Practical No:02

- Aim :- Create a database and collection using MongoDB environment. For example, a document collection meant for analyzing. Restaraunt records can have fields like rest-id, restaurant name, customer name, locality, date, cusiine, grade, comment etc
Create database using Insert, Update, Index, Upserts, delete.

- Theory :-
MongoDB is a NoSQL database that stores data & created automatically when we insert the first document.

Database and Collection Creation
In MongoDB data is stored data in BSON format.

→ Create Database :- "use" command used to create database / switch databases.
eg
```
use dataDB
```

→ Create Collection :- Collection is created implicitely when we insert the data

Insert Operation
Used to add new document to the collection
→ Insert Single Document →
• InsertOne used to insert one document

eg db.collection.insertOne({
 name: "Raj",
 age: 12 });

Insert Multiple Document :-

eg

```
db.collection.insertMany([
    {name : "raj",
    age: 12 },
    { name : "Ajay",
    age: 13 }
    ]);
```

- Read Operations :- Methods like find()
used to retrive data.

eg

```
db.collection.find();
```

return all data in a collection

- Additionally, we can provide filters in parameters to retrieve specific ~~records~~ documents.

- Update Operations :-
The "UpdateOne" and "updateMany" methods modify existing documents Also modifies existing documents based on filters.

eg

```
db.collection.updateOne(
    { _id: "120"
    }, {
    $set: {grade: "A"}
    });
```

updates the grade of document with id 120

- Upsert Operation :- An upsert performs an update if a documents exists or Insert new document if it does not.

eg

```
db.collectionName.updateOne(
 {email: "test@gmail.com"},
 {
  $set :{name: "John", age: 30}
 },
 {upsert : true}
)
```

- Delete Operation :- The "deleteOne" and "deleteMany" methods remove documents

eg

```
db.collectionName.deleteOne({id = 1002});
```

- Index :- An index in MongoDB improves query Performance by allowing faster data retrieval, similar to a index in a book.

eg db.collectionName.createIndex({email:1})
i.e 1 means ascending Order (-1 descending) helps speed up queries.

- Result :- Successfully implemented Insert, Update, Upsert, Index & delete commands.

P	T	D	K	Total
3M	3M	3M	6M	15M
3	2	3	5	13

[illegible]
13/01/25

Practical No 03

Aim:- Experiment with MongoDB and explain comparision & logical Query Operator $gt, $gte, $lt, $lte, $nin, $ne, $and, $or, $not

Comparision Operator in MongoDB.
Used to compare the value of a field to a specific value.

→ $gt (greater than)
eg {"age":{$gt:30}}

→ $gte (greater than or equal to)
eg {"age": {$gte: 30}}

→ $lt (Less Than)
eg {"age": {$lt: 30}}

→ $lte (less than or equal)
eg {"age": {$lte: 30}}

→ {$ne (not equal to)
eg {"age": {$ne: 30}}

→ $in (in an array)
{"age": {$in: [12, 30, 40]}}

→ $nin (not in an array)
{"age": {$nin: [12, 30, 40]}}

- Logical Operators In MongoDB.

The Queries combine multiple conditions in a query.

→ $AND (Logical AND)

eg { $AND : [{"age":{$gte:20}, {"age" :{$lte : 40}]}

→ $OR (Logical OR)

{ $OR :[{"age":{$gte:30}, {"age" :{$lte:20}]}

→ $NOT (Logical NOT)

~~{ $NOT~~ : '

db.collection.find {age : {$not : {$in:[20,25,30]}}})

- Result:-

Hence we succesfully perfomed queries with logical & comparison operator.

P	T	D	K	Total
3M	3M	3M	6M	15M
3	3	3	5	14
[illegible]				

[illegible]
20/01/25