

Class 4: Projection, Limit & Selectors

Agenda:-

- Understand Projection:
 - Develop a MongoDB query to select certain fields and ignore some fields of the documents from
- Understand LIMIT:
 - Develop a MongoDB query to display the first 5 documents from the results obtained.

PROJECTION:-

1. projection refers to the process of selecting specific fields to return from a query, effectively controlling which fields of the documents are included in the result set. This can help improve performance by reducing the amount of data transferred over the network and processed by the application.

2. The projection work based on:

- 1 indicates that the field should be included name to be returned.
- 0 indicates that the field should be excluded them from the result.

Ex:

```
{ "_id": 1,  
  "name": "Alice",  
  "age": 30,  
  "email": "alice@example.com",  
  "address": {  
    "city": "New York",  
    "zipcode": "10001"  
  }  
}
```

Get Selected Attributes:-

To select specific fields in a MongoDB query, you use the find() method with a projection document. The projection document specifies which fields to include (using 1) or exclude (using 0).

```
db. students. find ({}, { _id: 0 });
```

```
db> db.students.find({}, {_id:0});
[
  {
    name: 'Student 328',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English']",
    gpa: 3.42,
    home_city: 'City 2',
    blood_group: 'AB-',
    is_hotel_resident: true
  },
  {
    name: 'Student 468',
    age: 21,
    courses: "['Computer Science', 'Physics', 'Mathematics', 'History']",
    gpa: 3.97,
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    name: 'Student 504',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English', 'Mathematics']",
    gpa: 2.92,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    name: 'Student 915',
    age: 22,
    courses: "['Computer Science', 'History', 'Physics', 'English']",
    gpa: 3.37,
    blood_group: 'AB+',
    is_hotel_resident: true
  },
  {
    name: 'Student 367',
    age: 25,
    courses: "['History', 'Physics', 'Computer Science']",
    gpa: 3.11,
  }
]
```

Ignore attributes:-

```
Db. students. find({}, {_id:0});
```

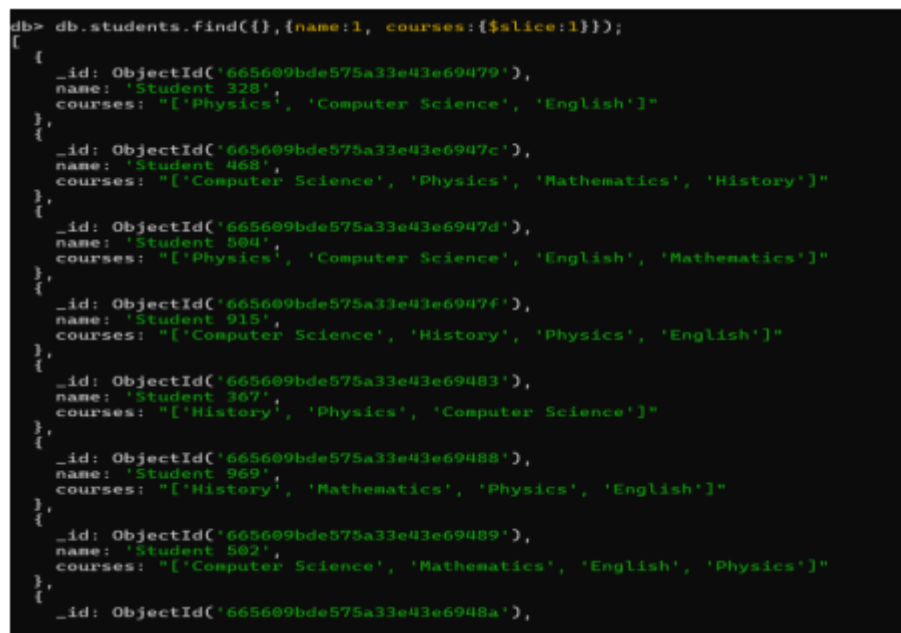
```
db> db.students.find({}, {_id:0});
[
  {
    name: 'Student 328',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English']",
    gpa: 3.42,
    home_city: 'City 2',
    blood_group: 'AB-',
    is_hotel_resident: true
  },
  {
    name: 'Student 468',
    age: 21,
    courses: "['Computer Science', 'Physics', 'Mathematics', 'History']",
    gpa: 3.97,
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    name: 'Student 504',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English', 'Mathematics']",
    gpa: 2.92,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    name: 'Student 915',
    age: 22,
    courses: "['Computer Science', 'History', 'Physics', 'English']",
    gpa: 3.37,
    blood_group: 'AB+',
    is_hotel_resident: true
  },
  {
    name: 'Student 367',
    age: 25,
    courses: "['History', 'Physics', 'Computer Science']",
    gpa: 3.11,
  }
]
```

Retrieving Specific Fields from Nested Objects :-

The \$slice operator in MongoDB is used to select a subset of an array. It is particularly useful when you have large arrays stored in your documents and you only need to retrieve certain elements, optimizing data retrieval and reducing overhead.

Ex:

```
db.Students.find({}, {
  name:1,
  courses:{$slice:1}
});
```



```
db> db.students.find({}, {name:1, courses:{$slice:1}});
[
  {
    _id: ObjectId('665609bde575a33e43e69479'),
    name: 'Student 328',
    courses: ["Physics", "Computer Science", "English"]
  },
  {
    _id: ObjectId('665609bde575a33e43e6947c'),
    name: 'Student 468',
    courses: ["Computer Science", "Physics", "Mathematics", "History"]
  },
  {
    _id: ObjectId('665609bde575a33e43e6947d'),
    name: 'Student 504',
    courses: ["Physics", "Computer Science", "English", "Mathematics"]
  },
  {
    _id: ObjectId('665609bde575a33e43e6947f'),
    name: 'Student 915',
    courses: ["Computer Science", "History", "Physics", "English"]
  },
  {
    _id: ObjectId('665609bde575a33e43e69483'),
    name: 'Student 367',
    courses: ["History", "Physics", "Computer Science"]
  },
  {
    _id: ObjectId('665609bde575a33e43e69488'),
    name: 'Student 969',
    courses: ["History", "Mathematics", "Physics", "English"]
  },
  {
    _id: ObjectId('665609bde575a33e43e69489'),
    name: 'Student 502',
    courses: ["Computer Science", "Mathematics", "English", "Physics"]
  },
  {
    _id: ObjectId('665609bde575a33e43e6948a'),
```

Limit:-

The limit method is used to restrict the number of documents returned by a query. This can be particularly useful when dealing with large collections, as it allows you to control the amount of data transferred and processed, thereby improving performance and reducing resource consumption.

Syntax:-

```
db.collection.find({filter},{projection}).limit(number)
```

Get first 5 document:-

```
db.Students.find({}, {_id:0}).limit(5);
```

```

db> db.students.find({}, {_id:0}).limit(5);
[
  {
    name: 'Student 328',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English']",
    gpa: 3.42,
    home_city: 'City 2',
    blood_group: 'AB-',
    is_hotel_resident: true
  },
  {
    name: 'Student 468',
    age: 21,
    courses: "['Computer Science', 'Physics', 'Mathematics', 'History']",
    gpa: 3.97,
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    name: 'Student 584',
    age: 21,
    courses: "['Physics', 'Computer Science', 'English', 'Mathematics']",
    gpa: 2.92,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    name: 'Student 915',
    age: 22,
    courses: "['Computer Science', 'History', 'Physics', 'English']",
    gpa: 3.37,
    blood_group: 'AB+',
    is_hotel_resident: true
  },
  {
    name: 'Student 367',
    age: 25,
    courses: "['History', 'Physics', 'Computer Science']",
    gpa: 3.11,
  }
]

```

Limiting Results:-

Limiting results in MongoDB is an important technique for controlling the amount of data returned by a query. This can help in optimizing performance, managing large datasets, and implementing pagination.

Syntax:-

```
db. collection. find().limit()
```

Ex :-

```

db. students. find({
  gpa :{$gt:3.5} },
  {_id:0}).limit(2);

```

```

db> db.students.find({gpa:{$gt:3.5}}, {_id:0}).limit(2);
[
  {
    name: 'Student 468',
    age: 21,
    courses: "['Computer Science', 'Physics', 'Mathematics', 'History']",
    gpa: 3.97,
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    name: 'Student 969',
    age: 24,
    courses: "['History', 'Mathematics', 'Physics', 'English']",
    gpa: 3.71,
    blood_group: 'B+',
    is_hotel_resident: true
  }
]
db> |

```

If we want 10 top students result with condition id=0, and limit =5:-

```
db.students.find({}, {_id:0}).sort({_id:-1}).limit(5);
```

```
db> db.students.find({}, {_id:0}).sort({_id:-1}).limit(5);
[
  {
    name: 'Student 933',
    age: 18,
    courses: ["Mathematics", "English", "Physics", "History"],
    gpa: 3.84,
    home_city: 'City 10',
    blood_group: 'B-',
    is_hotel_resident: true
  },
  {
    name: 'Student 831',
    age: 20,
    courses: ["Mathematics", "Computer Science"],
    gpa: 3.49,
    home_city: 'City 3',
    blood_group: 'AB+',
    is_hotel_resident: true
  },
  {
    name: 'Student 143',
    age: 21,
    courses: ["Mathematics", "Computer Science", "English", "History"],
    gpa: 2.78,
    home_city: 'City 4',
    blood_group: 'O-',
    is_hotel_resident: true
  },
  {
    name: 'Student 718',
    age: 21,
    courses: ["Computer Science", "English"],
    gpa: 2.75,
    home_city: 'City 5',
    blood_group: 'O-',
    is_hotel_resident: true
  },
  {
    name: 'Student 839',
```

SELECTORS:-

In MongoDB, selectors are query expressions used to specify criteria for selecting documents within a collection. They allow you to define conditions that documents must meet to be included in the query results, using key-value pairs and various operators. For example, a selector like `{ age: { \$gt: 18 } }` finds all documents where the `age` field is greater than 18.

COMPARISON gt lt:

```

db> db.students.find({age:{$gt:20}});
[
  {
    _id: ObjectId('6649bb89b51b15a423b44ad0'),
    name: 'Student 346',
    age: 25,
    courses: "['Mathematics', 'History', 'English']",
    gpa: 3.31,
    home_city: 'City 8',
    blood_group: 'O-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6649bb89b51b15a423b44ad1'),
    name: 'Student 930',
    age: 25,
    courses: "['English', 'Computer Science', 'Mathematics', 'History']",
    gpa: 3.63,
    home_city: 'City 3',
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6649bb89b51b15a423b44ad2'),
    name: 'Student 305',
    age: 24,
    courses: "['History', 'Physics', 'Computer Science', 'Mathematics']",
    gpa: 3.4,
    home_city: 'City 6',
    blood_group: 'O+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6649bb89b51b15a423b44ad5'),
    name: 'Student 440',
    age: 21,
    courses: "['History', 'Physics', 'Computer Science']",
    gpa: 2.56,
    home_city: 'City 10',
    blood_group: 'O-',
    is_hotel_resident: true
  }
]

```

Greater Than (\$gt):-

- **Functionality:** The \$gt operator selects documents where the value of a specified field is greater than (but not equal to) a given value.
- **Example:** db. collection. find({ field: { \$gt: value } }) selects documents where the field value is greater than value.
- **Use Cases:** Filtering documents with values exceeding a certain threshold, selecting the latest entries based on a timestamp, or retrieving documents with numerical values above a specific limit.

Less Than (\$lt):-

- **Functionality:** The \$lt operator selects documents where the value of a specified field is less than (but not equal to) a given value.
- **Example:** db. collection. find({ field: { \$lt: value } }) selects documents where the field value is less than value.
- **Use Cases:** Filtering documents with values below a certain threshold, selecting older entries based on a timestamp, or retrieving documents with numerical values below a specific limit.

Ex:

```
db.students.find({age:{ $gt:20 } })
```

```
db> db.students.find({age:{ $gt:20 }});
[
  {
    _id: ObjectId('665689bde575a33e43e69479'),
    name: 'Student 328',
    age: 21,
    courses: ["Physics", "Computer Science", "English"],
    gpa: 3.42,
    home_city: 'City 2',
    blood_group: 'AB-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('665689bde575a33e43e6947c'),
    name: 'Student 468',
    age: 21,
    courses: ["Computer Science", "Physics", "Mathematics", "History"],
    gpa: 3.97,
    blood_group: 'A-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('665689bde575a33e43e6947d'),
    name: 'Student 584',
    age: 21,
    courses: ["Physics", "Computer Science", "English", "Mathematics"],
    gpa: 2.92,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('665689bde575a33e43e6947f'),
    name: 'Student 915',
    age: 22,
    courses: ["Computer Science", "History", "Physics", "English"],
    gpa: 3.37,
    blood_group: 'AB+',
    is_hotel_resident: true
  }
]
```

This query gives the students collection with age is greater than 20.

AND OPERATOR:-

```
db> db.students.find({
... $and:[
... {home_city:"City 2"},
... {blood_group:"B+"}
... ]
... });
[
  {
    _id: ObjectId('6649bb89b51b15a423b44ae5'),
    name: 'Student 504',
    age: 21,
    courses: ["Physics", "Computer Science", "English", "Mathematics"],
    gpa: 2.92,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6649bb89b51b15a423b44c93'),
    name: 'Student 872',
    age: 24,
    courses: ["English", "Mathematics", "History"],
    gpa: 3.36,
    home_city: 'City 2',
    blood_group: 'B+',
    is_hotel_resident: true
  }
]
db>
```

The MongoDB query uses the `find` method with an `\$and` operator to retrieve documents from the `students` collection where both conditions `home_city` equal to "City 2" and `blood_group` equal to "B+" are met. The query returns documents that match these criteria, displaying the `_id`, `name`, `age`, `courses`, `gpa`, `home_city`, `blood_group`, and

`is_hotel_resident` fields for the matching students. In this case, the result includes two students who live in "City 2" and have a blood group of "B+", providing a way to filter and obtain specific data based on multiple conditions.

OR OPERATOR:-

```
db> db.students.find({
...   $or:[
...     {is_hotel_resident:true},
...     {gpa:{$lt:3.0}}
...   ]
... })
[
  {
    _id: ObjectId('6649bb89b51b15a423b44acd'),
    name: 'Student 948',
    age: 19,
    courses: ['English', 'Computer Science', 'Physics', 'Mathematics'],
    gpa: 3.44,
    home_city: 'City 2',
    blood_group: 'O+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6649bb89b51b15a423b44ace'),
    name: 'Student 157',
    age: 20,
    courses: ['Physics', 'English'],
    gpa: 2.77,
    home_city: 'City 4',
    blood_group: 'O-',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6649bb89b51b15a423b44acf'),
    name: 'Student 316',
    age: 20,
    courses: ['Physics', 'Computer Science', 'Mathematics', 'History'],
    gpa: 2.82,
    blood_group: 'B+',
    is_hotel_resident: true
  }
]
```

The result includes multiple students with various attributes such as `_id`, `name`, `age`, `courses`, `gpa`, `home_city`, `blood_group`, and `is_hotel_resident`. The students in the result set either reside in the hotel or have a GPA below 3.0, indicating that the query successfully combined the two conditions using the `\$or` operator to provide a list of students meeting at least one of the specified criteria.

LET'S TAKE NEW DATASET:-

New students_permission dataset

Explanation:-Collection name: students_permission

name: Student's name (string)

age: Student's age (number)

GEOSPATIAL:-

MongoDB, geospatial refers to the capability to store and query data based on its geographic location. MongoDB supports various geospatial queries, including finding points within a specified distance of a location, finding objects within a specified polygon, and finding the nearest objects to a location. Geospatial indexes can be created to efficiently perform these types of queries on geospatial data.

```
_id: 1
name : "Coffee Shop A"
▼ location : Object
  type : "Point"
  ► coordinates : Array (2)
```


GEOSPATIAL QUERY:-

In MongoDB, a geospatial query is used to retrieve documents based on their geographical location. These queries utilize special operators like \$geoNear, \$geoWithin, and \$near to find documents near a specific point, within a specified area, or based on proximity. Geospatial queries are particularly useful for applications that require location-based searches, such as mapping or location-based services.

```
db> db.locations.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: 3,
    name: 'Park E',
    location: { type: 'Point', coordinates: [ -74.006, 40.705 ] }
  }
]
db>
```

DATATYPES:-

In MongoDB, the data types include String, Number, Boolean, Object, Array, Null, and Date, among others.

Point:- A point is a GeoJSON object representing a single geographic coordinate.

LineString:- A LineString is a GeoJSON object representing a sequence of connected line segments.

Polygon:- A Polygon is a GeoJSON object representing a closed, two-dimensional shape with three or more vertices.

Data types and Operations:-

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