1. Basic Operations in MongoDB

a. Where Clause, AND, OR Example

```
db.students.find({ $and: [ { age: { $gt: 18 } }, { grade: "A" } ] })
db.students.find({ $or: [ { age: { $lt: 18 } }, { grade: "B" } ] })
```

b. CRUD Operations

```
// Insert
db.students.insertOne({ name: "John", age: 20, grade: "A" })

// Query
db.students.find({ name: "John" })

// Update
db.students.updateOne({ name: "John" }, { $set: { grade: "B" } })

// Delete
db.students.deleteOne({ name: "John" })

// Projection
db.students.find({}, { name: 1, age: 1, _id: 0 })
```

2. Field Selection and Limiting Results

a. Select & Ignore Fields

```
db.students.find({}, { name: 1, grade: 1, _id: 0 }) // Excludes _id
```

b. Display First 5 Documents

```
db.students.find({}, { name: 1, grade: 1, _id: 0 }).limit(5)
```

3. Query Selectors

a. Comparison & Logical Selectors

```
// Comparison
db.students.find({ age: { $gte: 18 } })

// Logical
db.students.find({ $or: [ { grade: "A" }, { age: { $lt: 20 } } ] })
```

b. Geospatial & Bitwise Selectors

```
// Geospatial example (Assume 'location' field with GeoJSON)
db.places.find({
  location: {
    $near: {
      $geometry: { type: "Point", coordinates: [40, -74] },
      $maxDistance: 5000
    }
  }
})
```

```
// Bitwise selector
db.devices.find({ flags: { $bitsAllSet: [1, 3] } })
```

4. Projection Operators

```
// $slice
db.students.find({}, { marks: { $slice: 2 } })

// $elemMatch
db.students.find({ scores: { $elemMatch: { subject: "Math", marks: { $gt:
70 } } } })

// Example with $
db.students.find({ "scores.subject": "Math" }, { "scores.$": 1 })
```

5. Aggregation Operators

```
db.students.aggregate([
  {
    $group: {
      _id: "$grade",
      averageAge: { $avg: "$age" },
      maxAge: { $max: "$age" },
      minAge: { $min: "$age" },
      allNames: { $push: "$name" },
      uniqueNames: { $addToSet: "$name" }
    }
  }
])
```

6. Aggregation Pipeline Example

```
db.students.aggregate([
  { $match: { age: { $gte: 18 } } },
  { $group: { _id: "$grade", count: { $sum: 1 } } },
  { $sort: { count: -1 } },
  { $project: { _id: 0, grade: "$_id", count: 1 } },
  { $skip: 0 },
  { $limit: 5 }
])
```

7. Real Dataset Queries

a. Listings with specific fields

```
db.listingsAndReviews.find(
  { host_picture_url: { $exists: true, $ne: "" } },
  { listing_url: 1, name: 1, address: 1, host_picture_url: 1, _id: 0 }
)
```

b. E-commerce Review Summary

```
db.reviews.aggregate([
```

```
  { $group: { _id: "$product_id", avgRating: { $avg: "$rating" }, count: {
$sum: 1 } } }
])
```

8. Indexing

a. Create Indexes

```
db.products.createIndex({ sku: 1 }, { unique: true })          // Unique
db.products.createIndex({ discount: 1 }, { sparse: true })     // Sparse
db.orders.createIndex({ userId: 1, date: -1 })                 // Compound
db.products.createIndex({ "tags": 1 })                         // Multikey
(on array field)
```

b. Query Optimization

```
// Use explain to analyze performance
db.products.find({ sku: "1001" }).explain("executionStats")
```

9. Text Search

a. Text Search Query

```
// Create text index
db.catalog.createIndex({ name: "text", description: "text" })

// Search for word
db.catalog.find({ $text: { $search: "laptop" } })
```

b. Exclude Words

```
db.catalog.find({ $text: { $search: "laptop -used" } }) // excludes 'used'
```

10. Text Search Using Aggregation Pipeline

```
db.catalog.aggregate([
  {
    $match: {
      $text: { $search: "gaming" }
    }
  },
  {
    $project: {
      name: 1,
      description: 1,
      score: { $meta: "textScore" }
    }
  },
  {
    $sort: { score: { $meta: "textScore" } }
  }
])
```