EXPERIMENT- 10

Student Name: Ashutosh Yadav **UID: 23BCS11023**

Branch: BE-CSE **Section/Group: KRG 1-B**

Semester: 05 **Date of Performance: 28/10/25**

Subject Name: ADBMS **Subject Code: 23CSP-333**

1. Aim:
To perform CRUD operations and aggregation using **MongoDB**, a NoSQL document-based database.

2. Objective:
- Learn creation of databases and collections in MongoDB.
- Execute Insert, Read, Update, and Delete operations.

3. Tools / Software
- MongoDB
- Mongo Shell
- Sample Dataset: Car Dealership Data

4. Program:

```
C:\Users\ashue>mongosh
Current Mongosh Log ID: 69098359d6b8a1508663b111
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.co
m/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

------
   The server generated these startup warnings when booting
   2025-11-04T01:20:26.256+05:30: Access control is not enabled for the database. Read and write access to data and conf
iguration is unrestricted
```

--Create database car_dealership

```
test> use car_dealership
switched to db car_dealership
```

--Creating a collection cars

```
car_dealership> db.createCollection("cars")
{ ok: 1 }
```

INSERTION OPERATION:

```
db.cars.insertMany([
  { maker: "Hyundai", model: "i20", fuel_type: "Petrol" },
  { maker: "Tata", model: "Nexon", fuel_type: "Diesel" },
  { maker: "Kia", model: "Seltos", fuel_type: "Petrol" },
  { maker: "Maruti", model: "Swift", fuel_type: "CNG" }
])
```

```
car_dealership> db.cars.insertMany([ { maker: "Hyundai", model: "i20", fuel_type: "Petrol" }, { maker: "Tata", model: "
exon", fuel_type: "Diesel" }, { maker: "Kia", model: "Seltos", fuel_type: "Petrol" }, { maker: "Maruti", model: "Swift"
 fuel_type: "CNG" } ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6909865ce69b03cda463b112'),
    '1': ObjectId('6909865ce69b03cda463b113'),
    '2': ObjectId('6909865ce69b03cda463b114'),
    '3': ObjectId('6909865ce69b03cda463b115')
  }
}
```

READ OPERATION:

```
db.cars.find()
db.cars.find({maker:'Maruti'},{fuel_type:0})
```

```
car_dealership> db.cars.find({maker:'Maruti'},{fuel_type:0});
[
  {
    _id: ObjectId('6909865ce69b03cda463b115'),
    maker: 'Maruti',
    model: 'Swift'
  }
]
```

UPDATE OPERATION:

```
db.cars.updateOne({ model: "i20" }, { $set: { fuel_type: "Hybrid" } })
db.cars.updateMany({}, { $set: { color: "White" } })
db.cars.updateOne({ model: "Nexon" }, { $push: { features: "Sunroof" } })
```

```
car_dealership> db.cars.updateOne({ model: "i20" }, { $set: { fuel_type: "Hybrid" } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
car_dealership> db.cars.updateMany({}, { $set: { color: "White" } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 4,
  modifiedCount: 4,
  upsertedCount: 0
}
car_dealership> db.cars.updateMany({}, { $set: { color: "White" } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 4,
  modifiedCount: 0,
  upsertedCount: 0
}
```

```
car_dealership> db.cars.find()
[
  {
    _id: ObjectId('6909865ce69b03cda463b112'),
    maker: 'Hyundai',
    model: 'i20',
    fuel_type: 'Hybrid',
    color: 'White'
  },
  {
    _id: ObjectId('6909865ce69b03cda463b113'),
    maker: 'Tata',
    model: 'Nexon',
    fuel_type: 'Diesel',
    color: 'White'
  },
  {
    _id: ObjectId('6909865ce69b03cda463b114'),
    maker: 'Kia',
    model: 'Seltos',
    fuel_type: 'Petrol',
    color: 'White'
  },
  {
    _id: ObjectId('6909865ce69b03cda463b115'),
    maker: 'Maruti',
    model: 'Swift',
    fuel_type: 'CNG',
    color: 'White'
```

DELETE OPERATION:

db.cars.deleteOne({ model: "Nexon" })

```
car_dealership> db.cars.deleteOne({ model: "Nexon" })
{ acknowledged: true, deletedCount: 1 }
car_dealership>
```

AGGREGATION:

db.cars.aggregate([{ $match: { fuel_type: { $in: ["Petrol", "Diesel"] } } }, { $group: { _id: "$maker", carcount: { $sum: 1 } } }])

```
car_dealership> db.cars.aggregate([ { $match: { fuel_type: { $in: ["Petrol", "Diesel"] } } }, { $group: { _id: "$maker"
 carcount: { $sum: 1 } } } ])
[ { _id: 'Kia', carcount: 1 } ]
car_dealership>
```

db.cars.aggregate([{ $group: { _id: "$color", totalCars: { $sum: 1 }, makers: { $addToSet: "$maker" } } }])

```
car_dealership> db.cars.aggregate([ { $group: { _id: "$color", totalCars: { $sum: 1 }, makers: { $addToSet: "$maker" } }
 } ])
[
  {
    _id: 'White',
    totalCars: 3,
    makers: [ 'Maruti', 'Hyundai', 'Kia' ]
  }
]
```

Learning Outcomes:

1. **Implement CRUD operations:**
   Learners will be able to create, read, update, and delete documents in MongoDB collections using appropriate commands and query operators.
2. **Apply aggregation pipelines:**
   Learners will be able to design and execute aggregation pipelines to filter, group, and analyze data for deriving meaningful insights.
3. **Understand data modeling in MongoDB:**
   Learners will be able to structure and organize data in collections using appropriate document models and field types suitable for NoSQL databases.