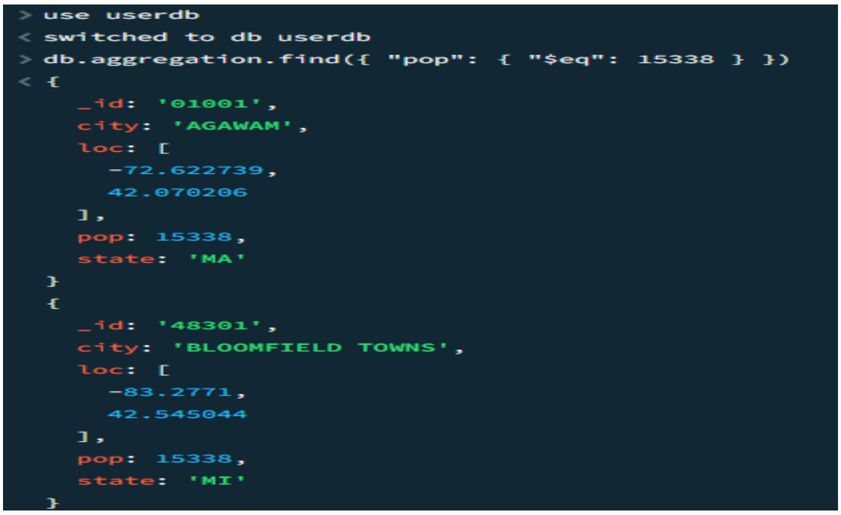
PRACTICAL 5

Aggregation with MONGODB

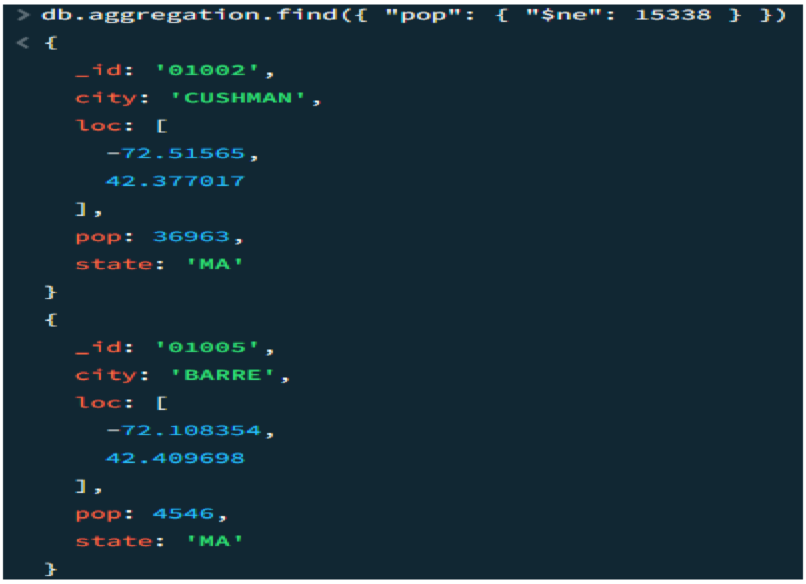
Comparison Operators

1. Find cities where the population is exactly 15,338



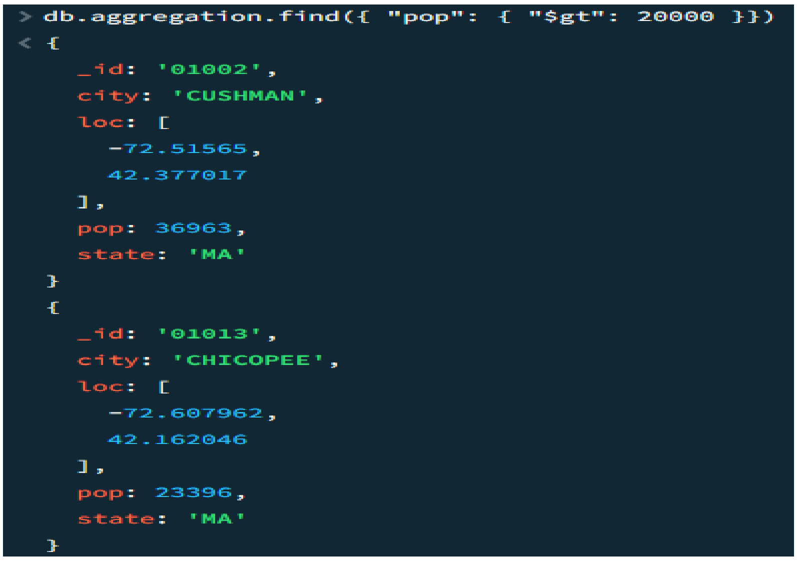
```
> use userdb
< switched to db userdb
> db.aggregation.find({ "pop": { "$eq": 15338 } })
< {
    _id: '01001',
    city: 'AGAWAM',
    loc: [
      -72.622739,
      42.070206
    ],
    pop: 15338,
    state: 'MA'
  }
  {
    _id: '48301',
    city: 'BLOOMFIELD TOWNS',
    loc: [
      -83.2771,
      42.545044
    ],
    pop: 15338,
    state: 'MI'
  }
```

2. Find cities where the population is not 15,338.



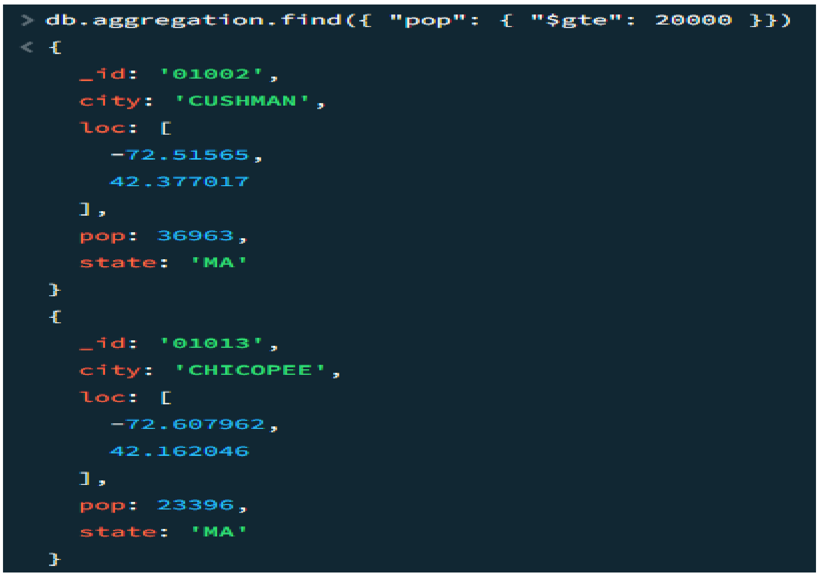
```
> db.aggregation.find({ "pop": { "$ne": 15338 } })
< {
    _id: '01002',
    city: 'CUSHMAN',
    loc: [
      -72.51565,
      42.377017
    ],
    pop: 36963,
    state: 'MA'
  }
  {
    _id: '01005',
    city: 'BARRE',
    loc: [
      -72.108354,
      42.409698
    ],
    pop: 4546,
    state: 'MA'
  }
```

3. Find cities where the population is greater than 20,000.



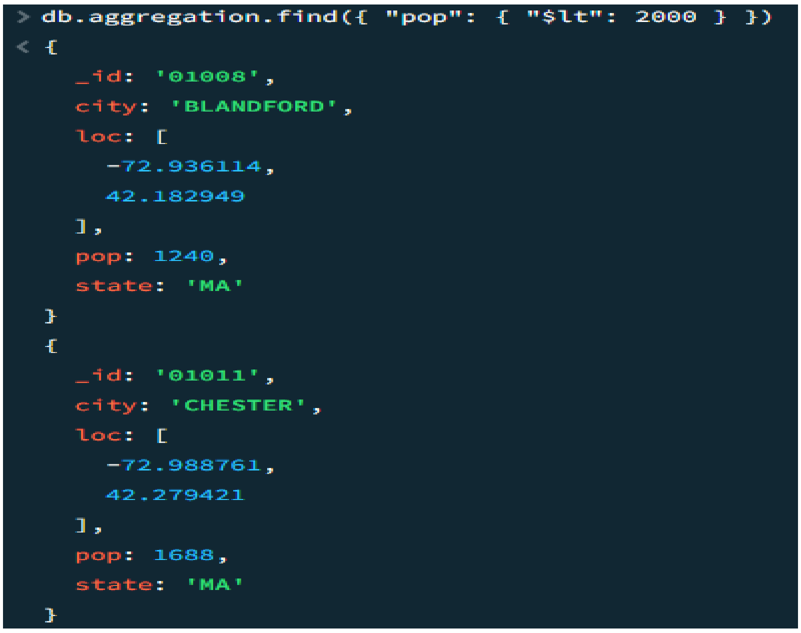
```
> db.aggregation.find({ "pop": { "$gt": 20000 }})
< {
    _id: '01002',
    city: 'CUSHMAN',
    loc: [
      -72.51565,
      42.377017
    ],
    pop: 36963,
    state: 'MA'
  }
  {
    _id: '01013',
    city: 'CHICOPEE',
    loc: [
      -72.607962,
      42.162046
    ],
    pop: 23396,
    state: 'MA'
  }
```

4. Find cities where the population is at least 20,000.



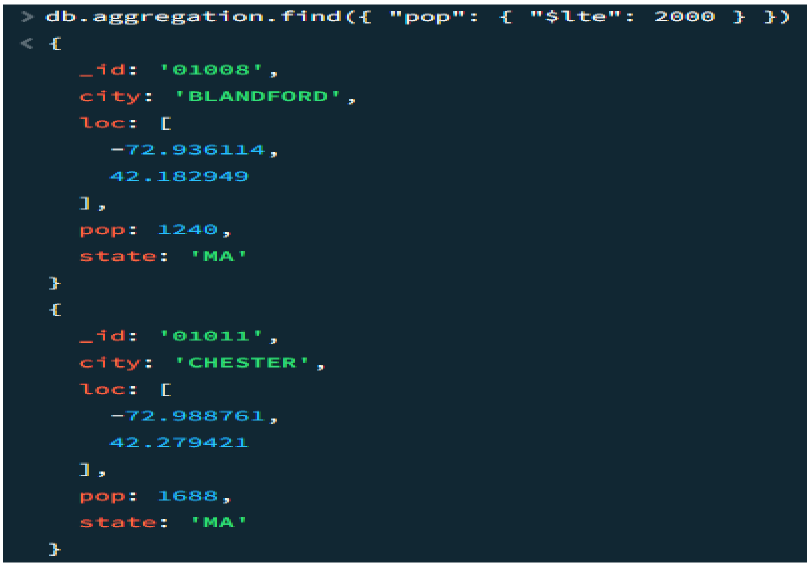
```
> db.aggregation.find({ "pop": { "$gte": 20000 }})
< {
    _id: '01002',
    city: 'CUSHMAN',
    loc: [
      -72.51565,
      42.377017
    ],
    pop: 36963,
    state: 'MA'
  }
  {
    _id: '01013',
    city: 'CHICOPEE',
    loc: [
      -72.607962,
      42.162046
    ],
    pop: 23396,
    state: 'MA'
  }
```

5. Find cities where the population is less than 2,000.



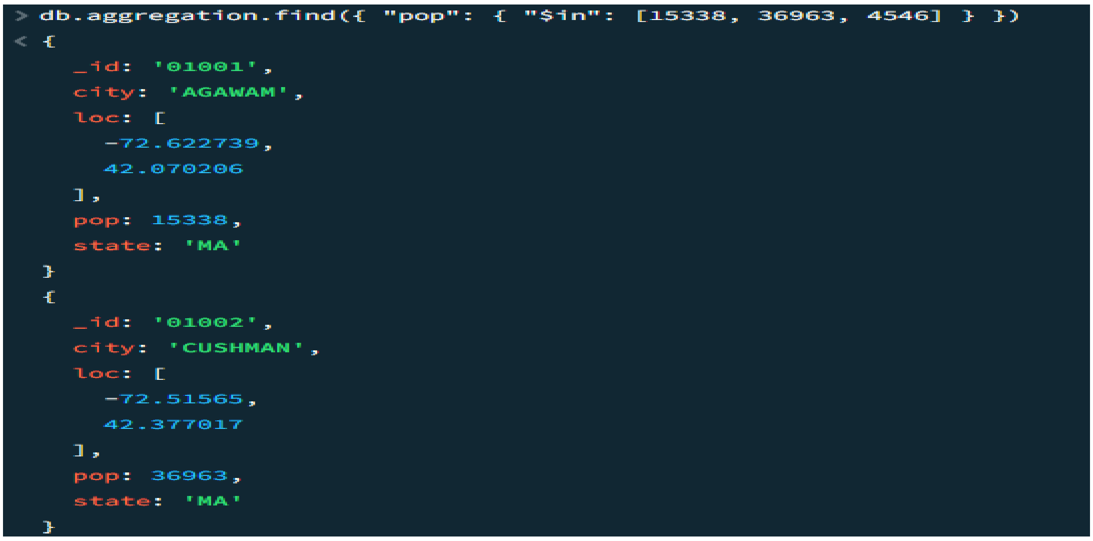
```
> db.aggregation.find({ "pop": { "$lt": 2000 } })
< {
    _id: '01008',
    city: 'BLANDFORD',
    loc: [
      -72.936114,
      42.182949
    ],
    pop: 1240,
    state: 'MA'
  }
  {
    _id: '01011',
    city: 'CHESTER',
    loc: [
      -72.988761,
      42.279421
    ],
    pop: 1688,
    state: 'MA'
  }
```

6. Find cities where the population is at most 2,000.



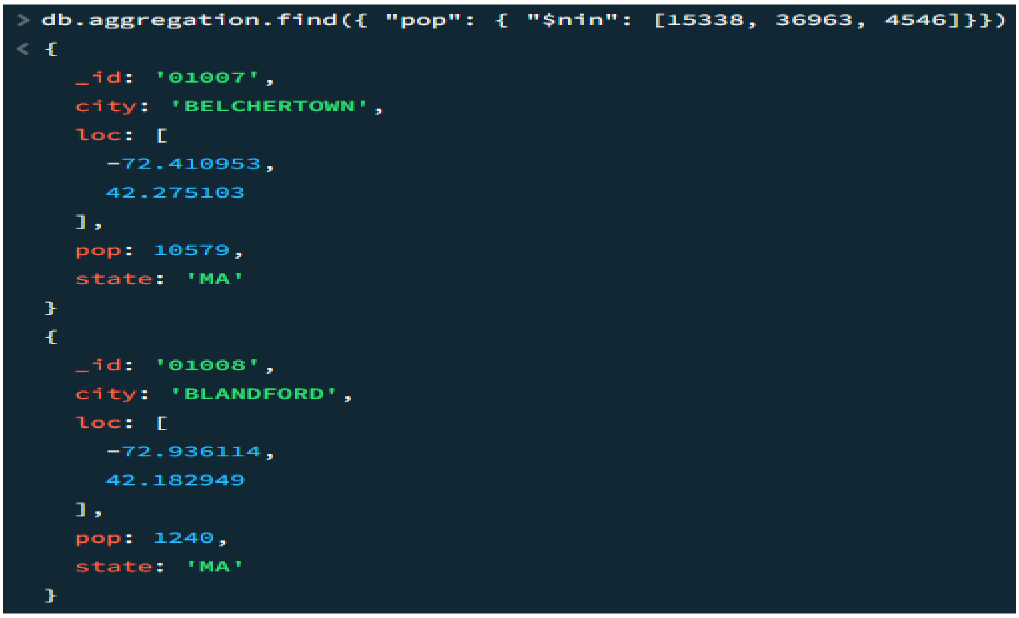
```
> db.aggregation.find({ "pop": { "$lte": 2000 } })
< {
    _id: '01008',
    city: 'BLANDFORD',
    loc: [
      -72.936114,
      42.182949
    ],
    pop: 1240,
    state: 'MA'
  }
  {
    _id: '01011',
    city: 'CHESTER',
    loc: [
      -72.988761,
      42.279421
    ],
    pop: 1688,
    state: 'MA'
  }
```

7. Find cities where the population is either **15,338, 36,963, or 4,546**.



```
> db.aggregation.find({ "pop": { "$in": [15338, 36963, 4546] } })
< {
    _id: '01001',
    city: 'AGAWAM',
    loc: [
      -72.622739,
      42.070206
    ],
    pop: 15338,
    state: 'MA'
  }
  {
    _id: '01002',
    city: 'CUSHMAN',
    loc: [
      -72.51565,
      42.377017
    ],
    pop: 36963,
    state: 'MA'
  }
```

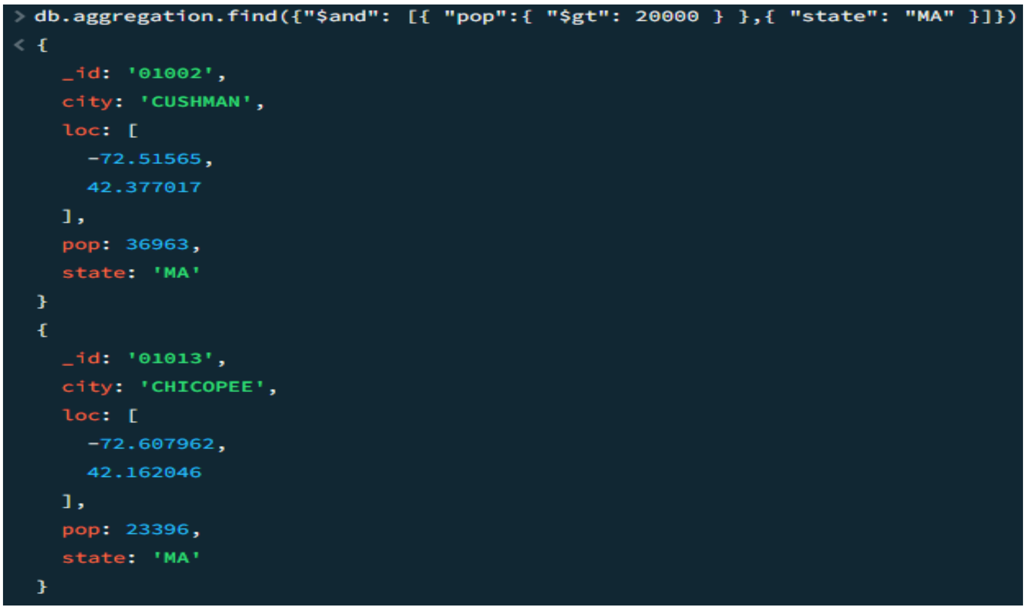
8. Find cities where the population is **not 15,338, 36,963, or 4,546**



```
> db.aggregation.find({ "pop": { "$nin": [15338, 36963, 4546]}})
< {
    _id: '01007',
    city: 'BELCHERTOWN',
    loc: [
      -72.410953,
      42.275103
    ],
    pop: 10579,
    state: 'MA'
  }
  {
    _id: '01008',
    city: 'BLANDFORD',
    loc: [
      -72.936114,
      42.182949
    ],
    pop: 1240,
    state: 'MA'
  }
```

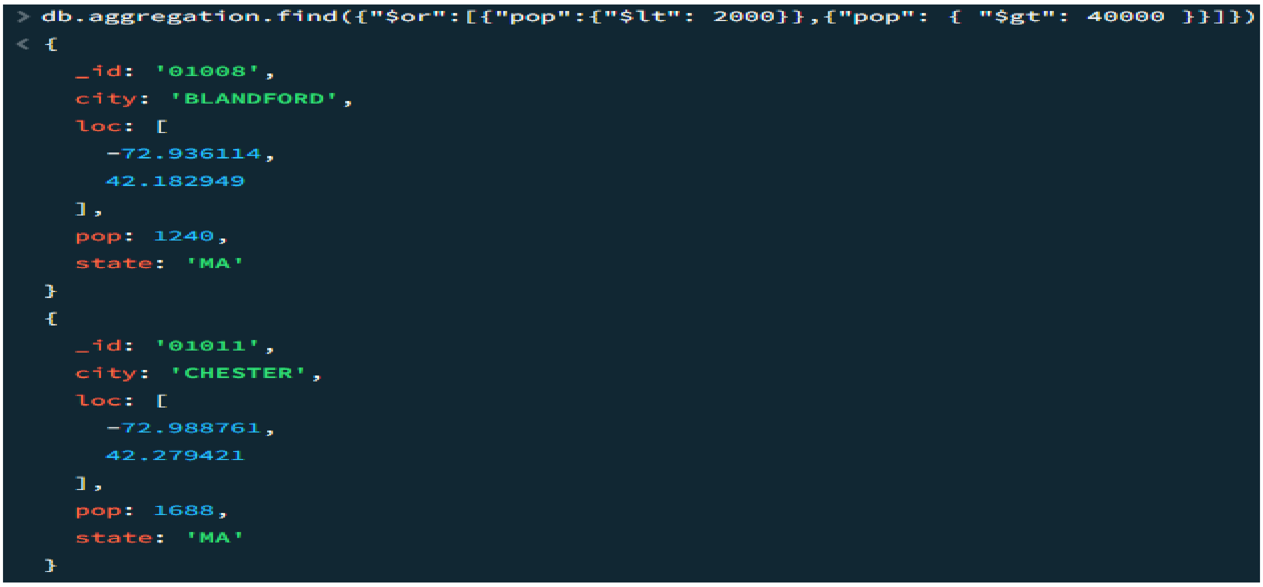
Logical Operators

1. Find cities where the population is greater than 20,000 **AND** the state is "MA".



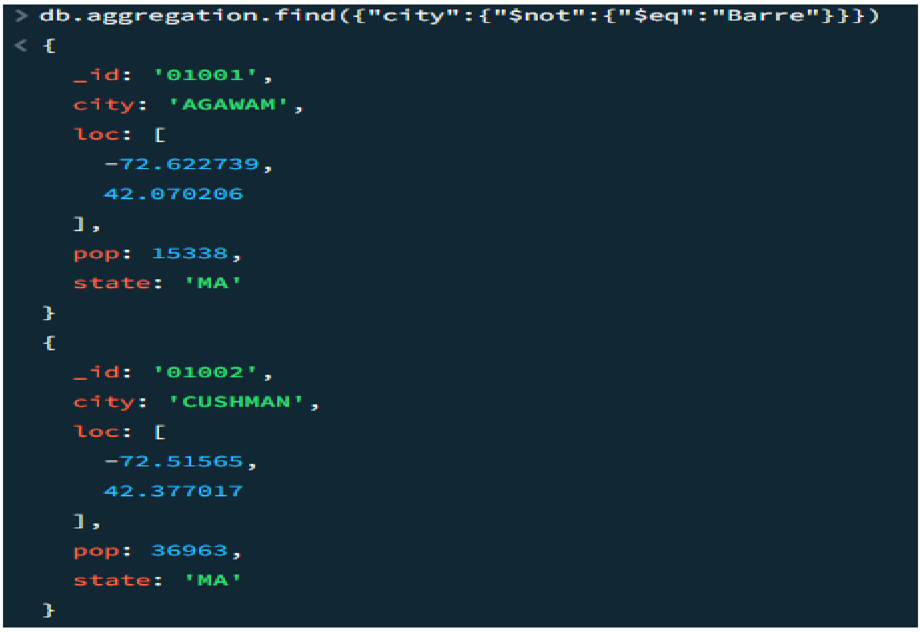
```
> db.aggregation.find({"$and": [{ "pop":{ "$gt": 20000 } },{ "state": "MA" }]})
< {
    _id: '01002',
    city: 'CUSHMAN',
    loc: [
      -72.51565,
      42.377017
    ],
    pop: 36963,
    state: 'MA'
  }
  {
    _id: '01013',
    city: 'CHICOPEE',
    loc: [
      -72.607962,
      42.162046
    ],
    pop: 23396,
    state: 'MA'
  }
```

2. Find cities where the population is either **less than 2,000 OR** greater than **40,000**.



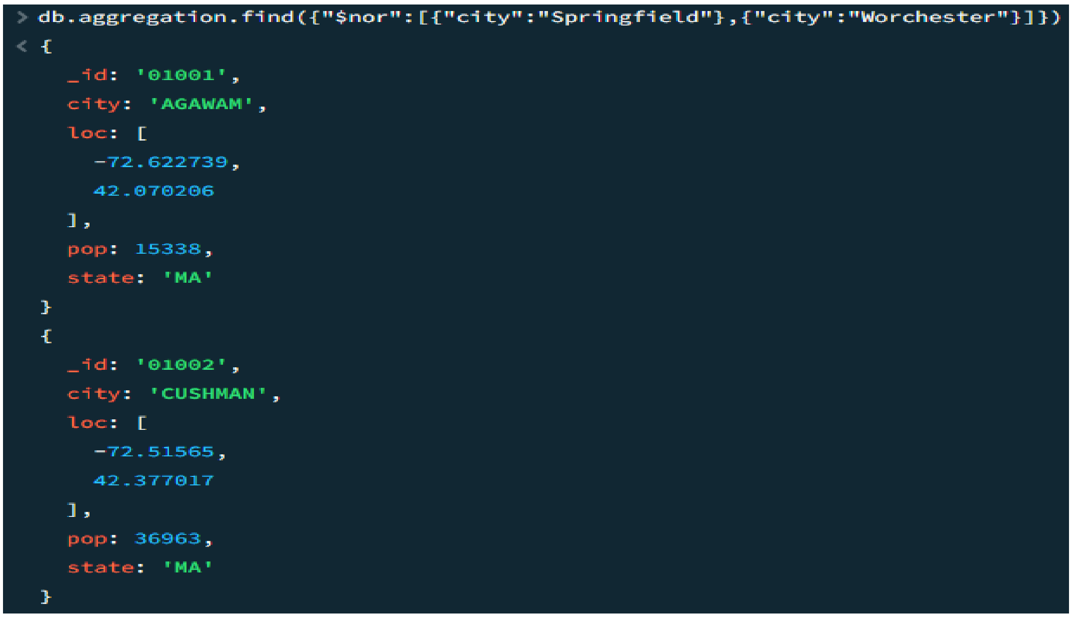
```
> db.aggregation.find({"$or":[{"pop":{"$lt": 2000}},{"pop": { "$gt": 40000 }}]})
< {
    _id: '01008',
    city: 'BLANDFORD',
    loc: [
      -72.936114,
      42.182949
    ],
    pop: 1240,
    state: 'MA'
  }
  {
    _id: '01011',
    city: 'CHESTER',
    loc: [
      -72.988761,
      42.279421
    ],
    pop: 1688,
    state: 'MA'
  }
```

3. Find cities where city is not equal to “Barre”.



```
> db.aggregation.find({"city":{"$not":{"$eq":"Barre"}}})
< {
    _id: '01001',
    city: 'AGAWAM',
    loc: [
      -72.622739,
      42.070206
    ],
    pop: 15338,
    state: 'MA'
  }
  {
    _id: '01002',
    city: 'CUSHMAN',
    loc: [
      -72.51565,
      42.377017
    ],
    pop: 36963,
    state: 'MA'
  }
```

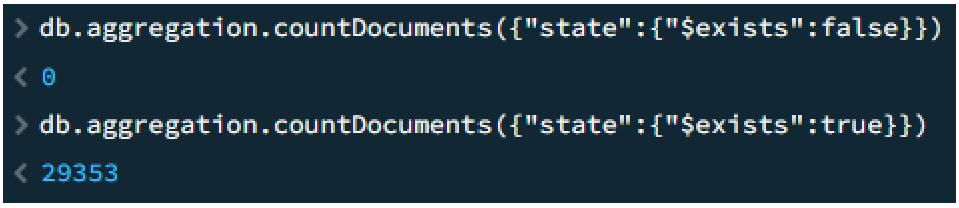
4. Find cities that are **NOT** "SPRINGFIELD" **AND NOT** "WORCESTER".



```
> db.aggregation.find({"$nor":[{"city":"Springfield"},{"city":"Worchester"}]})
< {
    _id: '01001',
    city: 'AGAWAM',
    loc: [
      -72.622739,
      42.070206
    ],
    pop: 15338,
    state: 'MA'
  }
  {
    _id: '01002',
    city: 'CUSHMAN',
    loc: [
      -72.51565,
      42.377017
    ],
    pop: 36963,
    state: 'MA'
  }
```

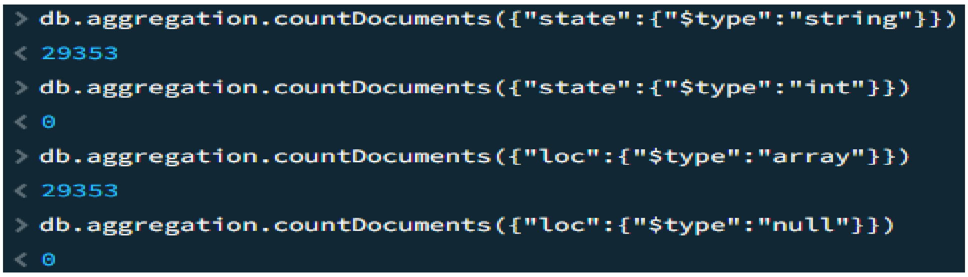
Element Operator

1.exists



```
> db.aggregation.countDocuments({"state":{"$exists":false}})
< 0
> db.aggregation.countDocuments({"state":{"$exists":true}})
< 29353
```

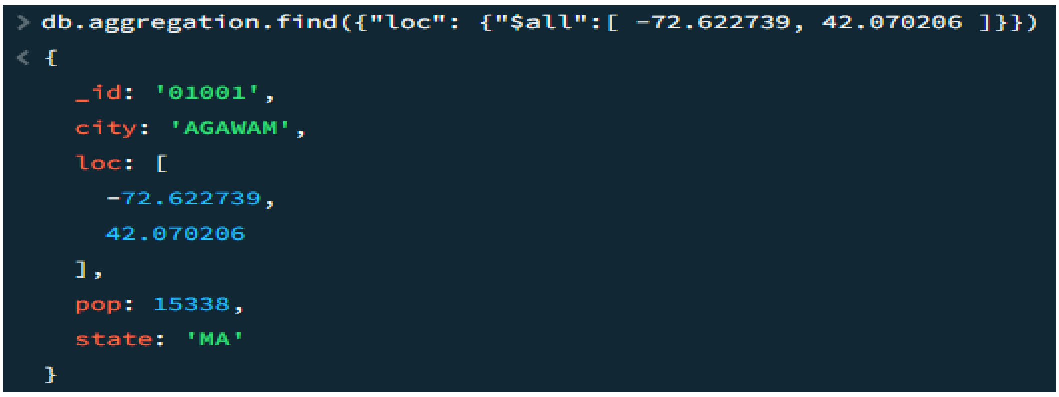
2.type



```
> db.aggregation.countDocuments({"state":{"$type":"string"}})
< 29353
> db.aggregation.countDocuments({"state":{"$type":"int"}})
< 0
> db.aggregation.countDocuments({"loc":{"$type":"array"}})
< 29353
> db.aggregation.countDocuments({"loc":{"$type":"null"}})
< 0
```

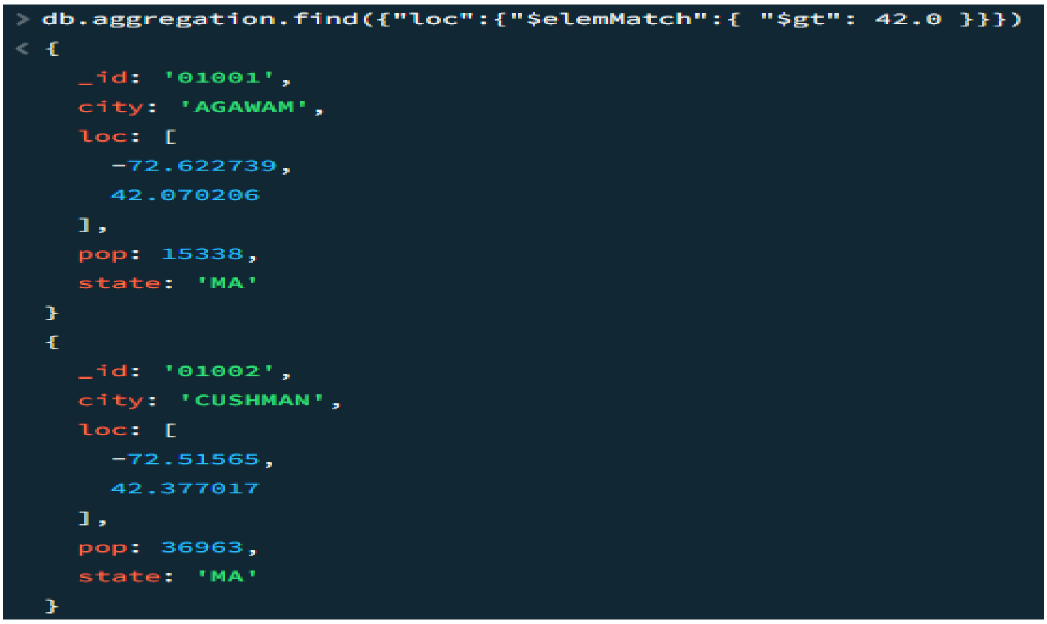
Array Operators.

1.all



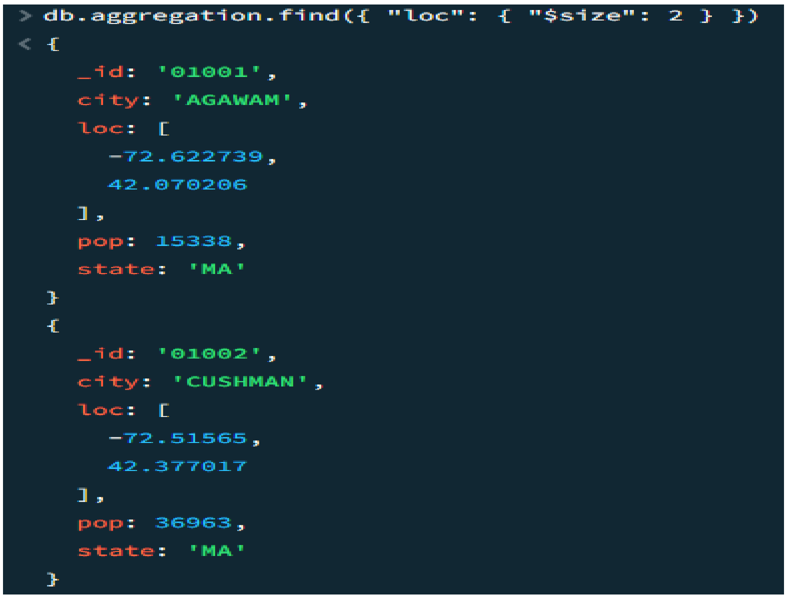
```
> db.aggregation.find({"loc": {"$all":[ -72.622739, 42.070206 ]}})
< {
    _id: '01001',
    city: 'AGAWAM',
    loc: [
      -72.622739,
      42.070206
    ],
    pop: 15338,
    state: 'MA'
  }
```

2. elemmatch



```
> db.aggregation.find({"loc":{"$elemMatch":{ "$gt": 42.0 }}})
< {
    _id: '01001',
    city: 'AGAWAM',
    loc: [
      -72.622739,
      42.070206
    ],
    pop: 15338,
    state: 'MA'
  }
  {
    _id: '01002',
    city: 'CUSHMAN',
    loc: [
      -72.51565,
      42.377017
    ],
    pop: 36963,
    state: 'MA'
  }
```

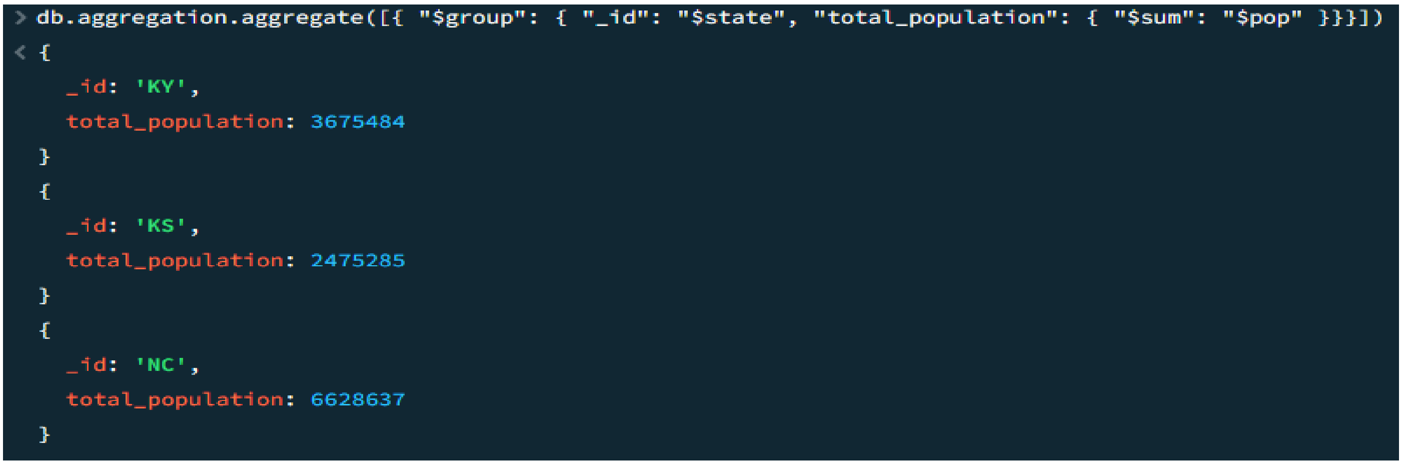
3.size



```
> db.aggregation.find({ "loc": { "$size": 2 } })
< {
    _id: '01001',
    city: 'AGAWAM',
    loc: [
      -72.622739,
      42.070206
    ],
    pop: 15338,
    state: 'MA'
  }
  {
    _id: '01002',
    city: 'CUSHMAN',
    loc: [
      -72.51565,
      42.377017
    ],
    pop: 36963,
    state: 'MA'
  }
```

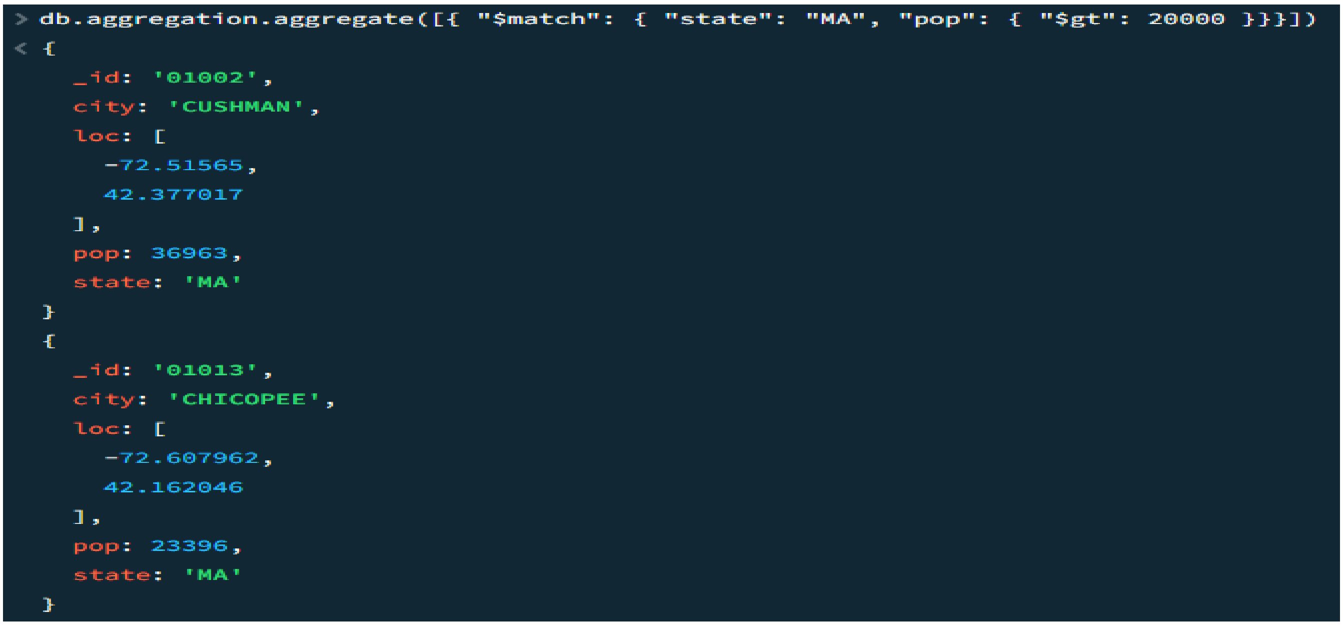
ADVANCED OPERATORS.

1.group - Find the **total population** of each state.



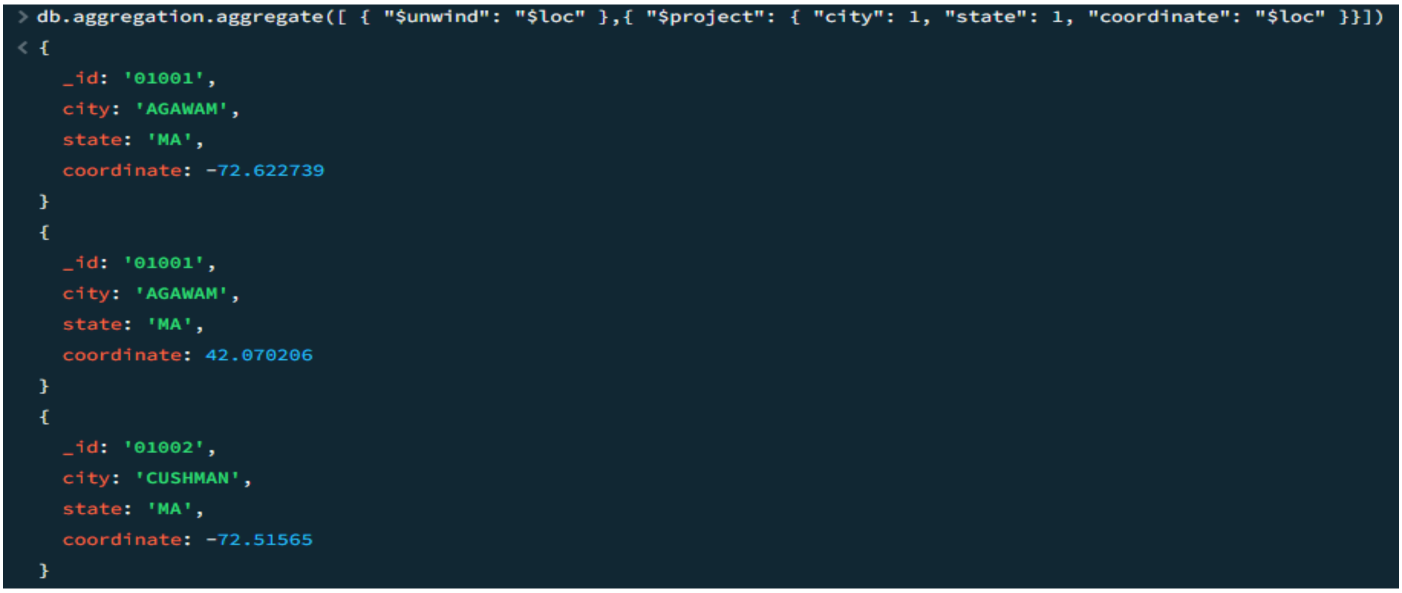
```
> db.aggregation.aggregate([{ "$group": { "_id": "$state", "total_population": { "$sum": "$pop" }}}])
< {
    _id: 'KY',
    total_population: 3675484
  }
  {
    _id: 'KS',
    total_population: 2475285
  }
  {
    _id: 'NC',
    total_population: 6628637
  }
```

2.match - Find cities in **state "MA"** with a population greater than **20,000**.



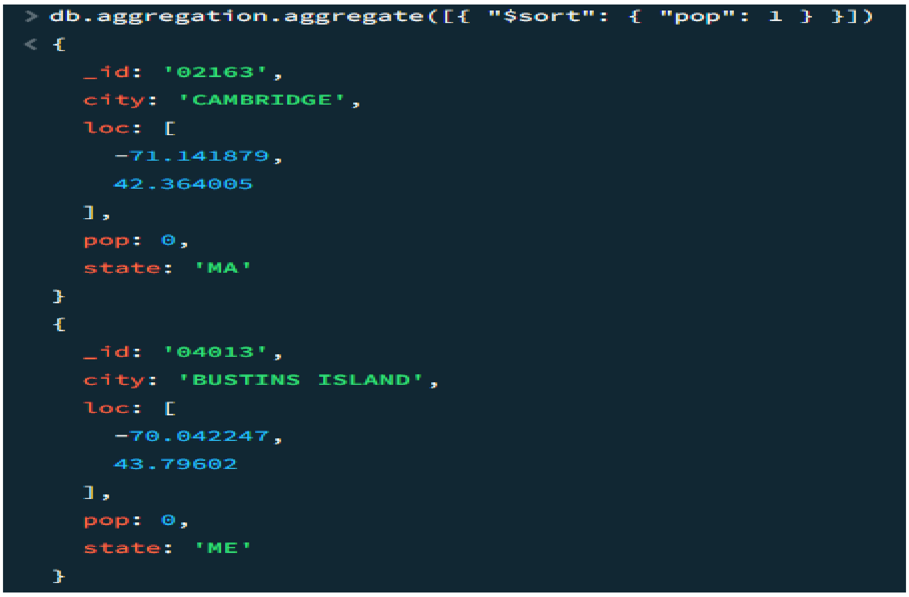
```
> db.aggregation.aggregate([{ "$match": { "state": "MA", "pop": { "$gt": 20000 }}}])
< {
    _id: '01002',
    city: 'CUSHMAN',
    loc: [
      -72.51565,
      42.377017
    ],
    pop: 36963,
    state: 'MA'
  }
  {
    _id: '01013',
    city: 'CHICOPEE',
    loc: [
      -72.607962,
      42.162046
    ],
    pop: 23396,
    state: 'MA'
  }
```

3. unwind



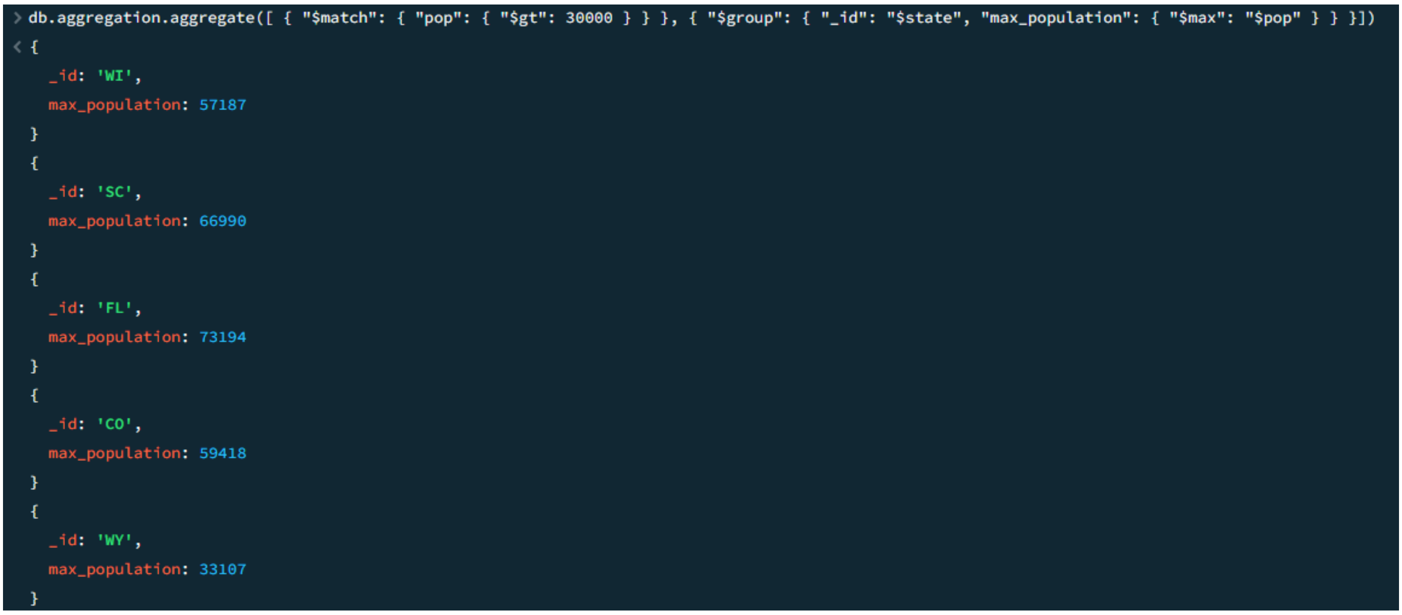
```
> db.aggregation.aggregate([ { "$unwind": "$loc" },{ "$project": { "city": 1, "state": 1, "coordinate": "$loc" }}])
< {
    _id: '01001',
    city: 'AGAWAM',
    state: 'MA',
    coordinate: -72.622739
  }
  {
    _id: '01001',
    city: 'AGAWAM',
    state: 'MA',
    coordinate: 42.070206
  }
  {
    _id: '01002',
    city: 'CUSHMAN',
    state: 'MA',
    coordinate: -72.51565
  }
```

4.sort - Find all cities, sorted by population from lowest to highest



```
> db.aggregation.aggregate([{ "$sort": { "pop": 1 } }])
< {
    _id: '02163',
    city: 'CAMBRIDGE',
    loc: [
      -71.141879,
      42.364005
    ],
    pop: 0,
    state: 'MA'
  }
  {
    _id: '04013',
    city: 'BUSTINS ISLAND',
    loc: [
      -70.042247,
      43.79602
    ],
    pop: 0,
    state: 'ME'
  }
```

5. Combining all operators.



```
> db.aggregation.aggregate([ { "$match": { "pop": { "$gt": 30000 } } }, { "$group": { "_id": "$state", "max_population": { "$max": "$pop" } } }])
< {
    _id: 'WI',
    max_population: 57187
  }
  {
    _id: 'SC',
    max_population: 66990
  }
  {
    _id: 'FL',
    max_population: 73194
  }
  {
    _id: 'CO',
    max_population: 59418
  }
  {
    _id: 'WY',
    max_population: 33107
  }
```