MongoDB Short Notes

Basic Database Operations

```bash
show dbs                # Display list of databases
use company              # Switch to database (creates if doesn't exist)
db.createCollection("name") # Create collection
show collections          # List collections
db.employee.drop()        # Drop collection (returns true/false)
db.dropDatabase()         # Drop database
```

CRUD Operations

Create (Insert)

```javascript
// Insert single document
db.employee.insertOne({name: "raju", age: 25, department: "accounts"})

// Insert multiple documents
db.employee.insertMany([
    {name: "neha", age: 35, department: "HR"},
    {name: "sachin", age: 28, department: "HR"}
])
```

Read (Find)

```javascript
db.employee.find()                    # Fetch all records
db.employee.findOne({name:"sachin"})  # Fetch single record
```

Update

```javascript
```

```
// Update single document
db.Employee.updateOne({"name":"Sourav"}, {$set:{designation:"Software developer"}})

// Update multiple documents
db.Employee.updateMany({"dept_id":"001"}, {$set:{designation:"HR Executive"}})

// Array operations
db.Employee.updateOne({"name":"Sourav"}, {$push:{age:30}})       # Add to array
db.Employee.updateOne({"name":"Sourav"}, {$pull:{age:30}})       # Remove from array
db.Employee.updateOne({"name":"Sourav"}, {$unset:{hobby:""}})     # Remove property

// Add multiple values to array
db.Employee.updateOne(
    {"name":"Sourav"},
    {$push:{hobby:{$each: ["Playing Cricket","Swimming","Cooking"]}}}
)
```

Delete

```javascript
db.employee.deleteOne({department:"001"})   # Delete first matching document
db.employee.deleteMany({department:"001"})  # Delete all matching documents
db.employee.deleteMany({})                  # Delete all documents
```

Upsert

Combination of update and insert - creates new document if no match found:

```javascript
db.students.updateMany(
    {name: "Amit"},
    {$set: {age:23, course: "Math"}},
    {upsert: true}
)
```

Data Types

1. ObjectId 2. String 3. Integer 4. Double 5. Boolean
2. Array 7. Object 8. Date 9. Null 10. Timestamp

```javascript
```

```
db.cart.insertOne({order_date: new Date()})
db.cart.insertOne({ts: new Timestamp()})
```

Query Operators

Relational Operators

```javascript
{age: {$eq:25}}   # Equal (=)
{age: {$lt:25}}   # Less than (<)
{age: {$gt:25}}   # Greater than (>)
{age: {$lte:25}}  # Less than or equal (<=)
{age: {$gte:25}}  # Greater than or equal (>=)
{age: {$neq:25}}  # Not equal (!=)

{age: {$in:[24,26]}}   # IN
{age: {$nin:[24,26]}}  # NOT IN
```

Logical Operators

```javascript
// AND
db.collection.find({
   $and: [
      {age: {$eq:28}},
      {name: "raju"}
   ]
})

// OR
db.collection.find({
   $or: [
      {age: {$eq:28}},
      {name: "raju"}
   ]
})
```

Element Operators

```javascript
db.collection.find({color:{$exists:true}})        # Check if field exists
db.collection.find({is_enable:{$type:"string"}})  # Filter by data type
```

Cursor Methods

```javascript
db.employee.find().count()     # Count documents
db.employee.find().sort({"name":1})    # Sort (1: ascending, -1: descending)
db.employee.find().limit(2)     # Limit results
db.employee.find().skip(3)     # Skip records from start
```

Indexing in MongoDB

What is Indexing?

- Special data structure (B-Tree) that stores collection data in searchable format
- Like book index - jump directly to information instead of scanning every page
- **Without indexes**: Collection scan (checks every document) - slow for large collections
- **With indexes**: Quick document lookup using index structure - drastically reduces query time

Types of Indexes

1. Single Field Index

```javascript
db.users.createIndex({name: 1})    # 1 = ascending, -1 = descending
```

Use Case: Queries filtering/sorting by one field

2. Compound Index

```javascript
db.products.createIndex({productId: 1, productCategory: -1})
```

Use Case: Queries involving multiple fields together

3. Multikey Index

```javascript
db.products.createIndex({tags: 1})  # Automatically created for array fields
```

Note: Each array element gets its own index entry

4. Text Index

```javascript
db.products.createIndex({description: "text"})
```

Note: Supports $text queries, only one text index per collection

Index Properties

Unique Index

```javascript
db.users.createIndex({email: 1}, {unique: true})  # Prevents duplicate values
```

TTL Index (Time To Live)

```javascript
db.sessions.createIndex({createdAt: 1}, {expireAfterSeconds: 3600})  # Auto-deletes after time
```

Managing Indexes

```javascript
db.collection.getIndexes()          # View all indexes
db.collection.dropIndex("indexName") # Drop specific index
db.collection.dropIndexes()         # Drop all indexes
```

Key Points

- MongoDB is NoSQL document database
- Collections store documents (similar to tables storing rows)
- Documents are in BSON format (Binary JSON)
- ObjectId is automatically generated unique identifier
- Use dot notation to access nested fields
- Indexes dramatically improve query performance but use storage space