Introduction to the MongoDB $eq operator

The $eq operator is a comparison query operator that allows you to match documents where the value of a field equals a specified value.

The following shows the syntax of $eq operator:

```
{ <field>: { $eq: <value> } }
```

Code language: HTML, XML (xml)

The query is equivalent to the following:

```
{<field>: <value>}
```

Code language: HTML, XML (xml)

MongoDB $eq operator examples

We’ll use the following products collection for the demonstration:

```
db.products.insertMany([
   { "_id" : 1, "name" : "xPhone", "price" : 799, "releaseDate": ISODate("2011-05-14"), "spec" : { "ram" : 4, "screen" : 6.5, "cpu" : 2.66 },"color":["white","black"],"storage":[64,128,256]},
   { "_id" : 2, "name" : "xTablet", "price" : 899, "releaseDate": ISODate("2011-09-01") , "spec" : { "ram" : 16, "screen" : 9.5, "cpu" : 3.66 },"color":["white","black","purple"],"storage":[128,256,512]},
   { "_id" : 3, "name" : "SmartTablet", "price" : 899, "releaseDate": ISODate("2015-01-14"), "spec" : { "ram" : 12, "screen" : 9.7, "cpu" : 3.66 },"color":["blue"],"storage":[16,64,128]},
   { "_id" : 4, "name" : "SmartPad", "price" : 699, "releaseDate": ISODate("2020-05-14"),"spec" : { "ram" : 8, "screen" : 9.7, "cpu" : 1.66 },"color":["white","orange","gold","gray"],"storage":[128,256,1024]},
   { "_id" : 5, "name" : "SmartPhone", "price" : 599,"releaseDate": ISODate("2022-09-14"), "spec" : { "ram" : 4, "screen" : 9.7, "cpu" : 1.66 },"color":["white","orange","gold","gray"],"storage":[128,256]}
 ])
```

Code language: JavaScript (javascript)

1) Using $eq operator to check if a field equals a specified value

The following example uses the $eq operator to query the products collection to select all documents where the value of the price field equals 899:

```
db.products.find({
    price: {
        $eq: 899
    }
}, {
    name: 1,
    price: 1
})
```

Code language: CSS (css)

The query is equivalent to the following:

```
db.products.find({
    price: 899
}, {
    name: 1,
    price: 1
})
```

Code language: CSS (css)

They both match the following documents:

```
[
  { _id: 2, name: 'xTablet', price: 899 },
  { _id: 3, name: 'SmartTablet', price: 899 }
]
```

Code language: JavaScript (javascript)

2) Using the $eq operator to check if a field in an embedded document equals a value

The following example uses the $eq operator to search for documents where the value of the ram field in the spec document equals 4:

```
db.products.find({
    "spec.ram": {
        $eq: 4
    }
}, {
    name: 1,
    "spec.ram": 1
})
```

Code language: PHP (php)

It is equivalent to the following:

```
db.products.find({
    "spec.ram": 4


}, {
    name: 1,
    "spec.ram": 1
})
```

Code language: JavaScript (javascript)

Both of these queries returns the following documents:

```
[
  { _id: 1, name: 'xPhone', spec: { ram: 4 } },
  { _id: 5, name: 'SmartPhone', spec: { ram: 4 } }
]
```

Code language: JavaScript (javascript)

3) Using $eq operator to check if an array element equals a value

The following example uses the $eq operator to query the products collection to find all documents where the array color contains an element with the value "black":

```
db.products.find({
    color: {
        $eq: "black"
    }
}, {
   name: 1,
   color: 1
})
```

Code language: CSS (css)

It’s equivalent to:

```
db.products.find({
   color: "black"


}, {
   name: 1,
   color: 1
})
```

Code language: CSS (css)

Both queries return the following matching documents:

```
[
  { _id: 1, name: 'xPhone', color: [ 'white', 'black' ] },
  { _id: 2, name: 'xTablet', color: [ 'white', 'black', 'purple' ] }
]
```

Code language: JavaScript (javascript)

4) Using $eq operator to check if a field equals a date

The following example uses the $eq operator to select documents in the widget collection with the published date is 2020-05-14:

```
db.products.find({
   releaseDate: {
      $eq: new ISODate("2020-05-14")
   }
}, {
   name: 1,
   releaseDate: 1
})
```

Code language: CSS (css)

It returned the following document:

```
[
  {
    _id: 4,
    name: 'SmartPad',
    releaseDate: ISODate("2020-05-14T00:00:00.000Z")
  }
]
```

Code language: JavaScript (javascript)

Summary

- Use the $eq operator to specify an equality condition.

Introduction to the MongoDB $lt operator

The $lt operator is a comparison query operator that allows you to select the documents where the value of a field is less than a specified value.

Here is the syntax of the $lt operator:

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

Code language: CSS (css)

MongoDB $lt operator examples

We'll use the following products collection:

```
db.products.insertMany([
   { "_id" : 1, "name" : "xPhone", "price" : 799, "releaseDate": ISODate("2011-05-14"), "spec" : { "ram" : 4, "screen" : 6.5, "cpu" : 2.66 },"color":["white","black"],"storage":[64,128,256]},
   { "_id" : 2, "name" : "xTablet", "price" : 899, "releaseDate": ISODate("2011-09-01") , "spec" : { "ram" : 16, "screen" : 9.5, "cpu" : 3.66 },"color":["white","black","purple"],"storage":[128,256,512]},

   { "_id" : 3, "name" : "SmartTablet", "price" : 899, "releaseDate": ISODate("2015-01-14"), "spec" : { "ram" : 12, "screen" : 9.7, "cpu" : 3.66 },"color":["blue"],"storage":[16,64,128]},

   { "_id" : 4, "name" : "SmartPad", "price" : 699, "releaseDate": ISODate("2020-05-14"),"spec" : { "ram" : 8, "screen" : 9.7, "cpu" : 1.66 },"color":["white","orange","gold","gray"],"storage":[128,256,1024]},

   { "_id" : 5, "name" : "SmartPhone", "price" : 599,"releaseDate": ISODate("2022-09-14"), "spec" : { "ram" : 4, "screen" : 9.7, "cpu" : 1.66 },"color":["white","orange","gold","gray"],"storage":[128,256]}

])
```

Code language: JavaScript (javascript)

1) Using $lt operator to select documents where a field is less than a specified value

The following example uses the $lt operator to select documents from the products collection where price is less than 799:

```
db.products.find({
    price: {
        $lt: 799
    }
}, {
    name: 1,
    price: 1
})
```

Code language: CSS (css)

Output:

```
[
  { _id: 4, name: 'SmartPad', price: 699 },
  { _id: 5, name: 'SmartPhone', price: 599 }
]
```

Code language: JavaScript (javascript)

2) Using the $lt operator to check if a field in an embedded document is less than a value

The following query uses $lt operator to select documents where the value of the screen field in the spec document is less than 7:

```
db.products.find({
   "spec.screen": {
      $lt: 7
   }
}, {
   name: 1,
   "spec.screen": 1
})
```

Code language: PHP (php)

Output:

```
[ { _id: 1, name: 'xPhone', spec: { screen: 6.5 } } ]
```

Code language: CSS (css)

3) Using the $lt operator to check if an array element is greater than or equal to a value

The following example uses the $lt operator to query the products collection to find all documents where the array storage has at least one element less than 128:

```
db.products.find({
   storage: {
      $lt: 128
   }
}, {
   name: 1,
   storage: 1
})
```

Code language: CSS (css)

The query returned the following documents:

```
[
  { _id: 1, name: 'xPhone', storage: [ 64, 128, 256 ] },
  { _id: 3, name: 'SmartTablet', storage: [ 16, 64, 128 ] }
]
```

Code language: JavaScript (javascript)

4) Using the $lt operator to check if a field is before a date

The following query uses the $lt operator to select documents from the products collection to find all documents where the release date before 2015-01-01:

```
db.products.find({
    "releaseDate": {
        $lt: new ISODate('2015-01-01')
    }
}, {
    name: 1,
    releaseDate: 1
})
```

Code language: PHP (php)

The query returned the following documents:

```
[
  {
    _id: 1,
    name: 'xPhone',
    releaseDate: ISODate("2011-05-14T00:00:00.000Z")
  },
  {
    _id: 2,
    name: 'xTablet',
    releaseDate: ISODate("2011-09-01T00:00:00.000Z")
  }
]
```

Code language: JavaScript (javascript)

Summary

- Use the $lt operator to select documents where a field is less a specified value.

Introduction to the MongoDB $lte operator

The $lte is a comparison query operator that allows you to select documents where the value of a field is less than or equal to (<=) a specified value.

The following shows the $lte syntax:

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

Code language: CSS (css)

MongDB $lte operator examples

We'll use the following products collection:

```
db.products.drop();

db.products.insertMany([

   { "_id" : 1, "name" : "xPhone", "price" : 799, "releaseDate": ISODate("2011-05-14"), "spec" : { "ram" : 4, "screen" : 6.5, "cpu" : 2.66 },"color":["white","black"],"storage":[64,128,256]},

   { "_id" : 2, "name" : "xTablet", "price" : 899, "releaseDate": ISODate("2011-09-01") , "spec" : { "ram" : 16, "screen" : 9.5, "cpu" : 3.66 },"color":["white","black","purple"],"storage":[128,256,512]},

   { "_id" : 3, "name" : "SmartTablet", "price" : 899, "releaseDate": ISODate("2015-01-14"), "spec" : { "ram" : 12, "screen" : 9.7, "cpu" : 3.66 },"color":["blue"],"storage":[16,64,128]},

   { "_id" : 4, "name" : "SmartPad", "price" : 699, "releaseDate": ISODate("2020-05-14"),"spec" : { "ram" : 8, "screen" : 9.7, "cpu" : 1.66 },"color":["white","orange","gold","gray"],"storage":[128,256,1024]},

   { "_id" : 5, "name" : "SmartPhone", "price" : 599,"releaseDate": ISODate("2022-09-14"), "spec" : { "ram" : 4, "screen" : 5.7, "cpu" : 1.66 },"color":["white","orange","gold","gray"],"storage":[128,256]}

 ]);
```

Code language: JavaScript (javascript)

1) Using $lte operator to select documents where the value of a field is less than or equal to a specified value

The following example uses the $gte operator to select documents from
the products collection where price is less than 799:

```
db.products.find({
    price: {
        $lte: 799
    }
}, {
    name: 1,
    price: 1
})
```

Code language: CSS (css)

Output:

```
[
  { _id: 1, name: 'xPhone', price: 799 },
  { _id: 4, name: 'SmartPad', price: 699 },
  { _id: 5, name: 'SmartPhone', price: 599 }
]
```

Code language: JavaScript (javascript)

2) Using the $lte operator to check if the value of a field in an embedded document is less than or equal to a value

The following query uses $lte operator to select documents where the value of
the screen field in the spec document is less than or equal to 6.5:

```
db.products.find({
    "spec.screen": {
        $lte: 6.5
    }
}, {
    name: 1,
    "spec.screen": 1
})
```

Code language: PHP (php)

Output:

```
[
  { _id: 1, name: 'xPhone', spec: { screen: 6.5 } },
  { _id: 5, name: 'SmartPhone', spec: { screen: 5.7 } }
]
```

Code language: JavaScript (javascript)

3) Using the $lte operator to check if an array element is less than or equal to a value

The following example uses the $lte operator to query the products collection to find all documents where the array storage has at least one element less than or equal to 64:

```
db.products.find({
    storage: {
        $lte: 64
    }
}, {
    name: 1,
    storage: 1
})
```

Code language: CSS (css)

The query returned the following documents:

```
[
  { _id: 1, name: 'xPhone', storage: [ 64, 128, 256 ] },
  { _id: 3, name: 'SmartTablet', storage: [ 16, 64, 128 ] }
]
```

Code language: JavaScript (javascript)

4) Using the $lte operator to check if the value of a field is before or on the same date

The following query uses the $lte operator to select documents from the products collection to find all documents where the release date is before or on 2015-01-11:

```
db.products.find({
  "releaseDate": {
    $lte: new ISODate('2015-01-01')
  }
}, {
  name: 1,
  releaseDate: 1
});
```

Code language: PHP (php)

The query returned the following documents:

```
[
 {
  _id: 1,
  name: 'xPhone',
  releaseDate: ISODate("2011-05-14T00:00:00.000Z")
 },
 {
  _id: 2,
  name: 'xTablet',
  releaseDate: ISODate("2011-09-01T00:00:00.000Z")
 }
]
```

Code language: JavaScript (javascript)

Summary

- Use the $lte operator to select documents where the value of a field is less than or equal to a specified value.

Introduction to the MongoDB $gt operator

The $gt operator is a comparison query operator that allows you to select documents where the value of a field is greater than (>) a specified value.

The following shows the syntax of the $gt operator:

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

Code language: CSS (css)

MongoDB $gt operator example

We'll use the following widget collection:

```
db.products.insertMany([
   { "_id" : 1, "name" : "xPhone", "price" : 799, "releaseDate": ISODate("2011-05-14"), "spec" : { "ram" : 4, "screen" : 6.5, "cpu" : 2.66 },"color":["white","black"],"storage":[64,128,256]},
   { "_id" : 2, "name" : "xTablet", "price" : 899, "releaseDate": ISODate("2011-09-01") , "spec" : { "ram" : 16, "screen" : 9.5, "cpu" : 3.66 },"color":["white","black","purple"],"storage":[128,256,512]},
   { "_id" : 3, "name" : "SmartTablet", "price" : 899, "releaseDate": ISODate("2015-01-14"), "spec" : { "ram" : 12, "screen" : 9.7, "cpu" : 3.66 },"color":["blue"],"storage":[16,64,128]},
   { "_id" : 4, "name" : "SmartPad", "price" : 699, "releaseDate": ISODate("2020-05-14"),"spec" : { "ram" : 8, "screen" : 9.7, "cpu" : 1.66 },"color":["white","orange","gold","gray"],"storage":[128,256,1024]},
   { "_id" : 5, "name" : "SmartPhone", "price" : 599,"releaseDate": ISODate("2022-09-14"), "spec" : { "ram" : 4, "screen" : 9.7, "cpu" : 1.66 },"color":["white","orange","gold","gray"],"storage":[128,256]}
 ])
```

Code language: JavaScript (javascript)

1) Using $gt to select documents where the value of a field is greater than a specified value

The following example uses the $gt operator to select documents from the products collection where price is greater than 699:

```
db.products.find({
   price: {
      $gt: 699
```

```
    }
}, {
    name: 1,
    price: 1
})
```

Code language: CSS (css)

The query returned the following documents:

```
[
  { _id: 1, name: 'xPhone', price: 799 },
  { _id: 2, name: 'xTablet', price: 899 },
  { _id: 3, name: 'SmartTablet', price: 899 }
]
```

Code language: JavaScript (javascript)

2) Using the $gt operator to check if the value of a field in an embedded document is greater than a value

The following example uses $gt operator to select documents where the value of the ram field in the spec document is greater than 8:

```
db.products.find({
    "spec.ram": {
        $gt: 8
    }
}, {
    name: 1,
    "spec.ram": 1
});
```

Code language: PHP (php)

Output:

```
[
  { _id: 2, name: 'xTablet', spec: { ram: 16 } },
  { _id: 3, name: 'SmartTablet', spec: { ram: 12 } }
]
```

Code language: JavaScript (javascript)

3) Using the $gt operator to check if an array element is greater than a value

The following example uses the $gt operator to query the products collection to find all documents where the storage array has at least one element greater than 128:

```
db.products.find({
    storage: {
        $gt: 128
    }
}, {
    name: 1,
    storage: 1
})
```

Code language: CSS (css)

The query returned the following documents:

```
[
  { _id: 1, name: 'xPhone', storage: [ 64, 128, 256 ] },
  { _id: 2, name: 'xTablet', storage: [ 128, 256, 512 ] },
  { _id: 4, name: 'SmartPad', storage: [ 128, 256, 1024 ] },
  { _id: 5, name: 'SmartPhone', storage: [ 128, 256 ] }
]
```

Code language: JavaScript (javascript)

4) Using the $gt operator to check if the value of a field is after a date

The following example uses the $gt operator to query documents from
the products collection to find all documents where the release date is after 2015-01-01:

```
db.products.find({
    "releaseDate": {
        $gt: new ISODate('2015-01-01')
    }
}, {
    name: 1,
    releaseDate: 1
});
```

Code language: PHP (php)

The query returned the following documents:

```
[
  {
    _id: 3,
    name: 'SmartTablet',
    releaseDate: ISODate("2015-01-14T00:00:00.000Z")
  },
  {
    _id: 4,
    name: 'SmartPad',
    releaseDate: ISODate("2020-05-14T00:00:00.000Z")
  },
  {
    _id: 5,
    name: 'SmartPhone',
    releaseDate: ISODate("2022-09-14T00:00:00.000Z")
  }
]
```

Code language: JavaScript (javascript)

Summary

- Use the $gt operator to select documents where a field is greater than a specified value.

Introduction to the MongoDB $gte operator

The $gte is a comparison query operator that allows you to select documents where a value of a field is greater than or equal to (i.e. >=) a specified value.

The $gte operator has the following syntax:

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

Code language: CSS (css)

MongDB $gte operator examples

We’ll use the following products collection:

```
db.products.insertMany([
   { "_id" : 1, "name" : "xPhone", "price" : 799, "releaseDate": ISODate("2011-05-14"), "spec" : { "ram" : 4, "screen" : 6.5, "cpu" : 2.66 },"color":["white","black"],"storage":[64,128,256]},
   { "_id" : 2, "name" : "xTablet", "price" : 899, "releaseDate": ISODate("2011-09-01") , "spec" : { "ram" : 16, "screen" : 9.5, "cpu" : 3.66 },"color":["white","black","purple"],"storage":[128,256,512]},
   { "_id" : 3, "name" : "SmartTablet", "price" : 899, "releaseDate": ISODate("2015-01-14"), "spec" : { "ram" : 12, "screen" : 9.7, "cpu" : 3.66 },"color":["blue"],"storage":[16,64,128]},
   { "_id" : 4, "name" : "SmartPad", "price" : 699, "releaseDate": ISODate("2020-05-14"),"spec" : { "ram" : 8, "screen" : 9.7, "cpu" : 1.66 },"color":["white","orange","gold","gray"],"storage":[128,256,1024]},
   { "_id" : 5, "name" : "SmartPhone", "price" : 599,"releaseDate": ISODate("2022-09-14"), "spec" : { "ram" : 4, "screen" : 5.7, "cpu" : 1.66 },"color":["white","orange","gold","gray"],"storage":[128,256]}
 ])
```

Code language: JavaScript (javascript)

1) Using $gte operator to select documents where a field is greater than or equal to a specified value

The following example uses the $gte operator to select documents from the products collection where price is greater than 799:

```
db.products.find({
   price: {
      $gte: 799
   }
}, {
   name: 1,
   price: 1
})
```

Code language: CSS (css)

The query returned the following documents:

```
[
  { _id: 1, name: 'xPhone', price: 799 },
  { _id: 2, name: 'xTablet', price: 899 },
  { _id: 3, name: 'SmartTablet', price: 899 }
]
```

Code language: JavaScript (javascript)

2) Using the $gte operator to check if a field in an embedded document is greater than or equal to a value

The following query uses $gte operator to select documents where the value of the screen field in the spec document is greater than or equal to 9.5:

```
db.products.find({
   "spec.screen": {
       $gte: 9.5
   }
}, {
   name: 1,
   "spec.screen": 1
})
```

Code language: PHP (php)

Output:

```
[
  { _id: 2, name: 'xTablet', spec: { screen: 9.5 } },
  { _id: 3, name: 'SmartTablet', spec: { screen: 9.7 } },
  { _id: 4, name: 'SmartPad', spec: { screen: 9.7 } },
  { _id: 5, name: 'SmartPhone', spec: { screen: 9.7 } }
]
```

Code language: JavaScript (javascript)

3) Using the $gte operator to check if an array element is greater than or equal to a value

The following example uses the $gte operator to query the products collection to find all documents where the array storage has at least one element greater than or equal to 512:

```
db.products.find({
   storage: {
      $gte: 512
   }
}, {
   name: 1,
   storage: 1
})
```

Code language: CSS (css)

The query returned the following documents:

```
[
 { _id: 2, name: 'xTablet', storage: [ 128, 256, 512 ] },
 { _id: 4, name: 'SmartPad', storage: [ 128, 256, 1024 ] }
]
```

Code language: JavaScript (javascript)

4) Using the $gte operator to check if a field is after or on the same date

The following query uses the $gte operator to select documents from the products collection to find all documents where the release date is after or on 2020-05-14:

```
db.products.find({
   "releaseDate": {
     $gte: new ISODate('2020-05-14')
   }
}, {
   name: 1,
   releaseDate: 1
});
```

Code language: PHP (php)

The query returned the following documents:

```
[
  {
    _id: 4,
    name: 'SmartPad',
    releaseDate: ISODate("2020-05-14T00:00:00.000Z")
  },
  {
    _id: 5,
    name: 'SmartPhone',
    releaseDate: ISODate("2022-09-14T00:00:00.000Z")
  }
]
```

Code language: JavaScript (javascript)

Summary

- Use the $gte operator to select documents where a field is greater than or equal to a specified value.

Introduction to MongoDB $ne operator

The $ne is a comparison query operator that allows you to select documents where the value of a filed is **not equal to** a specified value. It also includes documents that **don't contain the field**.

The $ne is called the **inequality operator**. Here is the syntax of the $ne operator:

{ field: {$ne: value}}

Code language: CSS (css)

MongoDB $ne operator examples

We'll use the following products collection:

db.products.insertMany([

{ "_id" : 1, "name" : "xPhone", "price" : 799, "releaseDate": ISODate("2011-05-14"), "spec" : { "ram" : 4, "screen" : 6.5, "cpu" : 2.66 },"color":["white","black"],"storage":[64,128,256]},

{ "_id" : 2, "name" : "xTablet", "price" : 899, "releaseDate": ISODate("2011-09-01") , "spec" : { "ram" : 16, "screen" : 9.5, "cpu" : 3.66 },"color":["white","black","purple"],"storage":[128,256,512]},

{ "_id" : 3, "name" : "SmartTablet", "price" : 899, "releaseDate": ISODate("2015-01-14"), "spec" : { "ram" : 12, "screen" : 9.7, "cpu" : 3.66 },"color":["blue"],"storage":[16,64,128]},

{ "_id" : 4, "name" : "SmartPad", "price" : 699, "releaseDate": ISODate("2020-05-14"),"spec" : { "ram" : 8, "screen" : 9.7, "cpu" : 1.66 },"color":["white","orange","gold","gray"],"storage":[128,256,1024]},

{ "_id" : 5, "name" : "SmartPhone", "price" : 599,"releaseDate": ISODate("2022-09-14"), "spec" : { "ram" : 4, "screen" : 9.7, "cpu" : 1.66 },"color":["white","orange","gold","gray"],"storage":[128,256]}

,

{ "_id" : 6, "name" : "xWidget", "spec" : { "ram" : 64, "screen" : 9.7, "cpu" : 3.66 },"color":["black"],"storage":[1024]}

])

Code language: JavaScript (javascript)

1) Using the $ne operator to select documents where the value of a field is greater than a specified value

The following example uses the $ne operator to select documents from the products collection where the price is not equal to 899:

db.products.find({

price: {

$ne: 899

```
    }
}, {
    name: 1,
    price: 1
})
```

Code language: CSS (css)

It matches the following documents:

```
[
  { _id: 1, name: 'xPhone', price: 799 },
  { _id: 4, name: 'SmartPad', price: 699 },
  { _id: 5, name: 'SmartPhone', price: 599 },
  { _id: 6, name: 'xWidget' }
]
```

Code language: JavaScript (javascript)

2) Using the $ne operator to check if a field in an embedded document is not equal to a value

The following example uses $ne operator to select documents where the value of the screen field in the spec document is not equal to 9.7:

```
db.products.find({
    "spec.screen": {
        $ne: 9.7
    }
}, {
    name: 1,
    "spec.screen": 1
})
```

Code language: PHP (php)

Output:

```
[
  { _id: 1, name: 'xPhone', spec: { screen: 6.5 } },
  { _id: 2, name: 'xTablet', spec: { screen: 9.5 } }
]
```

Code language: JavaScript (javascript)

3) Using the $ne operator to check if an array element is not equal to a value

The following example uses the $ne operator to query the products collection to find documents where the array storage does not have any element that equals 128:

```
db.products.find({
    storage: {
        $ne: 128
    }
}, {
    name: 1,
    storage: 1
});
```

Code language: CSS (css)

It matched the following documents:

```
[ { _id: 6, name: 'xWidget', storage: [ 1024 ] } ]
```

Code language: CSS (css)

4) Using the $ne operator to check if the value of a field is not equal to a date

The following query uses the $ne operator to find documents from the products collection where the release date is not 2015-01-14:

```
db.products.find({
    releaseDate: {
        $ne: new ISODate('2015-01-14')
    }
}, {
    name: 1,
    releaseDate: 1
});
```

Code language: CSS (css)

It returns the documents whose release dates are not 2015-01-14 and also the document that does not include the field releaseDate :

```
[
 {
   _id: 1,
   name: 'xPhone',
   releaseDate: ISODate("2011-05-14T00:00:00.000Z")
 },
 {
   _id: 2,
   name: 'xTablet',
   releaseDate: ISODate("2011-09-01T00:00:00.000Z")
 },
 {
   _id: 4,
   name: 'SmartPad',
   releaseDate: ISODate("2020-05-14T00:00:00.000Z")
 },
 {
   _id: 5,
   name: 'SmartPhone',
   releaseDate: ISODate("2022-09-14T00:00:00.000Z")
 },
 { _id: 6, name: 'xWidget' }
]
```

Code language: JavaScript (javascript)

Summary

- Use the $ne operator to check if the value of a field **is not equal** to a specified value.

Introduction to the MongoDB $in operator

The $in is a comparison query operator that allows you to select documents where the value of a field is equal to any value in an array.

The following shows the syntax of the $in operator:

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

Code language: CSS (css)

If the field holds a single value, then the $in operator selects documents where the value of the field is equal to any value such as <value1>, <value2>.

In case the field holds an array, the $in operator selects documents where the array contains at least one element that equals any value (<value1>, <value2>).

The value list <value1>, <value2>, etc., can be a list of literal values or regular expressions.

A regular expression is a set of characters that defines a search pattern e.g., /\d+/ any digits such as 1, 123, and 1234.

MongoDB $in operator examples

We'll use this products collections in the following examples:

```
db.products.insertMany([
   { "_id" : 1, "name" : "xPhone", "price" : 799, "releaseDate": ISODate("2011-05-14"), "spec" : { "ram" : 4, "screen" : 6.5, "cpu" : 2.66 },"color":["white","black"],"storage":[64,128,256]},
   { "_id" : 2, "name" : "xTablet", "price" : 899, "releaseDate": ISODate("2011-09-01") , "spec" : { "ram" : 16, "screen" : 9.5, "cpu" : 3.66 },"color":["white","black","purple"],"storage":[128,256,512]},
   { "_id" : 3, "name" : "SmartTablet", "price" : 899, "releaseDate": ISODate("2015-01-14"), "spec" : { "ram" : 12, "screen" : 9.7, "cpu" : 3.66 },"color":["blue"],"storage":[16,64,128]},
   { "_id" : 4, "name" : "SmartPad", "price" : 699, "releaseDate": ISODate("2020-05-14"),"spec" : { "ram" : 8, "screen" : 9.7, "cpu" : 1.66 },"color":["white","orange","gold","gray"],"storage":[128,256,1024]},
   { "_id" : 5, "name" : "SmartPhone", "price" : 599,"releaseDate": ISODate("2022-09-14"), "spec" : { "ram" : 4, "screen" : 5.7, "cpu" : 1.66 },"color":["white","orange","gold","gray"],"storage":[128,256]}
```

```
 ])
```

Code language: JavaScript (javascript)

1) Using the $in opeator to match values

The following example uses the $in operator to select documents from
the products collection whose the price is either 599 or 799:

```
db.products.find({
   price: {
      $in: [699, 799]
   }
}, {
   name: 1,
   price: 1
})
```

Code language: CSS (css)

It returned the following documents:

```
[
  { _id: 1, name: 'xPhone', price: 799 },
  { _id: 4, name: 'SmartPad', price: 699 }
]
```

Code language: JavaScript (javascript)

2) Using the $in operator to match values in an array

The products collection has the color array that contains some colors.

The following example uses the $in operator to select documents where the color array has at least one element either "black" or "white":

```
db.products.find({
   color: {
      $in: ["black", "white"]
   }
}, {
   name: 1,
   color: 1
})
```

Code language: CSS (css)

The query returned the following documents:

```
[
  { _id: 1, name: 'xPhone', color: [ 'white', 'black' ] },
  { _id: 2, name: 'xTablet', color: [ 'white', 'black', 'purple' ] },
  {
    _id: 4,
    name: 'SmartPad',
    color: [ 'white', 'orange', 'gold', 'gray' ]
  },
  {
    _id: 5,
    name: 'SmartPhone',
    color: [ 'white', 'orange', 'gold', 'gray' ]
  }
]
```

Code language: JavaScript (javascript)

3) Using the $in operator with regular expressions

The following query uses the $in operator to find documents where the color array has at least one element that matches either /^g+/ or /^w+/ regular expression:

```
db.products.find({
    color: {
        $in: [/^g+/, /^w+/]
    }
}, {
    name: 1,
    color: 1
})
```

Code language: CSS (css)

It returned the following documents:

```
[
  { _id: 1, name: 'xPhone', color: [ 'white', 'black' ] },
  { _id: 2, name: 'xTablet', color: [ 'white', 'black', 'purple' ] },
  {
    _id: 4,
    name: 'SmartPad',
    color: [ 'white', 'orange', 'gold', 'gray' ]
  },
  {
    _id: 5,
    name: 'SmartPhone',
    color: [ 'white', 'orange', 'gold', 'gray' ]
  }
]
```

Code language: JavaScript (javascript)

The /^g+/ regular expression matches any string that begins with the letter g and is followed by any number of characters (+). Similarly, the /^w+/ regular expression matches any string that starts with the letter w and is followed by any number of characters (+). This tutorial explains the regular expressions in JavaScript in detail.

Summary

- Use the MongoDB $in operator to select documents where the value of a field is equal to any values in an array.
- The values can be a list of literal values or regular expressions.

Introduction to the MongoDB $nin operator

The $nin is a query comparison operator that allows you to find documents where:

- the value of the field is not equal to any value in an array
- or the field does not exist.

Here is the syntax of the $nin operator:

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

Code language: CSS (css)

Like the $in operator, the value list (<value1>, <value2>,…) can be a list of literal values or regular expressions.

MongoDB $nin operator examples

We'll use this products collections:

db.products.insertMany([

{ "_id" : 1, "name" : "xPhone", "price" : 799, "releaseDate": ISODate("2011-05-14"), "spec" : { "ram" : 4, "screen" : 6.5, "cpu" : 2.66 },"color":["white","black"],"storage":[64,128,256]},

{ "_id" : 2, "name" : "xTablet", "price" : 899, "releaseDate": ISODate("2011-09-01") , "spec" : { "ram" : 16, "screen" : 9.5, "cpu" : 3.66 },"color":["white","black","purple"],"storage":[128,256,512]},

{ "_id" : 3, "name" : "SmartTablet", "price" : 899, "releaseDate": ISODate("2015-01-14"), "spec" : { "ram" : 12, "screen" : 9.7, "cpu" : 3.66 },"color":["blue"],"storage":[16,64,128]},

{ "_id" : 4, "name" : "SmartPad", "price" : 699, "releaseDate": ISODate("2020-05-14"),"spec" : { "ram" : 8, "screen" : 9.7, "cpu" : 1.66 },"color":["white","orange","gold","gray"],"storage":[128,256,1024]},

{ "_id" : 5, "name" : "SmartPhone", "price" : 599,"releaseDate": ISODate("2022-09-14"), "spec" : { "ram" : 4, "screen" : 5.7, "cpu" : 1.66 },"color":["white","orange","gold","gray"],"storage":[128,256]}

])

Code language: JavaScript (javascript)

1) Using the MongoDB $nin opeator to match values

The following query uses the $nin operator to select documents from the products collection whose price is neither 599 or 799:

db.products.find({

```
   price: {
      $nin: [699, 799]
   }
}, {
   name: 1,
   price: 1
})
```

Code language: CSS (css)

It returned the following documents:

```
[
  { _id: 2, name: 'xTablet', price: 899 },
  { _id: 3, name: 'SmartTablet', price: 899 },
  { _id: 5, name: 'SmartPhone', price: 599 }
]
```

Code language: JavaScript (javascript)

2) Using the MongoDB $nin operator to match values in an array

The following example uses the $nin operator to select documents where the color array doesn’t have an element that is either "black" or "white":

```
db.products.find({
   color: {
      $nin: ["black", "white"]
   }
}, {
   name: 1,
   color: 1
})
```

Code language: CSS (css)

The query returned the following documents:

```
[ { _id: 3, name: 'SmartTablet', color: [ 'blue' ] } ]
```

Code language: CSS (css)

3) Using the MongoDB $nin operator with regular expressions

The following query uses the $nin operator to find documents where the color array doesn’t have an element that matches /^g+/ and /^w+/ regular expression:

```
db.products.find({
  color: {
    $nin: [/^g+/, /^w+/]
  }
}, {
  name: 1,
  color: 1
})
```

Code language: CSS (css)

It returned the following documents:

```
[ { _id: 3, name: 'SmartTablet', color: [ 'blue' ] } ]
```

Code language: CSS (css)

Summary

- Use the MongoDB $nin operator to select documents where the value of a field is not equal to any values in an array.
- The value list can contain literal values or regular expressions.