Basic Commands

- **Show databases**:

```
show dbs
```

- **Switch database**:

```
use dbname
```

- **Show collections**:

```
show collections
```

CRUD Operations

Create (Insert)

- **Insert a document**:

```
db.collection.insertOne ({ key: value })
```

- **Insert multiple documents**:

```
db.collection.insertMany([{ key: value }, { key: value2 }])
```

Read (Find)

- **Find all documents**:

```
db.collection.find()
```

- **Find with condition**:

```
db.collection.find({ key: value })
```

- **Find with projection (select specific fields)**:

```
db.collection.find({ key: value }, { field1: 1, field2: 1 })
```

- **Find one document**:

```
db.collection.findOne({ key: value })
```

- **Sort documents**:

```
db.collection.find().sort({ key: 1 }) // 1 for ascending, -1 for
descending
```

- **Limit results**:

```
db.collection.find().limit(5)
```

- **Skip results**:

```
db.collection.find().skip(5)
```

Update

- **Update one document**:

```
db.collection.updateOne({ key: value }, { $set: { key2: value2 } })
```

- **Update multiple documents**:

```
db.collection.updateMany({ key: value }, { $set: { key2: value2 } })
```

- **Replace a document**:

```
db.collection.replaceOne({ key: value }, { key1: value1, key2: value2
})
```

Delete

- **Delete one document**:

```
db.collection.deleteOne({ key: value })
```

- **Delete multiple documents**:

```
db.collection.deleteMany({ key: value })
```

3. Operators

Comparison Operators

- **Equal**:

```
{ key: value }
```

- **Not equal**:

```
{ key: { $ne: value } }
```

- **Greater than / Less than**:

```
{ key: { $gt: value } }
{ key: { $lt: value } }
```

- **Greater than or equal / Less than or equal**:

```
{ key: { $gte: value } }
{ key: { $lte: value } }
```

Logical Operators

- **AND condition**:

```
{ $and: [{ key1: value1 }, { key2: value2 }] }
```

- **OR condition**:

```
{ $or: [{ key1: value1 }, { key2: value2 }] }
```

- **NOT condition**:

```
{ key: { $not: { $lt: value } } }
```

Array Operators

- **Find documents with element in an array**:

```
db.collection.find({ key: { $in: [value1, value2] } })
```

- **Find documents with element not in an array**:

```
db.collection.find({ key: { $nin: [value1, value2] } })
```

- **Match array elements**:

```
db.collection.find({ key: { $all: [value1, value2] } })
```

- **Array size**:

```
db.collection.find({ key: { $size: 3 } })
```

Element Operators

- **Field exists**:

```
db.collection.find({ key: { $exists: true } })
```

- **Field is of a specific type**:

```
db.collection.find({ key: { $type: "string" } })
```

Aggregation Operations

Basic Aggregation

- **Aggregation Pipeline**:

```
db.collection.aggregate([
  { $match: { key: value } },
  { $group: { _id: "$key", total: { $sum: 1 } } }
])
```

Pipeline Stages

- **$match** – Filter documents:

```
{ $match: { key: value } }
```

- **$group** – Group documents by a key:

```
{ $group: { _id: "$key", total: { $sum: "$amount" } } }
```

- **$sort** – Sort documents:

```
{ $sort: { key: 1 } } // 1 for ascending, -1 for descending
```

- **$limit** – Limit number of documents:

```
{ $limit: 5 }
```

- **$project** – Select specific fields:

```
{ $project: { field1: 1, field2: 1 } }
```

Aggregation Operators

- **$sum** – Calculate sum:

```
{ $group: { _id: null, total: { $sum: "$field" } } }
```

- **$avg** – Calculate average:

```
{ $group: { _id: null, average: { $avg: "$field" } } }
```

- **$min / $max** – Get minimum or maximum value:

```
{ $group: { _id: null, minVal: { $min: "$field" }, maxVal: { $max:
"$field" } } }
```

- **$count** – Count documents:

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

Array Unwinding

- **Unwind array fields**:

```
db.collection.aggregate([{ $unwind: "$arrayField" }])
```