# PRACTICAL-3

AIM: Performing queries based on AND, OR, Limit, Sort and Projection and apply some queries to get specified output.

## ❖ PRACTICE QUESTION

**1.Find all the student details whose city is Ahmedabad and age is 20.**

**INPUT:**

```
db.student_21012011074.find({$and:[{"city":"Ahmedabad"},{"Age":20}]}).pretty()
```

**Output :-**

```
guru> db.student_21012011074.find({$and:[{"city":"Ahmedabad"},{"Age":20}]}).pretty()

guru>
```

**2.Display enrolment number of students whose enrolment number is greater than 3 or age is 20.**

**INPUT:**

```
db.student_21012011074.find({$or:[{en_no:{$gt:3}},{"Age":20}]}).pretty()
```

**OUTPUT:**

```
guru> db.student_21012011074.find({$or:[{en_no:{$gt:3}},{"Age":20}]}).pretty()
[
  {
    _id: ObjectId("63e3ace23d95ce055cdb477a"),
    en_no: 4,
    department: 'IT',
    City: 'Surat',
    Age: 18,
    Pincode: 383312
  },
  {
    _id: ObjectId("63e3acfc3d95ce055cdb477b"),
    en_no: 5,
    department: 'CE',
    City: 'Surat',
    Age: 19,
    Pincode: 383311
  },
  {
    _id: ObjectId("63e86392cb89e588f21e132f"),
    en_no: 7,
    department: 'IT',
    City: 'Ahmedabad',
    Pincode: 383313
  },
  {
    _id: ObjectId("63e86392cb89e588f21e1330"),
    en_no: 8,
    department: 'IT',
    City: 'Modasa',
    Pincode: 383315
  }
]
guru>
```

**Q.3 Count the number of documents whose age is > 20.**

**INPUT:**

```
db.student_21012011074.count()
```

**OUTPUT:**

```
guru> db.student_21012011074.count()
DeprecationWarning: Collection.count() is deprecated. Use countDocuments or estimatedDocumentCount.
7
guru>
```

**Q.4 Only display first 5 records of student collection.**

INPUT:

```
db.student_21012011074.find().limit(5)
```

OUTPUT:

```
guru> db.student_21012011074.find().limit(5)
[
  {
    _id: ObjectId("63e3aa9e3d95ce055cdb4777"),
    en_no: 1,
    department: 'CE',
    City: 'Modasa',
    Age: 19,
    Pincode: 383315
  },
  {
    _id: ObjectId("63e3ac573d95ce055cdb4778"),
    en_no: 2,
    department: 'CE',
    City: 'Gandhinagar',
    Age: 19,
    Pincode: 383314
  },
  {
    _id: ObjectId("63e3ac773d95ce055cdb4779"),
    en_no: 3,
    department: 'IT',
    City: 'Ahemdabad',
    Age: 18,
    Pincode: 383313
  },
  {
    _id: ObjectId("63e3ace23d95ce055cdb477a"),
    en_no: 4,
    department: 'IT',
    City: 'Surat',
    Age: 18,
    Pincode: 383312
  },
  {
    _id: ObjectId("63e3acfc3d95ce055cdb477b"),
    en_no: 5,
    department: 'CE',
    City: 'Surat',
    Age: 19,
    Pincode: 383311
  }
]
guru>
```

**Q.5 Display records of students whose age is 21, skip first 2 records.**

INPUT:

```
db.student_21012011074.find({"Age":21}).skip(2).pretty()
```

OUTPUT:

```
guru> db.student_21012011074.find({"Age":21}).skip(2).pretty()

guru>
```

**Q.6 Sort & display records of students based on ascending order of name.**

**INPUT:**

```
db.student_21012011074.find({},{"name":1,"_id":0})
```

**OUTPUT:**

```
guru> db.student_21012011074.find({},{"name":1,"_id":0})
[
  {}, {}, {}, {},
  {}, {}, {}
]
guru>
```

## ❖ EXERCISE QUESTION

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

   INPUT:

   db.res_21012011074.find()

   OUTPUT:

```
res_guru> db.res_21012011074.find()
[
  {
    _id: ObjectId("63db61b798e42a90c09e2cfc"),
    address: {
      building: '8825',
      coord: [ -73.8803827, 40.7643124 ],
      street: 'Astoria Boulevard',
      zipcode: '11369'
    },
    borough: 'Queens',
    cuisine: 'American ',
    grades: [
      {
        date: ISODate("2014-11-15T00:00:00.000Z"),
        grade: 'Z',
        score: 38
      },
      {
        date: ISODate("2014-05-02T00:00:00.000Z"),
        grade: 'A',
        score: 10
      },
      {
        date: ISODate("2013-03-02T00:00:00.000Z"),
        grade: 'A',
        score: 7
      },
      {
        date: ISODate("2012-02-10T00:00:00.000Z"),
        grade: 'A',
        score: 13
      }
    ],
    name: 'Brunos On The Boulevard',
```

   And Continue……

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

   INPUT;

   db.res_21012011074.find({},{"restaurant_id":1,"name":1,"borough":1,"cuisine":1,"_id":0})

OUTPUT:

```
res_guru> db.res_21012011074.find({},{"restaurant_id":1,"name":1,"borough":1,"cuisine":1,"_id":0})
[
  {
    borough: 'Queens',
    cuisine: 'American ',
    name: 'Brunos On The Boulevard',
    restaurant_id: '40356151'
  },
  {
    borough: 'Brooklyn',
    cuisine: 'Hamburgers',
    name: "Wendy'S",
    restaurant_id: '30112340'
  },
  {
    borough: 'Brooklyn',
    cuisine: 'Chinese',
    name: 'May May Kitchen',
    restaurant_id: '40358429'
  },
  {
    borough: 'Manhattan',
    cuisine: 'American ',
    name: '1 East 66Th Street Kitchen',
    restaurant_id: '40359480'
  },
  {
    borough: 'Queens',
    cuisine: 'Jewish/Kosher',
    name: 'Tov Kosher Kitchen',
    restaurant_id: '40356068'
  },
  {
    borough: 'Bronx',
    cuisine: 'American ',
    name: 'Wild Asia',
    restaurant_id: '40357217'
  },
```

**And Continue……..**

3. 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. (USING PROJECTION)

INPUT:
db.res_21012011074.find({},{"restaurant_id":1,"name":1,"borough":1,"address.zipcode":1,"_id":0})

OUTPUT:

```
res_guru> db.res_21012011074.find({},{"restaurant_id":1,"name":1,"borough":1,"address.zipcode":1,"_id":0})
[
  {
    address: { zipcode: '11369' },
    borough: 'Queens',
    name: 'Brunos On The Boulevard',
    restaurant_id: '40356151'
  },
  {
    address: { zipcode: '11225' },
    borough: 'Brooklyn',
    name: "Wendy'S",
    restaurant_id: '30112340'
  },
  {
    address: { zipcode: '11208' },
    borough: 'Brooklyn',
    name: 'May May Kitchen',
    restaurant_id: '40358429'
  },
  {
    address: { zipcode: '10065' },
    borough: 'Manhattan',
    name: '1 East 66Th Street Kitchen',
    restaurant_id: '40359480'
  },
  {
```

**And Continue…….**

4. Write a MongoDB query to display the first 5 restaurant which is in the borough Bronx. (USING LIMIT)

INPUT:

db.res_21012011074.find({"borough":"Bronx"}).limit(5)

OUTPUT:

```
res_guru> db.res_21012011074.find({"borough":"Bronx"}).limit(5)
[
  {
    _id: ObjectId("63db61b798e42a90c09e2d01"),
    address: {
      building: '2300',
      coord: [ -73.8786113, 40.8502883 ],
      street: 'Southern Boulevard',
      zipcode: '10460'
    },
    borough: 'Bronx',
    cuisine: 'American ',
    grades: [
      {
        date: ISODate("2014-05-28T00:00:00.000Z"),
        grade: 'A',
        score: 11
      },
      {
        date: ISODate("2013-06-19T00:00:00.000Z"),
        grade: 'A',
        score: 4
      },
      {
        date: ISODate("2012-06-15T00:00:00.000Z"),
        grade: 'A',
        score: 3
      }
    ],
    name: 'Wild Asia',
    restaurant_id: '40357217'
  },
  {
    _id: ObjectId("63db61b798e42a90c09e2d0f"),
    address: {
      building: '1007',
      coord: [ -73.856077, 40.848447 ],
```

**And Continue…..**

5. Write a MongoDB query to display the next 5 restaurants after skipping first 5 which are in the borough Bronx. (USING SKIP)

INPUT:

db.res_21012011074.find({"borough":"Bronx"},{"_id":0}).skip(5)

OUTPUT:

```
res_guru> db.res_21012011074.find({"borough":"Bronx"},{"_id":0}).skip(5)
[
  {
    address: {
      building: '658',
      coord: [ -73.81363999999999, 40.82941100000001 ],
      street: 'Clarence Ave',
      zipcode: '10465'
    },
    borough: 'Bronx',
    cuisine: 'American ',
    grades: [
      {
        date: ISODate("2014-06-21T00:00:00.000Z"),
        grade: 'A',
        score: 5
      },
      {
        date: ISODate("2012-07-11T00:00:00.000Z"),
        grade: 'A',
        score: 10
      }
    ],
    name: 'Manhem Club',
    restaurant_id: '40364363'
  },
  {
    address: {
      building: '72',
      coord: [ -73.92506, 40.8275556 ],
      street: 'East  161 Street',
      zipcode: '10451'
    },
    borough: 'Bronx',
    cuisine: 'American ',
    grades: [
      {
```

**And Continue……**

6. 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. (USING AND)

INPUT:

db.res_21012011074.find({"cuisine":{$ne:"American"},"grades.score":{$gt:70},"address.coord":{$lt:-65.754168}})

OUTPUT:

```
res_guru> db.res_21012011074.find({"cuisine":{$ne:"American"},"grades.score":{$gt:70},"address.coord":{$lt:-65.754168}})
[
  {
    _id: ObjectId("63db61b798e42a90c09e2e4f"),
    address: {
      building: '65',
      coord: [ -73.9782725, 40.7624022 ],
      street: 'West   54 Street',
      zipcode: '10019'
    },
    borough: 'Manhattan',
    cuisine: 'American ',
    grades: [
      {
        date: ISODate("2014-08-22T00:00:00.000Z"),
        grade: 'A',
        score: 11
      },
      {
        date: ISODate("2014-03-28T00:00:00.000Z"),
        grade: 'C',
        score: 131
      },
      {
        date: ISODate("2013-09-25T00:00:00.000Z"),
        grade: 'A',
        score: 11
      },
      {
        date: ISODate("2013-04-08T00:00:00.000Z"),
        grade: 'B',
        score: 25
      },
      {
        date: ISODate("2012-10-15T00:00:00.000Z"),
        grade: 'A',
        score: 11
```

**And Continue…….**

7. Write a MongoDB query to find the restaurants which belong to the borough Bronx and prepared either American or Chinese dish. (USING OR)

   INPUT:

   db.res_21012011074.find({"borough":"Bronx",$or:[{"cuisine":"American"},{"cuisine":"Chinese"}]})

   OUTPUT:

```
res_guru> db.res_21012011074.find({"borough":"Bronx",$or:[{"cuisine":"American"},{"cuisine":"Chinese"}]})
[
  {
    _id: ObjectId("63db61b798e42a90c09e2d1e"),
    address: {
      building: '1236',
      coord: [ -73.8893654, 40.81376179999999 ],
      street: '238 Spofford Ave',
      zipcode: '10474'
    },
    borough: 'Bronx',
    cuisine: 'Chinese',
    grades: [
      {
        date: ISODate("2013-12-30T00:00:00.000Z"),
        grade: 'A',
        score: 8
      },
      {
        date: ISODate("2013-01-08T00:00:00.000Z"),
        grade: 'A',
        score: 10
      },
      {
        date: ISODate("2012-06-12T00:00:00.000Z"),
        grade: 'B',
        score: 15
      }
    ],
    name: 'Happy Garden',
    restaurant_id: '40363289'
  },
  {
    _id: ObjectId("63db61b798e42a90c09e2d2b"),
    address: {
```

**And Continue…….**

**8.** Write a MongoDB query to arrange the name of the restaurants in ascending / descending order along with all the columns. (USING SORT)

INPUT:

db.res_21012011074.find().sort({"name":1})

OUTPUT:

```
res_guru> db.res_21012011074.find().sort({"name":1})
[
  {
    _id: ObjectId("63db61b798e42a90c09e3997"),
    address: {
      building: '129',
      coord: [ -73.962943, 40.685007 ],
      street: 'Gates Avenue',
      zipcode: '11238'
    },
    borough: 'Brooklyn',
    cuisine: 'Italian',
    grades: [
      {
        date: ISODate("2014-03-06T00:00:00.000Z"),
        grade: 'A',
        score: 5
      },
      {
        date: ISODate("2013-08-29T00:00:00.000Z"),
        grade: 'A',
        score: 2
      },
      {
        date: ISODate("2013-03-08T00:00:00.000Z"),
        grade: 'A',
        score: 7
      },
      {
        date: ISODate("2012-06-27T00:00:00.000Z"),
        grade: 'A',
        score: 7
      },
      {
        date: ISODate("2011-11-17T00:00:00.000Z"),
```