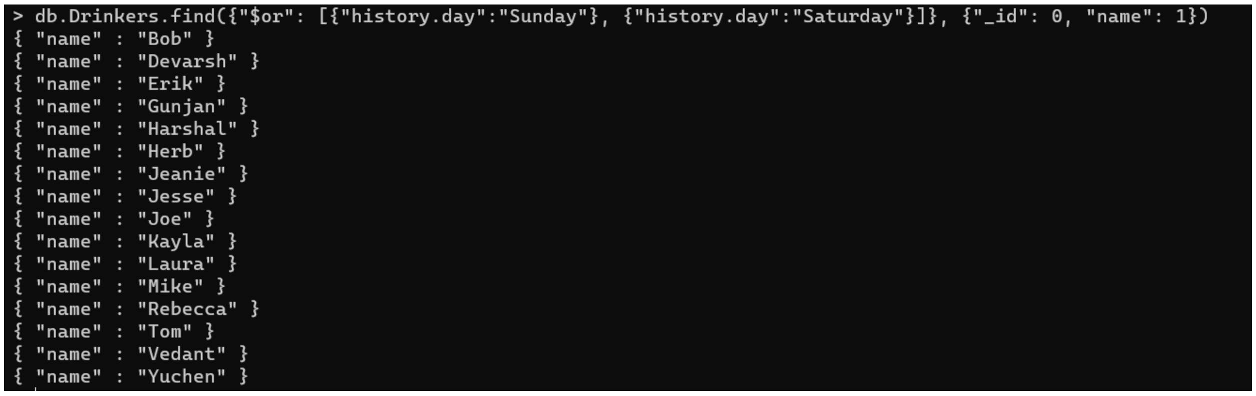

```
> db.Bars.aggregate(
... [
...       {"$project": {"_id":0,"name":1,
...                      "number_of_beers": {"$size": "$beers"}
...                     }
...       },
...       {"$match": {"number_of_beers": {"$gt": 5}}}, {"$count": "count"}
... ])
{ "count" : 18 }
```

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

db.Drinkers.find({"history": {"$elemMatch": {"day": {"$in": ["Sunday", "Saturday"]}}}}, {"_id": 0, "name": 1})

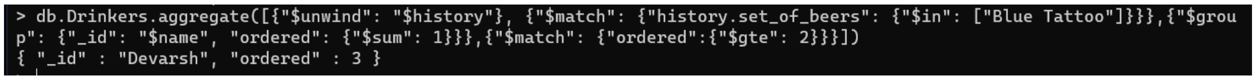

```
> db.Drinkers.find({"history": {"$elemMatch": {"day": {"$in": ["Sunday", "Saturday"]}}}}, {"_id": 0, "name": 1})
{ "name" : "Bob" }
{ "name" : "Devarsh" }
{ "name" : "Erik" }
{ "name" : "Gunjan" }
{ "name" : "Harshal" }
{ "name" : "Herb" }
{ "name" : "Jeanie" }
{ "name" : "Jesse" }
{ "name" : "Joe" }
{ "name" : "Kayla" }
{ "name" : "Laura" }
{ "name" : "Mike" }
{ "name" : "Rebecca" }
{ "name" : "Tom" }
{ "name" : "Vedant" }
{ "name" : "Yuchen" }
```

db.Drinkers.find({"$or": [{"history.day":"Sunday"}, {"history.day":"Saturday"}]}, {"_id": 0, "name": 1})


```
> db.Drinkers.find({"$or": [{"history.day":"Sunday"}, {"history.day":"Saturday"}]}, {"_id": 0, "name": 1})
{ "name" : "Bob" }
{ "name" : "Devarsh" }
{ "name" : "Erik" }
{ "name" : "Gunjan" }
{ "name" : "Harshal" }
{ "name" : "Herb" }
{ "name" : "Jeanie" }
{ "name" : "Jesse" }
{ "name" : "Joe" }
{ "name" : "Kayla" }
{ "name" : "Laura" }
{ "name" : "Mike" }
{ "name" : "Rebecca" }
{ "name" : "Tom" }
{ "name" : "Vedant" }
{ "name" : "Yuchen" }
```

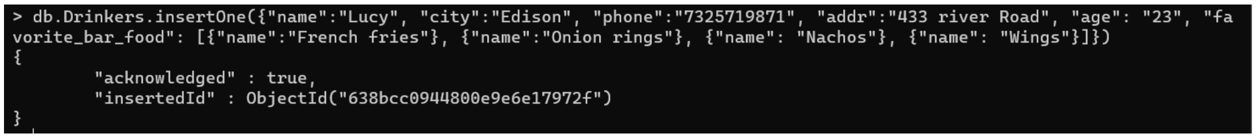
5. Find the drinker who has ordered "Blue Tattoo" beer more than once (in the drinkers history)

db.Drinkers.aggregate([{"$unwind": "$history"}, {"$match": {"history.set_of_beers": {"$in": ["Blue Tattoo"]}}},{"$group": {"_id": "$name", "ordered": {"$sum": 1}}},{"$match": {"ordered":{"$gte": 2}}}])


```
> db.Drinkers.aggregate([{"$unwind": "$history"}, {"$match": {"history.set_of_beers": {"$in": ["Blue Tattoo"]}}},{"$grou
p": {"_id": "$name", "ordered": {"$sum": 1}}},{"$match": {"ordered":{"$gte": 2}}}])
{ "_id" : "Devarsh", "ordered" : 3 }
```

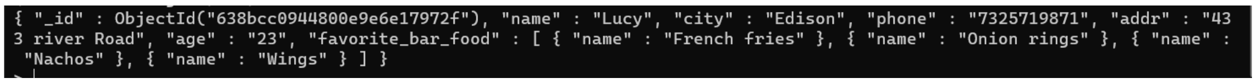
6. Insert Lucy to Drinker collection. Lucy is from Edison, lives at "433 river Road" with phone number 732-571-9871,
she is 23 years old and her list of favorite bar foods consists of French fries, Onion rings, Nachos. and Wings

db.Drinkers.insertOne({"name":"Lucy", "city":"Edison", "phone":"7325719871", "addr":"433 river Road", "age": "23", "favorite_bar_food": [{"name":"French fries"}, {"name":"Onion rings"}, {"name": "Nachos"}, {"name": "Wings"}]})



```
> db.Drinkers.insertOne({"name":"Lucy", "city":"Edison", "phone":"7325719871", "addr":"433 river Road", "age": "23", "fa
vorite_bar_food": [{"name":"French fries"}, {"name":"Onion rings"}, {"name": "Nachos"}, {"name": "Wings"}]})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("638bcc0944800e9e6e17972f")
}
```

db.Drinkers.find({"name": {$eq: "Lucy"}})



```
{ "_id" : ObjectId("638bcc0944800e9e6e17972f"), "name" : "Lucy", "city" : "Edison", "phone" : "7325719871", "addr" : "43
3 river Road", "age" : "23", "favorite_bar_food" : [ { "name" : "French fries" }, { "name" : "Onion rings" }, { "name" :
 "Nachos" }, { "name" : "Wings" } ] }
```

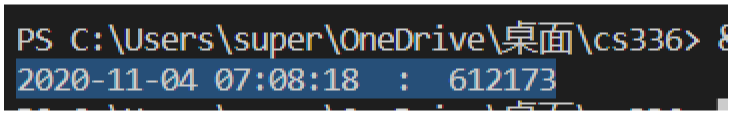
7. Value Count (lecture on Nov 18th, we called it value space, but I prefer value count)

J = [{"AB": 11, "CD": [00, 10]},
        {"AB": 01, "CD": [10, 00, 11]}]
The value count is 6 + 8 = 14.

8. [Python/any other programming language plus Mongo] – Penna collection.
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).
"This stored procedure will create a table newPenna, showing for each precinct how many votes were added to totalvotes, Trump, Biden between timestamp T and the last
timestamp directly preceding T. In other words, create a table like Penna but replace totalvotes with newvotes, Trump with new_Trump and Biden with new_Biden."
Finds timestamp(s) with largest value of TotIncrement along with this largest value.

```
# README !!!
# run this line in command prompt first before running the pymongo,
because the data is too big
# > use admin
# > db.adminCommand({setParameter: 1,
internalQueryExecMaxBlockingSortBytes:501514320})

import pymongo

client = pymongo.MongoClient()

```

```python
mydb = client["db"]

mycol = mydb["Penna"]

total = 0
count = 0
lis = []
time = []
cur = list(mycol.find().sort("Timestamp", pymongo.ASCENDING))

for i in range(len(cur)):
    if i == len(cur)-1:
        total += cur[i]["totalvotes"]
        count += 1
        break
    elif cur[i]["Timestamp"] == cur[i+1]["Timestamp"]:
        total += cur[i]["totalvotes"]
        count += 1
    else:
        key = cur[i]["Timestamp"]
        total += cur[i]["totalvotes"]
        lis.append(total)
        time.append(key)
        total = 0
        count += 1

Increment = []
Increment.append(lis[0])
for x in range(1, len(lis)):
    diff = lis[x] - lis[x-1]
    Increment.append(diff)

max = max(Increment)
index = Increment.index(max)
# print(len(time), len(Increment))
# print(index)
print(time[index], " : ", max)
```



```
PS C:\Users\super\OneDrive\桌面\cs336> 8
2020-11-04 07:08:18  :  612173
```