Homework 4 Report - COMP306

Sarp Çağan Kelleci

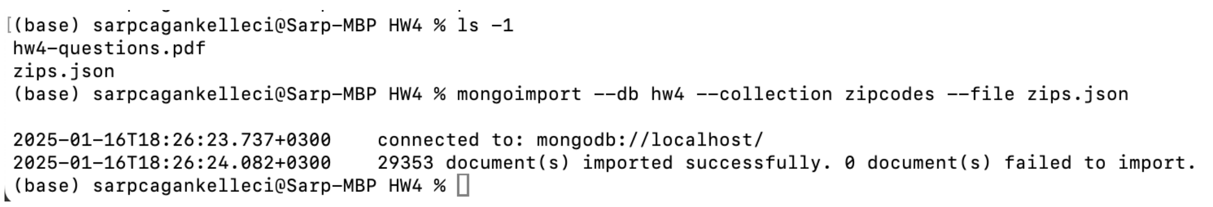
79482

Question 1

I imported the `zips.json` file into the `hw4` database and displayed the collections and the first 10 documents from the `zipcodes` collection. Commands and results are below:

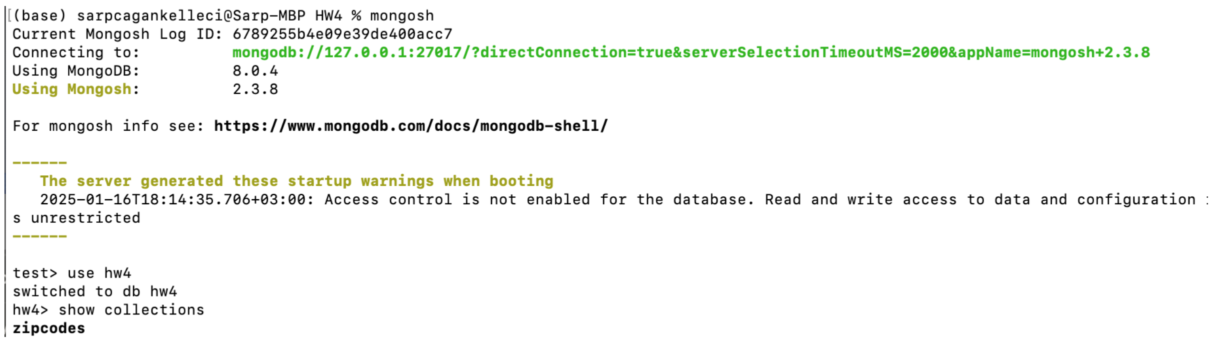
Commands:

```
mongoimport --db hw4 --collection zipcodes --file zips.json
db.zipcodes.find().limit(10).pretty()
```

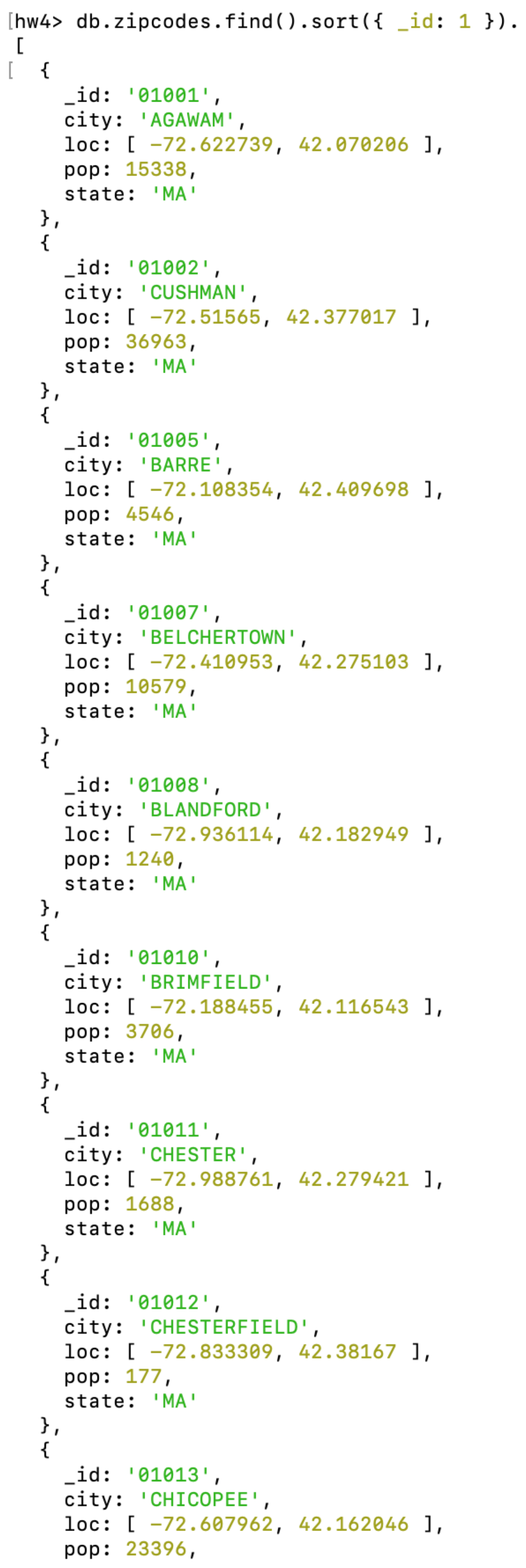


```
(base) sarpcagankelleci@Sarp-MBP HW4 % ls -1
hw4-questions.pdf
zips.json
(base) sarpcagankelleci@Sarp-MBP HW4 % mongoimport --db hw4 --collection zipcodes --file zips.json

2025-01-16T18:26:23.737+0300    connected to: mongodb://localhost/
2025-01-16T18:26:24.082+0300    29353 document(s) imported successfully. 0 document(s) failed to import.
(base) sarpcagankelleci@Sarp-MBP HW4 % 
```

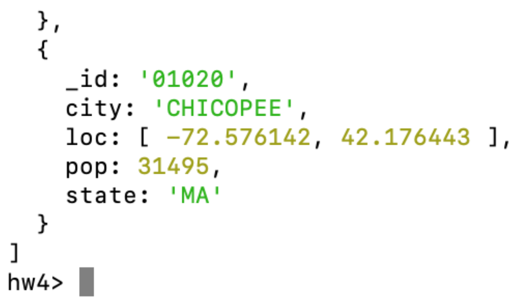
Figure 1: Command to import `zips.json` and list collections.



```
(base) sarpcagankelleci@Sarp-MBP HW4 % mongosh
Current Mongosh Log ID: 6789255b4e09e39de400acc7
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.3.8
Using MongoDB:          8.0.4
Using Mongosh:          2.3.8

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

------
   The server generated these startup warnings when booting
   2025-01-16T18:14:35.706+03:00: Access control is not enabled for the database. Read and write access to data and configuration i
s unrestricted
------

test> use hw4
switched to db hw4
hw4> show collections
zipcodes
```



```
[hw4> db.zipcodes.find().sort({ _id: 1 }).
 [
[  {
     _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'
   },
   {
     _id: '01005',
     city: 'BARRE',
     loc: [ -72.108354, 42.409698 ],
     pop: 4546,
     state: 'MA'
   },
   {
     _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'
   },
   {
     _id: '01010',
     city: 'BRIMFIELD',
     loc: [ -72.188455, 42.116543 ],
     pop: 3706,
     state: 'MA'
   },
   {
     _id: '01011',
     city: 'CHESTER',
     loc: [ -72.988761, 42.279421 ],
     pop: 1688,
     state: 'MA'
   },
   {
     _id: '01012',
     city: 'CHESTERFIELD',
     loc: [ -72.833309, 42.38167 ],
     pop: 177,
     state: 'MA'
   },
   {
     _id: '01013',
     city: 'CHICOPEE',
     loc: [ -72.607962, 42.162046 ],
     pop: 23396,
```



```
  },
  {
    _id: '01020',
    city: 'CHICOPEE',
    loc: [ -72.576142, 42.176443 ],
    pop: 31495,
    state: 'MA'
  }
]
hw4> 
```

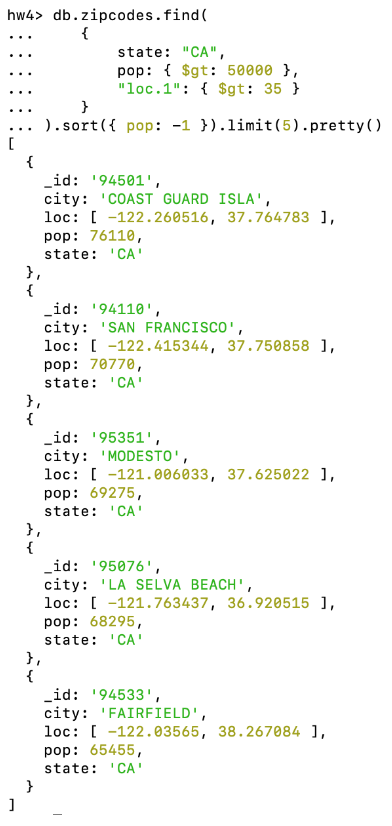
Figure 2: Result: Collections and first 10 documents displayed.

Question 2

I wrote a query to find documents in California with a population over 50,000 and latitude greater than 35. The results were limited to 5 and sorted by population in descending order. Commands and results are below:

Commands:

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



```
hw4> db.zipcodes.find(
...     {
...         state: "CA",
...         pop: { $gt: 50000 },
...         "loc.1": { $gt: 35 }
...     }
... ).sort({ pop: -1 }).limit(5).pretty()
[
  {
    _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'
  }
]
```

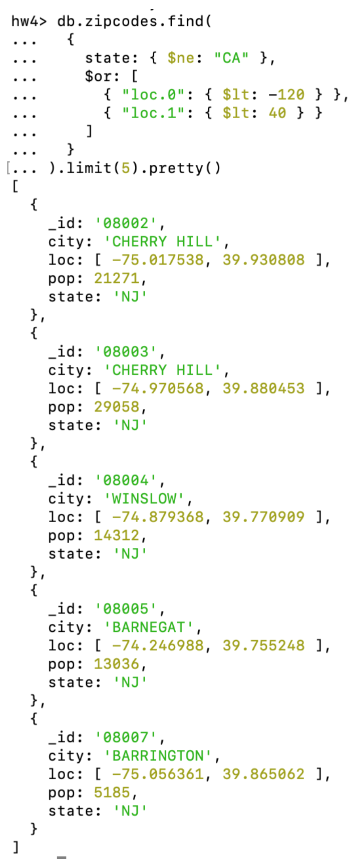
Figure 3: Query and results for Question 2.

Question 3

I wrote a query to find documents where the state is not California, and either longitude is less than -120 or latitude is less than 40. The results were limited to 5. Commands and results are below:

Commands:

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



```
hw4> db.zipcodes.find(
...   {
...     state: { $ne: "CA" },
...     $or: [
...       { "loc.0": { $lt: -120 } },
...       { "loc.1": { $lt: 40 } }
...     ]
...   }
... ).limit(5).pretty()
[
  {
    _id: '08002',
    city: 'CHERRY HILL',
    loc: [ -75.017538, 39.930808 ],
    pop: 21271,
    state: 'NJ'
  },
  {
    _id: '08003',
    city: 'CHERRY HILL',
    loc: [ -74.970568, 39.880453 ],
    pop: 29058,
    state: 'NJ'
  },
  {
    _id: '08004',
    city: 'WINSLOW',
    loc: [ -74.879368, 39.770909 ],
    pop: 14312,
    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'
  }
]
```

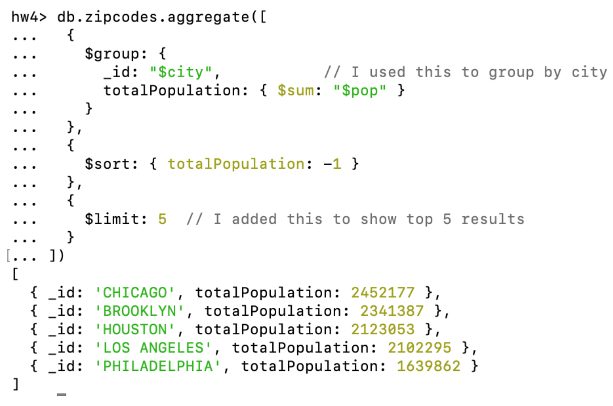
Figure 4: Query and results for Question 3.

Question 4

I wrote an aggregation query to find the top 5 most populated cities with their respective populations, sorted in descending order. Commands and results are below:

Commands:

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



```
hw4> db.zipcodes.aggregate([
...   {
...     $group: {
...       _id: "$city",           // I used this to group by city
...       totalPopulation: { $sum: "$pop" }
...     }
...   },
...   {
...     $sort: { totalPopulation: -1 }
...   },
...   {
...     $limit: 5  // I added this to show top 5 results
...   }
... ])
[
  { _id: 'CHICAGO', totalPopulation: 2452177 },
  { _id: 'BROOKLYN', totalPopulation: 2341387 },
  { _id: 'HOUSTON', totalPopulation: 2123053 },
  { _id: 'LOS ANGELES', totalPopulation: 2102295 },
  { _id: 'PHILADELPHIA', totalPopulation: 1639862 }
]
```

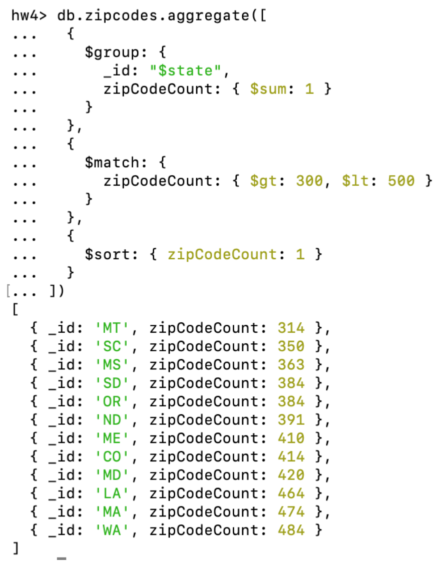
Figure 5: Aggregation query and results for Question 4.

Question 5

I wrote an aggregation query to find states with between 300 and 500 zip codes, displaying the state and count, sorted in ascending order of count. Commands and results are below:

Commands:

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



```
hw4> 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 }
]
```

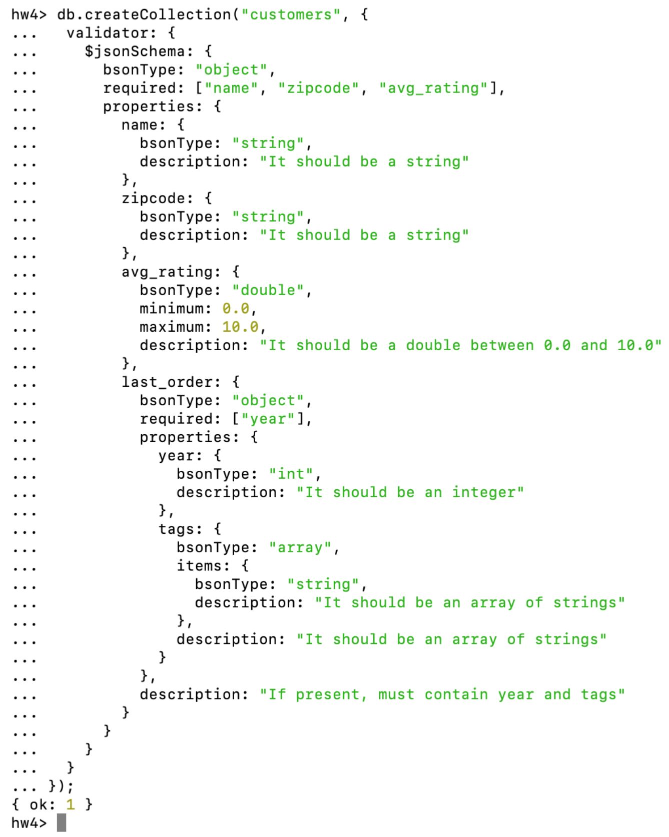
Figure 6: Aggregation query and results for Question 5.

Question 6

I created a `customers` collection with a schema validator. The schema includes fields such as `name`, `zipcode`, `avg_rating`, and an optional `last_order` field. Commands and results are below:

Commands:

```
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" } }
          }
        }
      }
    }
  }
})
```



```
hw4> db.createCollection("customers", {
...   validator: {
...     $jsonSchema: {
...       bsonType: "object",
...       required: ["name", "zipcode", "avg_rating"],
...       properties: {
...         name: {
...           bsonType: "string",
...           description: "It should be a string"
...         },
...         zipcode: {
...           bsonType: "string",
...           description: "It should be a string"
...         },
...         avg_rating: {
...           bsonType: "double",
...           minimum: 0.0,
...           maximum: 10.0,
...           description: "It should be a double between 0.0 and 10.0"
...         },
...         last_order: {
...           bsonType: "object",
...           required: ["year"],
...           properties: {
...             year: {
...               bsonType: "int",
...               description: "It should be an integer"
...             },
...             tags: {
...               bsonType: "array",
...               items: {
...                 bsonType: "string",
...                 description: "It should be an array of strings"
...               },
...               description: "It should be an array of strings"
...             }
...           },
...           description: "If present, must contain year and tags"
...         }
...       }
...     }
...   }
... });
{ ok: 1 }
hw4> 
```

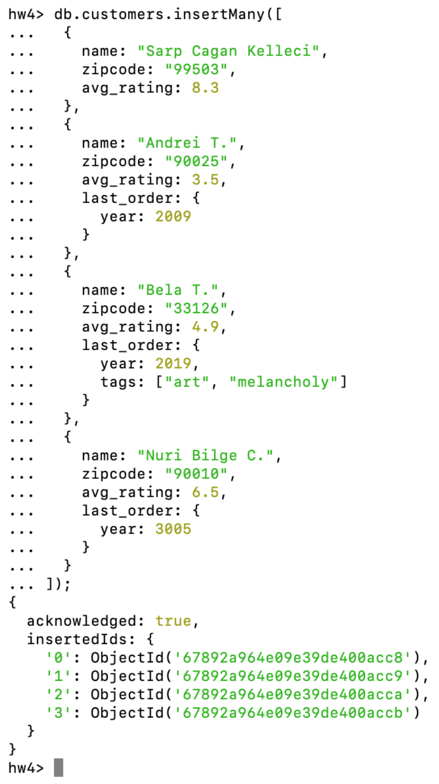
Figure 7: Schema validator creation for the `customers` collection.

Question 7

I inserted 4 documents into the `customers` collection in a single query and displayed all documents in the collection. Commands and results are below:

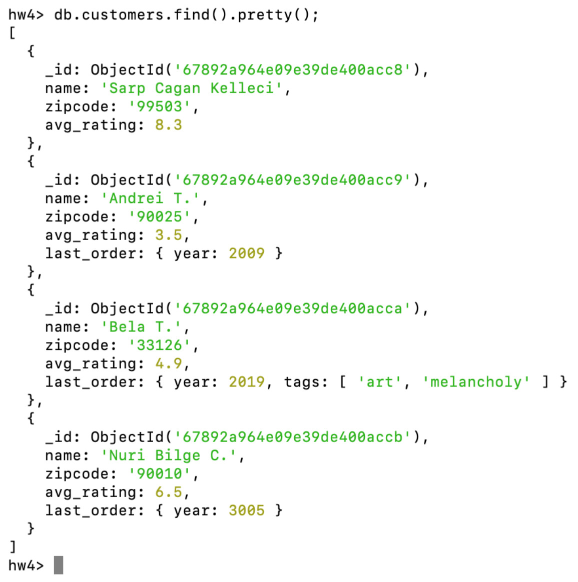
Commands:

```
db.customers.insertMany([
  { name: "Sarp Cagan Kelleci", 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 } }
])
```



```
hw4> db.customers.insertMany([
...   {
...     name: "Sarp Cagan Kelleci",
...     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('67892a964e09e39de400acc8'),
    '1': ObjectId('67892a964e09e39de400acc9'),
    '2': ObjectId('67892a964e09e39de400acca'),
    '3': ObjectId('67892a964e09e39de400accb')
  }
}
hw4> 
```

Figure 8: Insertion of 4 documents into the customers collection.



```
hw4> db.customers.find().pretty();
[
  {
    _id: ObjectId('67892a964e09e39de400acc8'),
    name: 'Sarp Cagan Kelleci',
    zipcode: '99503',
    avg_rating: 8.3
  },
  {
    _id: ObjectId('67892a964e09e39de400acc9'),
    name: 'Andrei T.',
    zipcode: '90025',
    avg_rating: 3.5,
    last_order: { year: 2009 }
  },
  {
    _id: ObjectId('67892a964e09e39de400acca'),
    name: 'Bela T.',
    zipcode: '33126',
    avg_rating: 4.9,
    last_order: { year: 2019, tags: [ 'art', 'melancholy' ] }
  },
  {
    _id: ObjectId('67892a964e09e39de400accb'),
    name: 'Nuri Bilge C.',
    zipcode: '90010',
    avg_rating: 6.5,
    last_order: { year: 3005 }
  }
]
hw4> 
```

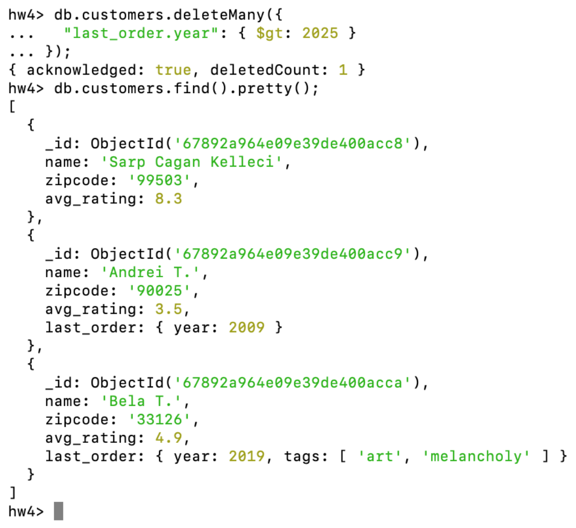
Figure 9: Display of all documents in the customers collection.

Question 8

I deleted documents in the `customers` collection where `last_order.year` is greater than 2025 and displayed the remaining documents. Commands and results are below:

Commands:

```
db.customers.deleteMany({ "last_order.year": { $gt: 2025 } })
db.customers.find().pretty()
```



```
hw4> db.customers.deleteMany({
...   "last_order.year": { $gt: 2025 }
... });
{ acknowledged: true, deletedCount: 1 }
hw4> db.customers.find().pretty();
[
  {
    _id: ObjectId('67892a964e09e39de400acc8'),
    name: 'Sarp Cagan Kelleci',
    zipcode: '99503',
    avg_rating: 8.3
  },
  {
    _id: ObjectId('67892a964e09e39de400acc9'),
    name: 'Andrei T.',
    zipcode: '90025',
    avg_rating: 3.5,
    last_order: { year: 2009 }
  },
  {
    _id: ObjectId('67892a964e09e39de400acca'),
    name: 'Bela T.',
    zipcode: '33126',
    avg_rating: 4.9,
    last_order: { year: 2019, tags: [ 'art', 'melancholy' ] }
  }
]
hw4>
```

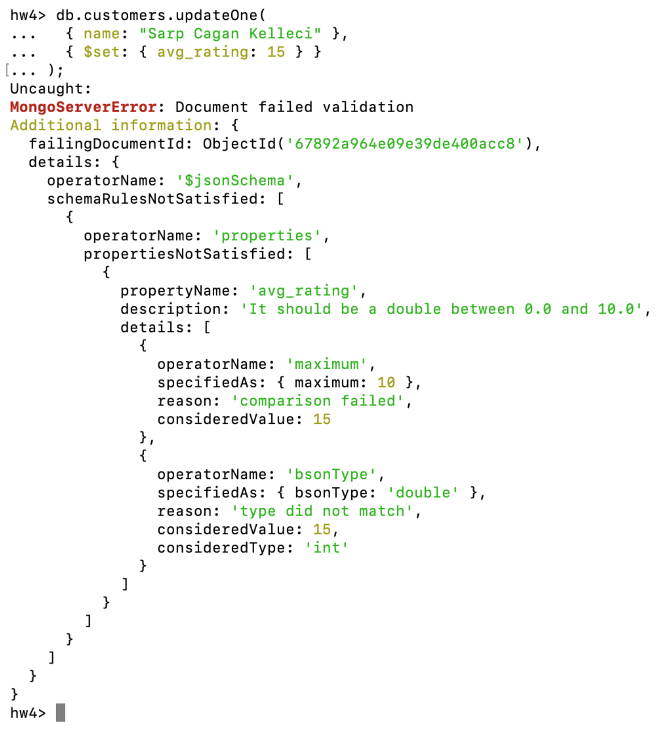
Figure 10: Deletion query and display of remaining documents.

Question 9

I attempted to update the `avg_rating` field of my document to 15. The operation failed because the value exceeds the schema validation rule, which restricts `avg_rating` to be between 0.0 and 10.0. The response was received because the value 15 failed the schema validation rules defined in the customers collection. Commands and results are below:

Commands:

```
db.customers.updateOne(
  { name: "Sarp Cagan Kelleci" },
  { $set: { avg_rating: 15 } }
)
```



```
hw4> db.customers.updateOne(
...   { name: "Sarp Cagan Kelleci" },
...   { $set: { avg_rating: 15 } }
... );
Uncaught:
MongoServerError: Document failed validation
Additional information: {
  failingDocumentId: ObjectId('67892a964e09e39de400acc8'),
  details: {
    operatorName: '$jsonSchema',
    schemaRulesNotSatisfied: [
      {
        operatorName: 'properties',
        propertiesNotSatisfied: [
          {
            propertyName: 'avg_rating',
            description: 'It should be a double between 0.0 and 10.0',
            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'
              }
            ]
          }
        ]
      }
    ]
  }
}
hw4> 
```

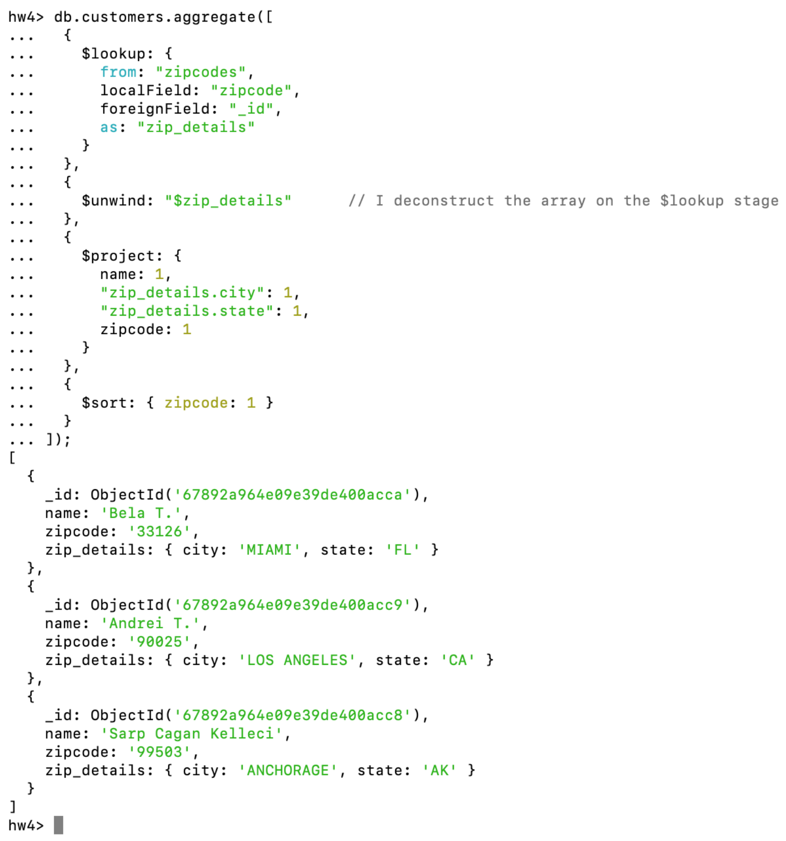
Figure 11: Attempt to update `avg_rating` to 15 and error message.

Question 10

I wrote an aggregation query to display the name, city, and state of each customer, using the connection between the `customers` and `zipcodes` collections. Commands and results are below:

Commands:

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



```
hw4> db.customers.aggregate([
...   {
...     $lookup: {
...       from: "zipcodes",
...       localField: "zipcode",
...       foreignField: "_id",
...       as: "zip_details"
...     }
...   },
...   {
...     $unwind: "$zip_details"      // I deconstruct the array on the $lookup stage
...   },
...   {
...     $project: {
...       name: 1,
...       "zip_details.city": 1,
...       "zip_details.state": 1,
...       zipcode: 1
...     }
...   },
...   {
...     $sort: { zipcode: 1 }
...   }
... ]);
[
  {
    _id: ObjectId('67892a964e09e39de400acca'),
    name: 'Bela T.',
    zipcode: '33126',
    zip_details: { city: 'MIAMI', state: 'FL' }
  },
  {
    _id: ObjectId('67892a964e09e39de400acc9'),
    name: 'Andrei T.',
    zipcode: '90025',
    zip_details: { city: 'LOS ANGELES', state: 'CA' }
  },
  {
    _id: ObjectId('67892a964e09e39de400acc8'),
    name: 'Sarp Cagan Kelleci',
    zipcode: '99503',
    zip_details: { city: 'ANCHORAGE', state: 'AK' }
  }
]
hw4>
```

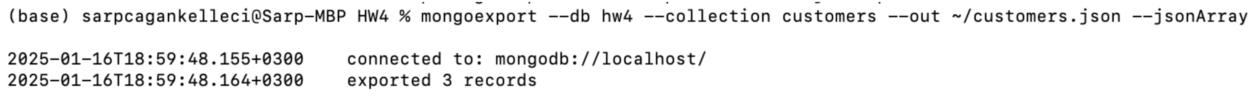
Figure 12: Aggregation query using `customers` and `zipcodes` collections.

Question 11

I exported the `customers` collection to a JSON file named `customers.json`. The command used is shown below.

Commands:

```
mongoexport --db hw4 --collection customers --out ~/customers.json --jsonArray
```



```
(base) sarpcagankelleci@Sarp-MBP HW4 % mongoexport --db hw4 --collection customers --out ~/customers.json --jsonArray

2025-01-16T18:59:48.155+0300    connected to: mongodb://localhost/
2025-01-16T18:59:48.164+0300    exported 3 records
```

Figure 13: Export command and confirmation message.