Comparison Operators:

$eq (Equal):

Find all documents where the "age" field is equal to 30.

```
db.users.find({ age: { $eq: 30 } })
```

$ne (Not Equal):

Find all documents where the "status" field is not equal to "inactive".

```
db.accounts.find({ status: { $ne: "inactive" } })
```

$gt (Greater Than):

Find all documents where the "score" field is greater than 80.

```
db.students.find({ score: { $gt: 80 } })
```

$gte (Greater Than or Equal To):

Find all documents where the "price" field is greater than or equal to 100.

```
db.products.find({ price: { $gte: 100 } })
```

$lt (Less Than):

Find all documents where the "quantity" field is less than 5.

```
db.inventory.find({ quantity: { $lt: 5 } })
```

$lte (Less Than or Equal To):

Find all documents where the "rating" field is less than or equal to 3.5.

```
db.reviews.find({ rating: { $lte: 3.5 } })
```

$in (In Array):

Find all documents where the "category" field is either "Tech" or "Science".

```
db.products.find({ category: { $in: ["Tech", "Science"] } })
```

$nin (Not In Array):

Find all documents where the "role" field is not one of the specified roles.

```
db.users.find({ role: { $nin: ["admin", "editor"] } })
```

Logical Operators:

$and (Logical AND):

Find all documents where both "age" is 25 and "city" is "New York".

```
db.users.find({ $and: [{ age: 25 }, { city: "New York" }] })
```

$or (Logical OR):

Find all documents where either "status" is "active" or "role" is "admin".

```
db.users.find({ $or: [{ status: "active" }, { role: "admin" }] })
```

$not (Inverts Expression):

Find all documents where "age" is not equal to 30.

```
db.users.find({ age: { $not: { $eq: 30 } } })
```

$nor (Logical NOR):

Find all documents where "age" is neither 25 nor 30.

```
db.users.find({ $nor: [{ age: 25 }, { age: 30 }] })
```

Element Operators:

$exists (Field Exists):

Find all documents where the "email" field exists.

```
db.contacts.find({ email: { $exists: true } })
```

$type (Field Type):

Find all documents where the "age" field is of type "number".

```
db.users.find({ age: { $type: "number" } })
```

Array Operators:

$all (All Elements Match):

Find all documents where the "tags" array contains both "mongodb" and "nodejs".

```
db.articles.find({ tags: { $all: ["mongodb", "nodejs"] } })
```

$elemMatch (Array Element Matches):

Find all documents where the "scores" array contains at least one score greater than 90.

```
db.students.find({ scores: { $elemMatch: { $gt: 90 } } })
```

$size (Array Size):

Find all documents where the "comments" array has exactly 5 elements.

```
db.posts.find({ comments: { $size: 5 } })
```

Evaluation Operators:

$expr (Aggregation Expression):

Find all documents where "price" is less than 0.8 times "cost".

```
db.products.find({ $expr: { $lt: ["$price", { $multiply: ["$cost", 0.8] }] } })
```

$jsonSchema (JSON Schema Validation):

Find all documents that match a specified JSON schema.

```
db.data.find({ $jsonSchema: { type: "object", properties: { name: { type: "string" } } } })
```

$mod (Modulo):

Find all documents where "quantity" modulo 5 equals 0.

```
db.inventory.find({ quantity: { $mod: [5, 0] } })
```

Geospatial Operators:

$geoWithin (Within a Geometry):

Find all documents that are within a specified polygon.

```
db.locations.find({ geometry: { $geoWithin: { $geometry: { type: "Polygon", coordinates: [[ ... ]] } } } })
```

$geoIntersects (Intersects with a Geometry):

Find all documents that intersect with a specified circle.

```
db.places.find({ location: { $geoIntersects: { $geometry: { type: "Point", coordinates: [ ... ] } } } })
```

$near (Near a Point):

Find all documents near a specified point, sorted by distance.

```
db.locations.find({ location: { $near: { type: "Point", coordinates: [ ... ] } } })
```

Text Operators:

$text (Text Search):

Perform a text search on the "content" field for the word "MongoDB".

```
db.articles.find({ $text: { $search: "MongoDB" } })
```

$search (Specify Search String):

Find all documents that match the specified search string.

```
db.products.find({ $text: { $search: "electronics" } })
```

Array Update Operators:

$push (Append to Array):

Add a new phone number to the "contactNumbers" array.

```
db.contacts.update({ _id: 1 }, { $push: { contactNumbers: "+1234567890" } })
```

$pull (Remove from Array):

Remove all instances of the tag "deprecated" from the "tags" array.

```
db.posts.update({}, { $pull: { tags: "deprecated" } }, { multi: true })
```

$addToSet (Add to Set):

Add a new subject "chemistry" to the "subjects" array if it doesn't already exist.

```
db.students.update({ _id: 1 }, { $addToSet: {
```