# Lab 12: MongoDB CRUD Operations

The learning objectives of this lab are to:

- **Query  Documents**
- **Update Documents**
- **Delete  Documents**

## 1. MongoDB Query Document using find()

The method of fetching or getting data from a MongoDB database is carried out by using queries. While performing a query operation, one can also use criteria's or conditions which can be used to retrieve specific data from the database.

MongoDB provides a function called db.collection.find () which is used for retrieval of documents from a MongoDB database.

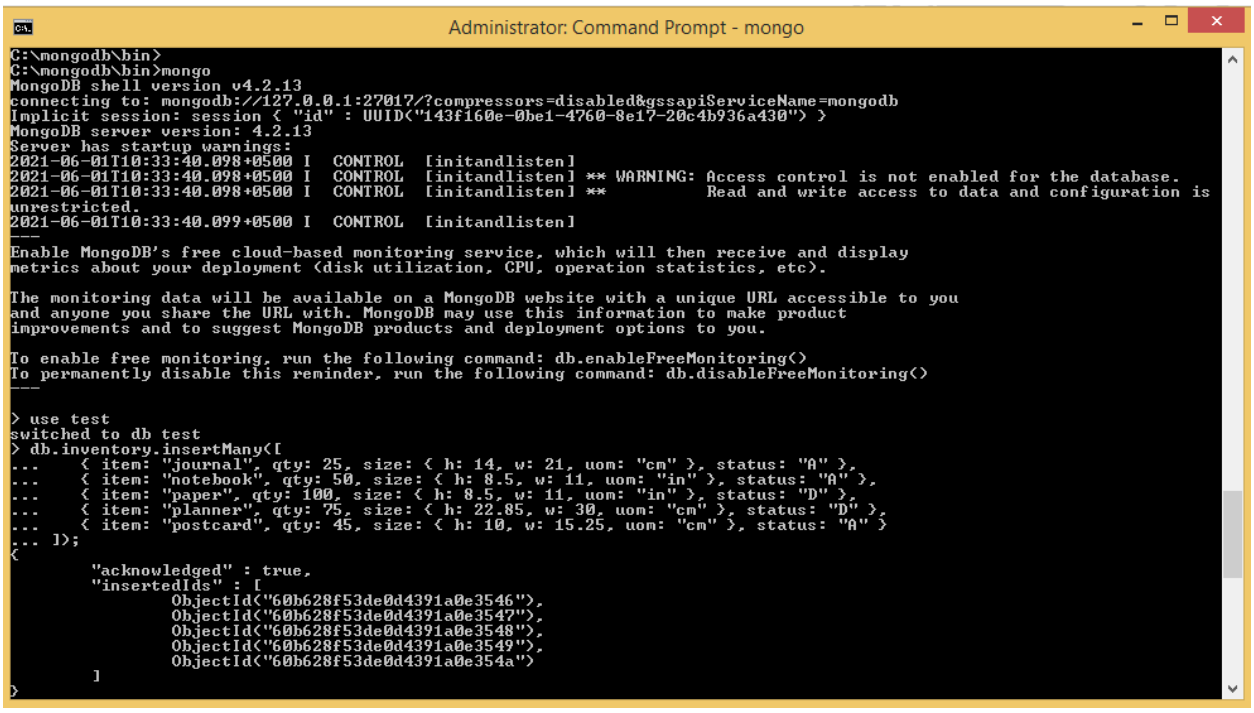
During the course of this lab, you will see how this function is used in various ways to achieve the purpose of document retrieval.

Consider that we have a collection named 'inventory' in our MongoDB database

To populate the inventory collection, run the following:

```
db.inventory.insertMany([

  { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },

  { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "A" },

  { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },

  { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },

  { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }

]);
```

You can run the operation in the web shell below:




### Select All Documents in a Collection

To select all documents in the collection, pass an empty document as the query filter parameter to the find method. The query filter parameter determines the select criteria:

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

This operation corresponds to the following SQL statement:

```
SELECT * FROM inventory
```

For more information on the syntax of the method

```
db.collection.find(query, projection)
```

| Parameter | Type | Description |
| --- | --- | --- |
| `query` | document | Optional. Specifies selection filter using query operators. To return all documents in a collection, omit this parameter or pass an empty document (`{}`). |
| `projection` | document | Optional. Specifies the fields to return in the documents that match the query filter. To return all fields in the matching documents, omit this parameter. For details, see Projection. |

### Specify Equality Condition

To specify equality conditions, use `<field>:<value>` expressions in the query filter document:

```
{ <field1>: <value1>, ... }
```

The following example selects from the inventory collection all documents where the status equals "D":

```
db.inventory.find( { status: "D" } )
```

This operation corresponds to the following SQL statement:

```
SELECT * FROM inventory WHERE status = "D"
```

### Specify Conditions Using Query Operators

A query filter document can use the query operators to specify conditions in the following form:

```
{ <field1>: { <operator1>: <value1> }, ... }
```

The following example retrieves all documents from the inventory collection where status equals either "A" or "D":

```
db.inventory.find( { status: { $in: [ "A", "D" ] } } )
```

The operation corresponds to the following SQL statement:

```
SELECT * FROM inventory WHERE status in ("A", "D")
```

**Query Selectors:**

**Comparison**

| Name | Description |
|---|---|
| $eq | Matches values that are equal to a specified value. |
| $gt | Matches values that are greater than a specified value. |
| $gte | Matches values that are greater than or equal to a specified value. |
| $in | Matches any of the values specified in an array. |
| $lt | Matches values that are less than a specified value. |
| $lte | Matches values that are less than or equal to a specified value. |
| $ne | Matches all values that are not equal to a specified value. |
| $nin | Matches none of the values specified in an array. |

**Logical**

| Name | Description |
|---|---|
| $and | Joins query clauses with a logical AND returns all documents that match the conditions of both clauses. |
| $not | Inverts the effect of a query expression and returns documents that do not match the query expression. |
| $nor | Joins query clauses with a logical NOR returns all documents that fail to match both clauses. |
| $or | Joins query clauses with a logical OR returns all documents that match the conditions of either clause. |

Refer to the Query and Projection Operators document for the complete list of MongoDB query 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.

The following example retrieves all documents in the collection where the status equals "A" **or** qty is less than ($lt) 30:

```
db.inventory.find( { $or: [ { status: "A" }, { qty: { $lt: 30 } } ] }
)
```

The operation corresponds to the following SQL statement:

```
SELECT * FROM inventory WHERE status = "A" OR qty < 30
```

**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.

The following example retrieves all documents in the inventory collection where the status equals "A" and qty is less than ($lt) 30:

```
db.inventory.find( { status: "A", qty: { $lt: 30 } } )
```

The operation corresponds to the following SQL statement:

```
SELECT * FROM inventory WHERE status = "A" AND qty < 30
```

**Specify AND as well as OR Conditions**

In the following example, the compound query document selects all documents in the collection where the status equals "A" **and** *either* qty is less than ($lt) 30 *or* item starts with the character p:

```
db.inventory.find( {

     status: "A",

     $or: [ { qty: { $lt: 30 } }, { item: /^p/ } ]

} )
```

The operation corresponds to the following SQL statement:

```
SELECT * FROM inventory WHERE status = "A" AND ( qty < 30 OR item LIKE
"p%")
```

MongoDB supports regular expressions $regex queries to perform string pattern matches.

**Query on Embedded/Nested Documents**

This section provides examples of query operations on embedded/nested documents using the db.collection.find() method in the mongo shell.

**Match an Embedded/Nested Document**

To specify an equality condition on a field that is an embedded/nested document, use the query filter document { <field>: <value> } where <value> is the document to match.

For example, the following query selects all documents where the field size equals the document { h: 14, w: 21, uom: "cm" }:

```
db.inventory.find( { size: { h: 14, w: 21, uom: "cm" } } )
```

Equality matches on the whole embedded document require an *exact* match of the specified <value> document, including the field order. For example, the following query does not match any documents in the inventory collection:

```
db.inventory.find(  { size: { w: 21, h: 14, uom: "cm" } }  )
```

**Query on Nested Field**

To specify a query condition on fields in an embedded/nested document, use dot notation ("field.nestedField").

**NOTE**

When querying using dot notation, the field and nested field must be inside quotation marks.

**Specify Equality Match on a Nested Field**

The following example selects all documents where the field uom nested in the size field equals "in":

```
db.inventory.find( { "size.uom": "in" } )
```

**Specify Match using Query Operator**

A query filter document can use the query operators to specify conditions in the following form:

```
{ <field1>: { <operator1>: <value1> }, ... }
```

The following query uses the less than operator ($lt) on the field h embedded in the size field:

```
db.inventory.find( { "size.h": { $lt: 15 } } )
```

**Specify AND Condition**

The following query selects all documents where the nested field h is less than 15, the nested field uom equals "in", and the status field equals "D":

```
db.inventory.find(

{ "size.h": { $lt: 15 }, "size.uom": "in", status: "D" } )
```

**Query an Array**

This section provides examples of query operations on array fields using the db.collection.find() method in the mongo shell. The examples in this section use the inventory collection. To populate the inventory collection, run the following:

```
db.inventory.insertMany([

   { item: "journal", qty: 25, tags: ["blank", "red"], dim_cm: [ 14,
21 ] },

   { item: "notebook", qty: 50, tags: ["red", "blank"], dim_cm: [ 14,
21 ] },

   { item: "paper", qty: 100, tags: ["red", "blank", "plain"], dim_cm:
[ 14, 21 ] },

   { item: "planner", qty: 75, tags: ["blank", "red"], dim_cm: [
22.85, 30 ] },
```

```
   { item: "postcard", qty: 45, tags: ["blue"], dim_cm: [ 10, 15.25 ]
}

]);
```

**Match an Array**

To specify equality condition on an array, use the query document { <field>: <value> } where <value> is the exact array to match, including the order of the elements.

The following example queries for all documents where the field tags value is an array with exactly two elements, "red" and "blank", in the specified order:

```
db.inventory.find( { tags: ["red", "blank"] } )
```

If, instead, you wish to find an array that contains both the elements "red" and "blank", without regard to order or other elements in the array, use the $all operator:

```
db.inventory.find( { tags: { $all: ["red", "blank"] } } )
```

**Query an Array for an Element**

To query if the array field contains at least *one* element with the specified value, use the filter { <field>: <value> } where <value> is the element value.

The following example queries for all documents where tags is an array that contains the string "red".

To specify conditions on the elements in the array field, use query operators in the query filter document: as one of its elements:

```
{ <array field>: { <operator1>: <value1>, ... } }
```

For example, the following operation queries for all documents where the array dim_cm contains at least one element whose value is greater than 25.

```
db.inventory.find( { dim_cm: { $gt: 25 } } )
```

**Specify Multiple Conditions for Array Elements**

When specifying compound conditions on array elements, you can specify the query such that either a single array element meets these condition or any combination of array elements meets the conditions.

**Query an Array with Compound Filter Conditions on the Array Elements**

The following example queries for documents where the dim_cm array contains elements that in some combination satisfy the query conditions; e.g., one element can satisfy the greater than 15 condition and another element can satisfy the less than 20 condition, or a single element can satisfy both:

```
db.inventory.find( { dim_cm: { $gt: 15, $lt: 20 } } )
```

**Query for an Array Element that Meets Multiple Criteria**

Use $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 dim_cm array contains at least one element that is both greater than ($gt) 22 and less than ($lt) 30:

```
db.inventory.find( { dim_cm: { $elemMatch: { $gt: 15, $lt: 20 } } } )
```

**Query for an Element by the Array Index Position**

Using dot notation, you can specify query conditions for an element at a particular index or position of the array. The array uses zero-based indexing.

***NOTE***

*When querying using dot notation, the field and nested field must be inside quotation marks.*

The following example queries for all documents where the second element in the array dim_cm is greater than 25:

```
db.inventory.find( { "dim_cm.1": { $gt: 25 } } )
```

**Query an Array by Array Length**

Use the $size operator to query for arrays by number of elements. For example, the following selects documents where the array tags has 3 elements.

```
db.inventory.find( { "tags": { $size: 3 } } )
```

**Project Fields to Return from Query**

By default, queries in MongoDB return all fields in matching documents. To limit the amount of data that MongoDB sends to applications, you can include a projection document to specify or restrict fields to return.

This section provides examples of query operations with projection using the db.collection.find() method in the mongo shell. The examples in this section use the inventory collection. To populate the inventory collection, run the following:

```
db.inventory.insertMany( [

  { item: "journal", status: "A", size: { h: 14, w: 21, uom: "cm" }, instock: [ { warehouse: "A", qty: 5 } ] },

  { item: "notebook", status: "A",  size: { h: 8.5, w: 11, uom: "in" }, instock: [ { warehouse: "C", qty: 5 } ] },

  { item: "paper", status: "D", size: { h: 8.5, w: 11, uom: "in" }, instock: [ { warehouse: "A", qty: 60 } ] },

  { item: "planner", status: "D", size: { h: 22.85, w: 30, uom: "cm" }, instock: [ { warehouse: "A", qty: 40 } ] },

  { item: "postcard", status: "A", size: { h: 10, w: 15.25, uom: "cm" }, instock: [ { warehouse: "B", qty: 15 }, { warehouse: "C", qty: 35 } ] }

]);
```

**Return All Fields in Matching Documents**

If you do not specify a projection document, the db.collection.find() method returns all fields in the matching documents.

The following example returns all fields from all documents in the inventory collection where the status equals "A":

```
db.inventory.find( { status: "A" } )
```

The operation corresponds to the following SQL statement:

```
SELECT * from inventory WHERE status = "A"
```

Return the Specified Fields and the _id Field Only

A projection can explicitly include several fields by setting the <field> to 1 in the projection document. The following operation returns all documents that match the query. In the result set, only the item, status and, by default, the _id fields return in the matching documents.

```
db.inventory.find( { status: "A" }, { item: 1, status: 1 } )
```

The operation corresponds to the following SQL statement:

```
SELECT _id, item, status from inventory WHERE status = "A"
```

Suppress _id Field

You can remove the _id field from the results by setting it to 0 in the projection, as in the following example:

```
db.inventory.find( { status: "A" }, { item: 1, status: 1, _id: 0 } )
```

The operation corresponds to the following SQL statement:

```
SELECT item, status from inventory WHERE status = "A"
```

***NOTE***

*With the exception of the _id field, you cannot combine inclusion and exclusion statements in projection documents.*

**Return All But the Excluded Fields**

Instead of listing the fields to return in the matching document, you can use a projection to exclude specific fields. The following example which returns all fields except for the status and the instock fields in the matching documents:

```
db.inventory.find( { status: "A" }, { status: 0, instock: 0 } )
```

**Return Specific Fields in Embedded Documents**

You can return specific fields in an embedded document. Use the dot notation to refer to the embedded field and set to 1 in the projection document.

The following example returns:

- The _id field (returned by default),
- The item field,
- The status field,
- The uom field in the size document.

The uom field remains embedded in the size document.

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

```
   { status: "A" },
   { item: 1, status: 1, "size.uom": 1 }
)
```

**Suppress Specific Fields in Embedded Documents**

You can suppress specific fields in an embedded document. Use the dot notation to refer to the embedded field in the projection document and set to 0.

The following example specifies a projection to exclude the uom field inside the size document. All other fields are returned in the matching documents:

```
db.inventory.find(
   { status: "A" },
   { "size.uom": 0 }
)
```

**Projection on Embedded Documents in an Array**

Use dot notation to project specific fields inside documents embedded in an array.

The following example specifies a projection to return:

- The _id field (returned by default),
- The item field,
- The status field,
- The qty field in the documents embedded in the instock array.

```
db.inventory.find( { status: "A" }, { item: 1, status: 1,
"instock.qty": 1 } )
```

**Project Specific Array Elements in the Returned Array**

For fields that contain arrays, MongoDB provides the following projection operators for manipulating arrays: $elemMatch, $slice, and $.

The following example uses the $slice projection operator to return the last element in the instock array:

```
db.inventory.find( { status: "A" }, { item: 1, status: 1, instock: { $slice: -1 } } )
```

$elemMatch, $slice, and $ are the *only* way to project specific elements to include in the returned array. For instance, you *cannot* project specific array elements using the array index; e.g. { "instock.0": 1 } projection will *not* project the array with the first element.

**Query for Null or Missing Fields**

Different query operators in MongoDB treat null values differently.

This section provides examples of operations that query for null values using the db.collection.find() method in the mongo shell. The examples in this section use the inventory collection. To populate the inventory collection, run the following:

```
db.inventory.insertMany([

   { _id: 1, item: null },

   { _id: 2 }

])
```

**Equality Filter**

The { item : null } query matches documents that either contain the item field whose value is null *or* that do not contain the item field.

```
db.inventory.find( { item: null } )
```

The query returns both documents in the collection.

**Type Check**

The { item : { $type: 10 } } query matches *only* documents that contain the item field whose value is null; i.e. the value of the item field is of BSON Type Null (type number 10) :

db.inventory.find( { item : { $type: 10 } } )

The query returns only the document where the item field has a value of null.

**Existence Check**

The following example queries for documents that do not contain a field.

The { item : { $exists: false } } query matches documents that do not contain the item field:

db.inventory.find( { item : { $exists: **false** } } )

The query only returns the document that does *not* contain the item field.

2. **Update Documents**

MongoDB provides the update() command to update the documents of a collection. To update only the documents you want to update, you can add a criteria to the update statement so that only selected documents are updated.

The basic parameters in the command is a condition for which document needs to be updated, and the next is the modification which needs to be performed.

This section uses the following mongo shell methods:

- db.collection.updateOne(<filter>, <update>, <options>)
- db.collection.updateMany(<filter>, <update>, <options>)
- db.collection.replaceOne(<filter>, <update>, <options>)

The examples in this section uses the inventory collection. To create and/or populate the inventory collection, run the following:

```
db.inventory.insertMany( [

   { item: "canvas", qty: 100, size: { h: 28, w: 35.5, uom: "cm" },
status: "A" },

   { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" },
status: "A" },

   { item: "mat", qty: 85, size: { h: 27.9, w: 35.5, uom: "cm" },
status: "A" },

   { item: "mousepad", qty: 25, size: { h: 19, w: 22.85, uom: "cm" },
status: "P" },

   { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" },
status: "P" },

   { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" },
status: "D" },

   { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" },
status: "D" },
```

```
   { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" },

   { item: "sketchbook", qty: 80, size: { h: 14, w: 21, uom: "cm" }, status: "A" },

   { item: "sketch pad", qty: 95, size: { h: 22.85, w: 30.5, uom: "cm" }, status: "A" }

] );
```

**Update Documents in a Collection**

To update a document, MongoDB provides update operators, such as $set, to modify field values.

To use the update operators, pass to the update methods an update document of the form:

```
{

  <update operator>: { <field1>: <value1>, ... },

  <update operator>: { <field2>: <value2>, ... },

  ...

}
```

Some update operators, such as $set, will create the field if the field does not exist.

**Update a Single Document**

The following example uses the db.collection.updateOne() method on the inventory collection to update the *first* document where item equals "paper":

```
db.inventory.updateOne(

   { item: "paper" },

   {

     $set: { "size.uom": "cm", status: "P"}

   }

)
```

The update operation:

- uses the $set operator to update the value of the size.uom field to "cm" and the value of the status field to "P",

**Update Multiple Documents**

The following example uses the db.collection.updateMany() method on the inventory collection to update all documents where qty is less than 50:

```
db.inventory.updateMany(
   { "qty": { $lt: 50 } },
   {
     $set: { "size.uom": "in", status: "P" }
   }
)
```

**Replace a Document**

To replace the entire content of a document except for the _id field, pass an entirely new document as the second argument to db.collection.replaceOne().

When replacing a document, the replacement document must consist of only field/value pairs; i.e. do not include update operators expressions.

The replacement document can have different fields from the original document. In the replacement document, you can omit the _id field since the _id field is immutable; however, if you do include the _id field, it must have the same value as the current value.

The following example replaces the *first* document from the inventory collection where item: "paper":

```
db.inventory.replaceOne(
   { item: "paper" },
   { item: "paper", instock: [ { warehouse: "A", qty: 60 }, { warehouse: "B", qty: 40 } ] }
)
```

Behavior

**_id Field**

Once set, you cannot update the value of the _id field nor can you replace an existing document with a replacement document that has a different _id field value.

**Upsert Option**

If updateOne(), updateMany(), or replaceOne() includes upsert : true **and** no documents match the specified filter, then the operation creates a new document and inserts it. If there are matching documents, then the operation modifies or replaces the matching document or documents.

3. **Delete Documents**

This section uses the following mongo shell methods:

db.collection.deleteMany()

db.collection.deleteOne()

The examples in this section uses the inventory collection. To populate the inventory collection, run the following:

```
db.inventory.insertMany( [

   { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" },
status: "A" },

   { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" },
status: "P" },

   { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" },
status: "D" },

   { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" },
status: "D" },

   { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" },
status: "A" },

] );
```

**Delete All Documents**

To delete all documents from a collection, pass an empty filter document {} to the db.collection.deleteMany() method.

The following example deletes all documents from the inventory collection:

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

**Delete All Documents that Match a Condition**

You can specify criteria, or filters, that identify the documents to delete. The filters use the same syntax as read operations.

To specify equality conditions, use <field>:<value> expressions in the query filter document:

```
{ <field1>: <value1>, ... }
```

A query filter document can use the query operators to specify conditions in the following form:

```
{ <field1>: { <operator1>: <value1> }, ... }
```

To delete all documents that match a deletion criteria, pass a filter parameter to the deleteMany() method.

The following example removes all documents from the inventory collection where the status field equals "A":

```
db.inventory.deleteMany({ status : "A" })
```

**Delete Only One Document that Matches a Condition**

To delete at most a single document that matches a specified filter (even though multiple documents may match the specified filter) use the db.collection.deleteOne() method.

The following example deletes the first document where status is "D":

```
db.inventory.deleteOne( { status: "D" } )
```

For more details

References:

*https://docs.mongodb.com/v4.2/crud/*

*https://www.guru99.com/mongodb-query-document-using-find.html*