Insert Operation – Insert One Document

```
Microsoft Windows [Version 10.0.26200.7462]
(c) Microsoft Corporation. All rights reserved.

C:\Users\manik>mongosh
Current Mongosh Log ID: 693e7e9a32376d7d001e2620
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.10
Using MongoDB:          8.2.2
Using Mongosh:          2.5.10

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

------
   The server generated these startup warnings when booting
   2025-12-12T12:41:22.027+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

test> use academyDB
switched to db academyDB
academyDB> db.createCollection("learners")
{ ok: 1 }
academyDB> show collections
learners
academyDB>
```

The MongoDB shell is started using mongosh, and a new database **academyDB** with a collection **learners** is created successfully.

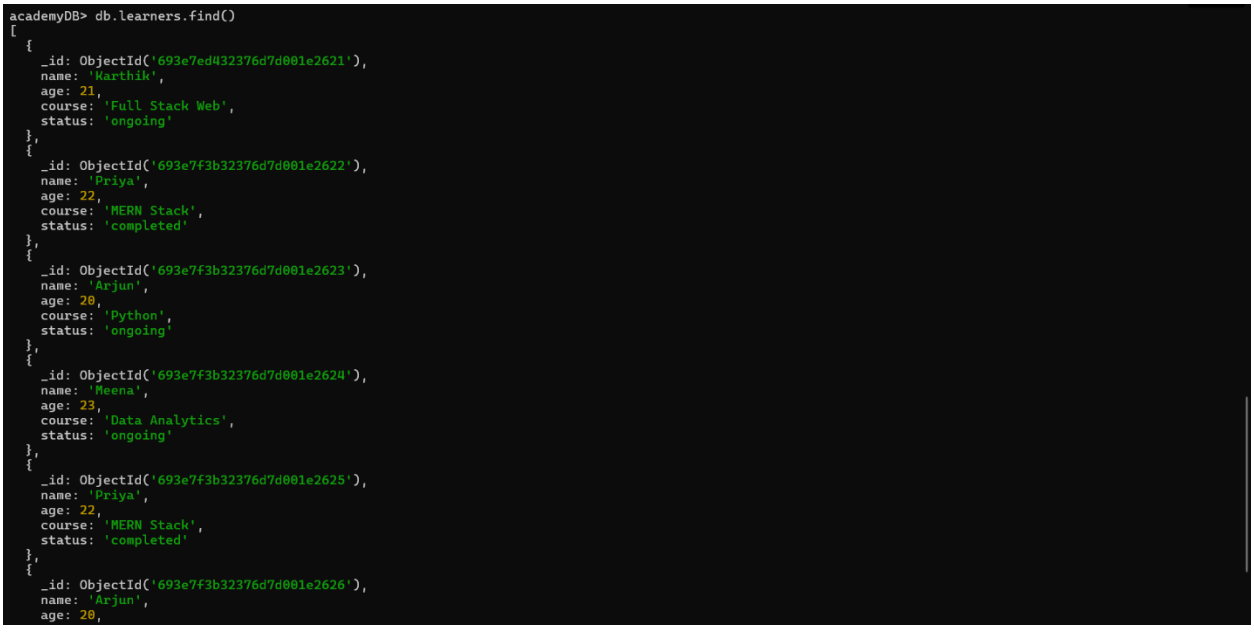
```
academyDB> db.learners.insertOne({
...     name: "Karthik",
...     age: 21,
...     course: "Full Stack Web",
...     status: "ongoing"
... })
...
{
  acknowledged: true,
  insertedId: ObjectId('693e7ed432376d7d001e2621')
}
academyDB>
```

A single record is inserted into the learners collection using insertOne(), and the successful insertion is confirmed by MongoDB.

Insert Operation – Insert Multiple Documents

This screenshot demonstrates inserting multiple records into the learners collection using the insertMany() method. MongoDB confirms the successful insertion by generating ObjectIds for each document.

Read Operation – Fetch All Documents



This screenshot shows the use of the find() method to retrieve all documents from the learners collection. It displays complete student details stored in the academyDB database.

Read Operation – Filter Documents

This screenshot shows filtering documents using the find() method with a condition. Only learners enrolled in the MERN Stack course are retrieved from the learners collection.

Update Operation – Update One Document

This screenshot shows updating a single document in the learners collection using the updateOne() method. The status of the learner named Arjun is successfully changed to completed.

Update Operation – Update Multiple Documents

This screenshot shows updating multiple documents in the learners collection using the updateMany() method. All learners with the status ongoing are updated to completed successfully.

Delete Operation – Delete One Document

```
academyDB> db.learners.deleteOne({ name: "Meena" })
{ acknowledged: true, deletedCount: 1 }
academyDB>
```

This screenshot shows deleting a single document from the learners collection using the deleteOne() method. The record with the name **Meena** is successfully removed from the database.

Delete Operation – Delete Multiple Documents

```
academyDB> db.learners.deleteMany({})
{ acknowledged: true, deletedCount: 6 }
```

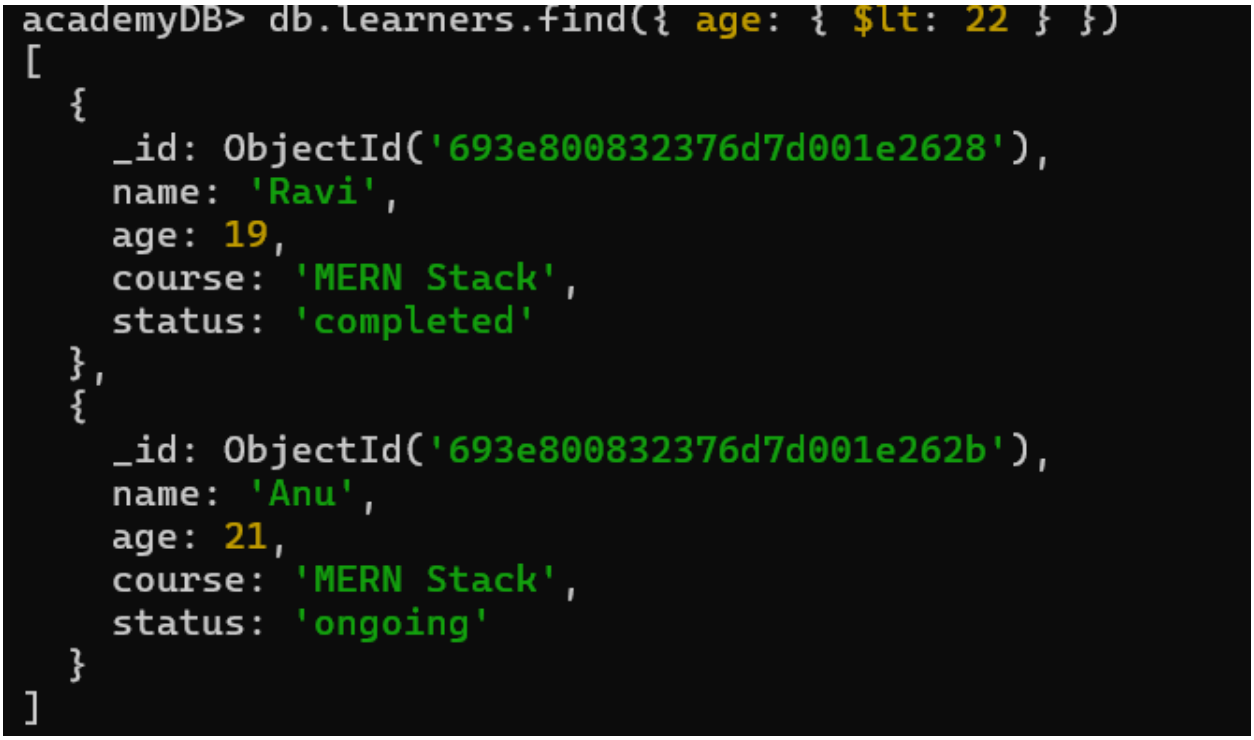
This screenshot shows deleting multiple documents from the learners collection using the deleteMany() method. All records in the collection are removed successfully for practice purposes.

Query Operations – Greater Than Operator ($gt)

```
academyDB> db.learners.find({ age: { $gt: 21 } })
[
  {
    _id: ObjectId('693e800832376d7d001e2629'),
    name: 'Sneha',
    age: 24,
    course: 'Python',
    status: 'completed'
  },
  {
    _id: ObjectId('693e800832376d7d001e262a'),
    name: 'Vikram',
    age: 22,
    course: 'Java',
    status: 'ongoing'
  }
]
```

This screenshot demonstrates the use of the $gt (greater than) operator to filter documents. It retrieves learners whose age is greater than 21 from the learners collection.

Query Operators – Using $gt Operator



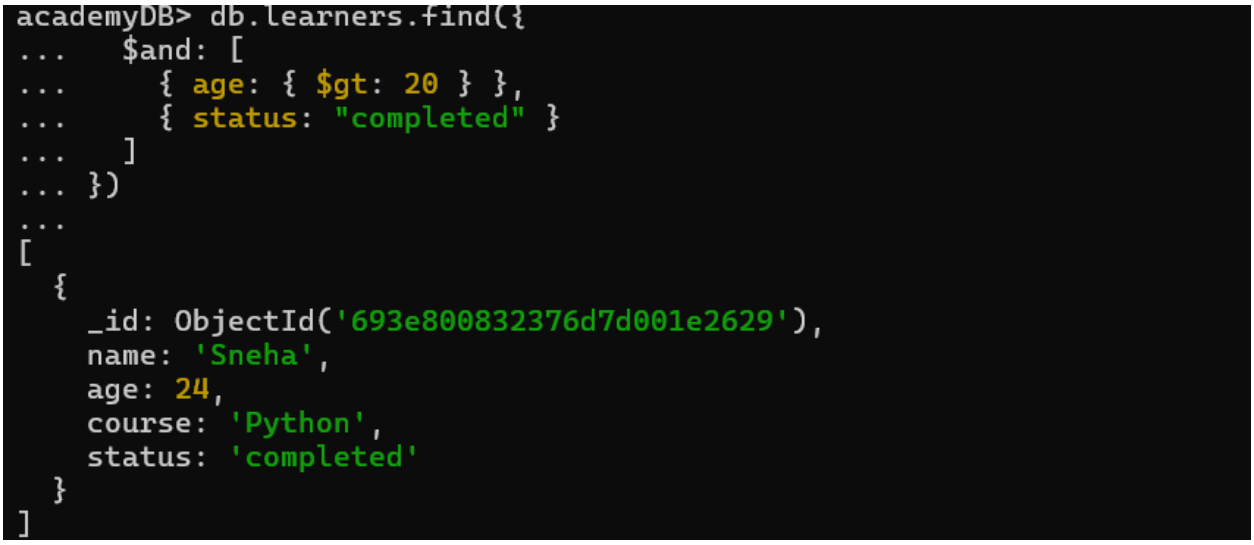
This screenshot demonstrates the use of the $gt (greater than) operator to filter documents. It retrieves learners whose age is greater than 21 from the learners collection.

Query Operators – Using $in Operator

```
academyDB> db.learners.find({
...   course: { $in: ["MERN Stack", "Java"] }
... })
...
[
  {
    _id: ObjectId('693e800832376d7d001e2628'),
    name: 'Ravi',
    age: 19,
    course: 'MERN Stack',
    status: 'completed'
  },
  {
    _id: ObjectId('693e800832376d7d001e262a'),
    name: 'Vikram',
    age: 22,
    course: 'Java',
    status: 'ongoing'
  },
  {
    _id: ObjectId('693e800832376d7d001e262b'),
    name: 'Anu',
    age: 21,
    course: 'MERN Stack',
    status: 'ongoing'
  }
]
```

This screenshot demonstrates the use of the $in operator to filter documents. It retrieves learners enrolled in either

Query Operators – Using $and Operator



This screenshot demonstrates the use of the $and operator to apply multiple conditions in a query. It retrieves learners whose age is greater than 20 and whose status is completed.

Query Operators – Using $or Operator

This screenshot demonstrates the use of the $or operator to filter documents. It retrieves learners enrolled in either **Python** or **Java** courses from the learners collection.

Query Operators – Using $exists Operator

```
academyDB> db.learners.find({
...   status: { $exists: true }
... })
...
[
  {
    _id: ObjectId('693e800832376d7d001e2628'),
    name: 'Ravi',
    age: 19,
    course: 'MERN Stack',
    status: 'completed'
  },
  {
    _id: ObjectId('693e800832376d7d001e2629'),
    name: 'Sneha',
    age: 24,
    course: 'Python',
    status: 'completed'
  },
  {
    _id: ObjectId('693e800832376d7d001e262a'),
    name: 'Vikram',
    age: 22,
    course: 'Java',
    status: 'ongoing'
  },
  {
    _id: ObjectId('693e800832376d7d001e262b'),
    name: 'Anu',
    age: 21,
    course: 'MERN Stack',
    status: 'ongoing'
  }
]
```

This screenshot demonstrates the use of the $exists operator to check whether the status field is present in documents. It retrieves all learners where the status field exists in the learners collection.