

MongoDB

14.Restaurant Collection

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

```
db.restaurants.find({ $or: [ { cuisine: 'Chinese' }, { cuisine: { $nin: ['American', 'Chinese'] }, name: { $regex: '^Wil', $options: 'i' } } ] }, { _id: 1, name: 1, borough: 1, cuisine: 1 })
```

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 "2014-08-11 T00:00:00Z" among many of survey dates.

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

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": { // Accessing the 2nd element (index 1) of the grades array
      "grade": "A", // Grade must be "A"
      "score": 9, // Score must be 9
      "date": ISODate("2014-08-11T00:00:00Z") // Date must match the specified
ISODate
    } },
  {
    _id: 1, // Retrieve the restaurant ID name:
    1, // Retrieve the restaurant name grades: 1
    // Retrieve the
    grades
  }
)
```

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

```
db.restaurants.find(
  {
```

```

    "coord.1": { $gt: 42, $lte: 52 } // Accessing the 2nd element (index 1) of the coord
array
  }, {
    _id: 1,    // Retrieve the restaurant ID name:
    1, // Retrieve the restaurant name
    address: 1, // Retrieve the address
    coord: 1    // Retrieve the geographical location (coord)
  }
)

```

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 })

```

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

```

db.restaurants.find().sort({ name: -1 })

```

7. **Write a MongoDB query to arrange 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 })

```

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

```

db.restaurants.find({ "address.street": { $exists: false } })

```

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

```

db.restaurants.find({
  "coord": { $type: "double" } // or you can use $type: 1
})

```

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

```

db.restaurants.find(
  {
    "grades": {
      $elemMatch: {
        $expr: {
          $eq: [{ $mod: ["$score", 7] }, 0] // Check if score % 7 == 0
        }
      }
    }
  }
),
{

```

```

    _id: 1,    // Retrieve the restaurant ID name:
    1, // Retrieve the restaurant name grades: 1
    // Retrieve the grades
  }
)

```

- 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 } // Regex to find 'mon' anywhere in the
    name (case-insensitive)
  },
  {
    name: 1,    // Retrieve the restaurant
    name borough: 1,    // Retrieve
    the borough
    "coord.0": 1, // Retrieve longitude (assuming longitude is the first element in the
    coord array)
    "coord.1": 1, // Retrieve latitude (assuming latitude is the second element in the
    coord array) cuisine: 1, // Retrieve the cuisine
    _id: 0      // Exclude the restaurant ID from the results
  }
)

```

- 12. Write a mongodb query to find the restaurant name, borough, longitude and attitude and cuisine for those restaurants which contains 'Mad' as first three letters in its name.**

```

db.restaurants.find({ name: { $regex: /^Mad/i } }, { name: 1, borough: 1, "coord.0":
1, "coord.1": 1, cuisine: 1, _id: 0 })

```

- 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 } // Score must be
        less than 5
      }
    }
  }
)

```

- 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", // Condition to filter by borough
    "grades": {
      $elemMatch: { score: { $lt: 5 } // Condition to filter grades with score
        less than 5
      }
    }
  }
)

```

- 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(
  {
    $or: [
      { borough: "Manhattan" }, // Condition to filter by borough Manhattan
      { borough: "Brooklyn" } // Condition to filter by borough Brooklyn
    ],
    "grades": {
      $elemMatch: { score: { $lt: 5 } // Condition to filter grades with score
        less than 5
      }
    }
  }
)

```

Movies Collection

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

```

db.movies.find(
  {
    releaseYear: 1893 // Assuming the field for the release year is named 'releaseYear'
  }
)

```

```
}  
)
```

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 } // Assuming the field for runtime is named 'runtime'  
  }  
)
```

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

```
db.movies.find(  
  {  
    genres: "Short" // Assuming the field for genres is an array named 'genres'  
  }  
)
```

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" // Assuming the field for the director is named  
    'director' }  
)
```

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" // Assuming the field for the release country is named 'country'  
  }  
)
```

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

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

```
    {
      rating: "UNRATED" // Assuming the field for the rating is named 'rating'
    }
  )
}
```

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

```
db.movies.find(
  {
    votes: { $gt: 1000 } // Assuming the field for votes is named 'votes'
  }
)
```

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

```
db.movies.find(
  {
    imdbRating: { $gt: 7 } // Assuming the field for IMDb rating is named 'imdbRating'
  }
)
```

- 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: { $gt: 4 } } // Assuming the viewer rating is nested within a
    'tomatoes' object
  }
)
```