PROJECT TRAINING WORKSHOP

MongoDB

RAAFI
Builds You Best

UNDERSTANDING NoSQL

NoSQL - "Not Only SQL" or "non-relational" database system. NoSQL databases are designed to handle a variety of data types and high volumes of data while offering flexibility, scalability, and performance.

Non-Tabular Data Model	Schema Flexibility	Horizontal Scalability	High Performance
No Complex Joins	No ACID Transactions	Distributed Architectures	

Document Stores

Key-Value Stores

Column-Family Stores

Graph Database

UNDERSTANDING MongoDB

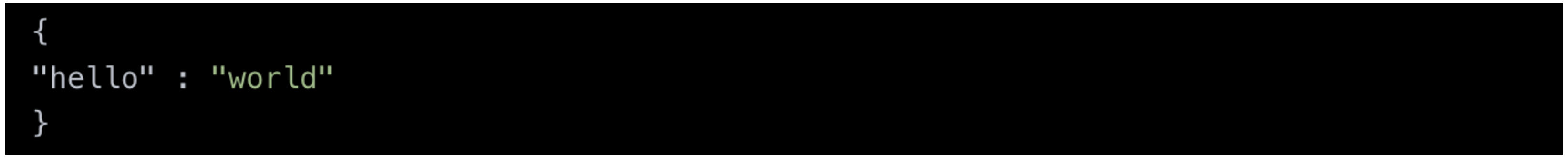
MongoDB, a NoSQL database system, plays several key roles in modern software development and data management

Data Storage	Scalability	High Performance	Replication
Automatic Sharding	Geospatial Capabilities	Aggregation Framework	Ad Hoc Queries
Document-Oriented Storage			

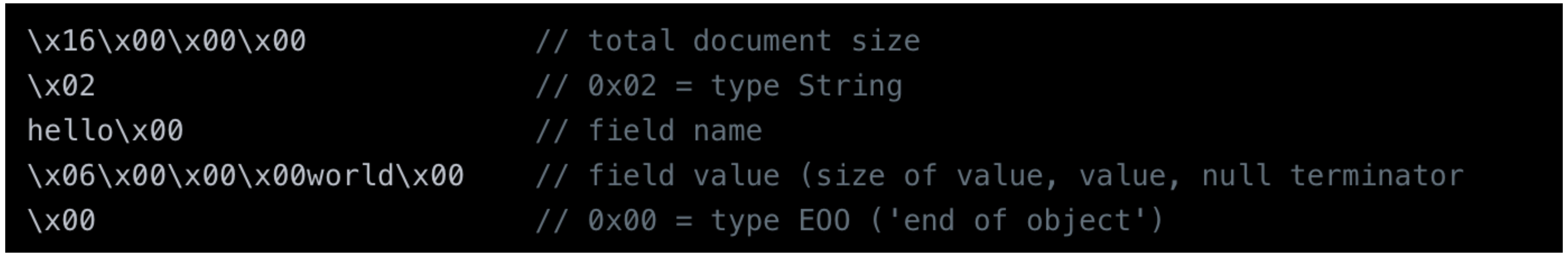
UNDERSTANDING MongoDB

Key Concepts

Document	Collection	Database	Document ID (_id)
Field	Index	Query	Aggregation
Replication	Sharding	Transactions	NoSQL
BSON	Compass (GUI)	Operators	View



```
{
"hello" : "world"
}
```



```
\x16\x00\x00\x00            // total document size
\x02                        // 0x02 = type String
hello\x00                   // field name
\x06\x00\x00\x00world\x00   // field value (size of value, value, null terminator
\x00                        // 0x00 = type EOO ('end of object')
```

UNDERSTANDING MongoDB - CRUD

Create

```
db.createCollection(”students")
```

Insert

```
db.students.insertOne(
{ name: ”Ravi", grade: ”2", section: ”C" rollnumber: 1}
)
```

```
db.students.insertMany(
{ name: ”Ravi", grade: ”2", section: ”C" rollnumber: 1},
{ name: ”Vani", grade: ”1", section: ”B" rollnumber: 2}
)
```

Find

```
db.students.find()
```

```
db.students.findOne()
```

```
db.students.find( {grade: 2} )
```

```
db.students.find({}, {name: 1, grade: 1})
```

```
db.students.find({}, {section: 0})
```

```
db.students.find({}, {title: 1, date: 0})
```

RAAFI
Builds You Best

UNDERSTANDING MongoDB - CRUD

Update

```
db.students.find( { rollnumber: 2 } )

db.students.updateOne( { rollnumber: 2 }, { $set: { grade: “2” } } )

db.students.updateOne(
{ rollnumber: 3 },
{ $set: { name: ”Mark", grade: ”3", section: ”C" rollnumber: 3},
{ upsert: true } )

db.students.updateMany({}, { $inc: { rollnumber: 100 } })
```

Delete

```
db.students.deleteOne({ name: ”Mark" })

db.students.deleteMany({ section: ”C" })
```

UNDERSTANDING MongoDB - OPERATORS

Comparison	Logical	Evaluation	Fields	Arrays
• $eq	• $and	• $regex	• $currentDate	• $addToSet
• $ne	• $or	• $text	• $inc	• $pop
• $gt	• $nor	• $where	• $rename	• $pull
• $gte	• $not		• $set	• $push
• $lt			• $unset	
• $lte				
• $in				

Query Operators

Update Operators

RAAFI
Builds You Best

UNDERSTANDING MongoDB - Aggregations

```
[
  { "_id": 1, "product": "A", "quantity": 5, "revenue": 100 },
  { "_id": 2, "product": "B", "quantity": 3, "revenue": 60 },
  { "_id": 3, "product": "A", "quantity": 2, "revenue": 40 },
  { "_id": 4, "product": "C", "quantity": 4, "revenue": 80 },
  { "_id": 5, "product": "B", "quantity": 6, "revenue": 120 }
]
```

```
db.sales.aggregate([
  {
    $group: {
      _id: "$product",
      totalRevenue: { $sum: "$revenue" }
    }
  }
])
```

```
[
  { "_id": "A", "totalRevenue": 140 },
  { "_id": "B", "totalRevenue": 180 },
  { "_id": "C", "totalRevenue": 80 }
]
```

- Group
- Limit
- Project
- Sort
- Match
- Add Fields
- Count
- Lookup
- Out

RAAFI
Builds You Best

UNDERSTANDING MongoDB - Aggregations

```
[
  { "_id": 1, "product": "A", "quantity": 5, "revenue": 100 },
  { "_id": 2, "product": "B", "quantity": 3, "revenue": 60 },
  { "_id": 3, "product": "A", "quantity": 2, "revenue": 40 },
  { "_id": 4, "product": "C", "quantity": 4, "revenue": 80 },
  { "_id": 5, "product": "B", "quantity": 6, "revenue": 120 }
]
```

```
db.sales.aggregate([
  {
    $group: {
      _id: "$product",
      totalRevenue: { $sum: "$revenue" }
    }
  },
  {
    $sort: { totalRevenue: -1 }
  },
  {
    $limit: 2
  }
])
```

```
[
  { "_id": "B", "totalRevenue": 180 },
  { "_id": "A", "totalRevenue": 140 }
]
```

- Group
- Limit
- Project
- Sort
- Match
- Add Fields
- Count
- Lookup
- Out

RAAFI
Builds You Best

UNDERSTANDING MongoDB - Aggregations

```
[
  { "_id": 1, "product": "A", "quantity": 5, "revenue": 100 },
  { "_id": 2, "product": "B", "quantity": 3, "revenue": 60 },
  { "_id": 3, "product": "A", "quantity": 2, "revenue": 40 },
  { "_id": 4, "product": "C", "quantity": 4, "revenue": 80 },
  { "_id": 5, "product": "B", "quantity": 6, "revenue": 120 }
]
```

```
db.collection.aggregate([
  {
    $project: {
      product: 1,
      revenue: 1
    }
  }
])
```

```
[
  { "_id": 1, "product": "A", "revenue": 100 },
  { "_id": 2, "product": "B", "revenue": 60 },
  { "_id": 3, "product": "A", "revenue": 40 },
  { "_id": 4, "product": "C", "revenue": 80 },
  { "_id": 5, "product": "B", "revenue": 120 }
]
```

- Group
- Limit
- Project
- Sort
- Match
- Add Fields
- Count
- Lookup
- Out

RAAFI
Builds You Best

UNDERSTANDING MongoDB - Aggregations

```
[
  { "_id": 1, "product": "A", "quantity": 5, "revenue": 100 },
  { "_id": 2, "product": "B", "quantity": 3, "revenue": 60 },
  { "_id": 3, "product": "A", "quantity": 2, "revenue": 40 },
  { "_id": 4, "product": "C", "quantity": 4, "revenue": 80 },
  { "_id": 5, "product": "B", "quantity": 6, "revenue": 120 }
]
```

```
db.collection.aggregate([
  {
    $match: {
      revenue: { $gt: 80 }
    }
  }
])
```

```
[
  { "_id": 1, "product": "A", "quantity": 5, "revenue": 100 },
  { "_id": 4, "product": "C", "quantity": 4, "revenue": 80 },
  { "_id": 5, "product": "B", "quantity": 6, "revenue": 120 }
]
```

- Group
- Limit
- Project
- Sort
- Match
- Add Fields
- Count
- Lookup
- Out

UNDERSTANDING MongoDB - Aggregations

```
[
  { "_id": 1, "product": "A", "quantity": 5, "revenue": 100 },
  { "_id": 2, "product": "B", "quantity": 3, "revenue": 60 },
  { "_id": 3, "product": "A", "quantity": 2, "revenue": 40 },
  { "_id": 4, "product": "C", "quantity": 4, "revenue": 80 },
  { "_id": 5, "product": "B", "quantity": 6, "revenue": 120 }
]
```

```
db.collection.aggregate([
  {
    $addFields: {
      totalCost: { $multiply: ["$quantity", "$revenue"] }
    }
  }
])
```

```
[
  { "_id": 1, "product": "A", "quantity": 5, "revenue": 100, "totalCost": 500 },
  { "_id": 2, "product": "B", "quantity": 3, "revenue": 60, "totalCost": 180 },
  { "_id": 3, "product": "A", "quantity": 2, "revenue": 40, "totalCost": 80 },
  { "_id": 4, "product": "C", "quantity": 4, "revenue": 80, "totalCost": 320 },
  { "_id": 5, "product": "B", "quantity": 6, "revenue": 120, "totalCost": 720 }
]
```

- Group
- Limit
- Project
- Sort
- Match
- Add Fields
- Count
- Lookup
- Out

RAAFI
Builds You Best

UNDERSTANDING MongoDB - Aggregations

```
[
  { "_id": 1, "product": "A", "quantity": 5, "revenue": 100 },
  { "_id": 2, "product": "B", "quantity": 3, "revenue": 60 },
  { "_id": 3, "product": "A", "quantity": 2, "revenue": 40 },
  { "_id": 4, "product": "C", "quantity": 4, "revenue": 80 },
  { "_id": 5, "product": "B", "quantity": 6, "revenue": 120 }
]
```

```
db.collection.aggregate([
  {
    $count: "totalDocuments"
  }
])
```

```
[
  { "totalDocuments": 5 }
]
```

- Group
- Limit
- Project
- Sort
- Match
- Add Fields
- Count
- Lookup
- Out

RAAFI
Builds You Best

UNDERSTANDING MongoDB - Aggregations

```
[
  { "_id": 1, "productName": "Laptop", "categoryId": 101 },
  { "_id": 2, "productName": "Smartphone", "categoryId": 102 },
  { "_id": 3, "productName": "Book", "categoryId": 103 },
  { "_id": 4, "productName": "Headphones", "categoryId": 102 },
  { "_id": 5, "productName": "Chair", "categoryId": 104 }
]
```

```
db.products.aggregate([
  {
    $lookup: {
      from: "categories",     // The target collection to join with
      localField: "categoryId",   // The field from the "products" collection
      foreignField: "_id",    // The field from the "categories" collection
      as: "categoryInfo"     // The alias for the joined data
    }
  }
])
```

```
[
  { "_id": 101, "categoryName": "Electronics" },
  { "_id": 102, "categoryName": "Electronics" },
  { "_id": 103, "categoryName": "Books" },
  { "_id": 104, "categoryName": "Furniture" }
]
```

```
[
  {
    "_id": 1,
    "productName": "Laptop",
    "categoryId": 101,
    "categoryInfo": [
      { "_id": 101, "categoryName": "Electronics" }
    ]
  },
  {
    "_id": 2,
    "productName": "Smartphone",
    "categoryId": 102,
    "categoryInfo": [
      { "_id": 102, "categoryName": "Electronics" }
    ]
  },
```

Group

Limit

Project

Sort

Match

Add Fields

Count

Lookup

Out

RAAFI
Builds You Best

UNDERSTANDING MongoDB - Aggregations

```
[
  { "_id": 1, "product": "A", "quantity": 5, "revenue": 100 },
  { "_id": 2, "product": "B", "quantity": 3, "revenue": 60 },
  { "_id": 3, "product": "A", "quantity": 2, "revenue": 40 },
  { "_id": 4, "product": "C", "quantity": 4, "revenue": 80 },
  { "_id": 5, "product": "B", "quantity": 6, "revenue": 120 }
]
```

```
db.collection.aggregate([
  {
    $group: {
      _id: "$product",
      totalRevenue: { $sum: "$revenue" }
    }
  },
  {
    $out: "aggregatedSales" // Create a new collection called "aggregatedSale
  }
])
```

```
db.aggregatedSales.find()
```

```
[
  { "_id": "A", "totalRevenue": 140 },
  { "_id": "B", "totalRevenue": 180 },
  { "_id": "C", "totalRevenue": 80 }
]
```

- Group
- Limit
- Project
- Sort
- Match
- Add Fields
- Count
- Lookup
- Out

RAAFI
Builds You Best

UNDERSTANDING MongoDB - Index

```
[
  { "_id": 1, "product": "A", "quantity": 5, "revenue": 100 },
  { "_id": 2, "product": "B", "quantity": 3, "revenue": 60 },
  { "_id": 3, "product": "A", "quantity": 2, "revenue": 40 },
  { "_id": 4, "product": "C", "quantity": 4, "revenue": 80 },
  { "_id": 5, "product": "B", "quantity": 6, "revenue": 120 }
]
```

```
db.collection.createIndex({ product: 1 });
```

```
db.collection.getIndexes();
```

```
db.collection.find({ product: "A" });
```

RAAFI
Builds You Best

UNDERSTANDING MongoDB - Validations

```
db.createCollection("books", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["title", "author", "publicationYear", "price"],
      properties: {
        // JSON Schema properties here...
      }
    }
  }
});
```