Assignment No. 2

Aim: Implement MongoDB Clauses on College/Airline database schema.

Pre-Requisites: DBMS, MongoDB

Objective: The objective of implementing MongoDB clauses on a college database schema is to efficiently query and manipulate data within the database. MongoDB provides powerful query and aggregation capabilities through various clauses that allow for flexible and precise retrieval of information. The goal is to leverage these clauses to perform operations like filtering, sorting, aggregating, and updating data in the College database schema.

Outcomes:

The expected outcome is a well-optimized and responsive system that enables effective data retrieval and manipulation based on specific criteria. By using MongoDB clauses on the College database schema, we aim to achieve the following outcomes:

Data Retrieval:

Data Filtering:

Data Aggregation:

Data Sorting:

Theory:

Query with the $eq Operator:

Objective: Retrieve students enrolled in a specific course.

Example:

db.Students.find({ courseId: { $eq: ObjectId("courseIdHere") } });

Theory: The $eq operator matches documents where the value of the specified field is equal to the provided value.

2. Range Queries with $gt and $lt Operators:

Objective: Retrieve students older than 20 but younger than 25.

Example:

db.Students.find({ age: { $gt: 20, $lt: 25 } });

Theory: The $gt and $lt operators are used to perform range queries, allowing the retrieval of documents with values greater than or less than the specified thresholds.

3. Sorting Documents with sort Method:

Objective: Get a list of courses sorted by the number of enrolled students.

Example:

db.Courses.find().sort({ studentsEnrolled: -1 });

Theory: The sort method allows the sorting of documents based on the specified field and order. In this case, it sorts courses in descending order of the number of enrolled students.

4. Aggregation with $group Stage:

Objective: Calculate the average age of students in each department.

Example:

```
db.Students.aggregate([
  { $group: { _id: "$departmentId", avgAge: { $avg: "$age" } } }
]);
```

Theory: The $group stage in the aggregation pipeline groups documents by a specified field and calculates aggregate values, such as the average age in this case.

5. $in Operator for Matching Multiple Values:

Objective: Retrieve students enrolled in Computer Science or Mathematics courses.

Example:

```
db.Students.find({ courseId: { $in: [ObjectId("CS101"), ObjectId("MATH202")] } });
```

Theory: The $in operator matches documents where the value of a field equals any value in the specified array.

Conclusion:

Hence, we have successfully studied Hospital/Company database schema and implement all MongoDB methods on it.

Frequently Asked Questions:

1. What is the purpose of the $eq operator in MongoDB?

2. How can I perform range queries in MongoDB, and which operators are commonly used for this purpose?

3. How can I calculate aggregate values using MongoDB clauses?

4. Can I use multiple clauses in a single MongoDB query?

5. What is the purpose of the $exists operator in MongoDB?

6. How can I handle complex queries and transformations in MongoDB?