



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment - 10

Student Name: Nishant Rejra

UID: 23BCS12961

Branch: BE-CSE

Section/Group: KRG-3B

Semester: 5th

Date of Performance: 31 Oct, 2025

Subject Name: ADBMS

Subject Code: 23CSP-333

1. 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.

2. 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.

3. 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**.

4. 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.

5. Code:

```
C:\Users\Armaa>mongosh
Current Mongosh Log ID: 690a30cdd72ea166af63b111
Connecting to:
    mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.9
Using MongoDB:      8.2.1
Using Mongosh:     2.5.9
```

For mongosh info see: <https://www.mongodb.com/docs/mongodb-shell/>

To help improve our products, anonymous usage data is collected and sent to MongoDB periodically
(<https://www.mongodb.com/legal/privacy-policy>).
You can opt-out by running the disableTelemetry() command.

```
-----
The server generated these startup warnings when booting
2025-11-04T22:18:13.154+05:30: Access control is not enabled for the database. Read and write access to data and
configuration is unrestricted
-----
```

```
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()
MongoshInvalidInputError: [COMMON-10001] Missing required argument at position 0 (Collection.insertOne)
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
... },
... {
...   "maker": "Mahindra",
...   "model": "XUV500",
...   "fuel_type": "Diesel",
...   "transmission": "Manual",
...   "engine": {
...     "type": "Turbocharged",
...     "cc": 2179,
...     "torque": "360 Nm"
...   },
...   "features": [
...     "All-Wheel Drive",
...     "Navigation System",
...     "Cruise Control"
...   ],
...   "sunroof": true,
...   "airbags": 6
... },
... {
...   "maker": "Honda",
...   "model": "City",
...   "fuel_type": "Petrol",
...   "transmission": "Automatic",
...   "engine": {
...     "type": "Naturally Aspirated",
...     "cc": 1330,
...     "torque": "100 Nm"
...   },
...   "features": [
...     "ABS",
...     "Traction Control System",
...     "Power Steering"
...   ],
...   "sunroof": false,
...   "airbags": 2
... }
]
```

```
... "transmission": "Automatic",
... "engine": {
... "type": "Naturally Aspirated",
... "cc": 1498,
... "torque": "145 Nm"
... },
... "features": [
... "Keyless Entry",
... "Auto AC",
... "Multi-angle Rearview Camera"
... ],
... "sunroof": false,
... "airbags": 4
... }
... ]
{
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
},
{
_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('690a31e6d72ea166af63b115'),
maker: 'Mahindra',
```

```

model: 'XUV500',
fuel_type: 'Diesel',
transmission: 'Manual',
engine: { type: 'Turbocharged', cc: 2179, torque: '360 Nm' },
features: [ 'All-Wheel Drive', 'Navigation System', 'Cruise Control' ],
sunroof: true,
airbags: 6
},
{
_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.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")
Uncaught:
SyntaxError: Unexpected token, expected "," (1:22)

> 1 | db.cars.find(fuel_type:"Petrol")
|          ^
2 |

test> db.cars.find("fuel_type":"Petrol")
Uncaught:
SyntaxError: Unexpected token, expected "," (1:24)

```

```

> 1 | db.cars.find("fuel_type":"Petrol")
|   ^
2 |

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.updateOne(
...  { model: "Nexon" },
...  { $set: { color: "Red" } }
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
test> db.cars.updateOne(
...  { model: "Nexon" },
...  { $push: { features: "Heated Seats" } }
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
test> db.cars.updateOne(
...  { model: "Nexon" },
...  { $pull: { features: "Heated Seats" } }
...

```

```
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
test> db.cars.updateMany(
... { fuel_type: "Diesel" },
... { $set: { alloys: "yes" } }
...
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
test> db.cars.updateOne(
... { model: "Creta" },
... { $set: { "engine.torque": "270 Nm" } }
...
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
test> db.cars.updateOne(
... { model: "Nexon" },
... { $push: { features: "Heated Steering Wheel" } }
...
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
test> db.cars.updateOne(
... { model: "Nexon" },
... { $pull: { features: "Bluetooth Connectivity" } }
...
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
test> db.cars.updateOne(
... { model: "Nexon" },
... { $push: { features: { $each: ["Wireless charging", "Voice Control"] } } }
...
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
test> db.cars.updateOne(  
...   { model: "Nexon" },  
...   { $unset: { color: "" } }  
... )  
...  
{  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 1,  
  modifiedCount: 1,  
  upsertedCount: 0  
}  
test> db.cars.updateMany(  
...   {},  
...   { $set: { color: "Blue" } }  
... )  
...  
{  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 5,  
  modifiedCount: 5,  
  upsertedCount: 0  
}  
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 }  
...     }  
...   }  
... ])  
...  
[ { _id: 'Petrol', TotalCars: 3 }, { _id: 'Diesel', TotalCars: 2 } ]  
test> db.collection.aggregate([  
...   {  
...     $group: {  
...       _id: "$category",  
...       totalAmount: { $sum: "$amount" },  
...       averageAmount: { $avg: "$amount" },  
...       minAmount: { $min: "$amount" },  
...       maxAmount: { $max: "$amount" },  
...       amountsList: { $push: "$amount" },  
...       uniqueAmounts: { $addToSet: "$amount" }  
...     }  
...   }  
... ])  
...  
test>
```

6. 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.