Read Documents using find()

```
training> db.trainee.find()
[
  {
    _id: ObjectId("61bfcdd9df9d3c572c601e18"),
    admission_no: 101,
    first_name: 'James'
  },
  {
    _id: ObjectId("61bfcdd9df9d3c572c601e19"),
    admission_no: 102,
    first_name: 'Rock'
  },
  {
    _id: ObjectId("61bfcdd9df9d3c572c601e1a"),
    admission_no: 106,
    first_name: 'Hanks'
  }
]
```

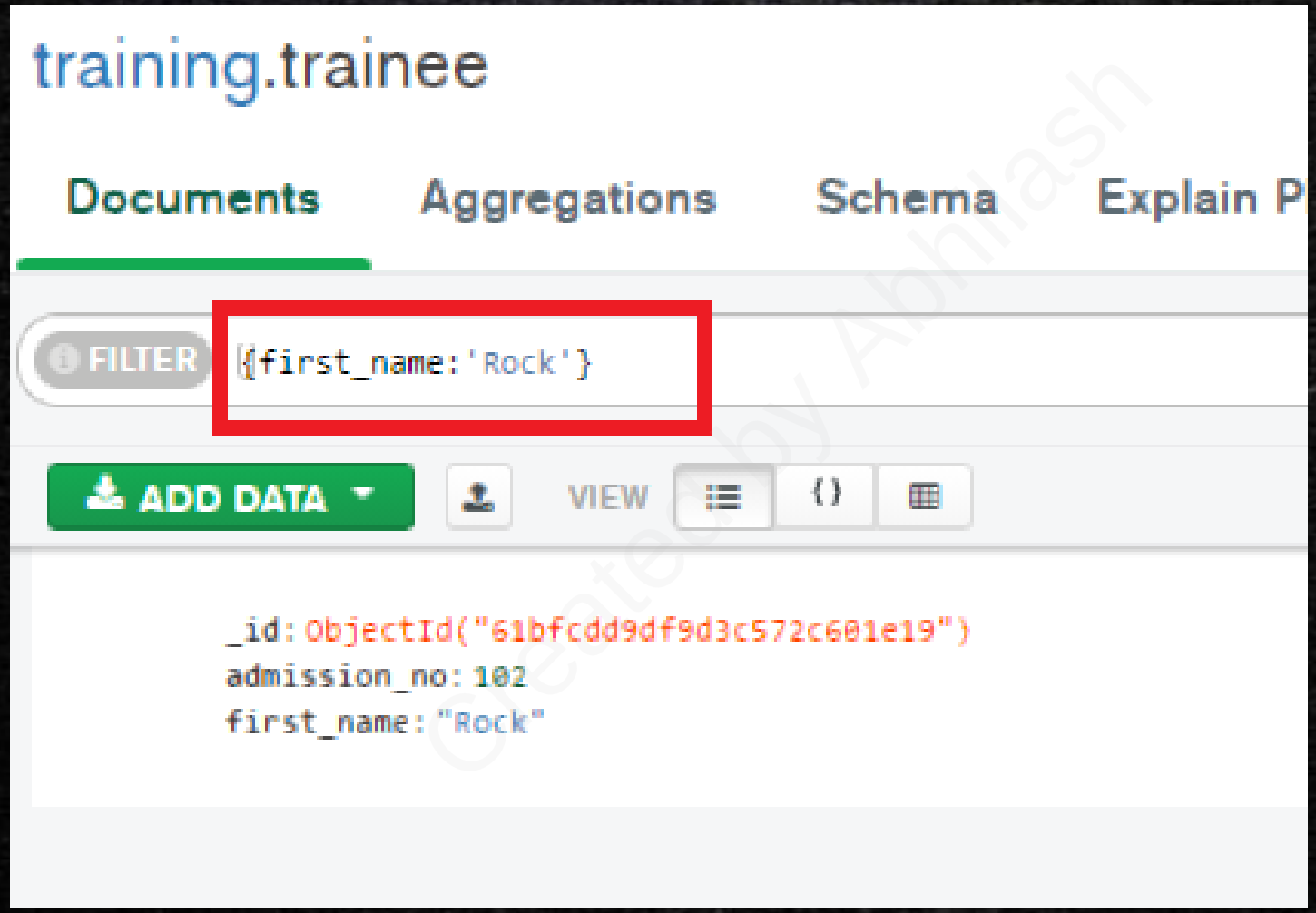
SYNTAX:
db.collection_name.find()

Read and Filter Documents using find()

```
training> db.trainee.find({"admission_no":106})
[
  {
    _id: ObjectId("61bfcdd9df9d3c572c601e1a"),
    admission_no: 106,
    first_name: 'Hanks'
  }
]
training> db.trainee.find({"admission_no":106, "first_name":'Hanks'})
[
  {
    _id: ObjectId("61bfcdd9df9d3c572c601e1a"),
    admission_no: 106,
    first_name: 'Hanks'
  }
]
```

SYNTAX:
db.collection_name.
find({"fieldname":"value"})

Read and Filter Documents in Compass



Updating Document using updateOne()

The updateOne() method finds the first document that matches the filter and applies the specified update modifications.

The syntax is:

```
db.collection.updateOne(
   <filter>,
   <update>,
   {
     upsert: <boolean>,
     writeConcern: <document>,
     collation: <document>,
     arrayFilters: [ <filterdocument1>, ... ],
     hint:  <document|string>
      // Available starting in MongoDB 4.2.1
   }
)
```

Updating Document using updateOne()

The updateOne() parameters include:

Parameter	Type	Description
filter	document	The selection criteria for the update. The same query selectors as in the `find()` method are available. Specify an empty document `{ }` to update the first document returned in the collection.
update	document or pipeline	The modifications to apply. Can be one of the following: Update document — Contains only update operator expressions.

Updating Document using updateOne()

The updateOne() parameters include:

`upsert`	boolean	Optional. When `true`, `updateOne()` either: • Creates a new document if no documents match the `filter`. For more details see upsert behavior. • Updates a single document that matches the `filter`. To avoid multiple upserts, ensure that the `filter` field(s) are uniquely indexed. Defaults to `false`, which does *not* insert a new document when no match is found.
`writeConcern`	document	Optional. A document expressing the write concern. Omit to use the default write concern.

Updating Document using updateOne()

The updateOne() parameters include:

`collation`	document	Optional. Specifies the collation to use for the operation. Collation allows users to specify language-specific rules for string comparison, such as rules for lettercase and accent marks.
`arrayFilters`	array	Optional. An array of filter documents that determine which array elements to modify for an update operation on an array field.
hint	Document or string	Optional. A document or string that specifies the index to use to support the query predicate. The option can take an index specification document or the index name string.

For more details see :
https://docs.mongodb.com/manual/reference/method/db.collection.updateOne/#mongodb-method-db.collection.updateOne

Updating Document using updateOne()

```
training>    db.trainee.updateOne(
...          {"admission_no" : 103},
...          {$set: {"admission_no" : 106,"first_name":"Hanks"} }
...      );
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
training>
```

Updating Document using updateMany()

The updateMany() Updates all documents that match the specified filter for a collection.

The syntax is

```
db.collection.updateMany(
   <filter>,
   <update>,
   {
     upsert: <boolean>,
     writeConcern: <document>,
     collation: <document>,
     arrayFilters: [ <filterdocument1>, ... ],
     hint:  <document|string>        // Available starting in MongoDB 4.2.1
   }
)
```

Updating Document using updateMany()

```
training>    db.trainee.updateMany(
...             {"admission_no" : 100},
...             {$set: {"last_name" : "Sunil"} }
...         );
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 3,
  modifiedCount: 3,
  upsertedCount: 0
}
```

Updating Document using findAndModify()

The findAndModify() updates and returns an existing document sorted by field where the document matches the query criteria:

```
training> db.trainee.findAndModify({
...       query: { admission_no: 106, first_name: "Hanks" },
...       sort: { admission_no: 1 },
...       update: { admission_no: 106, last_name: "Hanks", first_name: "Tony" }
... })
{
  _id: ObjectId("61bfcdd9df9d3c572c601e1a"),
  admission_no: 106,
  first_name: 'Hanks'
}
training>
```

Using upsert: true

The findAndModify() combined with upsert : true will create the document if it's not existing in the collection:

```
training> db.trainee.findAndModify({
...       query: { admission_no: 107, first_name: "Bond" },
...       sort: { admission_no: 1 },
...       update: { admission_no: 107, last_name: "Bond", first_name: "James" },
...       upsert: true
... })
null
```

Using upsert: true

The findAndModify() combined with upsert : true will create the document if it's not existing in the collection:

```
training> db.trainee.find()
[
  {
    _id: ObjectId("61bfcdd9df9d3c572c601e1a"),
    admission_no: 106,
    last_name: 'Hanks',
    first_name: 'Tony'
  },
  {
    _id: ObjectId("61c11a691f356e2c404124f0"),
    admission_no: 107,
    last_name: 'Bond',
    first_name: 'James'
  }
]
```

Other Methods where we can use upsert: true

The following methods can also add new documents to a collection:

`db.collection.updateOne()` when used with the upsert: true option.

`db.collection.updateMany()` when used with the upsert: true option.

`db.collection.findAndModify()` when used with the upsert: true option.

`db.collection.findOneAndUpdate()` when used with the upsert: true option.

`db.collection.findOneAndReplace()` when used with the upsert: true option.

`db.collection.bulkWrite()` when used with the upsert: true option.

Updating Document using findOneAndUpdate ()

The findOneAndUpdate () returns an existing document (first one matching criteria) before the update happens:

```
training> db.trainee.findOneAndUpdate(
...     { admission_no: 107, last_name: "Bond" },
...     { $set : { admission_no: 108, last_name: "Bonding", first_name: "James"
} }
... )
{
  _id: ObjectId("61c11a691f356e2c404124f0"),
  admission_no: 107,
  last_name: 'Bond',
  first_name: 'James'
}
```


If upsert : true, add new document if the filtering criteria don’t match ,
If returnNewDocument : true, returns the modified document than the existing one

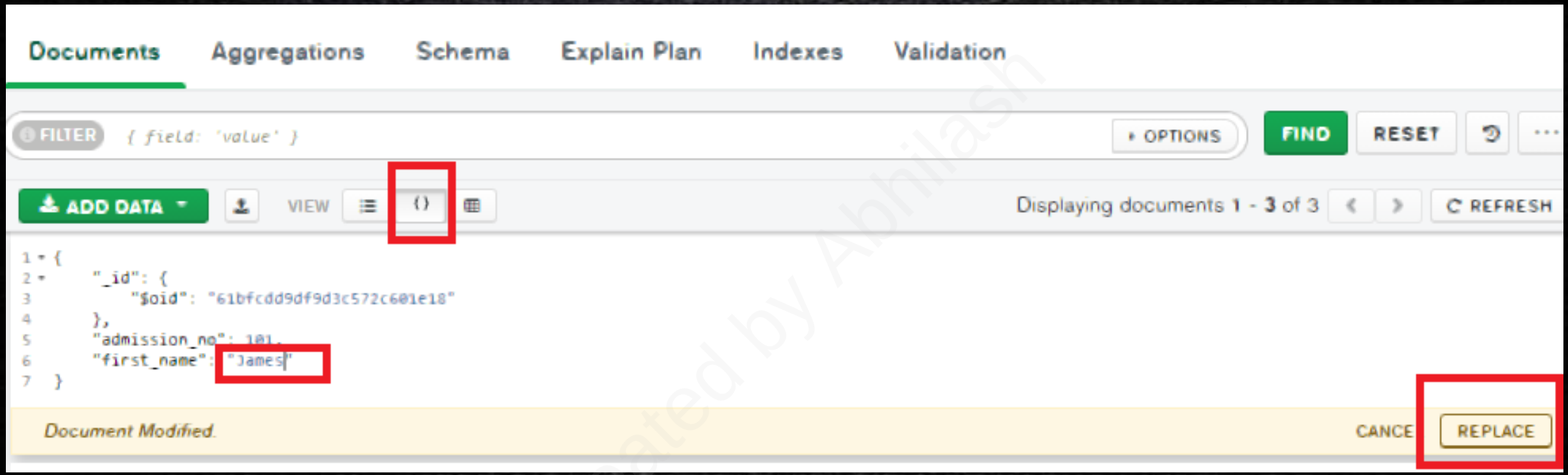
Updating Document using findOneAndReplace ()

The findOneAndReplace () returns an existing document (first one matching criteria) before the replace happens:

```
training> db.trainee.findOneAndReplace(
...       { admission_no: 108, first_name: "James" },
...       { admission_no: 109, last_name: "Bond", first_name: "James" }
... )
{
  _id: ObjectId("61c11a691f356e2c404124f0"),
  admission_no: 108,
  last_name: 'Bonding',
  first_name: 'James'
}
```


If upsert : true, add new document if the filtering criteria don't match ,
If returnNewDocument : true, returns the modified document than the existing one

Updating Document using Compass



Deleting Document using deleteOne()

The updateOne() method finds the first document that matches the filter and removes it from a collection.

The syntax is

```
db.collection.deleteOne(
   <filter>,
   {
      writeConcern: <document>,
      collation: <document>,
      hint: <document|string>        // Available starting in MongoDB 4.4
   }
)
```

Deleting Document using deleteOne()

```
training>     db.trainee.deleteOne(
...           {"admission_no" : 100, "last_name" : "Sunil"}
...       );
{ acknowledged: true, deletedCount: 1 }
training>
```

Deleting Document using deleteMany()

The deleteMany() deletes all documents that match the specified filter for a collection.

The syntax is

```
db.collection.deleteMany(
   <filter>,
   {
      writeConcern: <document>,
      collation: <document>
   }
)
```

Deleting Document using deleteMany()

```
training>     db.trainee.deleteMany(
...             {"admission_no" : 100, "last_name" : "Sunil"}
...         );
{ acknowledged: true, deletedCount: 2 }
```

Bulk Write Operations using bulkWrite()

The db.collection.bulkWrite() method provides the ability to perform bulk insert, update, and remove operations.

bulkWrite() supports the following write operations:

- insertOne
- updateOne
- updateMany
- replaceOne
- deleteOne
- deleteMany

Each write operation is passed to bulkWrite() as a document in an array.

Bulk Write Operations using bulkWrite()

```
training> db.trainee.bulkWrite(
...    [
...     {insertOne :{"document":{ "admission_no": 109, "last_name": "Bond", "first_name": "James" }}},
...     {insertOne :{"document": { "admission_no": 110, "last_name": "Murali", "first_name": "Minnal" }}},
...     {updateOne :{"filter": { "admission_no": 110 }, "update" : { $set : { "first_name": "Lightning" }}}}
...     ] );
{
  acknowledged: true,
  insertedCount: 2,
  insertedIds: {
    '0': ObjectId("61c12b3b337d81b5b4c7a068"),
    '1': ObjectId("61c12b3b337d81b5b4c7a069")
  },
  matchedCount: 1,
  modifiedCount: 1,
  deletedCount: 0,
  upsertedCount: 0,
  upsertedIds: {}
}
```

Assignment – MongoDB basic CRUD Operations

- Create a database called ‘news’
- Add Collection called ‘news_feed’
- Insert ten records (5 using shell and 5 using compass) with in the “articles": [{ }]
- manually from the feed https://newsapi.org/s/india-news-api
- List all the content in both shell and compass
- Modify all matching author: "I P Singh“ to “IPS”
- Delete all records matching author: “Sports Desk”
- List all the contents once again