MongoDB Notes with Coding Examples

### READ Operation in MongoDB

- **Basic `find` Query**:

```
  db.collection_name.find({ field_name: value });
  Example:
  db.users.find({ age: 25 }); // Fetch all documents where age is 25
```

- **Find a Single Document**:

```
  db.collection_name.findOne({ field_name: value });
  Example:
  db.users.findOne({ name: "Alice" }); // Fetch one document where name is "Alice"
```

### Advanced READ Operation in MongoDB

- **Projection**: Select specific fields to return.

```
  db.users.find({}, { name: 1, age: 1, _id: 0 }); // Show only name and age fields
```

- **Sorting**:

```
  db.users.find().sort({ age: 1 }); // Ascending order by age
  db.users.find().sort({ age: -1 }); // Descending order by age
```

- **Filtering with Conditions**:

```
  db.users.find({ age: { $gte: 25 } }); // Users aged 25 or above
```

### Importing and Exporting JSON in MongoDB

- **Import JSON**:

```
  mongoimport --db database_name --collection collection_name --file file.json --jsonArray
```

  Example:

```
  mongoimport --db test --collection users --file users.json --jsonArray
```

- **Export JSON**:

```
  mongoexport --db database_name --collection collection_name --out file.json
```

  Example:

```
  mongoexport --db test --collection users --out users.json
```

### Comparison Operators

Operator	Description	Example
`$eq`	Equal to	`{ age: { $eq: 25 } }`
`$ne`	Not equal to	`{ age: { $ne: 25 } }`
`$gt`	Greater than	`{ age: { $gt: 25 } }`
`$gte`	Greater than or equal to	`{ age: { $gte: 25 } }`
`$lt`	Less than	`{ age: { $lt: 25 } }`
`$lte`	Less than or equal to	`{ age: { $lte: 25 } }`
`$in`	Matches any in an array	`{ age: { $in: [25, 30, 35] } }`
`$nin`	Matches none in an array	`{ age: { $nin: [25, 30, 35] } }`

Example:

```
db.users.find({ age: { $gte: 25, $lte: 30 } }); // Users aged between 25 and 30
```

### Introduction to Cursors

- A **cursor** in MongoDB allows you to iterate over query results.

#### **Cursor Methods**:

1. **`count`**: Count the number of documents matching the query.

```
   db.users.find({ age: { $gte: 25 } }).count();
```

2. **`limit`**: Limit the number of results.

```
   db.users.find().limit(5); // First 5 documents
```

3. **`skip`**: Skip a number of documents.

```
   db.users.find().skip(5); // Skip the first 5 documents
```

4. **`sort`**: Sort results.

```
   db.users.find().sort({ age: -1 }); // Sort by age descending
```

### Logical Operators

Operator	Description	Example
`$and`	Match all conditions	`{ $and: [{ age: { $gt: 25 } }, { active: true }] }`
`$or`	Match any condition	`{ $or: [{ age: { $lt: 25 } }, { active: false }] }`
`$not`	Negates a condition	`{ age: { $not: { $gte: 30 } } }`
`$nor`	Match none of the conditions	`{ $nor: [{ age: { $lt: 25 } }, { active: false }] }`

Example:

```
db.users.find({ $and: [{ age: { $gte: 25 } }, { active: true }] });
```

### Complex Expressions (`$expr`)

- Use `$expr` to perform complex queries using aggregation expressions.
  Example:
```
    db.sales.find({ $expr: { $gt: ["$quantity", "$threshold"] } }); // quantity > threshold
```

$exists and $type

- **`$exists`**: Check if a field exists.
```
  db.users.find({ phone: { $exists: true } }); // Find users with a phone field
```

- **`$type`**: Match by BSON type.
```
   db.users.find({ age: { $type: "int" } }); // Find documents where age is an integer
```