DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

Experiment 10

Student Name: Sukhjinder Singh
UID: 23BAI70078

Branch: BE-AIT-CSE
Section/Group: 23AIT-KRG-G2

Semester: 5th
Date of Performance: 11th Nov, 2025

Subject Name: ADBMS
Subject Code: 23CSP-333

1. **AIM:** Create a Database on MongoDB.
2. **Tools Used:** MongoDB CLI.
3. **Experiment:** Create a Database in MongoDB.
4. **Solution:**

```
switched to db car_dealership
car_dealership>

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

insertOne:

```
car_dealership> 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
...   }
...
```

insertMany():

```
car_dealership> db.cars.insertMany(
... [
...   {
...   "maker": "Hyundai",
...   "model": "Creta",
...   "fuel_type": "Diesel",
...   "transmission": "Manual",
...   "engine": {
...   "type": "Naturally Aspirated",
...   "cc": 1493,
...   "torque": "250 Nm"
```

Find(): gives all data - depricate

```
car_dealership> db.cars.find()
[
  {
    _id: ObjectId('66b8a525d95b225bbc381167'),
    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('66b8a5b5d95b225bbc381168'),
    maker: 'Hyundai',
    model: 'Creta',
    fuel_type: 'Diesel',
    transmission: 'Manual',
    engine: { type: 'Naturally Aspirated', cc: 1493, torque: '250 Nm' },
    features: [ 'Sunroof', 'Leather Seats', 'Wireless Charging' ],
    sunroof: true,
    airbags: 6
  },
```

FindOne(): First found data from the collection.

```
car_dealership> db.cars.findOne()
{
  _id: ObjectId('66b8a525d95b225bbc381167'),
  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
}
```

Find(): showing specific columns, 1: true, 0: false

```
car_dealership> db.cars.find({},{model:1})
[
  { _id: ObjectId('66b8a525d95b225bbc381167'), model: 'Nexon' },
  { _id: ObjectId('66b8a5b5d95b225bbc381168'), model: 'Creta' },
  { _id: ObjectId('66b8a5b5d95b225bbc381169'), model: 'Baleno' },
  { _id: ObjectId('66b8a5b5d95b225bbc38116a'), model: 'XUV500' },
  { _id: ObjectId('66b8a5b5d95b225bbc38116b'), model: 'City' }
]
car_dealership>
```

```
car_dealership> db.cars.find({},{model:1,_id:0})
[
  { model: 'Nexon' },
  { model: 'Creta' },
  { model: 'Baleno' },
  { model: 'XUV500' },
  { model: 'City' }
]
```

```
car_dealership> 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' }
]
```

UpdateOne:

```
car_dealership> db.cars.updateOne(
... {model:"Nexon"},
... {$set:{color:"Red"}}
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

UpdateMany will do this for every record present inside the collection, while updateOne will only do this for the first matched record only.

```
car_dealership> db.cars.updateMany( { fuel_type: "Diesel" }, { $set: { alloys: "yes" } } )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 1,
  upsertedCount: 0
}
```

DeleteOne(): will delete the first matching record from the document.

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

Grouping

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

Find number of cars for each specific brand: $sum

```
car_dealership> db.cars.aggregate([
...     {$group:{
...         _id:"$maker",
...         TotalCars: {$sum:1}
...     }}
... ])
[
  { _id: 'Mahindra', TotalCars: 1 },
  { _id: 'Hyundai', TotalCars: 4 },
  { _id: 'Maruti Suzuki', TotalCars: 3 },
  { _id: 'Honda', TotalCars: 3 },
  { _id: 'Tata', TotalCars: 3 }
]
```

5. **Learning Outcomes:**
    - Learn't about MongoDB.
    - Learn't how to create a database on MongoDB using it's CLI.
    - Learn't how to perform various operations on MongoDB.