

**Practical 5**  
**31-01-25**  
**Aggregation USING Mangodb**

**Write- up:**

- Comparison Operators
- Logical Operators
- Element Operators
- Array Operators

**MONGO IMPORT**

- How to download and use mongodb import utility
- <https://www.mongodb.com/try/download/database-tools>
- download database-tools and unzip.
- Copy database tools to MongoDB bin location.
- start cmd. mongoimport
- Download sample json file from
- <https://media.mongodb.org/zips.json>
- mongoimport --db sampledata collection samplecollection --file C:\sample\_data\_from\_mongodb.json

**Solve the case from :**

[https://github.com/mattdavis0351/mongodb-labs/blob/master/exercises/02\\_intermediate-mongo-queries.md](https://github.com/mattdavis0351/mongodb-labs/blob/master/exercises/02_intermediate-mongo-queries.md)

## Create Database

x

Database Name

practical\_5

Collection Name

p5

☐ Time-Series

Time-series collections efficiently store sequences of measurements over a period of time. [Learn More](#)

➤ **Additional preferences** (e.g. Custom collation, Capped, Clustered collections)

Cancel

Create Database



### This collection has no data

It only takes a few seconds to import data from a JSON or CSV file.

Import Data

## Import

To collection practical\_5.p5

Import file: datap5.json 

### Options


☐ Stop on errors

Cancel

Import

+ ADD DATA ▾

 EXPORT DATA ▾

 UPDATE

 DELETE

```
_id: "01001"  
city: "AGAWAM"  
▸ loc: Array (2)  
pop: 15338  
state: "MA"
```

```
_id: "01002"  
city: "CUSHMAN"  
▸ loc: Array (2)  
pop: 36963  
state: "MA"
```

```
_id: "01005"  
city: "BARRE"  
▸ loc: Array (2)
```

```
db.p5.find().pretty()
{
  _id: '01001',
  city: 'AGAWAM',
  loc: [
    -72.622739,
    42.070206
  ],
  pop: 15338,
  state: 'MA'
}
...
```

### Comparison Operators

Name	Description
\$eq	Matches values that are equal to a specified value.
\$gt	Matches values that are greater than a specified value.
\$gte	Matches values that are greater than or equal to a specified value.
\$in	Matches any of the values specified in an array.
\$lt	Matches values that are less than a specified value.
\$lte	Matches values that are less than or equal to a specified value.
\$ne	Matches all values that are not equal to a specified value.
\$nin	Matches none of the values specified in an array.

```
db.p5.find({ "pop": { "$eq": 15338 } })
{
  _id: '01001',
  city: 'AGAWAM',
```

```
loc: [  
  -72.622739,  
  42.070206  
],  
pop: 15338,  
state: 'MA'  
}  
...  
  
db.p5.find({ "pop": { "$ne": 15338 } })  
{  
  _id: '01002',  
  city: 'CUSHMAN',  
  loc: [  
    -72.51565,  
    42.377017  
  ],  
  pop: 36963,  
  state: 'MA'  
}  
...  
  
db.p5.find({ "pop": { "$gt": 15000 } })  
{  
  _id: '01001',  
  city: 'AGAWAM',  
  loc: [  
    -72.622739,  
    42.070206  
  ],  
  pop: 15338,  
  state: 'MA'  
}
```

...

```
db.p5.find({ "pop": { "$lt": 5000 } })
```

```
{  
  _id: '01005',  
  city: 'BARRE',  
  loc: [  
    -72.108354,  
    42.409698  
  ],  
  pop: 4546,  
  state: 'MA'  
}
```

...

```
db.p5.find({ "state": { "$in": ["MA", "NY"] } })
```

```
{  
  _id: '01001',  
  city: 'AGAWAM',  
  loc: [  
    -72.622739,  
    42.070206  
  ],  
  pop: 15338,  
  state: 'MA'  
}
```

...

## Logical Operators

Name	Description
\$and	Joins query clauses with a logical AND returns all documents that match the conditions of both clauses.
\$or	Joins query clauses with a logical OR returns all documents that match the conditions of either clause.
\$not	Inverts the effect of a query expression and returns documents that do not match the query expression.
\$nor	Joins query clauses with a logical NOR returns all documents that fail to match both clauses.

```
db.p5.find({ "$and": [ { "pop": { "$gt": 5000 } }, { "state": "MA" } ] })  
{  
  _id: '01001',  
  city: 'AGAWAM',  
  loc: [  
    -72.622739,  
    42.070206  
  ],  
  pop: 15338,  
  state: 'MA'  
}  
...
```

```
db.p5.find({ "$or": [ { "pop": { "$gt": 5000 } }, { "state": "MA" } ] })  
{  
  _id: '01001',  
  city: 'AGAWAM',  
  loc: [  
    -72.622739,  
    42.070206  
  ],  
  pop: 15338,  
  state: 'MA'  
}
```

```

],
pop: 15338,
state: 'MA'
}
...

db.p5.find({ "pop": { "$not": { "$gt": 5000 } } })
{
  _id: '01005',
  city: 'BARRE',
  loc: [
    -72.108354,
    42.409698
  ],
  pop: 4546,
  state: 'MA'
}
...

```

## Element Operators

Name	Description
\$exists	Matches documents that have the specified field.
\$type	Selects documents if a field is of the specified type.

```

db.p5.find({ "loc": { "$exists": true } })
{
  _id: '01001',
  city: 'AGAWAM',
  loc: [
    -72.622739,
    42.070206
  ]
}

```



```
],  
  pop: 15338,  
  state: 'MA'  
}  
...  
  
db.p5.find({ "pop": { "$type": "int" } })  
{  
  _id: '01001',  
  city: 'AGAWAM',  
  loc: [  
    -72.622739,  
    42.070206  
  ],  
  pop: 15338,  
  state: 'MA'  
}  
...
```

### Array Operators

Name	Description
\$all	Matches arrays that contain all elements specified in the query.
\$elemMatch	Selects documents if element in the array field matches all the specified \$elemMatch conditions.
\$size	Selects documents if the array field is a specified size.

```
db.p5.find({ "loc": { "$all": [ -72.622739, 42.070206 ] } })  
{  
  _id: '01001',  
  city: 'AGAWAM',  
  loc: [  

```

```
-72.622739,  
42.070206  
],  
pop: 15338,  
state: 'MA'  
}
```

```
db.p5.find({ "loc": { "$elemMatch": { "$gt": -73, "$gt": 42 } } })  
{  
  _id: '01001',  
  city: 'AGAWAM',  
  loc: [  
    -72.622739,  
    42.070206  
  ],  
  pop: 15338,  
  state: 'MA'  
}  
...
```

```
db.p5.find({ "loc": { "$size": 2 } })  
{  
  _id: '01001',  
  city: 'AGAWAM',  
  loc: [  
    -72.622739,  
    42.070206  
  ],  
  pop: 15338,  
  state: 'MA'  
}  
...
```

**The "\$group" operator**

The \$group operator groups the documents by an identifier specified by `_id` field, and based on that distinct grouping, performs an aggregation like \$sum and returns the resulting documents.

```
db.p5.aggregate([
  {
    "$group": {
      "_id": "$state",      // Group by state
      "total_population": { "$sum": "$pop" } // Sum of population
    }
  }
])

{
  _id: 'IN',
  total_population: 5544136
}
{
  _id: 'NC',
  total_population: 6628637
}
...
```

**The "\$match" operator**

The \$match operator matches input documents to a given criteria and passes those matched documents to the next stage of the pipeline.

```
db.p5.aggregate([
  {
    "$match": { "pop": { "$gt": 10000 } } // Match cities where population is
    greater than 10,000
  }
])
```

```
)
```

```
{  
  _id: '01001',  
  city: 'AGAWAM',  
  loc: [  
    -72.622739,  
    42.070206  
  ],  
  pop: 15338,  
  state: 'MA'  
}  
...
```

**The "\$sort" operator**

```
db.p5.aggregate([  
  {  
    "$sort": { "pop": -1 } // -1 for descending, 1 for ascending  
  }  
])
```

```
{  
  _id: '60623',  
  city: 'CHICAGO',  
  loc: [  
    -87.7157,  
    41.849015  
  ],  
  pop: 112047,  
  state: 'IL'  
}
```

**The "\$unwind" operator**

The \$unwind operator deconstructs an array resulting in a document for each array element. The concept will become more evident through the exercise.

```
db.p5.aggregate([
  {
    "$unwind": "$loc" // Unwinds the 'loc' array, so each element gets its own
    document
  }
])

{
  _id: '01001',
  city: 'AGAWAM',
  loc: -72.622739,
  pop: 15338,
  state: 'MA'
}
...
```

**Combined Example: Using \$match, \$group, \$sort, and \$unwind together**

```
db.p5.aggregate([
  {
    "$match": { "pop": { "$gt": 5000 } } // First, match cities with population >
    5000
  },
  {
    "$group": {
      "_id": "$state", // Group by state
      "total_population": { "$sum": "$pop" } // Calculate total population for each
      state
    }
  }
])
```

```
    },  
    {  
      "$sort": { "total_population": -1 } // Sort states by total population in  
      descending order  
    }  
  ]  
}
```

```
{  
  _id: 'CA',  
  total_population: 29009632  
}  
{  
  _id: 'NY',  
  total_population: 16462946  
}  
...
```