EXPERIMENT 19

```
> use DBMS
< switched to db DBMS
> db.restaurants.insertOne({
    "address": {
      "building": "1007",
      "coord": [-73.856077, 40.848447],
      "street": "Morris Park Ave",
      "zipcode": "10462"
    },
    "borough": "Bronx",
    "cuisine": "Bakery",
    "grades": [
      {"date": new Date(1393804800000), "grade": "A", "score": 2},
      {"date": new Date(1378857600000), "grade": "A", "score": 6},
      {"date": new Date(1358985600000), "grade": "A", "score": 10},
      {"date": new Date(1322006400000), "grade": "A", "score": 9},
      {"date": new Date(1299715200000), "grade": "B", "score": 14}
    ],
    "name": "Morris Park Bake Shop",
    "restaurant_id": "30075445"
  })
```

```
< {
    acknowledged: true,
    insertedId: ObjectId('6648b2822ba022877671bc57')
  }
DBMS >
```

1. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which prepared dish except 'American' and 'Chinees' 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: {
          "date": ISODate("2014-08-11T00:00:00Z"),
          "grade": "A",
          "score": 11
        }
      }
    },
    {
      "restaurant_id": 1,
      "name": 1,
      "grades": 1,
      "_id": 0
    }
  )
<
```

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

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
    }
  )
<
DBMS>
```

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

```
> db.restaurants.find().sort({ "name": 1 })
< {
    _id: ObjectId('6648b2822ba022877671bc57'),
    address: {
      building: '1007',
      coord: [
        -73.856077,
        40.848447
      ],
      street: 'Morris Park Ave',
      zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
      {
        date: 2014-03-03T00:00:00.000Z,
        grade: 'A',
        score: 2
      },
```

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

```
> db.restaurants.find().sort({ "name": -1 })
< {
    _id: ObjectId('6648b2822ba022877671bc57'),
    address: {
      building: '1007',
      coord: [
        -73.856077,
        40.848447
      ],
      street: 'Morris Park Ave',
      zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
      {
        date: 2014-03-03T00:00:00.000Z,
```

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().sort({ "cuisine": 1, "borough": -1 })
< {
    _id: ObjectId('6648b2822ba022877671bc57'),
    address: {
      building: '1007',
      coord: [
        -73.856077,
        40.848447
      ],
      street: 'Morris Park Ave',
      zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
      {
        date: 2014-03-03T00:00:00.000Z,
        grade: 'A',
        score: 2
```

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

```
> db.restaurants.find({
    $and: [
      { "address.street": { $exists: true, $ne: null } },
      { "address.street": { $not: { $size: 0 } } }
    ]
  }).count() === db.restaurants.count()
< DeprecationWarning: Collection.count() is deprecated. Use countDocuments or estimatedDocumentCount.
< true
DBMS>
```

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" } })
< {
    _id: ObjectId('6648b2822ba022877671bc57'),
    address: {
      building: '1007',
      coord: [
        -73.856077,
        40.848447
      ],
      street: 'Morris Park Ave',
      zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
      {
        date: 2014-03-03T00:00:00.000Z,
        grade: 'A',
        score: 2
      },
```

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.

```
> db.restaurants.find(
    { $expr: { $eq: [{ $mod: ["$grades.score", 7] }, 0] } },
    { "restaurant_id": 1, "name": 1, "grades": 1, "_id": 0 }
  )
```

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.

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

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/i } },
    {
      "name": 1,
      "borough": 1,
      "address.coord": 1,
      "cuisine": 1,
      "_id": 0
    }
  )
<
DBMS>
```

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": { $elemMatch: { "score": { $lt: 5 } } } }
  )
< {
    _id: ObjectId('6648b2822ba022877671bc57'),
    address: {
      building: '1007',
      coord: [
        -73.856077,
        40.848447
      ],
      street: 'Morris Park Ave',
      zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
      {
        date: 2014-03-03T00:00:00.000Z,
```

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({
    $and: [
      { "borough": "Manhattan" },
      { "grades": { $elemMatch: { "score": { $lt: 5 } } } }
    ]
  })
<
DBMS >
```

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({
    $and: [
      { "borough": { $in: ["Manhattan", "Brooklyn"] } },
      { "grades": { $elemMatch: { "score": { $lt: 5 } } } }
    ]
  })
<
DBMS>
```

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.

```
> db.restaurants.find({
    $and: [
      { "borough": { $in: ["Manhattan", "Brooklyn"] } },
      { "grades": { $elemMatch: { "score": { $lt: 5 } } } },
      { "cuisine": { $ne: "American" } }
    ]
  })
<
DBMS>
```

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({
    $and: [
      { "borough": { $in: ["Manhattan", "Brooklyn"] } },
      { "grades": { $elemMatch: { "score": { $lt: 5 } } } },
      { "cuisine": { $nin: ["American", "Chinese"] } }
    ]
  })
<
DBMS>
```

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({
    $and: [
      { "grades": { $elemMatch: { "score": 2 } } },
      { "grades": { $elemMatch: { "score": 6 } } }
    ]
  })
< {
    _id: ObjectId('6648b2822ba022877671bc57'),
    address: {
      building: '1007',
      coord: [
        -73.856077,
        40.848447
      ],
      street: 'Morris Park Ave',
      zipcode: '10462'
    },
```

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({
    $and: [
      { "borough": "Manhattan" },
      { "grades": { $elemMatch: { "score": 2 } } },
      { "grades": { $elemMatch: { "score": 6 } } }
    ]
  })
<
DBMS>
```

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({
    $and: [
      { "borough": { $in: ["Manhattan", "Brooklyn"] } },
      { "grades": { $elemMatch: { "score": 2 } } },
      { "grades": { $elemMatch: { "score": 6 } } }
    ]
  })
<
DBMS>
```

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({
    $and: [
      { "borough": { $in: ["Manhattan", "Brooklyn"] } },
      { "grades": { $elemMatch: { "score": 2 } } },
      { "grades": { $elemMatch: { "score": 6 } } },
      { "cuisine": { $nin: ["American"] } }
    ]
  })
<
DBMS>
```

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({
    $and: [
      { "borough": { $in: ["Manhattan", "Brooklyn"] } },
      { "grades": { $elemMatch: { "score": 2 } } },
      { "grades": { $elemMatch: { "score": 6 } } },
      { "cuisine": { $nin: ["American", "Chinese"] } }
    ]
  })
<
DBMS>
```

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.

```
> db.restaurants.find({
    $or: [
      { "grades": { $elemMatch: { "score": 2 } } },
      { "grades": { $elemMatch: { "score": 6 } } }
    ]
  })
< {
    _id: ObjectId('6648b2822ba022877671bc57'),
    address: {
      building: '1007',
      coord: [
        -73.856077,
        40.848447
      ],
      street: 'Morris Park Ave',
      zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
```

EXPERIMENT 20

```
> 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"
    ]
  })
< {
    acknowledged: true,
    insertedId: ObjectId('573a1390f29313caabcd42e8')
  }
DBMS>
```

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

```
> db.movies.find({ "year": 1893 })
<
DBMS>
```

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

```
> db.movies.find({ "runtime": { $gt: 120 } })
<
DBMS>
```

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

```
> db.movies.find({ "genres": "Short" })
< {
    _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'
    ]
  }
DBMS>
```

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({ "director": "William K.L. Dickson" })
<
DBMS>
```

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

```
> db.movies.find({ "country": "USA" })
<
DBMS>
```

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

```
> db.movies.find({ "rating": "UNRATED" })
<
DBMS>
```

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 } })
<
DBMS>
```

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 } })
<
DBMS>
```

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({ "tomato_viewer_rating": { $gt: 4 } })
<
DBMS>
```

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

```
> db.movies.find({ "awards": { $exists: true, $ne: null } })
<
DBMS>
```

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.

```
> db.movies.find(
    { "awards.nominations": { $gt: 0 } }, // Find movies with at least one nomination
    {
      "title": 1,
      "languages": 1,
      "released": 1,
      "directors": 1,
      "writers": 1,
      "awards": 1,
      "year": 1,
      "genres": 1,
      "runtime": 1,
      "cast": 1,
      "countries": 1,
      "_id": 0 // Exclude the default _id field
    }
  )
<
DBMS>
```

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".

```
> db.movies.find(
    { "cast": "Charles Kayser" }, // Filter movies with Charles Kayser in the cast
    {
      "title": 1,
      "languages": 1,
      "released": 1,
      "directors": 1,
      "writers": 1,
      "awards": 1,
      "year": 1,
      "genres": 1,
      "runtime": 1,
      "cast": 1,
      "countries": 1,
      "_id": 0 // Exclude the default _id field
    }
  )
<
DBMS>
```

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("1893-05-09") }, // Filter movies released on May 9, 1893
    {
      "title": 1,
      "languages": 1,
      "released": 1,
      "directors": 1,
      "writers": 1,
      "countries": 1,
      "_id": 0 // Exclude the default _id field
    }
  )
<
DBMS>
```

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.

```
> db.movies.find(
    { "title": { $regex: /scene/i } }, // Filter movies with "scene" in the title
    {
      "title": 1,
      "languages": 1,
      "released": 1,
      "directors": 1,
      "writers": 1,
      "countries": 1,
      "_id": 0 // Exclude the default _id field
    }
  )
<
DBMS>
```