What You Need to Know Before MongoDB

You should already be comfortable with:

- **JavaScript / Node.js** (if using MongoDB with backend like Express.js)
- **JSON syntax** – MongoDB stores data in BSON (binary JSON)
- **Basic command line** – to interact with `mongosh` or run scripts

---

20% MongoDB That Does 80% of the Work

🧱 1. Core CRUD Operations

These are used in almost every application:

```
// Insert
db.users.insertOne({ name: "Ismail", age: 25 })
db.users.insertMany([{ name: "Ali" }, { name: "Sara" }])

// Read
db.users.find({ age: { $gte: 18 } })   // Filter with operators
db.users.findOne({ name: "Ismail" })

// Update
db.users.updateOne({ name: "Ismail" }, { $set: { age: 26 } })
db.users.updateMany({}, { $inc: { age: 1 } })  // Increase all ages

// Delete
db.users.deleteOne({ name: "Ali" })
db.users.deleteMany({ age: { $lt: 18 } })
```

🧩 2. Query Operators

These power your filtering logic:

- `$gt`, `$lt`, `$eq`, `$in`, `$or`, `$and`, `$ne`

```
db.users.find({ age: { $gt: 18, $lt: 30 } })
db.users.find({ $or: [{ name: "Sara" }, { age: 25 }] })
```

🧠 3. Indexing (Performance)

Indexes make your queries fast.

```
db.users.createIndex({ name: 1 }) // Ascending index on name
```

🔁 4. Aggregation Framework

Powerful for data analytics, transformations, and grouping.

```
db.orders.aggregate([
  { $match: { status: "delivered" } },
  { $group: { _id: "$customerId", total: { $sum: "$amount" } } },
  { $sort: { total: -1 } }
])
```

🔒 5. Schema Design / Embedded vs Referenced

Understand when to:

- Embed documents (nested data inside another)
- Reference documents (like foreign keys)

Example:

```
// Embedded
{
  name: "Order 1",
  items:  [ 'English', 'Calculus', 'Statistics', 'Geography' ]
}

// Referenced
{
  name: "Order 1",
  items: [ObjectId("..."), ObjectId("...")]
}
```

⚙️ 6. Working with Mongoose (if using Node.js)

Mongoose adds schema + model management.

```
const mongoose = require('mongoose');
const UserSchema = new mongoose.Schema({ name: String, age: Number });
const User = mongoose.model('User', UserSchema);

// Create
await User.create({ name: "Ismail", age: 25 });

// Read
const users = await User.find({ age: { $gt: 18 } });
```

🎯 Summary: Learn These First

Topic	Why It Matters
`insertOne`, `find`, `updateOne`, `deleteOne`	Core of all database work
Query Operators (`$gt`, `$in`, `$or`)	Powerful filtering
Aggregation (`$match`, `$group`, `$sort`)	Reporting & analytics
Indexing	Speeds up performance

Schema Design (Embed vs Reference)	Scalability and efficiency
Mongoose (optional)	Structure + simplicity in Node.js

📚 Bonus Resources

- MongoDB University — free courses
- MongoDB Docs
- `MongoDB Compass` — GUI to explore collections visually