



This version of the manual is no longer supported.

MongoDB CRUD Operations > MongoDB CRUD Tutorials > Query Documents

# Query Documents

## On this page

- [Select All Documents in a Collection](#)
- [Specify Equality Condition](#)
- [Specify Conditions Using Query Operators](#)
- [Specify AND Conditions](#)
- [Specify OR Conditions](#)
- [Specify AND as well as OR Conditions](#)
- [Embedded Documents](#)
- [Arrays](#)

In MongoDB, the `db.collection.find()` method retrieves documents from a collection. [1] The `db.collection.find()` method returns a cursor to the retrieved documents.

This tutorial provides examples of read operations using the `db.collection.find()` method in the mongo shell. In these examples, the retrieved documents contain all their fields. To restrict the fields to return in the retrieved documents, see [Limit Fields to Return from a Query](#).

[1] The `db.collection.findOne()` method also performs a read operation to return a single document. Internally, the `db.collection.findOne()` method is the `db.collection.find()` method with a limit of 1.

## Select All Documents in a Collection

An empty query document (`{}`) selects all documents in the collection:

```
db.inventory.find( {} )
```

Not specifying a query document to the `find()` is equivalent to specifying an empty query document. Therefore the following operation is equivalent to the previous operation:

```
db.inventory.find()
```

## Specify Equality Condition

To specify equality condition, use the query document `{ <field>: <value> }` to select all documents that contain the `<field>` with the specified `<value>`.

The following example retrieves from the `inventory` collection all documents where the `type` field has the value `snacks`:

```
db.inventory.find( { type: "snacks" } )
```

## Specify Conditions Using Query Operators

A query document can use the query operators to specify conditions in a MongoDB query.

The following example selects all documents in the `inventory` collection where the value of the `type` field is either `'food'` or `'snacks'`:

```
db.inventory.find( { type: { $in: [ 'food', 'snacks' ] } } )
```

Although you can express this query using the `$or` operator, use the `$in` operator rather than the `$or` operator when performing equality checks on the same field.

Refer to the [Query and Projection Operators](#) document for the complete list of query operators.

## Specify AND Conditions

A compound query can specify conditions for more than one field in the collection's documents. Implicitly, a logical **AND** conjunction connects the clauses of a compound query so that the query selects the documents in the collection that match all the conditions.

In the following example, the query document specifies an equality match on the field **type** **and** a less than (**\$lt**) comparison match on the field **price**:

```
db.inventory.find( { type: 'food', price: { $lt: 9.95 } } )
```

This query selects all documents where the **type** field has the value **'food'** **and** the value of the **price** field is less than **9.95**. See comparison operators for other comparison operators.

## Specify OR Conditions

Using the **\$or** operator, you can specify a compound query that joins each clause with a logical **OR** conjunction so that the query selects the documents in the collection that match at least one condition.


In the following example, the query document selects all documents in the collection where the field **qty** has a value greater than (**\$gt**) **100** **or** the value of the **price** field is less than (**\$lt**) **9.95**:

```
db.inventory.find(
  {
    $or: [ { qty: { $gt: 100 } }, { price: { $lt: 9.95 } } ]
  }
)
```

## Specify AND as well as OR Conditions

With additional clauses, you can specify precise conditions for matching documents.

In the following example, the compound query document selects all documents in the collection where the value of the **type** field is **'food'** **and** *either* the **qty** has a value greater than (**\$gt**) **100** *or* the value of the **price** field is less than (**\$lt**) **9.95**:



```
db.inventory.find(
  {
    type: 'food',
    $or: [ { qty: { $gt: 100 } }, { price: { $lt: 9.95 } } ]
  }
)
```

## Embedded Documents

When the field holds an embedded document, a query can either specify an exact match on the embedded document or specify a match by individual fields in the embedded document using the dot notation.

### Exact Match on the Embedded Document

To specify an equality match on the whole embedded document, use the query document { <field>: <value> } where <value> is the document to match. Equality matches on an embedded document require an *exact* match of the specified <value>, including the field order.

In the following example, the query matches all documents where the value of the field `producer` is an embedded document that contains *only* the field `company` with the value `'ABC123'` and the field `address` with the value `'123 Street'`, in the exact order:

```
db.inventory.find(
  {
    producer:
      {
        company: 'ABC123',
        address: '123 Street'
      }
  }
)
```

### Equality Match on Fields within an Embedded Document

Use the dot notation to match by specific fields in an embedded document. Equality matches for specific fields in an embedded document will select documents in the collection where the embedded document contains the specified fields with the specified values. The embedded document can contain additional fields.

In the following example, the query uses the dot notation to match all documents where the value of the field `producer` is an embedded document that contains a field `company` with the value `'ABC123'` and may contain other fields:

```
db.inventory.find( { 'producer.company': 'ABC123' } )
```

## Arrays

When the field holds an array, you can query for an exact array match or for specific values in the array. If the array holds embedded documents, you can query for specific fields in the embedded documents using dot notation.

If you specify multiple conditions using the `$elemMatch` operator, the array must contain at least one element that satisfies all the conditions. See [Single Element Satisfies the Criteria](#).

If you specify multiple conditions without using the `$elemMatch` operator, then some combination of the array elements, not necessarily a single element, must satisfy all the conditions; i.e. different elements in the array can satisfy different parts of the conditions. See [Combination of Elements Satisfies the Criteria](#).

Consider an `inventory` collection that contains the following documents:

```
{ _id: 5, type: "food", item: "aaa", ratings: [ 5, 8, 9 ] }  
{ _id: 6, type: "food", item: "bbb", ratings: [ 5, 9 ] }  
{ _id: 7, type: "food", item: "ccc", ratings: [ 9, 5, 8 ] }
```

### Exact Match on an Array

To specify equality match on an array, use the query document `{ <field>: <value> }` where `<value>` is the array to match. Equality matches on the array require that the array field match *exactly* the specified `<value>`, including the element order.

The following example queries for all documents where the field `ratings` is an array that holds exactly three elements, 5, 8, and 9, in this order:

```
db.inventory.find( { ratings: [ 5, 8, 9 ] } )
```

The operation returns the following document:

```
{ "_id" : 5, "type" : "food", "item" : "aaa", "ratings" : [ 5, 8, 9 ] }
```

## Match an Array Element

Equality matches can specify a single element in the array to match. These specifications match if the array contains at least *one* element with the specified value.

The following example queries for all documents where `ratings` is an array that contains 5 as one of its elements:

```
db.inventory.find( { ratings: 5 } )
```

The operation returns the following documents:

```
{ "_id" : 5, "type" : "food", "item" : "aaa", "ratings" : [ 5, 8, 9 ] }  
{ "_id" : 6, "type" : "food", "item" : "bbb", "ratings" : [ 5, 9 ] }  
{ "_id" : 7, "type" : "food", "item" : "ccc", "ratings" : [ 9, 5, 8 ] }
```

## Match a Specific Element of an Array

Equality matches can specify equality matches for an element at a particular index or position of the array using the dot notation.

In the following example, the query uses the dot notation to match all documents where the `ratings` array contains 5 as the first element:



```
db.inventory.find( { 'ratings.0': 5 } )
```

The operation returns the following documents:

```
{ "_id" : 5, "type" : "food", "item" : "aaa", "ratings" : [ 5, 8, 9 ] }  
{ "_id" : 6, "type" : "food", "item" : "bbb", "ratings" : [ 5, 9 ] }
```

## Specify Multiple Criteria for Array Elements

### Single Element Satisfies the Criteria

Use the `$elemMatch` operator to specify multiple criteria on the elements of an array such that at least one array element satisfies all the specified criteria.

The following example queries for documents where the `ratings` array contains at least one element that is greater than (`$gt`) 5 and less than (`$lt`) 9:

```
db.inventory.find( { ratings: { $elemMatch: { $gt: 5, $lt: 9 } } } )
```

The operation returns the following documents, whose `ratings` array contains the element 8 which meets the criteria:

```
{ "_id" : 5, "type" : "food", "item" : "aaa", "ratings" : [ 5, 8, 9 ] }  
{ "_id" : 7, "type" : "food", "item" : "ccc", "ratings" : [ 9, 5, 8 ] }
```

### Combination of Elements Satisfies the Criteria

The following example queries for documents where the `ratings` array contains elements that in some combination satisfy the query conditions; e.g., one element can satisfy the greater than 5 condition and another element can satisfy the less than 9 condition, or a single element can satisfy both:



```
db.inventory.find( { ratings: { $gt: 5, $lt: 9 } } )
```

The operation returns the following documents:

```
{ "_id" : 5, "type" : "food", "item" : "aaa", "ratings" : [ 5, 8, 9 ] }
{ "_id" : 6, "type" : "food", "item" : "bbb", "ratings" : [ 5, 9 ] }
{ "_id" : 7, "type" : "food", "item" : "ccc", "ratings" : [ 9, 5, 8 ] }
```

The document with the "ratings" : [ 5, 9 ] matches the query since the element 9 is greater than 5 (the first condition) and the element 5 is less than 9 (the second condition).

## Array of Embedded Documents

Consider that the `inventory` collection includes the following documents:

```
{
  _id: 100,
  type: "food",
  item: "xyz",
  qty: 25,
  price: 2.5,
  ratings: [ 5, 8, 9 ],
  memos: [ { memo: "on time", by: "shipping" }, { memo: "approved", by: "billing" } ]
}

{
  _id: 101,
  type: "fruit",
  item: "jkl",
  qty: 10,
  price: 4.25,
  ratings: [ 5, 9 ],
  memos: [ { memo: "on time", by: "payment" }, { memo: "delayed", by: "shipping" } ]
}
```



## Match a Field in the Embedded Document Using the Array Index



If you know the array index of the embedded document, you can specify the document using the embedded document's position using the dot notation.

The following example selects all documents where the `memos` contains an array whose first element (i.e. index is 0) is a document that contains the field `by` whose value is `'shipping'`:

```
db.inventory.find( { 'memos.0.by': 'shipping' } )
```

The operation returns the following document:

```
{
  _id: 100,
  type: "food",
  item: "xyz",
  qty: 25,
  price: 2.5,
  ratings: [ 5, 8, 9 ],
  memos: [ { memo: "on time", by: "shipping" }, { memo: "approved", by: "billing" } ]
}
```

## Match a Field Without Specifying Array Index

If you do not know the index position of the document in the array, concatenate the name of the field that contains the array, with a dot (.) and the name of the field in the embedded document.

The following example selects all documents where the `memos` field contains an array that contains at least one embedded document that contains the field `by` with the value `'shipping'`:

```
db.inventory.find( { 'memos.by': 'shipping' } )
```

The operation returns the following documents:


```
mongoDB
{
  _id: 100,
  type: "food",
  item: "xyz",
  qty: 25,
  price: 2.5,
  ratings: [ 5, 8, 9 ],
  memos: [ { memo: "on time", by: "shipping" }, { memo: "approved", by: "billing" } ]
}
{
  _id: 101,
  type: "fruit",
  item: "jkl",
  qty: 10,
  price: 4.25,
  ratings: [ 5, 9 ],
  memos: [ { memo: "on time", by: "payment" }, { memo: "delayed", by: "shipping" } ]
}
```

## Specify Multiple Criteria for Array of Documents

### Single Element Satisfies the Criteria

Use the `$elemMatch` operator to specify multiple criteria on an array of embedded documents such that at least one embedded document satisfies all the specified criteria.

The following example queries for documents where the `memos` array has at least one embedded document that contains both the field `memo` equal to `'on time'` and the field `by` equal to `'shipping'`:




```
db.inventory.find(
  {
    memos:
      {
        $elemMatch:
          {
            memo: 'on time',
            by: 'shipping'
          }
      }
  }
)
```

The operation returns the following document:

```
{
  _id: 100,
  type: "food",
  item: "xyz",
  qty: 25,
  price: 2.5,
  ratings: [ 5, 8, 9 ],
  memos: [ { memo: "on time", by: "shipping" }, { memo: "approved", by: "billing" } ]
}
```

### Combination of Elements Satisfies the Criteria

The following example queries for documents where the `memos` array contains elements that in some combination satisfy the query conditions; e.g. one element satisfies the field `memo` equal to `'on time'` condition and another element satisfies the field `by` equal to `'shipping'` condition, or a single element can satisfy both criteria:



```
db.inventory.find(  
  {  
    'memos.memo': 'on time',  
    'memos.by': 'shipping'  
  }  
)
```

The query returns the following documents:

```
{  
  _id: 100,  
  type: "food",  
  item: "xyz",  
  qty: 25,  
  price: 2.5,  
  ratings: [ 5, 8, 9 ],  
  memos: [ { memo: "on time", by: "shipping" }, { memo: "approved", by: "billing" } ]  
}  
{  
  _id: 101,  
  type: "fruit",  
  item: "jkl",  
  qty: 10,  
  price: 4.25,  
  ratings: [ 5, 9 ],  
  memos: [ { memo: "on time", by: "payment" }, { memo: "delayed", by: "shipping" } ]  
}
```

#### SEE ALSO:

[Limit Fields to Return from a Query](#)