1.
2.
3.
4.
5.
6.
7.
8.
9.
10.
11.

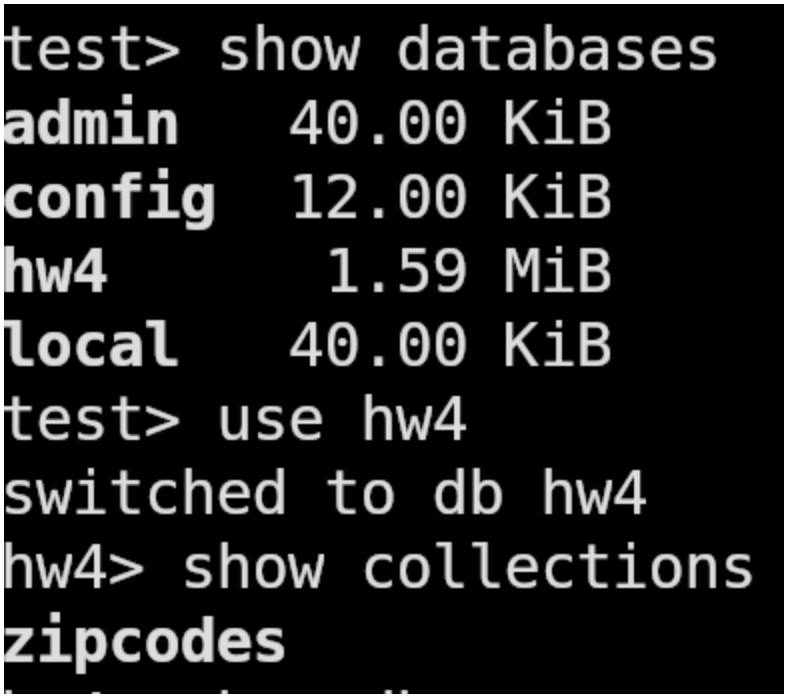
1.

Just File:

```
h := env_var("MONGODB_HOME")
p := env_var("MONGODB_PORT")


mongo-init:
  #!/usr/bin/env bash
  set -euxo pipefail
  if [ ! -d "$MONGODB_HOME" ]; then
    mkdir -p "$MONGODB_HOME"
  fi

mongo-serve *ARGS:
  mongod --dbpath {{h}} --port {{p}} {{ARGS}}

mongo *ARGS:
  mongosh --port {{p}} {{ARGS}}

mongo-import *ARGS:
  mongoimport --db hw4 --collection zipcodes --file {{ARGS}}
```

Commands I used:

```
      just mongo-serve
      just mongo-init
      just mongo-import zips
```




```
db.zipcodes.find().limit(15)
```

Output:

```
hw4> db.zipcodes.find().limit(15)
[
  {
    _id: '01007',
    city: 'BELCHERTOWN',
    loc: [ -72.410953, 42.275103 ],
    pop: 10579,
    state: 'MA'
  },
  {
    _id: '01031',
    city: 'GILBERTVILLE',
    loc: [ -72.198585, 42.332194 ],
    pop: 2385,
    state: 'MA'
  },
  {
    _id: '01002',
    city: 'CUSHMAN',
```

```
  loc: [ -72.51565, 42.377017 ],
  pop: 36963,
  state: 'MA'
},
{
  _id: '01020',
  city: 'CHICOPEE',
  loc: [ -72.576142, 42.176443 ],
  pop: 31495,
  state: 'MA'
},
{
  _id: '01032',
  city: 'GOSHEN',
  loc: [ -72.844092, 42.466234 ],
  pop: 122,
  state: 'MA'
},
{
  _id: '01033',
  city: 'GRANBY',
  loc: [ -72.520001, 42.255704 ],
  pop: 5526,
  state: 'MA'
},
{
  _id: '01036',
  city: 'HAMPDEN',
  loc: [ -72.431823, 42.064756 ],
  pop: 4709,
  state: 'MA'
},
{
  _id: '01012',
  city: 'CHESTERFIELD',
  loc: [ -72.833309, 42.38167 ],
  pop: 177,
  state: 'MA'
},
{
  _id: '01039',
  city: 'HAYDENVILLE',
  loc: [ -72.703178, 42.381799 ],
```

```
  pop: 1387,
  state: 'MA'
},
{
  _id: '01038',
  city: 'HATFIELD',
  loc: [ -72.616735, 42.38439 ],
  pop: 3184,
  state: 'MA'
},
{
  _id: '01040',
  city: 'HOLYOKE',
  loc: [ -72.626193, 42.202007 ],
  pop: 43704,
  state: 'MA'
},
{
  _id: '01034',
  city: 'TOLLAND',
  loc: [ -72.908793, 42.070234 ],
  pop: 1652,
  state: 'MA'
},
{
  _id: '01035',
  city: 'HADLEY',
  loc: [ -72.571499, 42.36062 ],
  pop: 4231,
  state: 'MA'
},
{
  _id: '01050',
  city: 'HUNTINGTON',
  loc: [ -72.873341, 42.265301 ],
  pop: 2084,
  state: 'MA'
},
{
  _id: '01053',
  city: 'LEEDS',
  loc: [ -72.703403, 42.354292 ],
  pop: 1350,
```

```
    state: 'MA'
  }
]
```

2.

```
db.zipcodes.find({ "state": "CA", "pop": { "$gt": 50000 }, "loc": { "$gt": 35 } }).sort({ "pop": -1 }).limit(5)
```


```
[
  {
    _id: '94501',
    city: 'COAST GUARD ISLA',
    loc: [ -122.260516, 37.764783 ],
    pop: 76110,
    state: 'CA'
  },
  {
    _id: '94110',
    city: 'SAN FRANCISCO',
    loc: [ -122.415344, 37.750858 ],
    pop: 70770,
    state: 'CA'
  },
  {
    _id: '95351',
    city: 'MODESTO',
    loc: [ -121.006033, 37.625022 ],
    pop: 69275,
    state: 'CA'
  },
  {
    _id: '95076',
    city: 'LA SELVA BEACH',
    loc: [ -121.763437, 36.920515 ],
    pop: 68295,
    state: 'CA'
  },
  {
    _id: '94533',
    city: 'FAIRFIELD',
    loc: [ -122.03565, 38.267084 ],
    pop: 65455,
    state: 'CA'
  }
]
```

3.

```
hw4> db.zipcodes.find({ "$and": [ { "state": { "$ne": "CA" } }, { "$or": [
{ "loc.0": { "$lt": -120 } }, { "loc.1": { "$lt": 40 } }] }] }).limit(5)\

[
  {
    _id: '08003',
    city: 'CHERRY HILL',
    loc: [ -74.970568, 39.880453 ],
    pop: 29058,
    state: 'NJ'
  },
  {
    _id: '08005',
    city: 'BARNEGAT',
    loc: [ -74.246988, 39.755248 ],
    pop: 13036,
    state: 'NJ'
  },
  {
    _id: '08007',
    city: 'BARRINGTON',
    loc: [ -75.056361, 39.865062 ],
    pop: 5185,
    state: 'NJ'
  },
  {
    _id: '08008',
    city: 'HARVEY CEDARS',
    loc: [ -74.189033, 39.636347 ],
    pop: 8647,
    state: 'NJ'
  },
  {
    _id: '08012',
    city: 'WASHINGTON',
    loc: [ -75.058747, 39.774104 ],
    pop: 35874,
    state: 'NJ'
  }
]
```

4.

```
db.zipcodes.aggregate([ { $group: { _id: "$city", totalPopulation: { $sum:
"$pop" } } }, { $sort: { totalPopulation: -1 } }, { $limit: 5 }, {
$project: { _id: 0, city: "$_id", population: "$totalPopulation" } }] )
```

```
[
  { city: 'CHICAGO', population: 2452177 },
  { city: 'BROOKLYN', population: 2341387 },
  { city: 'HOUSTON', population: 2123053 },
  { city: 'LOS ANGELES', population: 2102295 },
  { city: 'PHILADELPHIA', population: 1639862 }
]
```

5.

```
db.zipcodes.aggregate([ { $group: { _id: "$state", zipCodeCount: { $sum: 1 } } }, { $match: { zipCodeCount: { $gt: 300, $lt: 500 } } }, { $sort: { zipCodeCount: 1 } }] )
```

```
[
  { _id: 'MT', zipCodeCount: 314 },
  { _id: 'SC', zipCodeCount: 350 },
  { _id: 'MS', zipCodeCount: 363 },
  { _id: 'SD', zipCodeCount: 384 },
  { _id: 'OR', zipCodeCount: 384 },
  { _id: 'ND', zipCodeCount: 391 },
  { _id: 'ME', zipCodeCount: 410 },
  { _id: 'CO', zipCodeCount: 414 },
  { _id: 'MD', zipCodeCount: 420 },
  { _id: 'LA', zipCodeCount: 464 },
  { _id: 'MA', zipCodeCount: 474 },
  { _id: 'WA', zipCodeCount: 484 }
]
```

6.

```
db.createCollection("customers", {
    validator: {
        $jsonSchema: {
            bsonType: "object",
            required: ["name", "zipcode", "avg_rating"],
            properties: {
                name: { bsonType: "string" },
                zipcode: { bsonType: "string" },
                avg_rating: {
                    bsonType: "double",
                    minimum: 0.0,
                    maximum: 10.0
                },
                last_order: {
                    bsonType: "object",
                    required: ["year"],
                    properties: {
                        year: { bsonType: "int" },
                        tags: {
                            bsonType: "array",
                            items: { bsonType: "string" }
                        }
                    }
                }
            }
        }
    }
})
```

```
test> db.createCollection("customers", {
...     validator: {
...         $jsonSchema: {
...             bsonType: "object",
...             required: ["name", "zipcode", "avg_rating"],
...             properties: {
...                 name: { bsonType: "string" },
...                 zipcode: { bsonType: "string" },
...                 avg_rating: {
...                     bsonType: "double",
...                     minimum: 0.0,
...                     maximum: 10.0
...                 },
...                 last_order: {
...                     bsonType: "object",
...                     required: ["year"],
...                     properties: {
...                         year: { bsonType: "int" },
...                         tags: {
...                             bsonType: "array",
...                             items: { bsonType: "string" }
...                         }
...                     }
...                 }
...             }
...         }
...     }
... })
{ ok: 1 }
```

7.

```
db.customers.insertMany([
   { name: "Omar Al Asaad", zipcode: "99503", avg_rating: 8.3 },
   { name: "Andrei T.", zipcode: "90025", avg_rating: 3.5, last_order: {
year: 2009 } },
   { name: "Bela T.", zipcode: "33126", avg_rating: 4.9, last_order: {
year: 2019, tags: ["art", "melancholy"] } },
   { name: "Nuri Bilge C.", zipcode: "90010", avg_rating: 6.5, last_order:
{ year: 3005 } }
])
```

```
test> db.customers.insertMany([
...     { name: "Omar Al Asaad", zipcode: "99503", avg_rating: 8.3 },
...     { name: "Andrei T.", zipcode: "90025", avg_rating: 3.5, last_order: { year: 2009 } },
...     { name: "Bela T.", zipcode: "33126", avg_rating: 4.9, last_order: { year: 2019, tags: ["art", "melancholy"] } },
...     { name: "Nuri Bilge C.", zipcode: "90010", avg_rating: 6.5, last_order: { year: 3005 } }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('65981af492a8ba8c2131b724'),
    '1': ObjectId('65981af492a8ba8c2131b725'),
    '2': ObjectId('65981af492a8ba8c2131b726'),
    '3': ObjectId('65981af492a8ba8c2131b727')
  }
}
```

```
test> db.customers.find()
[
  {
    _id: ObjectId('65981af492a8ba8c2131b724'),
    name: 'Omar Al Asaad',
    zipcode: '99503',
    avg_rating: 8.3
  },
  {
    _id: ObjectId('65981af492a8ba8c2131b725'),
    name: 'Andrei T.',
    zipcode: '90025',
    avg_rating: 3.5,
    last_order: { year: 2009 }
  },
  {
    _id: ObjectId('65981af492a8ba8c2131b726'),
    name: 'Bela T.',
    zipcode: '33126',
    avg_rating: 4.9,
    last_order: { year: 2019, tags: [ 'art', 'melancholy' ] }
  },
  {
    _id: ObjectId('65981af492a8ba8c2131b727'),
    name: 'Nuri Bilge C.',
    zipcode: '90010',
    avg_rating: 6.5,
    last_order: { year: 3005 }
  }
]
```

8.

```
db.customers.deleteMany({ "last_order.year": { $gt: 2023 } })
```

```
test> db.customers.deleteMany({
...     "last_order.year": { $gt: 2023 }
... })
{ acknowledged: true, deletedCount: 1 }
test> db.customers.find()
[
  {
    _id: ObjectId('65981af492a8ba8c2131b724'),
    name: 'Omar Al Asaad',
    zipcode: '99503',
    avg_rating: 8.3
  },
  {
    _id: ObjectId('65981af492a8ba8c2131b725'),
    name: 'Andrei T.',
    zipcode: '90025',
    avg_rating: 3.5,
    last_order: { year: 2009 }
  },
  {
    _id: ObjectId('65981af492a8ba8c2131b726'),
    name: 'Bela T.',
    zipcode: '33126',
    avg_rating: 4.9,
    last_order: { year: 2019, tags: [ 'art', 'melancholy' ] }
  }
]
```

9.

```
test> db.customers.updateOne(
...     { name: "Omar Al Asaad" },
...     { $set: { avg_rating: 15 } }
... )
Uncaught:
MongoServerError: Document failed validation
Additional information: {
  failingDocumentId: ObjectId('65981af492a8ba8c2131b724'),
  details: {
    operatorName: '$jsonSchema',
    schemaRulesNotSatisfied: [
      {
        operatorName: 'properties',
        propertiesNotSatisfied: [
          {
            propertyName: 'avg_rating',
            details: [
              {
                operatorName: 'maximum',
                specifiedAs: { maximum: 10 },
                reason: 'comparison failed',
                consideredValue: 15
              },
              {
                operatorName: 'bsonType',
                specifiedAs: { bsonType: 'double' },
                reason: 'type did not match',
                consideredValue: 15,
                consideredType: 'int'
              }
            ]
          }
        ]
      }
    ]
  }
}
```

It gave an error because the avg_rating we defined the collection with was a double between 0.0 and 10.0 so it violated the conditions. Even if we were to put it in double it would still violate it since the value is more than 15.0.

10.

```
db.customers.aggregate([
    {
        $lookup: {
            from: "zipcodes",
            localField: "zipcode",
            foreignField: "_id",
            as: "zipcode_info"
        }
    },
    {
        $unwind: "$zipcode_info"
    },
    {
        $sort: { "name": 1 }
    },
    {
        $project: {
            _id: 0,
            name: 1,
            city: "$zipcode_info.city",
            state: "$zipcode_info.state",
            zipcode: "$zipcode"
        }
    }
])
```

```
[
  {
    name: 'Andrei T.',
    city: 'LOS ANGELES',
    state: 'CA',
    zipcode: '90025'
  },
  {
    name: 'Bela T.',
    city: 'MIAMI',
    state: 'FL',
    zipcode: '33126' },
  {
    name: 'Omar Al Asaad',
```

```
    city: 'ANCHORAGE',
    state: 'AK',
    zipcode: '99503'
  }
]
```

11.

```
mongoexport --collection=customers --db=test --out=customers.json
```