CLASS-4

New Data Set: link

Explanation: Collection name: sp

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

```
db> db.sp.find();
[
  {
    _id: ObjectId('666af8aaf187a483eb5aa10a'),
    name: 'Alice',
    age: 22,
    permissions: 0
  },
  {
    _id: ObjectId('666af8aaf187a483eb5aa10b'),
    name: 'Bob',
    age: 25,
    permissions: 1
  },
  {
    _id: ObjectId('666af8aaf187a483eb5aa10c'),
    name: 'Charlie',
    age: 20,
    permissions: 2
  },
  {
    _id: ObjectId('666af8aaf187a483eb5aa10d'),
    name: 'David',
    age: 28,
    permissions: 3
  },
  {
    _id: ObjectId('666af8aaf187a483eb5aa10e'),
    name: 'Eve',
    age: 19,
    permissions: 4
  },
  {
    _id: ObjectId('666af8aaf187a483eb5aa10f'),
    name: 'Fiona',
    age: 23,
    permissions: 5
  },
  {
    _id: ObjectId('666af8aaf187a483eb5aa110'),
    name: 'George',
```

Bitwise Types:

Bitwise

Name	Description
`$bitsAllClear`	Matches numeric or binary values in which a set of bit positions *all* have a value of `0`.
`$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`.

```
db> const lobby_p=1;

db> const campus_p=2;

db> db.sp.find({ permissions: { $bitsAllSet: [lobby_p, campus_p] } });
[
  {
    _id: ObjectId('666af8aaf187a483eb5aa110'),
    name: 'George',
    age: 21,
    permissions: 6
  },
  {
    _id: ObjectId('666af8aaf187a483eb5aa111'),
    name: 'Henry',
    age: 27,
    permissions: 7
  },
  {
    _id: ObjectId('666af8aaf187a483eb5aa112'),
    name: 'Isla',
    age: 18,
    permissions: 6
  }
]
```

Geospatial:

Geospatial queries in MongoDB are operations that allow to store, index, and query geographic data. This type of data includes points, lines, and polygons representing Earth's locations or shapes. MongoDB provides powerful tools to work with this data through geospatial indexes and queries.

2d Indexes 2d indexes support queries that calculate geometries on a two-dimensional plane. To create a 2d index, use the db.collection.createIndex() method, specifying the location field as the key and the string literal "2d" as the index type:

```
db.collection.createIndex( { <location field>: "2d" } )
```

2dsphere Indexes 2dsphere indexes support queries that calculate geometries on a sphere, such as the surface of the Earth. To create a 2dsphere index, use the db.collection.createIndex() method, specifying the location field as the key and the string literal "2dsphere" as the index type:

```
db.collection.createIndex( {<location field>:  : "2dsphere" } )
```

```
db> db.locations.find();
[
  {
    _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: 3,
    name: 'Library C',
    location: { type: 'Point', coordinates: [ -77.036, 38.907 ] }
  },
  {
    _id: 4,
    name: 'Museum D',
    location: { type: 'Point', coordinates: [ -80.843, 34.26 ] }
  },
  {
    _id: 5,
    name: 'Park E',
    location: { type: 'Point', coordinates: [ -74.006, 40.705 ] }
  }
]
```

Data types and Operations:

Name	Description
`$geoIntersects`	Selects geometries that intersect with a GeoJSON geometry. The `2dsphere` index supports `$geoIntersects`.
`$geoWithin`	Selects geometries within a bounding GeoJSON geometry. The `2dsphere` and `2d` indexes support `$geoWithin`.
`$near`	Returns geospatial objects in proximity to a point. Requires a geospatial index. The `2dsphere` and `2d` indexes support `$near`.
`$nearSphere`	Returns geospatial objects in proximity to a point on a sphere. Requires a geospatial index. The `2dsphere` and `2d` indexes support `$nearSphere`.

Example:

```
db> db.locations.find(
... { location: {
... $geoWithin: {
... $centerSphere: [[-74.006,40.713],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 ] }
  }
]
```