SELECTORS

"$lt", "$lte", "$gt", and "$gte" are all comparison operators, corresponding to <,<=, >, and >=, respectively. They can be combined to look for a range of values.

```
test> db.stu.find({age:{$gt:20}}).count();
310
test>
```

AND OPERATOR :

This operator is used to perform logical AND operation on the array of one or more expressions and select or retrieve only those documents that match all the given expression. You can use this operator in methods like find(), update(), etc.

```
test> db.stu.find({
...     $and:[
...       {home_city:"City 2"},
...       {blood_group:"B+"},
...      ]
... });
[
  {
    _id: ObjectId('6661e4dad45a6fc3f4eef54d'),
    name: 'Student 584',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English', 'Mathematics']",
    gpa: 2.42,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6661e4dad45a6fc3f4eef584'),
    name: 'Student 367',
    age: 19,
    courses: "['English', 'Physics', 'History', 'Mathematics']",
    gpa: 2.81,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6661e4dad45a6fc3f4eef610'),
    name: 'Student 255',
    age: 21,
    courses: "['English', 'Physics']",
    gpa: 2.85,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6661e4dad45a6fc3f4eef61c'),
    name: 'Student 281',
    age: 18,
    courses: "['History', 'Mathematics', 'Physics', 'Computer Science']",
    gpa: 2.2,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6661e4dad45a6fc3f4eef66a'),
    name: 'Student 289',
    age: 18,
    courses: "['History', 'Physics']",
    gpa: 2.89,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6661e4dad45a6fc3f4eef695'),
    name: 'Student 303',
    age: 20,
    courses: "['Physics', 'English']",
    gpa: 3.08,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6661e4dad45a6fc3f4eef6fb'),
    name: 'Student 872',
    age: 24,
    courses: "['English', 'Mathematics', 'History']",
    gpa: 3.36,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: true
  }
]
test>
```

OR OPERATOR : **There are two ways to do an OR query in MongoDB. "$in"**

can be used to query for a variety of values for a single key. "$or" is more general; it can be used to query for any of the given values across multiple keys.

```
test> db.stu.find({
...     $or:[
...         {is_hostel_resident:true},
...         {gpa:{$lt:3.0}}
...     ]
... }).count();
261
```

DOWNLOADING NEW DATA SET

- **New Students Permission dataset link**

Explanation: Collection name: student

- **name: Student's name (string)**
- **age: Student's age (number)**
- **permissions: Bitmask representing user permissions (number)**

Bitwise Value

- **In our example its a 32 bit each bit representing different things ●**
  Bitwise value 7 means all access 7 -> 111

Bit 3	Bit 2	Bit 1
cafe	campus	lobby

TYPES :

Name	Description
`$bitsAllClear`	Matches numeric or binary values in which set positions have a value of `0`.
`$bitsAllSet`	Matches numeric or binary values in which a set of bit positions *all* have a value of `1`.
`$bitsAnyClear`	Matches numeric or binary values in which *any* bit from a set of bit positions has a value of `0`.
`$bitsAnySet`	Matches numeric or binary values in which *any* bit from a set of bit positions has a value of `1`.

QUERY

```
test> db.student.find({
...       permissions:{$bitsAllSet:[LOBBY_PERMISSION,CAMPUS_PERMISSION]}
...   });
[
  {
    _id: ObjectId('665f361536cc83d394ce92cd'),
    name: 'George',
    age: 21,
    permissions: 6
  },
  {
    _id: ObjectId('665f361536cc83d394ce92ce'),
    name: 'Henry',
    age: 27,
    permissions: 7
  },
  {
    _id: ObjectId('665f361536cc83d394ce92cf'),
    name: 'Isla',
    age: 18,
    permissions: 6
  }
]
```

Geospatial :

- Official Documentation link
- Create collection called “locations”
- Upload the dataset using json link Geospatial

Query :

In MongoDB, you can store geospatial data as GeoJSON objects or as legacy coordinate pairs.

GeoJSON Objects To calculate geometry over an Earth-like sphere, store your location data as GeoJSON objects.To specify GeoJSON data, use an embedded document with:

- a field named `type` that specifies the GeoJSON object type, and
- a field named `coordinates` that specifies the object's coordinates.

```
<field>: { type: <GeoJSON type> , coordinates: <coordinates> }
```

If specifying latitude and longitude coordinates, list the longitude first and then latitude; i.e.

```
<field>: [<longitude>,<latitude> ]
```

Specify via an embedded document:

```
<field>: { <field1>: <x>, <field2>: <y> }
```

If specifying latitude and longitude coordinates, the first field, regardless of the field name, must contain the longitude value and the second field, the latitude value ; i.e.

```
<field>: { <field1>: <longitude>,<field2>:<latitude>}
```

EXAMPLE :

```
test> db.location.find({
...       location:{
...         $geoWithin:{
...           $centerSphere:[[-74.005,40.712],0.00621376]
...         }
...       }
... });
[
  {
    _id: 1,
    name: 'Coffee Shop A',
    location: { type: 'Point', coordinates: [ -73.985, 40.748 ] }
  },
  {
    _id: 2,
    name: 'Restaurant B',
    location: { type: 'Point', coordinates: [ -74.009, 40.712 ] }
  },
  {
    _id: 5,
    name: 'Park E',
    location: { type: 'Point', coordinates: [ -74.006, 40.705 ] }
  }
]
test>
```