EXPERIMENT - 10

Student Name: Dikshay Sharma   **UID: 23BCS11096**

Branch: BE-CSE   **Section/Group: KRG_1-A**

Semester: 5th   **Date of Performance: 1/11/2025**

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

1. INSERT OPERATION

- 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"
```

2. READ OPERATION

- Find(): gives all data - depricated

```
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' },                I
  { model: 'Baleno' },
  { model: 'XUV500' },
  { model: 'City' }
]
```

```
car_dealership> db.cars.find({},{model:1,maker:1,_id:0})
[
  { maker: 'Tata', model: 'Nexon' },I
  { maker: 'Hyundai', model: 'Creta' },
  { maker: 'Maruti Suzuki', model: 'Baleno' },
  { maker: 'Mahindra', model: 'XUV500' },
  { maker: 'Honda', model: 'City' }
]
```

- **Find function with condition**:

```
car_dealership> db.cars.find({fuel_type:"Petrol"})
```

- Find function for nested document:

```
car_dealership> db.cars.find({"engine.type":"Turbocharged"})
[
  {
    _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('66b8a5b5d95b225bbc38116a'),
    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
  }
]
```

3. UPDATING THE RECORDS

- UpdateOne:

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

- Update Many

```
car_dealership> db.cars.updateOne(
... {model:"Creta"},
... {$set:{"engine.torque":"270 Nm"}}
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

4. Delete Operation

- DeleteOne()

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

5. Groping in MONGO DB

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

- $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 }
]
```

Standardization Across Platforms

- SQL syntax is standardized (ANSI SQL), making it portable across systems like MySQL, PostgreSQL, Oracle, etc.

Easier Data Analysis and Reporting

- SQL databases integrate seamlessly with BI tools, analytics, and reporting systems.

- Excellent for OLAP and data warehousing.