**Practical-5: Aggregation USING Mangodb**

**Date:-31/01/2025 Submission Date:- 7/01/2025**

**Write- up:**

· Comparison Operators

· Logical Operators

· Element Operators

· Array Operators

How to download and use mongodbimport utility

http://www.mongodb.com/try/download/database-tools

download database-tools and unzip.

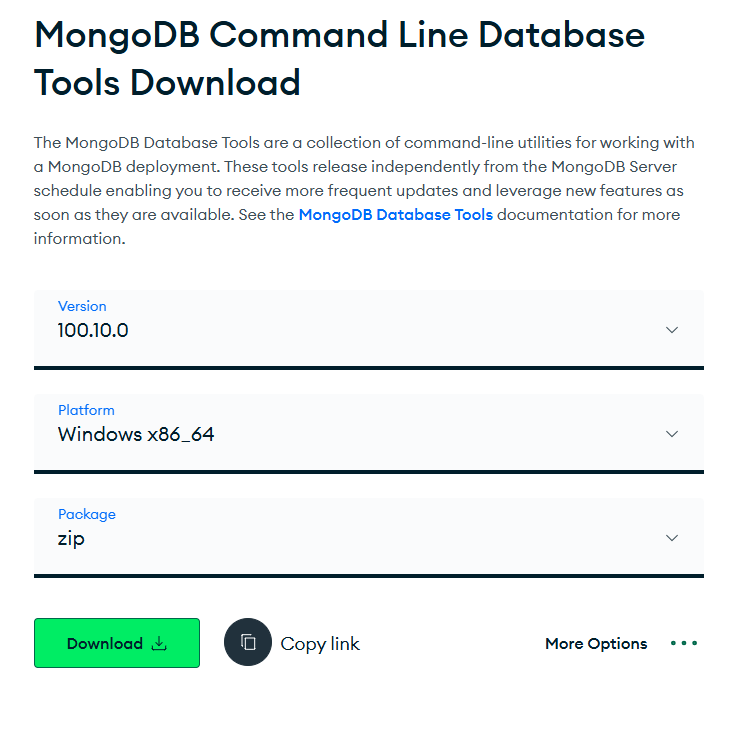
Copy database tools to MongoDB bin location.

start and mongoimport

2. Download sample jaon file from https://media.mongodb.org/zipe.jaon

mongoimport --db sampledata --collection samplecollection--file Cr\sample\_data\_from\_mongodb.jaon





**A.Comparison Query Operators**

**1. Match Cities with a Specific Population ($eq):**

**Code: db.aggregation.find({ "pop": { $eq: 15338 } })**

**output:**

{

\_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,

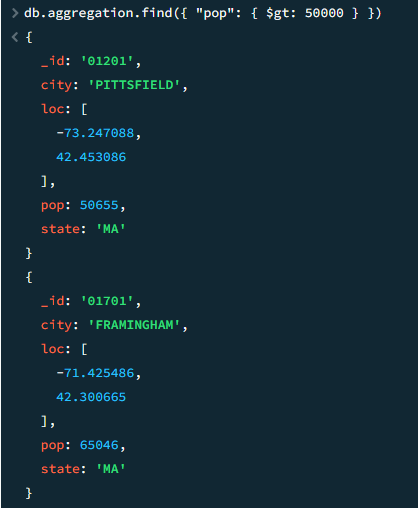
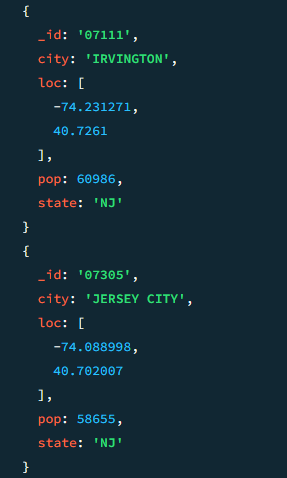
state: 'MI'

}

**2. Find Cities with Population Greater Than ($gt):**

**Code: db.aggregation.find({ "pop": { $gt: 50000 } })**

**Output:**

** **

3. Find Cities with Population Greater Than or Equal ($gte):

**Code: db.aggregation.find({ "pop": { $gte: 30000 } })**

**Output:**

** **

**4. Find Cities with Population Less Than ($lt)**

**Code: db.aggregation.find({ "pop": { $lt: 5000 } })**

**Output:**

** **

**5. Find Cities with Population Less Than or Equal ($lte)**

**Code:** db.aggregation.find({ "pop": { $lte: 5000 } })

**Output:**

** **

**6. Match Cities Excluding a Population ($ne)**

**Code:** db.aggregation.find({ "pop": { $ne: 10000 } })

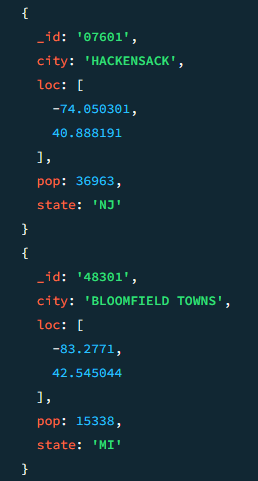
**Output:**

** **

**7. Match Cities Within a List ($in)**

**Code:** db.aggregation.find({ "pop": { $in: [15338,23396 , 36963] } })

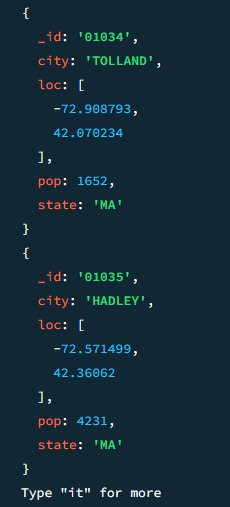
**Output:**

**8. Match Cities NOT in a List ($nin)**

**Code: db.aggregation.find({ "pop": { $nin: [10000,20000,30000] } })**

**Output:**

** **

**Logical Operators**

**OR operation:**

**1.Code:** **db.aggregation.find({**

**$or: [**

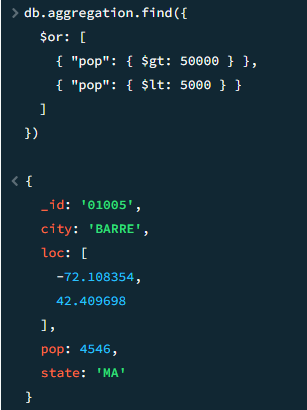
**{ "pop": { $gt: 50000 } },**

**{ "pop": { $lt: 5000 } }**

**]**

**})**

**Output:**

** **

**2. Find cities where population is greater than 50,000 OR city name is "SPRINGFIELD"**:

**code:**

**db.aggregation.find({**

**$or: [**

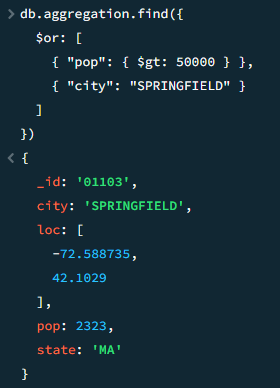
**{ "pop": { $gt: 50000 } },**

**{ "city": "SPRINGFIELD" }**

**]**

**})**

**Output:**

** **

**AND OPERATION:**

Find cities where **population is greater than 10,000** AND **state is "MA"**:

**Code:**

**db.aggregation.find({**

**$and: [**

**{ "pop": { $gt: 10000 } },**

**{ "state": "MA" }**

**]**

**})**

**OUTPUT:**

** **

## **NOT Operator**

## **Find cities not in Massachusetts ("MA"):**

**Code: db.aggregation.find({**

**"state": { $not: { $eq: "MA" } }**

**})**

**Output:**

**NOR OPERATOR**

**Find cities where population is NOT greater than 50,000 AND city is NOT "SPRINGFIELD":**

**Code: db.aggregation.find({**

**$nor: [**

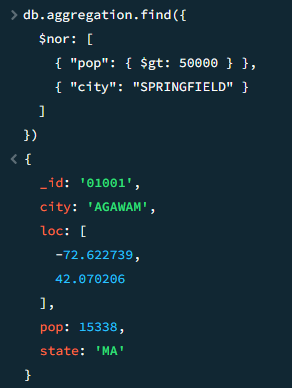
**{ "pop": { $gt: 50000 } },**

**{ "city": "SPRINGFIELD" }**

**]**

**})**

**output:**

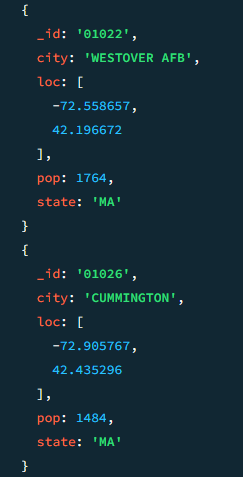
**Element Operators**

**EXITS OPERATOR:**

**Find documents where the "pop" field exists**

**CODE: db.aggregation.find({ "pop": { $exists: true } })**

**Output:**

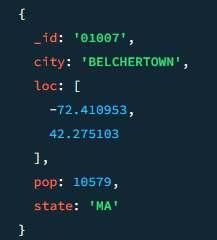
 

**TYPE OPERATOR**:

**Find documents where "pop" is a Number (int or double)**

**CODE: db.aggregation.find({ "pop": { $type: "number" } })**

**Output:**

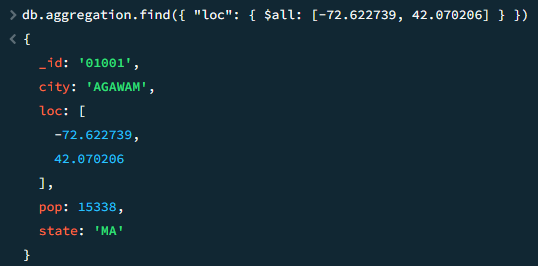
**ARRAY OPERATION**

**ALL OPERATOR**

Find Cities That Contain Both -72.622739 AND 42.070206 in loc Array

Code: db.aggregation.find({ "loc": { $all: [-72.622739, 42.070206] } })

Output:



**Element Match operator**

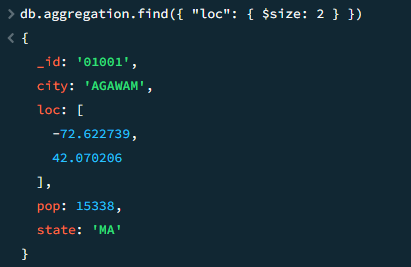
Find Cities Where loc Contains a Value Between -73 and -72 OR Population is Between 10,000 and 20,000

**Output:**



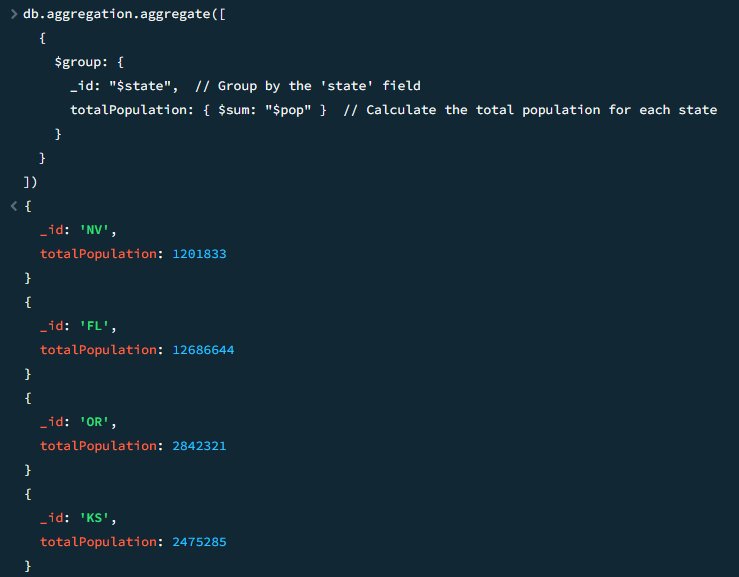
**Size operator**

Find Cities Where loc Has Exactly 2 Elements

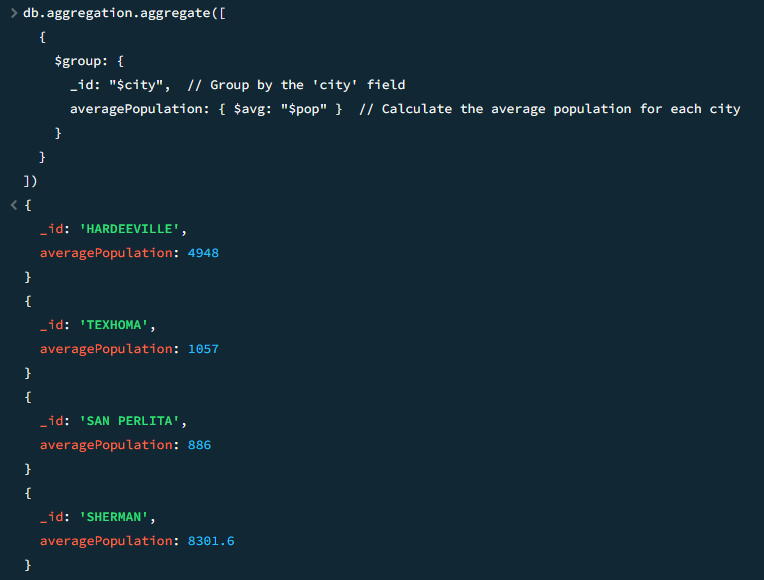


**Advanced Queries**

**Group operator**

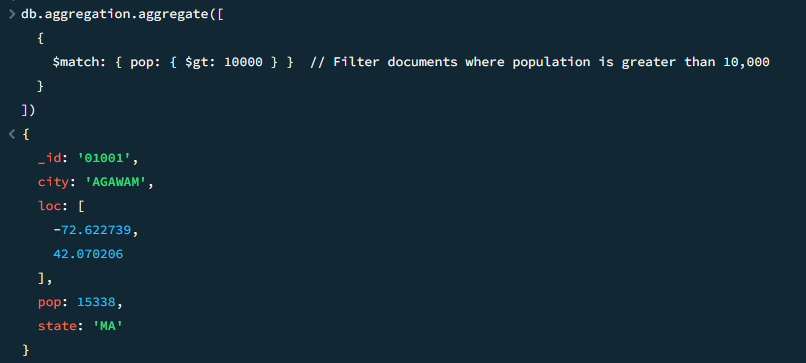


**Group by City and Calculate Average Population:**

****

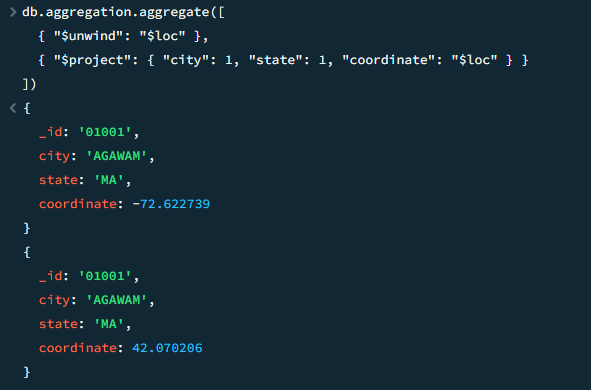
**Match operator:**

**Filter Documents by population**

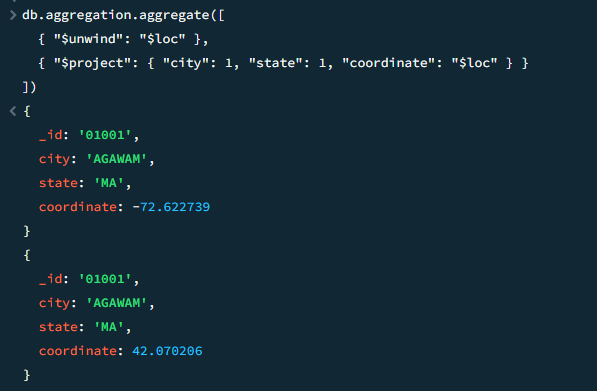


**SORT OERATOR:**

Find all cities, sorted by **population from lowest to highest**.



**Unwind operator:**

****

**Combination operation**

****