EXERCISE-18

MONGO DB

Name: Vedhasree S

Register Number: 240701580

Department: CSE

Structure of ‘restaurants’ collection

```
>_MONGOSH
> use mongodbexercise
< switched to db mongodbexercise
>
> db.restaurants.insertOne({
    "address": {
      "building": "1007",
      "coord": [ -73.856077, 40.848447 ],
      "street": "Morris Park Ave",
      "zipcode": "10462"
    },
    "borough": "Bronx",
    "cuisine": "Bakery",
    "grades": [
      { "date": ISODate("2014-03-03T00:00:00Z"), "grade": "A", "score": 2 },
      { "date": ISODate("2013-09-11T00:00:00Z"), "grade": "A", "score": 6 },
      { "date": ISODate("2013-01-24T00:00:00Z"), "grade": "A", "score": 10 },
      { "date": ISODate("2011-11-23T00:00:00Z"), "grade": "A", "score": 9 },
      { "date": ISODate("2011-03-10T00:00:00Z"), "grade": "B", "score": 14 }
    ],
    "name": "Morris Park Bake Shop",
    "restaurant_id": "30075445"
  })
< {
    acknowledged: true,
    insertedId: ObjectId('6900d510c26d2484028e68e7')
  }
```

1. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which prepared dish except 'American' and 'Chinese' or restaurant's name begins with letter 'Wil'.

```
> db.restaurants.find(
    {
      $or: [
        { cuisine: { $nin: ["American", "Chinese"] } },
        { name: { $regex: /^Wil/i } }
      ]
    },
    { restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0 }
  )
< {
    borough: 'Bronx',
    cuisine: 'Bakery',
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
```

2. Write a MongoDB query to find the restaurant Id, name, and grades for those restaurants which achieved a grade of "A" and scored 11 on an ISODate "2014-08-11T00:00:00Z" among many of survey dates.

```
> db.restaurants.find(
    {
      grades: {
        $elemMatch: {
          grade: "A",
          score: 11,
          date: ISODate("2014-08-11T00:00:00Z")
        }
      }
    },
    { restaurant_id: 1, name: 1, grades: 1, _id: 0 }
  )
<
```

```
> db.restaurants.insertOne({
    "name": "Wilshire Diner",
    "borough": "Brooklyn",
    "cuisine": "American",
    "grades": [
      { "date": ISODate("2014-08-11T00:00:00Z"), "grade": "A", "score": 11 },
      { "date": ISODate("2013-05-10T00:00:00Z"), "grade": "B", "score": 8 }
    ],
    "restaurant_id": "40012345"
  })
< {
    acknowledged: true,
    insertedId: ObjectId('6900d8e0c26d2484028e68e8')
  }
```

```
>_MONGOSH
> db.restaurants.find(
    {
      grades: {
        $elemMatch: {
          grade: "A",
          score: 11,
          date: ISODate("2014-08-11T00:00:00Z")
        }
      }
    },
    { restaurant_id: 1, name: 1, grades: 1, _id: 0 }
  ).pretty()
< {
    name: 'Wilshire Diner',
    grades: [
      {
        date: 2014-08-11T00:00:00.000Z,
        grade: 'A',
        score: 11
      },
      {
        date: 2013-05-10T00:00:00.000Z,
        grade: 'B',
        score: 8
      }
    ],
    restaurant_id: '40012345'
  }
```

3. Write a MongoDB query to find the restaurant Id, name and grades for those restaurants where the 2nd element of grades array contains a grade of "A" and score 9 on an ISODate "2014-08-11T00:00:00Z".

```
> db.restaurants.find(
    {
      "grades.1.grade": "A",
      "grades.1.score": 9,
      "grades.1.date": ISODate("2014-08-11T00:00:00Z")
    },
    { restaurant_id: 1, name: 1, grades: 1, _id: 0 }
  )
<
```

```
> db.restaurants.insertOne({
    restaurant_id: "50099999",
    name: "Test Bake House",
    borough: "Manhattan",
    cuisine: "Bakery",
    grades: [
      { "date": ISODate("2014-01-01T00:00:00Z"), "grade": "B", "score": 8 },
      { "date": ISODate("2014-08-11T00:00:00Z"), "grade": "A", "score": 9 }
    ]
  })
< {
    acknowledged: true,
    insertedId: ObjectId('6900da42c26d2484028e68e9')
  }
```

```
> db.restaurants.find(
    {
      "grades.1.grade": "A",
      "grades.1.score": 9,
      "grades.1.date": ISODate("2014-08-11T00:00:00Z")
    },
    { restaurant_id: 1, name: 1, grades: 1, _id: 0 }
  )
< {
    restaurant_id: '50099999',
    name: 'Test Bake House',
    grades: [
      {
        date: 2014-01-01T00:00:00.000Z,
        grade: 'B',
        score: 8
      },
      {
        date: 2014-08-11T00:00:00.000Z,
        grade: 'A',
        score: 9
      }
    ]
  }
```

4. Write a MongoDB query to find the restaurant Id, name, address and geographical location for those restaurants where 2nd element of coord array contains a value which is more than 42 and upto 52.

```
> db.restaurants.find(
    {
      "address.coord.1": { $gt: 42, $lte: 52 }
    },
    { restaurant_id: 1, name: 1, "address": 1, _id: 0 }
  )
<
```

```
> db.restaurants.insertOne({
    "restaurant_id": "60012345",
    "name": "Mountain View Café",
    "borough": "Brooklyn",
    "cuisine": "Continental",
    "address": {
      "building": "202",
      "coord": [ -70.123456, 47.987654 ],
      "street": "Park Street",
      "zipcode": "10001"
    }
  })
< {
    acknowledged: true,
    insertedId: ObjectId('6900dbabc26d2484028e68ea')
  }
```

```
> db.restaurants.find(
    {
      "address.coord.1": { $gt: 42, $lte: 52 }
    },
    { restaurant_id: 1, name: 1, address: 1, _id: 0 }
  ).pretty()
< {
    restaurant_id: '60012345',
    name: 'Mountain View Café',
    address: {
      building: '202',
      coord: [
        -70.123456,
        47.987654
      ],
      street: 'Park Street',
      zipcode: '10001'
    }
  }
```

5. Write a MongoDB query to arrange the name of the restaurants in ascending order along with all the columns.

```
> db.restaurants.find({}, { restaurant_id: 1, name: 1, _id: 0 }).sort({ name: 1 })
< {
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
  {
    restaurant_id: '60012345',
    name: 'Mountain View Café'
  }
  {
    restaurant_id: '50099999',
    name: 'Test Bake House'
  }
  {
    name: 'Wilshire Diner',
    restaurant_id: '40012345'
  }
```

6. Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns.

```
> db.restaurants.find({}, { restaurant_id: 1, name: 1, _id: 0 }).sort({ name: -1 })
< {
    name: 'Wilshire Diner',
    restaurant_id: '40012345'
  }
  {
    restaurant_id: '50099999',
    name: 'Test Bake House'
  }
  {
    restaurant_id: '60012345',
    name: 'Mountain View Café'
  }
  {
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
```

7. Write a MongoDB query to arranged the name of the cuisine in ascending order and for that same cuisine borough should be in descending order.

```
> db.restaurants.find({}, { restaurant_id: 1, name: 1, cuisine: 1, borough: 1, _id: 0 }).sort({ cuisine: 1, borough: -1 })
< {
    name: 'Wilshire Diner',
    borough: 'Brooklyn',
    cuisine: 'American',
    restaurant_id: '40012345'
  }
  {
    restaurant_id: '50099999',
    name: 'Test Bake House',
    borough: 'Manhattan',
    cuisine: 'Bakery'
  }
  {
    borough: 'Bronx',
    cuisine: 'Bakery',
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
  {
    restaurant_id: '60012345',
    name: 'Mountain View Café',
    borough: 'Brooklyn',
    cuisine: 'Continental'
  }
```

8. Write a MongoDB query to know whether all the addresses contains the street or not.

```
> db.restaurants.find({ "address.street": { $exists: true } }, { restaurant_id: 1, name: 1, "address.street": 1, _id: 0 })
< {
    address: {
      street: 'Morris Park Ave'
    },
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
  {
    restaurant_id: '60012345',
    name: 'Mountain View Café',
    address: {
      street: 'Park Street'
    }
  }
```

9. Write a MongoDB query which will select all documents in the restaurants collection where the coord field value is Double.

```
> db.restaurants.find({ "address.coord": { $type: "double" } }, { restaurant_id: 1, name: 1, "address.coord": 1, _id: 0 })
< {
    address: {
      coord: [
        -73.856077,
        40.848447
      ]
    },
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
  {
    restaurant_id: '60012345',
    name: 'Mountain View Café',
    address: {
      coord: [
        -70.123456,
        47.987654
      ]
    }
  }
```

10. Write a MongoDB query which will select the restaurant Id, name and grades for those restaurants which returns 0 as a remainder after dividing the score by 7.

```
>_MONGOSH
> db.restaurants.find(
    { "grades.score": { $mod: [7, 0] } },
    { restaurant_id: 1, name: 1, grades: 1, _id: 0 }
  )
< {
    grades: [
      {
        date: 2014-03-03T00:00:00.000Z,
        grade: 'A',
        score: 2
      },
      {
        date: 2013-09-11T00:00:00.000Z,
        grade: 'A',
        score: 6
      },
      {
        date: 2013-01-24T00:00:00.000Z,
        grade: 'A',
        score: 10
      },
      {
        date: 2011-11-23T00:00:00.000Z,
        grade: 'A',
        score: 9
      },
      {
        date: 2011-03-10T00:00:00.000Z,
        grade: 'B',
        score: 14
      }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
```

11. Write a MongoDB query to find the restaurant name, borough, longitude and attitude and cuisine for those restaurants which contains 'mon' as three letters somewhere in its name.

```
>_MONGOSH
> db.restaurants.find(
    { name: { $regex: "mon", $options: "i" } },
    { name: 1, borough: 1, "address.coord": 1, cuisine: 1, _id: 0 }
  )
<
```

```
> db.restaurants.insertOne({
    name: "Monarch Cafe",
    borough: "Manhattan",
    cuisine: "Continental",
    address: { coord: [ -73.95, 40.77 ] }
  })
< {
    acknowledged: true,
    insertedId: ObjectId('6900e644c26d2484028e68eb')
  }
```

```
> db.restaurants.find(
    { name: { $regex: "mon", $options: "i" } },
    { name: 1, borough: 1, "address.coord": 1, cuisine: 1, _id: 0 }
  )
< {
    name: 'Monarch Cafe',
    borough: 'Manhattan',
    cuisine: 'Continental',
    address: {
      coord: [
        -73.95,
        40.77
      ]
    }
  }
```

12. Write a MongoDB query to find the restaurant name, borough, longitude and latitude and cuisine for those restaurants which contain 'Mad' as first three letters of its name.

```
> db.restaurants.find(
    { name: { $regex: "^Mad", $options: "i" } },
    { name: 1, borough: 1, "address.coord": 1, cuisine: 1, _id: 0 }
  )
<
```

>_MONGOSH

```
> db.restaurants.insertMany([
    {
      restaurant_id: "40000123",
      name: "Madras Cafe",
      borough: "Manhattan",
      cuisine: "South Indian",
      address: {
        building: "22",
        street: "Lexington Ave",
        zipcode: "10010",
        coord: [ -73.982, 40.743 ]
      },
      grades: [
        { date: ISODate("2014-08-11T00:00:00Z"), grade: "A", score: 12 }
      ]
    },
    {
      restaurant_id: "40000124",
      name: "Madison Diner",
      borough: "Brooklyn",
      cuisine: "American",
      address: {
        building: "45",
        street: "Madison St",
        zipcode: "11201",
        coord: [ -73.987, 40.692 ]
      },
      grades: [
        { date: ISODate("2015-03-10T00:00:00Z"), grade: "B", score: 8 }
      ]
    },
    {
      restaurant_id: "40000125",
      name: "Madhura Sweets",
      borough: "Queens",
      cuisine: "Bakery",
      address: {
        building: "12",
        street: "Main Road",
        zipcode: "11373",
        coord: [ -73.876, 40.739 ]
      },
      grades: [
        { date: ISODate("2016-05-20T00:00:00Z"), grade: "A", score: 10 }
      ]
    }
  ])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('6900e820c26d2484028e68ec'),
      '1': ObjectId('6900e820c26d2484028e68ed'),
      '2': ObjectId('6900e820c26d2484028e68ee')
    }
  }
```

```
>_MONGOSH
> db.restaurants.find(
    { name: { $regex: "^Mad", $options: "i" } },
    { name: 1, borough: 1, "address.coord": 1, cuisine: 1, _id: 0 }
  )
< {
    name: 'Madras Cafe',
    borough: 'Manhattan',
    cuisine: 'South Indian',
    address: {
      coord: [
        -73.982,
        40.743
      ]
    }
  }
  {
    name: 'Madison Diner',
    borough: 'Brooklyn',
    cuisine: 'American',
    address: {
      coord: [
        -73.987,
        40.692
      ]
    }
  }
  {
    name: 'Madhura Sweets',
    borough: 'Queens',
    cuisine: 'Bakery',
    address: {
      coord: [
        -73.876,
        40.739
      ]
    }
  }
```

13. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5.

```
> db.restaurants.find(
    { "grades.score": { $lt: 5 } },
    { name: 1, _id: 0 }
  )
< {
    name: 'Morris Park Bake Shop'
  }
```

14. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan.

```
> db.restaurants.find(
    { borough: "Manhattan", "grades.score": { $lt: 5 } },
    { name: 1, _id: 0 }
  )
<
```

```
> db.restaurants.insertOne({
    restaurant_id: "99999",
    name: "Test Diner",
    borough: "Manhattan",
    cuisine: "American",
    grades: [
      { date: ISODate("2014-07-01T00:00:00Z"), grade: "B", score: 3 },
      { date: ISODate("2014-08-01T00:00:00Z"), grade: "A", score: 10 }
    ]
  })
< {
    acknowledged: true,
    insertedId: ObjectId('6900ea10c26d2484028e68ef')
  }
```

```
> db.restaurants.find(
    { borough: "Manhattan", "grades.score": { $lt: 5 } },
    { name: 1, _id: 0 }
  )
< {
    name: 'Test Diner'
  }
```

15. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn.

```
> db.restaurants.find(
    {
      borough: { $in: ["Manhattan", "Brooklyn"] },
      "grades.score": { $lt: 5 }
    },
    { name: 1, borough: 1, _id: 0 }
  )
< {
    name: 'Test Diner',
    borough: 'Manhattan'
  }
```

16. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn, and their cuisine is not American.

```
>_MONGOSH
> db.restaurants.find(
    {
      borough: { $in: ["Manhattan", "Brooklyn"] },
      "grades.score": { $lt: 5 },
      cuisine: { $ne: "American" }
    },
    { name: 1, borough: 1, cuisine: 1, _id: 0 }
  )
<
```

>_MONGOSH

```
> db.restaurants.insertMany([
    {
      restaurant_id: "R001",
      name: "Mad Spice Diner",
      borough: "Manhattan",
      cuisine: "Indian",
      grades: [
        { date: ISODate("2014-08-11T00:00:00Z"), grade: "B", score: 3 },
        { date: ISODate("2015-06-10T00:00:00Z"), grade: "A", score: 12 }
      ]
    },
    {
      restaurant_id: "R002",
      name: "Brooklyn Bites",
      borough: "Brooklyn",
      cuisine: "Italian",
      grades: [
        { date: ISODate("2014-08-11T00:00:00Z"), grade: "A", score: 4 },
        { date: ISODate("2015-07-15T00:00:00Z"), grade: "B", score: 10 }
      ]
    },
    {
      restaurant_id: "R003",
      name: "American Feast",
      borough: "Brooklyn",
      cuisine: "American",
```

>_MONGOSH

```
      grades: [
        { date: ISODate("2014-08-11T00:00:00Z"), grade: "C", score: 2 }
      ]
    }
  ])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('6900f25cc26d2484028e68f0'),
      '1': ObjectId('6900f25cc26d2484028e68f1'),
      '2': ObjectId('6900f25cc26d2484028e68f2')
    }
  }
```

```
> db.restaurants.find(
    {
      borough: { $in: ["Manhattan", "Brooklyn"] },
      "grades.score": { $lt: 5 },
      cuisine: { $ne: "American" }
    },
    { name: 1, borough: 1, cuisine: 1, _id: 0 }
  )
< {
    name: 'Mad Spice Diner',
    borough: 'Manhattan',
    cuisine: 'Indian'
  }
  {
    name: 'Brooklyn Bites',
    borough: 'Brooklyn',
    cuisine: 'Italian'
  }
```

17. Write a MongoDB query to find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn, and their cuisine is not American or Chinese.

```
> db.restaurants.find(
    {
      borough: { $in: ["Manhattan", "Brooklyn"] },
      "grades.score": { $lt: 5 },
      cuisine: { $nin: ["American", "Chinese"] }
    },
    { name: 1, borough: 1, cuisine: 1, _id: 0 }
  )
< {
    name: 'Mad Spice Diner',
    borough: 'Manhattan',
    cuisine: 'Indian'
  }
  {
    name: 'Brooklyn Bites',
    borough: 'Brooklyn',
    cuisine: 'Italian'
  }
```

18. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6.

```
> db.restaurants.find(
    {
      "grades.score": { $all: [2, 6] }
    },
    { name: 1, "grades.score": 1, _id: 0 }
  )
< {
    grades: [
      {
        score: 2
      },
      {
        score: 6
      },
      {
        score: 10
      },
      {
        score: 9
      },
      {
        score: 14
      }
    ],
    name: 'Morris Park Bake Shop'
  }
```

19. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan.

```
> db.restaurants.find(
    {
      borough: "Manhattan",
      "grades.score": { $all: [2, 6] }
    },
    { name: 1, borough: 1, "grades.score": 1, _id: 0 }
  )
<
```

```
>_MONGOSH
> db.restaurants.insertMany([
    {
      restaurant_id: "R010",
      name: "Manhattan Spice House",
      borough: "Manhattan",
      cuisine: "Indian",
      grades: [
        { date: ISODate("2014-08-11T00:00:00Z"), grade: "B", score: 2 },
        { date: ISODate("2015-05-21T00:00:00Z"), grade: "A", score: 6 },
        { date: ISODate("2016-01-12T00:00:00Z"), grade: "A", score: 10 }
      ]
    },
    {
      restaurant_id: "R011",
      name: "Brooklyn Bistro",
      borough: "Brooklyn",
      cuisine: "Italian",
      grades: [
        { date: ISODate("2014-08-11T00:00:00Z"), grade: "C", score: 2 },
        { date: ISODate("2015-03-14T00:00:00Z"), grade: "B", score: 8 }
      ]
    }
  ])
```

```
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('690165e4159b8873d4502b26'),
      '1': ObjectId('690165e4159b8873d4502b27')
    }
  }
```

```
> db.restaurants.find(
    {
      borough: "Manhattan",
      "grades.score": { $all: [2, 6] }
    },
    { name: 1, borough: 1, "grades.score": 1, _id: 0 }
  )
< {
    name: 'Manhattan Spice House',
    borough: 'Manhattan',
    grades: [
      {
        score: 2
      },
      {
        score: 6
      },
      {
        score: 10
      }
    ]
  }
```

20. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn.

```
> db.restaurants.find(
    {
      borough: { $in: ["Manhattan", "Brooklyn"] },
      "grades.score": { $all: [2, 6] }
    },
    { name: 1, borough: 1, "grades.score": 1, _id: 0 }
  )
< {
    name: 'Manhattan Spice House',
    borough: 'Manhattan',
    grades: [
      {
        score: 2
      },
      {
        score: 6
      },
      {
        score: 10
      }
    ]
  }
```

21. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn, and their cuisine is not American.

```
> db.restaurants.find(
    {
      borough: { $in: ["Manhattan", "Brooklyn"] },
      "grades.score": { $all: [2, 6] },
      cuisine: { $ne: "American" }
    },
    { name: 1, borough: 1, cuisine: 1, "grades.score": 1, _id: 0 }
  )
< {
    name: 'Manhattan Spice House',
    borough: 'Manhattan',
    cuisine: 'Indian',
    grades: [
      {
        score: 2
      },
      {
        score: 6
      },
      {
        score: 10
      }
    ]
  }
```

22. Write a MongoDB query to find the restaurants that have a grade with a score of 2 and a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn, and their cuisine is not American or Chinese.

```
> db.restaurants.find(
    {
      borough: { $in: ["Manhattan", "Brooklyn"] },
      "grades.score": { $all: [2, 6] },
      cuisine: { $nin: ["American", "Chinese"] }
    },
    { name: 1, borough: 1, cuisine: 1, "grades.score": 1, _id: 0 }
  )
< {
    name: 'Manhattan Spice House',
    borough: 'Manhattan',
    cuisine: 'Indian',
    grades: [
      {
        score: 2
      },
      {
        score: 6
      },
      {
        score: 10
      }
    ]
  }
```

23. Write a MongoDB query to find the restaurants that have a grade with a score of 2 or a grade with a score of 6.

```
>_MONGOSH
> db.restaurants.find(
    { "grades.score": { $in: [2, 6] } },
    { name: 1, "grades.score": 1, _id: 0 }
  )
< {
    grades: [
      {
        score: 2
      },
      {
        score: 6
      },
      {
        score: 10
      },
      {
        score: 9
      },
      {
        score: 14
      }
    ],
    name: 'Morris Park Bake Shop'
  }
```

```
{
  name: 'American Feast',
  grades: [
    {
      score: 2
    }
  ]
}
{
  name: 'Manhattan Spice House',
  grades: [
    {
      score: 2
    },
    {
      score: 6
    },
    {
      score: 10
    }
  ]
}
```

```
{
  name: 'Brooklyn Bistro',
  grades: [
    {
      score: 2
    },
    {
      score: 8
    }
  ]
}
```

Sample document of 'movies' collection

```
>_MONGOSH
> use mongodbexercise
< switched to db mongodbexercise
> db.movies.insertOne({
    _id: ObjectId("573a1390f29313caabcd42e8"),
    plot: "A group of bandits stage a brazen train hold-up, only to find a determined posse hot on their heels.",
    genres: [ "Short", "Western" ],
    runtime: 11,
    cast: [
      "A.C. Abadie",
      "Gilbert M. 'Broncho Billy' Anderson",
      "George Barnes",
      "Justus D. Barnes"
    ],
    poster: "https://m.media-amazon.com/images/M/MV5BMTU3NjE5NzYtYTYyNS00MDVmLWIwYjgtMmYwYWIxZDYyNzU2XkEyXkFqcGdeQXVyNzQzNzQxNzI@._V1_SY1000_SX677_AL_.jpg",
    title: "The Great Train Robbery",
    fullplot: "Among the earliest existing films in American cinema - notable as the first film that presented a narrative story to tell - it depicts a group of
    languages: [ "English" ],
    released: ISODate("1903-12-01T00:00:00.000Z"),
    directors: [ "Edwin S. Porter" ],
    rated: "TV-G",
    awards: { wins: 1, nominations: 0, text: "1 win." },
    lastupdated: "2015-08-13 00:27:59.177000000",
    year: 1903,
    imdb: { rating: 7.4, votes: 9847, id: 439 },
    countries: [ "USA" ],
    type: "movie",
    tomatoes: {
      viewer: { rating: 3.7, numReviews: 2559, meter: 75 },
      fresh: 6,
      critic: { rating: 7.6, numReviews: 6, meter: 100 },
      rotten: 0,
      lastUpdated: ISODate("2015-08-08T19:16:10.000Z")
    }
  })
< {
    acknowledged: true,
    insertedId: ObjectId('573a1390f29313caabcd42e8')
  }
```

1. Find all movies with full information from the 'movies' collection that released in the year 1893.

```
>_MONGOSH
> db.movies.find({ year: 1893 })
< {
    _id: ObjectId('690195cfe68fff49a00d3cda'),
    title: 'The Blacksmith Scene',
    year: 1893,
    runtime: 1,
    genres: [
      'Short'
    ],
    languages: [
      'English'
    ],
    countries: [
      'USA'
    ],
    directors: [
      'William K.L. Dickson'
    ],
    rated: 'UNRATED',
    awards: {
      wins: 0,
      nominations: 2,
      text: '2 nominations.'
    },
    imdb: {
      rating: 5.9,
      votes: 1250,
      id: 440
    },
    tomatoes: {
      viewer: {
        rating: 4.5,
        numReviews: 300
      }
    },
    released: 1893-05-09T00:00:00.000Z,
    cast: [
      'Charles Kayser',
      'John Ott'
    ]
  }
```

2. Find all movies with full information from the 'movies' collection that have a runtime greater than 120 minutes.

```
>_MONGOSH
> db.movies.find({ runtime: { $gt: 120 } })
< {
    _id: ObjectId('690195cfe68fff49a00d3cdb'),
    title: 'Epic Adventure Scene',
    year: 2015,
    runtime: 150,
    genres: [
      'Action',
      'Adventure'
    ],
    languages: [
      'English'
    ],
    countries: [
      'USA'
    ],
    directors: [
      'James Nolan'
    ],
    rated: 'PG-13',
    awards: {
      wins: 3,
      nominations: 5,
      text: '3 wins & 5 nominations.'
    },
    imdb: {
      rating: 8.2,
      votes: 10500,
      id: 441
    },
    tomatoes: {
      viewer: {
        rating: 4.8,
        numReviews: 5000
      }
    },
    released: 2015-07-20T00:00:00.000Z,
    cast: [
      'Tom Hardy',
      'Emily Clark'
    ]
  }
```

3. Find all movies with full information from the 'movies' collection that have "Short" genre.

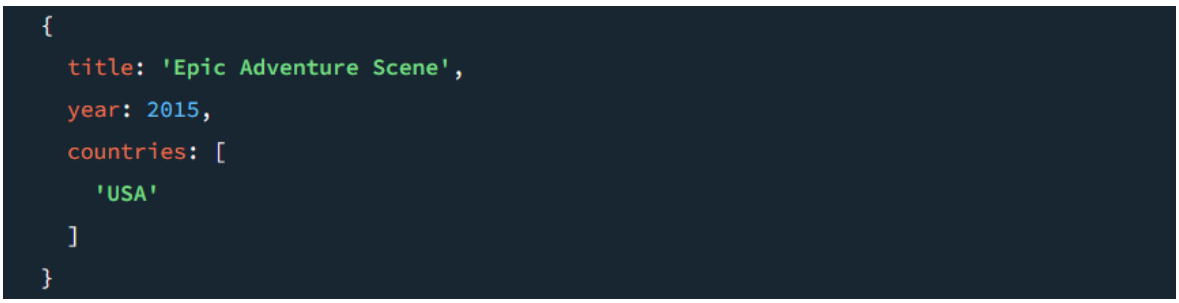
```
> db.movies.find(
    { genres: "Short" },
    { title: 1, year: 1, genres: 1, _id: 0 }
  ).pretty()
< {
    genres: [
      'Short',
      'Western'
    ],
    title: 'The Great Train Robbery',
    year: 1903
  }
  {
    title: 'The Great Train Robbery',
    year: 1903,
    genres: [
      'Short',
      'Western'
    ]
  }
  {
    title: 'The Blacksmith Scene',
    year: 1893,
    genres: [
      'Short'
    ]
  }
```

4. Retrieve all movies from the 'movies' collection that were directed by "William K.L. Dickson" and include complete information for each movie.

```
> db.movies.find(
    { directors: "William K.L. Dickson" },
    { title: 1, year: 1, directors: 1, countries: 1, _id: 0 }
  ).pretty()
< {
    title: 'The Blacksmith Scene',
    year: 1893,
    countries: [
      'USA'
    ],
    directors: [
      'William K.L. Dickson'
    ]
  }
```

5. Retrieve all movies from the 'movies' collection that were released in the USA and include complete information for each movie.

```
>_MONGOSH
> db.movies.find(
    { countries: "USA" },
    { title: 1, year: 1, countries: 1, _id: 0 }
  ).pretty()
< {
    title: 'The Great Train Robbery',
    year: 1903,
    countries: [
      'USA'
    ]
  }
  {
    title: 'The Great Train Robbery',
    year: 1903,
    countries: [
      'USA'
    ]
  }
  {
    title: 'The Blacksmith Scene',
    year: 1893,
    countries: [
      'USA'
    ]
  }
```



```
  {
    title: 'Epic Adventure Scene',
    year: 2015,
    countries: [
      'USA'
    ]
  }
```

6. Retrieve all movies from the 'movies' collection that have complete information and are rated as "UNRATED".

```
>_MONGOSH
> db.movies.find({ rated: "UNRATED" })
< {
    _id: ObjectId('690195cfe68fff49a00d3cda'),
    title: 'The Blacksmith Scene',
    year: 1893,
    runtime: 1,
    genres: [
      'Short'
    ],
    languages: [
      'English'
    ],
    countries: [
      'USA'
    ],
    directors: [
      'William K.L. Dickson'
    ],
    rated: 'UNRATED',
    awards: {
      wins: 0,
      nominations: 2,
      text: '2 nominations.'
    },
```

```
    imdb: {
      rating: 5.9,
      votes: 1250,
      id: 440
    },
    tomatoes: {
      viewer: {
        rating: 4.5,
        numReviews: 300
      }
    },
    released: 1893-05-09T00:00:00.000Z,
    cast: [
      'Charles Kayser',
      'John Ott'
    ]
  }
```

7. Retrieve all movies from the 'movies' collection that have complete information and have received more than 1000 votes on IMDb.

```
> db.movies.find(
    { "imdb.votes": { $gt: 1000 } },
    { title: 1, year: 1, "imdb.votes": 1, _id: 0 }
  ).pretty()
< {
    title: 'The Great Train Robbery',
    year: 1903,
    imdb: {
      votes: 9847
    }
  }
  {
    title: 'The Great Train Robbery',
    year: 1903,
    imdb: {
      votes: 9847
    }
  }
  {
    title: 'The Blacksmith Scene',
    year: 1893,
    imdb: {
      votes: 1250
    }
  }
```

```
{
  title: 'Epic Adventure Scene',
  year: 2015,
  imdb: {
    votes: 10500
  }
}
```

8. Retrieve all movies from the 'movies' collection that have complete information and have an IMDb rating higher than 7.

```
> db.movies.find(
    { "imdb.rating": { $gt: 7 } },
    { title: 1, year: 1, "imdb.rating": 1, _id: 0 }
  ).pretty()
< {
    title: 'The Great Train Robbery',
    year: 1903,
    imdb: {
      rating: 7.4
    }
  }
  {
    title: 'The Great Train Robbery',
    year: 1903,
    imdb: {
      rating: 7.4
    }
  }
  {
    title: 'Epic Adventure Scene',
    year: 2015,
    imdb: {
      rating: 8.2
    }
  }
```

9. Retrieve all movies from the 'movies' collection that have complete information and have a viewer rating higher than 4 on Tomatoes.

```
> db.movies.find(
    { "tomatoes.viewer.rating": { $gt: 4 } },
    { title: 1, year: 1, "tomatoes.viewer.rating": 1, _id: 0 }
  ).pretty()
< {
    title: 'The Blacksmith Scene',
    year: 1893,
    tomatoes: {
      viewer: {
        rating: 4.5
      }
    }
  }
  {
    title: 'Epic Adventure Scene',
    year: 2015,
    tomatoes: {
      viewer: {
        rating: 4.8
      }
    }
  }
```

10. Retrieve all movies from the 'movies' collection that have received an award.

```
>_MONGOSH
> db.movies.find(
    { "awards.wins": { $gt: 0 } },
    { title: 1, year: 1, "awards.wins": 1, "awards.text": 1, _id: 0 }
  ).pretty()
< {
    title: 'The Great Train Robbery',
    awards: {
      wins: 1,
      text: '1 win.'
    },
    year: 1903
  }
  {
    title: 'The Great Train Robbery',
    year: 1903,
    awards: {
      wins: 1,
      text: '1 win.'
    }
  }
  {
    title: 'Epic Adventure Scene',
    year: 2015,
    awards: {
      wins: 3,
      text: '3 wins & 5 nominations.'
    }
  }
```

11. Find all movies with title, languages, released, directors, writers, awards, year, genres, runtime, cast, countries from the 'movies' collection in MongoDB that have at least one nomination.

```
>_MONGOSH
> db.movies.find(
    { "awards.nominations": { $gt: 0 } },
    {
      title: 1,
      languages: 1,
      released: 1,
      directors: 1,
      writers: 1,
      awards: 1,
      year: 1,
      genres: 1,
      runtime: 1,
      cast: 1,
      countries: 1,
      _id: 0
    }
  ).pretty()
< {
    title: 'The Blacksmith Scene',
    year: 1893,
    runtime: 1,
    genres: [
      'Short'
    ],
    languages: [
      'English'
    ],
```

```
    countries: [
      'USA'
    ],
    directors: [
      'William K.L. Dickson'
    ],
    awards: {
      wins: 0,
      nominations: 2,
      text: '2 nominations.'
    },
    released: 1893-05-09T00:00:00.000Z,
    cast: [
      'Charles Kayser',
      'John Ott'
    ]
  }
  {
    title: 'Epic Adventure Scene',
    year: 2015,
    runtime: 150,
    genres: [
      'Action',
      'Adventure'
    ],
    languages: [
      'English'
    ],
```

```
    countries: [
      'USA'
    ],
    directors: [
      'James Nolan'
    ],
    awards: {
      wins: 3,
      nominations: 5,
      text: '3 wins & 5 nominations.'
    },
    released: 2015-07-20T00:00:00.000Z,
    cast: [
      'Tom Hardy',
      'Emily Clark'
    ]
  }
```

12. Find all movies with title, languages, released, directors, writers, awards, year, genres, runtime, cast, countries from the 'movies' collection in MongoDB with cast including "Charles Kayser".

```
>_MONGOSH
> db.movies.find(
    { cast: "Charles Kayser" },
    {
      title: 1,
      languages: 1,
      released: 1,
      directors: 1,
      writers: 1,
      awards: 1,
      year: 1,
      genres: 1,
      runtime: 1,
      cast: 1,
      countries: 1,
      _id: 0
    }
  ).pretty()
< {
    title: 'The Blacksmith Scene',
    year: 1893,
    runtime: 1,
    genres: [
      'Short'
    ],
    languages: [
      'English'
    ],
    countries: [
      'USA'
    ],
    directors: [
      'William K.L. Dickson'
    ],
    awards: {
      wins: 0,
      nominations: 2,
      text: '2 nominations.'
    },
    released: 1893-05-09T00:00:00.000Z,
    cast: [
      'Charles Kayser',
      'John Ott'
    ]
  }
```

13. Retrieve all movies with title, languages, released, directors, writers, countries from the 'movies' collection in MongoDB that released on May 9, 1893.

```
> db.movies.find(
    { released: ISODate("1893-05-09T00:00:00Z") },
    {
      title: 1,
      languages: 1,
      released: 1,
      directors: 1,
      writers: 1,
      countries: 1,
      _id: 0
    }
  ).pretty()
< {
    title: 'The Blacksmith Scene',
    languages: [
      'English'
    ],
    countries: [
      'USA'
    ],
    directors: [
      'William K.L. Dickson'
    ],
    released: 1893-05-09T00:00:00.000Z
  }
```

14. Retrieve all movies with title, languages, released, directors, writers, countries from the 'movies' collection in MongoDB that have a word "scene" in the title.

```
>_MONGOSH
> db.movies.find(
    { title: { $regex: "scene", $options: "i" } },
    {
      title: 1,
      languages: 1,
      released: 1,
      directors: 1,
      writers: 1,
      countries: 1,
      _id: 0
    }
  ).pretty()
< {
    title: 'The Blacksmith Scene',
    languages: [
      'English'
    ],
    countries: [
      'USA'
    ],
    directors: [
      'William K.L. Dickson'
    ],
    released: 1893-05-09T00:00:00.000Z
  }
```

```
{
  title: 'Epic Adventure Scene',
  languages: [
    'English'
  ],
  countries: [
    'USA'
  ],
  directors: [
    'James Nolan'
  ],
  released: 2015-07-20T00:00:00.000Z
}
```