Comparison Operators

Products Database: [
```
 {
            name: Wireless Mouse,
            category: Electronics,
            price: 25.50,
            stock: 200,
            tags: [accessories, wireless],
            details: { brand: ErgoGear, color: black },
            is_available: true
 },
 {
            name: Mechanical Keyboard,
            category: Electronics,
            price: 80.00,
            stock: 75,
            tags: [accessories, gaming, mechanical],
            details: { brand: ClickyKeys, layout: US },
            is_available: true
 },
 {
            name: Organic Coffee Beans,
            category: Groceries,
            price: 15.00,
            stock: 150,
            tags: [coffee, organic, beverages],
            details: { origin: Colombia, weight_kg: 0.5 },
            is_available: true
 },
 {
            name: Smart Speaker Echo,
            category: Smart Home,
            price: 99.99,
            stock: 120,
            tags: [smart-home, voice-assistant],
            details: { brand: HomeAI, assistant: Echo },
            is_available: false // Currently out of stock/unavailable
 },
 {
            name: Yoga Mat Pro,
            category: Fitness,
            price: 40.00,
            stock: 80,
            tags: [fitness, yoga, accessories],
            details: { material: TPE, thickness_mm: 6 },
            reviews: [
              { user: Charlie, rating: 5, comment: Very comfortable! }
            ],
            is_available: true
 }
]
```

$eq — Equal To Operator

The $eq operator in MongoDB is used to match documents where the value of a field is **equal to a specified value**.

Syntax: `{ field: { $eq: value } }`

Examples

1. Match Products with Price Equal to 25.5

```
db.products.find({ price: { $eq: 25.5 } })
```

Finds products where the price field is exactly 25.5.

Result: `{  name: "Wireless Mouse",  price: 25.5,  … }`

2. Match Products with Brand Equal to "ClickyKeys" (Nested Field)

```
db.products.find({ "details.brand": { $eq: "ClickyKeys" } })
```

Finds products where the nested field details.brand is exactly "ClickyKeys".

Result:
```
{

 name: "Mechanical Keyboard",
 details: {
   brand: "ClickyKeys",
   layout: "US"
 },
 ...
}
```

3. Match Products with Assistant Equal to "Echo" (Nested Field)

```
db.products.find({ "details.assistant": { $eq: "Echo" } })
```

Finds products where the nested field details.assistant is exactly "Echo".

Result:
```
{

 name: "Smart Speaker Echo",
 details: {
   brand: "HomeAI",
   assistant: "Echo"
 },
 ...
}
```

$gt — Greater Than Operator

The $gt operator in MongoDB is used to match documents where the value of a field is **greater than** a specified value.

Syntax

```
{ field: { $gt: value } }
```

Examples

1. Find products with stock greater than 100

```
db.products.find({ stock: { $gt: 100 } })
```

Retrieves all products where the stock field value is greater than 100.

Result: [

```
  { name: "Wireless Mouse", stock: 200, ... },
  { name: "Organic Coffee Beans", stock: 150, ... },
  { name: "Smart Speaker Echo", stock: 120, ... }
]
```

2. Find products with price greater than 50

```
db.products.find({ price: { $gt: 50 } })
```

Retrieves all products where the price field value is greater than 50.

Result: [

```
  { name: "Mechanical Keyboard", price: 80.0, ... },
  { name: "Smart Speaker Echo", price: 99.99, ... }
]
```

3. Find products with weight greater than 0.3 kg (nested field)

```
db.products.find({ "details.weight_kg": { $gt: 0.3 } })
```

Retrieves products where the nested field details.weight_kg has a value greater than 0.3.

Result:

```
{ name: "Organic Coffee Beans", details: { weight_kg: 0.5 }, ... }
```

4. Find products with thickness greater than 5 mm (nested field)

```
db.products.find({ "details.thickness_mm": { $gt: 5 } })
```

Retrieves products where the nested field `details.thickness_mm` has a value greater than 5.

Result:

```
{ name: "Yoga Mat Pro", details: { thickness_mm: 6 }, ... }
```

$gte — Greater Than or Equal To Operator

The `$gte` operator in MongoDB is used to match documents where the value of a field is **greater than or equal to** a specified value.

Syntax

```
{ field: { $gte: value } }
```

Examples

1. Find products with price greater than or equal to 80

```
db.products.find({ price: { $gte: 80 } })
```

Retrieves all products where the `price` field value is greater than or equal to 80.

Result: [

```
  { name: "Mechanical Keyboard", price: 80.0, ... },
  { name: "Smart Speaker Echo", price: 99.99, ... }
]
```

2. Find products with stock greater than or equal to 150

```
db.products.find({ stock: { $gte: 150 } })
```

Retrieves all products where the `stock` field value is greater than or equal to 150.

Result: [

```
  { name: "Wireless Mouse", stock: 200, ... },
  { name: "Organic Coffee Beans", stock: 150, ... }
]
```

3. Find products with thickness greater than or equal to 6 mm (nested field)

```
db.products.find({ "details.thickness_mm": { $gte: 6 } })
```

Retrieves products where the nested field `details.thickness_mm` has a value greater than or equal to 6.

Result:

```
{ name: "Yoga Mat Pro", details: { thickness_mm: 6 }, ... }
```

$lt — Less Than Operator

The `$lt` operator in MongoDB is used to match documents where the value of a field is **less than** a specified value.

Syntax

```
{ field: { $lt: value } }
```

Examples

1. Find products with price less than 50

```
db.products.find({ price: { $lt: 50 } })
```

Retrieves all products where the `price` field value is less than 50.

Result: [

```
  { name: "Wireless Mouse", price: 25.5, ... },
  { name: "Organic Coffee Beans", price: 15.0, ... },
  { name: "Yoga Mat Pro", price: 40.0, ... }
]
```

2. Find products with stock less than 100

```
db.products.find({ stock: { $lt: 100 } })
```

Retrieves all products where the `stock` field value is less than 100.

Result: [

```
  { name: "Mechanical Keyboard", stock: 75, ... },
  { name: "Yoga Mat Pro", stock: 80, ... }
]
```

3. Find products with weight less than 1 kg (nested field)

```
db.products.find({ "details.weight_kg": { $lt: 1 } })
```

Retrieves products where the nested field `details.weight_kg` has a value less than 1.

Result:

```
{ name: "Organic Coffee Beans", details: { weight_kg: 0.5 }, ... }
```

$lte — Less Than or Equal To Operator

The `$lte` operator in MongoDB is used to match documents where the value of a field is **less than or equal to** a specified value.

Syntax

```
{ field: { $lte: value } }
```

Examples

1. Find products with stock less than or equal to 75

```
db.products.find({ stock: { $lte: 75 } })
```

Retrieves products where the `stock` field value is less than or equal to 75.

Result:

```
{ name: "Mechanical Keyboard", stock: 75, ... }
```

2. Find products with price less than or equal to 40

```
db.products.find({ price: { $lte: 40 } })
```

Retrieves products where the `price` field value is less than or equal to 40.

Result: [

```
  { name: "Wireless Mouse", price: 25.5, ... },
  { name: "Organic Coffee Beans", price: 15.0, ... },
  { name: "Yoga Mat Pro", price: 40.0, ... }
]
```

3. Find products with weight less than or equal to 0.5 kg (nested field)

```
db.products.find({ "details.weight_kg": { $lte: 0.5 } })
```

Retrieves products where the nested field `details.weight_kg` has a value less than or equal to 0.5.

Result:

```
{ name: "Organic Coffee Beans", details: { weight_kg: 0.5 }, ... }
```

$ne — Not Equal To Operator

The `$ne` operator in MongoDB is used to match documents where the value of a field is **not equal to** a specified value.

Syntax

```
{ field: { $ne: value } }
```

Examples

1. Find products where category is not "Electronics"

```
db.products.find({ category: { $ne: "Electronics" } })
```

Retrieves products where the `category` field value is not `"Electronics"`.

Result: [

```
  { name: "Organic Coffee Beans", category: "Groceries", ... },
  { name: "Smart Speaker Echo", category: "Smart Home", ... },
  { name: "Yoga Mat Pro", category: "Fitness", ... }
]
```

2. Find products where brand is not "ErgoGear" (nested field)

```
db.products.find({ "details.brand": { $ne: "ErgoGear" } })
```

Retrieves products where the nested field `details.brand` value is not `"ErgoGear"`.

Result: [

```
  { name: "Mechanical Keyboard", details: { brand: "ClickyKeys" }, ... },
  { name: "Smart Speaker Echo", details: { brand: "HomeAI" }, ... }
]
```

3. Find products where is_available is not true

```
db.products.find({ is_available: { $ne: true } })
```

Retrieves products where the is_available field is not true.

Result:

```
{ name: "Smart Speaker Echo", is_available: false, ... }
```

$in — In Array Operator

The $in operator in MongoDB is used to match documents where the value of a field **equals any value in the specified array**.

Syntax

```
{ field: { $in: [value1, value2, ...] } }
```

Examples

1. Find products with tags containing "gaming" or "coffee"

```
db.products.find({ tags: { $in: ["gaming", "coffee"] } })
```

Retrieves products where the tags array contains either "gaming" or "coffee".

Result: [

```
  { name: "Mechanical Keyboard", tags: ["accessories", "gaming", "mechanical"], ... },
  { name: "Organic Coffee Beans", tags: ["coffee", "organic", "beverages"], ... }
]
```

2. Find products with category in ["Fitness", "Smart Home"]

```
db.products.find({ category: { $in: ["Fitness", "Smart Home"] } })
```

Retrieves products where the category field value is either "Fitness" or "Smart Home".

Result: [

```
 { name: "Smart Speaker Echo", category: "Smart Home", ... },
 { name: "Yoga Mat Pro", category: "Fitness", ... }
]
```

3. Find products where brand is in ["ClickyKeys", "HomeAI"] (nested field)

```
db.products.find({ "details.brand": { $in: ["ClickyKeys", "HomeAI"] } })
```

Retrieves products where the nested field `details.brand` value is either `"ClickyKeys"` or `"HomeAI"`.

Result: [

```
  { name: "Mechanical Keyboard", details: { brand: "ClickyKeys" }, ...},
 { name: "Smart Speaker Echo", details: { brand: "HomeAI" }, ... }
]
```

`$nin` — Not In Array Operator

The `$nin` operator in MongoDB is used to match documents where the value of a field **does not equal any value in the specified array**.

Syntax

```
{ field: { $nin: [value1, value2, ...] } }
```

Examples

1. Find products whose tags do not contain "organic"

```
db.products.find({ tags: { $nin: ["organic"] } })
```

Retrieves products where the `tags` array does not contain `"organic"`.

Result: [

```
   { name: "Wireless Mouse", tags: ["accessories", "wireless"], ... },
   { name: "Mechanical Keyboard", tags: ["accessories", "gaming", "mechanical"], ... },
   { name: "Smart Speaker Echo", tags: ["smart-home", "voice-assistant"], ... },
   { name: "Yoga Mat Pro", tags: ["fitness", "yoga", "accessories"], ... }
]
```

2. Find products with category not in ["Groceries", "Smart Home"]

```
db.products.find({ category: { $nin: ["Groceries", "Smart Home"] } })
```

Retrieves products where the `category` field value is neither `"Groceries"` nor `"Smart Home"`.

Result: [

```
 { name: "Wireless Mouse", category: "Electronics", ... },
 { name: "Mechanical Keyboard", category: "Electronics", ... },
 { name: "Yoga Mat Pro", category: "Fitness", ... }
]
```

3. Find products where brand is not in ["ErgoGear"] (nested field)

```
db.products.find({ "details.brand": { $nin: ["ErgoGear"] } })
```

Retrieves products where the nested field `details.brand` value is not `"ErgoGear"`.

Result:[

```
   { name: "Mechanical Keyboard", details: { brand: "ClickyKeys" }, ... },
 { name: "Smart Speaker Echo", details: { brand: "HomeAI" }, ... }
]
```