

01. Write a MongoDB query to display all the documents in the collection restaurants.

```
db.restaurants.find()
```

02. Write a MongoDB query to display the fields restaurant\_id, name, borough and cuisine for all the documents in the collection restaurant.

```
db.restaurants.find({}, {restaurant_id: 1, name: 1, borough: 1, cuisine: 1})
```

3. Write a MongoDB query to display the fields restaurant\_id, name, borough and cuisine, but exclude the field \_id for all the documents in the collection restaurant.

```
db.restaurants.find({}, { _id: 0, restaurant_id: 1, name: 1, borough: 1, cuisine: 1 })
```

4. Write a MongoDB query to display the fields restaurant\_id, name, borough and zip code, but exclude the field \_id for all the documents in the collection restaurant.

```
db.restaurants.find({}, { _id: 0, restaurant_id: 1, name: 1, borough: 1, "address.zipcode": 1 })
```

5. Write a MongoDB query to display all the restaurants which are in the borough Bronx.

```
db.restaurants.find({ borough: "Bronx" })
```

6. Write a MongoDB query to display the first 5 restaurants which are in the borough Bronx.

```
db.restaurants.find({ borough: "Bronx" }).limit(5)
```

7. Write a MongoDB query to display the next 5 restaurants after skipping the first 5 which are in the borough Bronx.

```
db.restaurants.find({ borough: "Bronx" }).skip(5).limit(5)
```

8. Write a MongoDB query to find the restaurants who achieved a score more than 90.

```
db.restaurants.find({ "grades.score": { $gt: 90 } })
```

9. Write a MongoDB query to find the restaurants that achieved a score, more than 80 but less than 100.

```
db.restaurants.find({score: {$gt: 80, $lt: 100}})
```

10. Write a MongoDB query to find the restaurants which locate in latitude value less than -95.754168.

```
db.restaurants.find({latitude: {$lt: -95.754168}})
```

11. Write a MongoDB query to find the restaurants that do not prepare any cuisine of 'American' and their grade score more than 70 and latitude less than -65.754168.

```
db.restaurants.find({
  cuisine: {$ne: 'American'},
  grade: {$elemMatch: {score: {$gt: 70}}},
  latitude: {$lt: -65.754168}
})
```

12. Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American' and achieved a score more than 70 and located in the longitude less than -65.754168.

Note : Do this query without using \$and operator.

```
db.restaurants.find({
  cuisine: {$ne: 'American'},
  score: {$gt: 70},
  longitude: {$lt: -65.754168}
})
```

13. Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American' and achieved a grade point 'A' not belonging to the borough

Brooklyn. The document must be displayed according to the cuisine in descending order.

```
db.restaurants.find({
  cuisine: {$ne: 'American'},
  'grades.grade': 'A',
  'grades.score': {$gte: 90},
  borough: {$ne: 'Brooklyn'}
}).sort({cuisine: -1})
```

14. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Wil' as first three letters for its name.

```
db.restaurants.find({
  name: {$regex: /^Wil/}
}),
{
  restaurant_id: 1,
  name: 1,
  borough: 1,
  cuisine: 1,
  _id: 0
})
```

15. Write a MongoDB query to find the restaurant ID, name, borough and cuisine for those restaurants which contain 'ces' as the last three letters for its name.

```
db.restaurants.find({
  name: {$regex: /ces$/}
}),
{
  restaurant_id: 1,
  name: 1,
  borough: 1,
  cuisine: 1,
  _id: 0
})
```

16. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Reg' as three letters somewhere in its name.

```
db.restaurants.find({
  name: {$regex: /Reg/}
}),
```

```
{
  restaurant_id: 1,
  name: 1,
  borough: 1,
  cuisine: 1,
  _id: 0
})
```

17. Write a MongoDB query to find the restaurants which belong to the borough Bronx and prepare either American or Chinese dishes.

```
db.restaurants.find({
  borough: "Bronx",
  cuisine: {$in: ["American", "Chinese"]}
})
```

18. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which belong to the borough Staten Island or Queens or Bronx or Brooklyn.

```
db.restaurants.find({
  borough: {$in: ["Staten Island", "Queens", "Bronx", "Brooklyn"]}
},
{
  restaurant_id: 1,
  name: 1,
  borough: 1,
  cuisine: 1,
  _id: 0
})
```

19. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which are not belonging to the borough Staten Island or Queens or Bronx or Brooklyn.

```
db.restaurants.find({
  borough: {$nin: ["Staten Island", "Queens", "Bronx", "Brooklyn"]}
},
{
  restaurant_id: 1,
  name: 1,
  borough: 1,
  cuisine: 1,
  _id: 0
})
```

20. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which achieved a score which is not more than 10.

```
db.restaurants.find({
  "grades.score": {$lte: 10}
},
{
  restaurant_id: 1,
  name: 1,
  borough: 1,
  cuisine: 1,
  _id: 0
})
```

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

```
db.restaurants.find({
  $or: [
    {name: /^Wil/},
    {cuisine: {$nin: ["American", "Chinese"]}}
  ]
},
{
  restaurant_id: 1,
  name: 1,
  borough: 1,
  cuisine: 1,
  _id: 0
})
```

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

23. 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
})
```

24. 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({
  "address.coord.1": { $gt: 42, $lte: 52 }
},
{
  restaurant_id: 1,
  name: 1,
  "address": 1,
  "coord": 1,
}
```

```
  _id: 0
})
```

25. 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 })
```

26. 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 })
```

27. 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 })
```

28. Write a MongoDB query to know whether all the addresses contain the street or not.

```
db.restaurants.find({address: {$not: /street/}}, {_id: 0, name: 1, address: 1})
```

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

```
db.restaurants.find({"address.coord": {"$type": "double"}})
```

30. 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( { "grades.score": { $mod: [7, 0] } }, { "restaurant_id": 1, "name": 1, "grades": 1 } )
```

31. 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
  }
)
```

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

```
db.restaurants.find(
  {
    "name": {
      $regex: /^Mad/i
    }
  },
  {
    "name": 1,
    "borough": 1,
```

```
    "address.coord": 1,  
    "cuisine": 1  
  }  
)
```