# Experiment 10

**Student Name:** Rishi Jain

**UID:** 23BAI70569

**Branch:** BE-AIT-CSE

**Section/Group:**23AML-1(A)

**Semester:**5th

**Date of Performance:**29 Oct,2025

**Subject Name:**ADBMS

**Subject Code:**23CSP-333

## Aim:

To study and perform basic CRUD (Create, Read, Update, Delete) operations in **MongoDB**, a NoSQL document-based database, and understand its key commands, such as creating databases, collections, inserting, updating, deleting records, and grouping data using aggregation.

## Objective:

- To understand the concept of NoSQL and document-based databases.
- To learn how to create and manage databases and collections in MongoDB.
- To perform CRUD operations — Create, Read, Update, and Delete — on MongoDB collections.
- To use operators like $push, $pull, $unset, and $upsert for document updates.
- To execute conditional queries and retrieve nested document data using find().
- To perform grouping and aggregation operations using $sum and other aggregation operators.
- To compare SQL and NoSQL databases in terms of structure, performance, and use cases.
- To apply MongoDB commands on a practical dataset (e.g., car dealership data) for hands-on understanding.

## Theory:

MongoDB is a NoSQL document-oriented database that stores data in JSON-like documents instead of tables and rows. It is designed for handling large, unstructured, or semi-structured data with flexibility and scalability.

A **MongoDB database** contains **collections**, and each collection holds multiple **documents** made up of key-value pairs. Unlike SQL databases, MongoDB allows documents in the same collection to have different structures, making it ideal for applications with changing data requirements.

**Basic Operations:**

- **Create:** use db_name, db.createCollection()
- **Insert:** insertOne(), insertMany()
- **Read:** find(), findOne()
- **Update:** updateOne(), updateMany() with $set, $push, $pull, $upsert
- **Delete:** deleteOne(), deleteMany()

MongoDB also supports **aggregation** (e.g., $sum, $group) for analyzing and summarizing data.

**Advantages:**

- Flexible and schema-less structure
- High performance and scalability
- Easy integration with modern applications

In short, MongoDB is preferred for applications requiring **speed, flexibility, and scalability**, while SQL remains suited for **structured and relational data**.

## Procedure:

- Start the MongoDB server and open the Mongo shell or MongoDB Compass.
- Create or switch to a new database.
- Create a new collection to store data.
- Insert one or more documents into the collection.
- Retrieve data from the collection using read operations.
- Apply filters or conditions to view specific data.
- Update existing records in the collection as required.
- Delete one or multiple records from the collection.
- Perform grouping or aggregation operations to analyze data.
- Drop collections or the entire database if no longer needed.

Code:

```
db.createCollection("cars")

db.cars.insertOne({
  "maker": "Tata",
  "model": "Nexon",
  "fuel_type": "Petrol",
  "transmission": "Automatic",
  "engine": {
    "type": "Turbocharged",
    "cc": 1199,
    "torque": "170 Nm"
  },
  "features": [
    "Touchscreen",
    "Reverse Camera",
    "Bluetooth Connectivity"
  ],
  "sunroof": false,
  "airbags": 2
})

db.cars.find().pretty()

db.cars.insertMany([{"maker":"Tata","model":"Nexon","fuel_type":"Petrol","transmission":"Automatic","engine":{"type":"Turbocharged","cc":1199,"torque":"170 Nm"},"features":["Touchscreen","Reverse Camera","Bluetooth Connectivity"],"sunroof":false,"airbags":2},{"maker":"Hyundai","model":"Creta","fuel_type":"Diesel","transmission":"Manual","engine":{"type":"CRDi","cc":1493,"torque":"250 Nm"},"features":["Sunroof","ABS","Touchscreen"],"sunroof":true,"airbags":6},{"maker":"Maruti","model":"Swift","fuel_type":"Petrol","transmission":"Manual","engine":{"type":"Naturally Aspirated","cc":1197,"torque":"113 Nm"},"features":["Bluetooth Connectivity","Power Windows"],"sunroof":false,"airbags":2}])
```

```
db.cars.find()
db.cars.findOne()
db.cars.find({}, { model: 1, _id: 0 })
db.cars.find({ "fuel_type": "Petrol" })
db.cars.updateOne(
  { model: "Nexon" },
  { $set: { color: "Red" } }
)
db.cars.find({ model: "Nexon" }).pretty()
db.cars.updateOne(
  { model: "Nexon" },
  { $push: { features: "Heated Seats" } }
)
db.cars.find({ model: "Nexon" }).pretty()
db.cars.updateOne(
  { model: "Nexon" },
  { $pull: { features: "Heated Seats" } }
)
db.cars.find({ model: "Nexon" }).pretty()
db.cars.updateMany(
  { fuel_type: "Diesel" },
  { $set: { alloys: "yes" } }
)
db.cars.find({ fuel_type: "Diesel" }).pretty()
db.cars.updateOne(
  { model: "Nexon" },
```

```
  { $push: { features: { $each: ["Wireless charging", "Voice Control"] } } }
)
db.cars.find({ model: "Nexon" }).pretty()
db.cars.updateOne(
  { model: "Nexon" },
  { $unset: { color: "" } }
)
db.cars.find({ model: "Nexon" }).pretty()
db.cars.aggregate([
  {
    $group: {
      _id: "$maker",
      TotalCars: { $sum: 1 }
    }
  }
])
db.cars.aggregate([
  {
    $group: {
      _id: "$fuel_type",
      TotalCars: { $sum: 1 }
    }
  }
])
```

Output:

```
test> db.createCollection("cars")
{ ok: 1 }
test> db.cars
test.cars
test> {
...   "maker": "Tata",
...   "model": "Nexon",
...   "fuel_type": "Petrol",
...   "transmission": "Automatic",
...   "engine": {
...   "type": "Turbocharged",
...   "cc": 1199,
...   "torque": "170 Nm"
...   },
...   "features": [
...   "Touchscreen",
...   "Reverse Camera",
...   "Bluetooth Connectivity"
...   ],
...   "sunroof": false,
...   "airbags": 2
...   }
...
...
```

```
test> db.cars.insertOne(
... {
...  "maker": "Tata",
...  "model": "Nexon",
...  "fuel_type": "Petrol",
...  "transmission": "Automatic",
...  "engine": {
...  "type": "Turbocharged",
...  "cc": 1199,
...  "torque": "170 Nm"
...  },
...  "features": [
...  "Touchscreen",
...  "Reverse Camera",
...  "Bluetooth Connectivity"
...  ],
...  "sunroof": false,
...  "airbags": 2
...  })
{
  acknowledged: true,
  insertedId: ObjectId('690a31c2d72ea166af63b112')
}
```

```
test> db.cars.insertMany(
... [
...  {
...  "maker": "Hyundai",
...  "model": "Creta",
...  "fuel_type": "Diesel",
...  "transmission": "Manual",
...  "engine": {
...  "type": "Naturally Aspirated",
...  "cc": 1493,
...  "torque": "250 Nm"
...  },
...  "features": [
...  "Sunroof",
...  "Leather Seats",
...  "Wireless Charging",
... "Ventilated Seats",
... "Bluetooth"
...  ],
...  "sunroof": true,
...  "airbags": 6
...  },
...  {
...  "maker": "Maruti Suzuki",
...  "model": "Baleno",
...  "fuel_type": "Petrol",
...  "transmission": "Automatic",
...  "engine": {
...  "type": "Naturally Aspirated",
...  "cc": 1197,
...  "torque": "113 Nm"
...  },
...  "features": [
...  "Projector Headlamps",
...  "Apple CarPlay",
...  "ABS"
...  ],
...  "sunroof": false,
...  "airbags": 2
```

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('690a31e6d72ea166af63b113'),
    '1': ObjectId('690a31e6d72ea166af63b114'),
    '2': ObjectId('690a31e6d72ea166af63b115'),
    '3': ObjectId('690a31e6d72ea166af63b116')
  }
}
```

```
test> db.cars.find()
[
  {
    _id: ObjectId('690a31c2d72ea166af63b112'),
    maker: 'Tata',
    model: 'Nexon',
    fuel_type: 'Petrol',
    transmission: 'Automatic',
    engine: { type: 'Turbocharged', cc: 1199, torque: '170 Nm' },
    features: [ 'Touchscreen', 'Reverse Camera', 'Bluetooth Connectivity' ],
    sunroof: false,
    airbags: 2
  },
  {
    _id: ObjectId('690a31e6d72ea166af63b113'),
    maker: 'Hyundai',
    model: 'Creta',
    fuel_type: 'Diesel',
    transmission: 'Manual',
    engine: { type: 'Naturally Aspirated', cc: 1493, torque: '250 Nm' },
    features: [
      'Sunroof',
      'Leather Seats',
      'Wireless Charging',
      'Ventilated Seats',
      'Bluetooth'
    ],
    sunroof: true,
    airbags: 6
  },
```

```
test> db.cars.findOne()
{
  _id: ObjectId('690a31c2d72ea166af63b112'),
  maker: 'Tata',
  model: 'Nexon',
  fuel_type: 'Petrol',
  transmission: 'Automatic',
  engine: { type: 'Turbocharged', cc: 1199, torque: '170 Nm' },
  features: [ 'Touchscreen', 'Reverse Camera', 'Bluetooth Connectivity' ],
  sunroof: false,
  airbags: 2
}
test> db.cars.find({},{model:1})
[
  { _id: ObjectId('690a31c2d72ea166af63b112'), model: 'Nexon' },
  { _id: ObjectId('690a31e6d72ea166af63b113'), model: 'Creta' },
  { _id: ObjectId('690a31e6d72ea166af63b114'), model: 'Baleno' },
  { _id: ObjectId('690a31e6d72ea166af63b115'), model: 'XUV500' },
  { _id: ObjectId('690a31e6d72ea166af63b116'), model: 'City' }
]
test> db.cars.find({},{model:1,_id:0})
[
  { model: 'Nexon' },
  { model: 'Creta' },
  { model: 'Baleno' },
  { model: 'XUV500' },
  { model: 'City' }
]
test> db.cars.find({},{model:1,maker:1,_id:0})
[
  { maker: 'Tata', model: 'Nexon' },
  { maker: 'Hyundai', model: 'Creta' },
  { maker: 'Maruti Suzuki', model: 'Baleno' },
  { maker: 'Mahindra', model: 'XUV500' },
  { maker: 'Honda', model: 'City' }
]
```

```
test> db.cars.find({"fuel_type":"Petrol"})
[
  {
    _id: ObjectId('690a31c2d72ea166af63b112'),
    maker: 'Tata',
    model: 'Nexon',
    fuel_type: 'Petrol',
    transmission: 'Automatic',
    engine: { type: 'Turbocharged', cc: 1199, torque: '170 Nm' },
    features: [ 'Touchscreen', 'Reverse Camera', 'Bluetooth Connectivity' ],
    sunroof: false,
    airbags: 2
  },
  {
    _id: ObjectId('690a31e6d72ea166af63b114'),
    maker: 'Maruti Suzuki',
    model: 'Baleno',
    fuel_type: 'Petrol',
    transmission: 'Automatic',
    engine: { type: 'Naturally Aspirated', cc: 1197, torque: '113 Nm' },
    features: [ 'Projector Headlamps', 'Apple CarPlay', 'ABS' ],
    sunroof: false,
    airbags: 2
  },
  {
    _id: ObjectId('690a31e6d72ea166af63b116'),
    maker: 'Honda',
    model: 'City',
    fuel_type: 'Petrol',
    transmission: 'Automatic',
    engine: { type: 'Naturally Aspirated', cc: 1498, torque: '145 Nm' },
    features: [ 'Keyless Entry', 'Auto AC', 'Multi-angle Rearview Camera' ],
    sunroof: false,
    airbags: 4
  }
]
```

```
test> db.cars.aggregate([
...   {
...     $group: {
...       _id: "$maker",
...       TotalCars: { $sum: 1 }
...     }
...   }
... ])
...
[
  { _id: 'Maruti Suzuki', TotalCars: 1 },
  { _id: 'Tata', TotalCars: 1 },
  { _id: 'Hyundai', TotalCars: 1 },
  { _id: 'Honda', TotalCars: 1 },
  { _id: 'Mahindra', TotalCars: 1 }
]
test> db.cars.aggregate([
...   {
...     $group: {
...       _id: "$fuel_type",
...       TotalCars: { $sum: 1 }
...     }
...   }
... ])
```

## 7. Learning Outcomes:

- Understand the concept of **NoSQL databases** and their differences from SQL databases.
- Gain knowledge of **MongoDB architecture**, including databases, collections, and documents.
- Learn to perform **CRUD operations** (Create, Read, Update, Delete) using MongoDB commands.
- Use operators such as `$set`, `$push`, `$pull`, `$unset`, and `$upsert` for data manipulation.
- Apply **aggregation and grouping operations** using operators like `$sum`, `$avg`, `$min`, and `$max`.
- Develop the ability to query and analyze data efficiently in MongoDB.
- Compare the advantages and use cases of **SQL vs NoSQL** database models.
- Gain practical experience handling real-world data (e.g., car dealership dataset) using MongoDB.