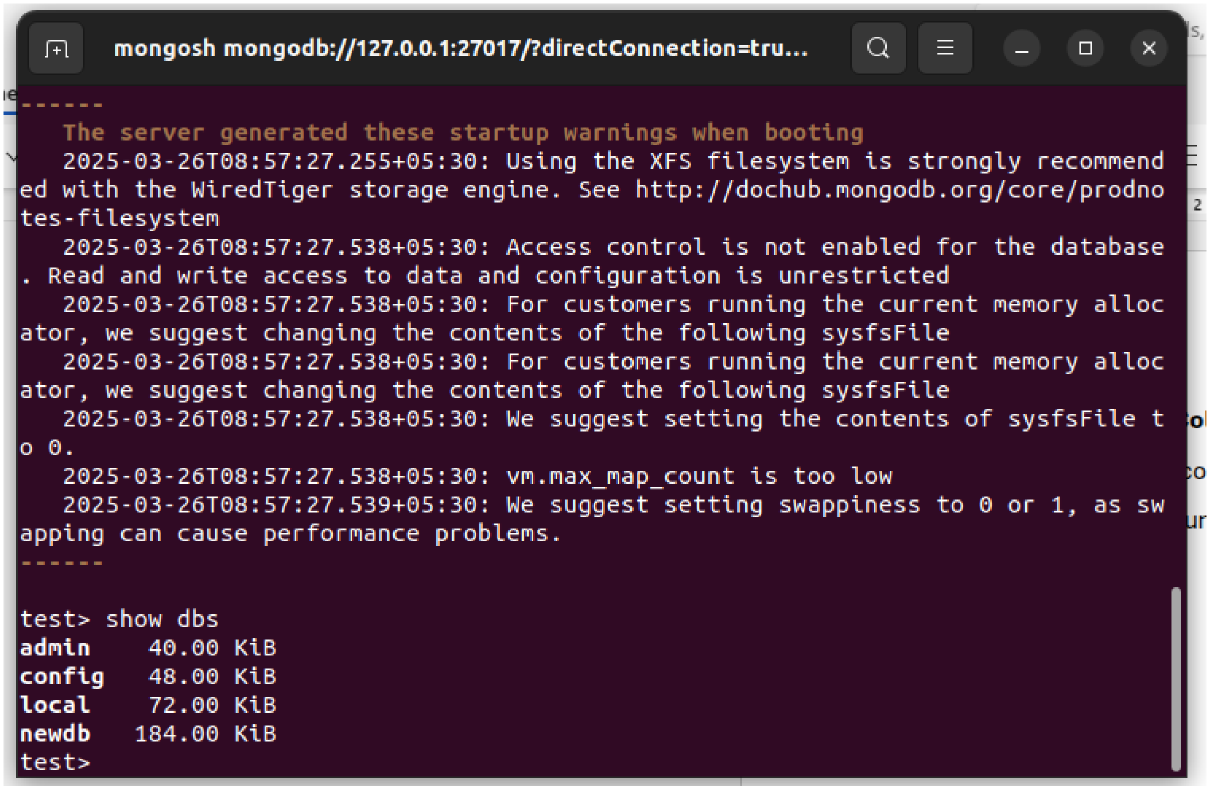
What is a Database and Collection in MongoDB?

Database: A container for collections, similar to a database in SQL.

Collection: A group of documents, like a table in SQL.



```
------
   The server generated these startup warnings when booting
   2025-03-26T08:57:27.255+05:30: Using the XFS filesystem is strongly recommend
ed with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodno
tes-filesystem
   2025-03-26T08:57:27.538+05:30: Access control is not enabled for the database
. Read and write access to data and configuration is unrestricted
   2025-03-26T08:57:27.538+05:30: For customers running the current memory alloc
ator, we suggest changing the contents of the following sysfsFile
   2025-03-26T08:57:27.538+05:30: For customers running the current memory alloc
ator, we suggest changing the contents of the following sysfsFile
   2025-03-26T08:57:27.538+05:30: We suggest setting the contents of sysfsFile t
o 0.
   2025-03-26T08:57:27.538+05:30: vm.max_map_count is too low
   2025-03-26T08:57:27.539+05:30: We suggest setting swappiness to 0 or 1, as sw
apping can cause performance problems.
------

test> show dbs
admin     40.00 KiB
config    48.00 KiB
local     72.00 KiB
newdb    184.00 KiB
test>
```

Show dbs – Create or switch to a database

db.createCollection("newdb") - Create a collection

Document in MongoDB

A document is the basic unit of data storage in MongoDB. It is a JSON-like structure that contains field-value pairs.

{

"_id": ObjectId("507f1f77bcf86cd799439011"),

"name": "Brindha",

}

Documents -

A **document** is the basic unit of data storage in MongoDB. It is a JSON-like structure that contains field-value pairs.

```
newdb> db.newproducts.find()
[
 { _id: ObjectId('67e3a4198ea3f781676b140b'), name: 'Laptop' },
 {
  _id: ObjectId('67e3a56a8ea3f781676b140c'),
  name: 'Laptop',
  price: 34300
 },
 {
  _id: ObjectId('67e3a5908ea3f781676b140d'),
  name: 'Laptop',
  price: 34300
 },
 {
  _id: ObjectId('67e3bfc58ea3f781676b140f'),
  name: 'Laptop',
  price: 34300
 },
 {
  _id: ObjectId('67e3bfc58ea3f781676b1410'),
  name: 'Laptop1',
  price: 34300
 },
 {
  _id: ObjectId('67e3bff68ea3f781676b1411'),
  name: 'Laptop1',
  price: 34300
 }
```

What is BSON?

- BSON (Binary JSON) is the internal storage format of MongoDB.
- It supports additional data types like Date, Decimal128, and Binary.

```
{ _id: new ObjectId('67e3a4198ea3f781676b140b'), name: 'Laptop' },
{ _id: new ObjectId('67e3a56a8ea3f781676b140c'), name: 'Laptop', price: 34300 },
{ _id: new ObjectId('67e3a5908ea3f781676b140d'), name: 'Laptop', price: 34300 },
{ _id: new ObjectId('67e3bfc58ea3f781676b140f'), name: 'Laptop', price: 34300 },
{ _id: new ObjectId('67e3bfc58ea3f781676b1410'), name: 'Laptop1', price: 34300 },
{ _id: new ObjectId('67e3bff68ea3f781676b1411'), name: 'Laptop1', price: 34300 };
```

Bson:

```
[
 { _id: ObjectId("67e3a4198ea3f781676b140b"), name: "Laptop" },
 { _id: ObjectId("67e3a56a8ea3f781676b140c"), name: "Laptop", price: NumberInt(34300) },
 { _id: ObjectId("67e3a5908ea3f781676b140d"), name: "Laptop", price: NumberInt(34300) },
 { _id: ObjectId("67e3bfc58ea3f781676b140f"), name: "Laptop", price: NumberInt(34300) },
 { _id: ObjectId("67e3bfc58ea3f781676b1410"), name: "Laptop1", price: NumberInt(34300) },
 { _id: ObjectId("67e3bff68ea3f781676b1411"), name: "Laptop1", price: NumberInt(34300) }
]
```

MongoDB installation:
In Linux:

```
sudo apt install -y mongodb-org
```

After installation, start the MongoDB service:

```
sudo systemctl start mongod
```

To check mongodb running status

sudo systemctl status mongod

Version

mongod –version

Connect to shell

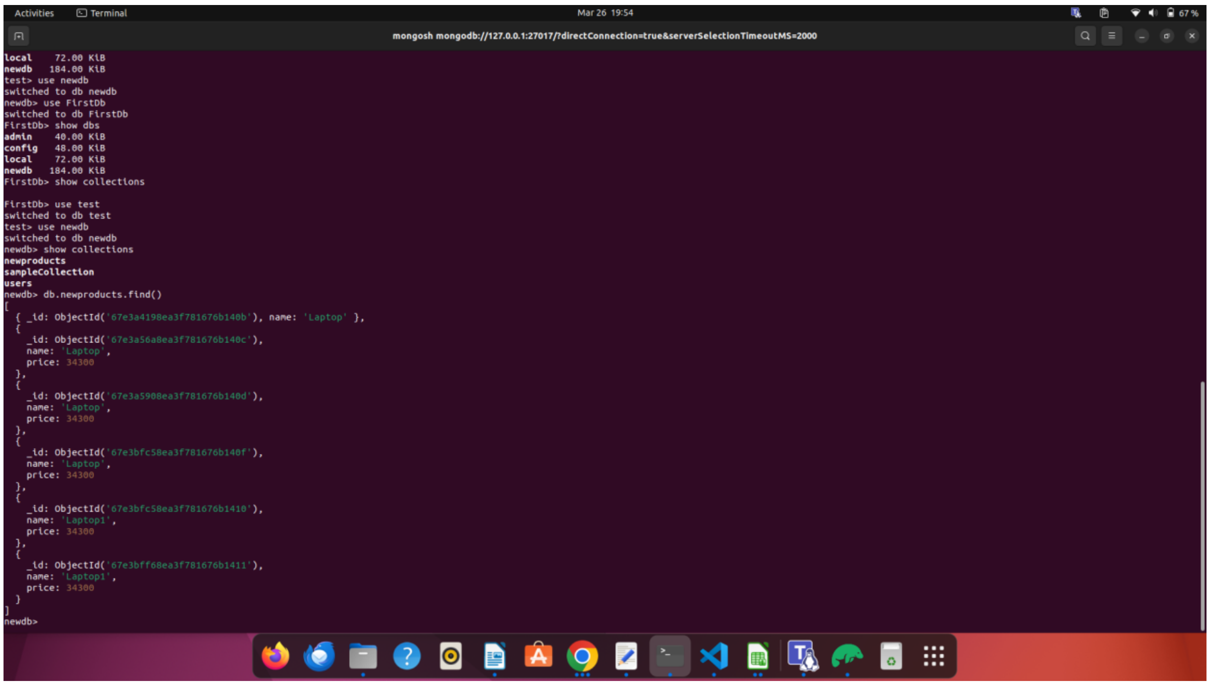
Mongosh

Restart the mongodb shell

sudo systemctl restart mongod

Commands;

Db.newproducts.find()- Retrieves all the data from the collection



Activities
Terminal
Mar 26 19:54
67 %
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

```
local    72.00 KiB
newdb   184.00 KiB
test> use newdb
switched to db newdb
newdb> use FirstDb
switched to db FirstDb
FirstDb> show dbs
admin    40.00 KiB
config   48.00 KiB
local    72.00 KiB
newdb   184.00 KiB
FirstDb> show collections

FirstDb> use test
switched to db test
test> use newdb
switched to db newdb
newdb> show collections
newproducts
sampleCollection
users
newdb> db.newproducts.find()
[
  { _id: ObjectId('67e3a4198ea3f781676b140b'), name: 'Laptop' },
  {
    _id: ObjectId('67e3a56a8ea3f781676b140c'),
    name: 'Laptop',
    price: 34300
  },
  {
    _id: ObjectId('67e3a5908ea3f781676b140d'),
    name: 'Laptop',
    price: 34300
  },
  {
    _id: ObjectId('67e3bfc58ea3f781676b140f'),
    name: 'Laptop',
    price: 34300
  },
  {
    _id: ObjectId('67e3bfc58ea3f781676b1410'),
    name: 'Laptop1',
    price: 34300
  },
  {
    _id: ObjectId('67e3bff68ea3f781676b1411'),
    name: 'Laptop1',
    price: 34300
  }
]
newdb>
```

How to Connect to MongoDB using a Connection String?

Local Connection

mongodb://localhost:27017

Remote Connection

mongodb://username:password@host:port/database

Example in Node.js:

```
const mongoose = require('mongoose');
mongoose.connect("mongodb://localhost:27017/newdb", { useNewUrlParser: true,
useUnifiedTopology: true})
  .then(() => console.log ("Connected to MongoDB"))
  .catch (err => console.log(err));
```

Database & Collection in MongoDB

Database in MongoDB

```
use  newdb;
```

Create collection

db.createCollection("users");

Inserting Data into a Collection (Creates Collection If Not Exists)
db.newproducts.insertOne({name: "Mobile", price:25,000.00});

MONGODB CRUD OPERATION

test> db.newProducts.find().pretty() - Find all documents

```
[
  {
    _id: ObjectId('67e4d0a1eb58cfae946b140b'),
    name: 'Laptop',
    description: 'High-end gaming laptop',
    price: 1500,
    category: 'ELectronics',
    inStock: true
  },
  {
    _id: ObjectId('67e4d1cdeb58cfae946b140c'),
    name: 'Smartphone',
    description: 'Latest model with high resolution camera',
    price: 800,
    category: 'Electronics',
    inStock: true
  },
  {
    _id: ObjectId('67e4d1cdeb58cfae946b140d'),
    name: 'Wireless Mouse',
    description: 'Ergonomic design for comfort',
    price: 25,
    category: 'Accessories',
    inStock: true
  }
]
```

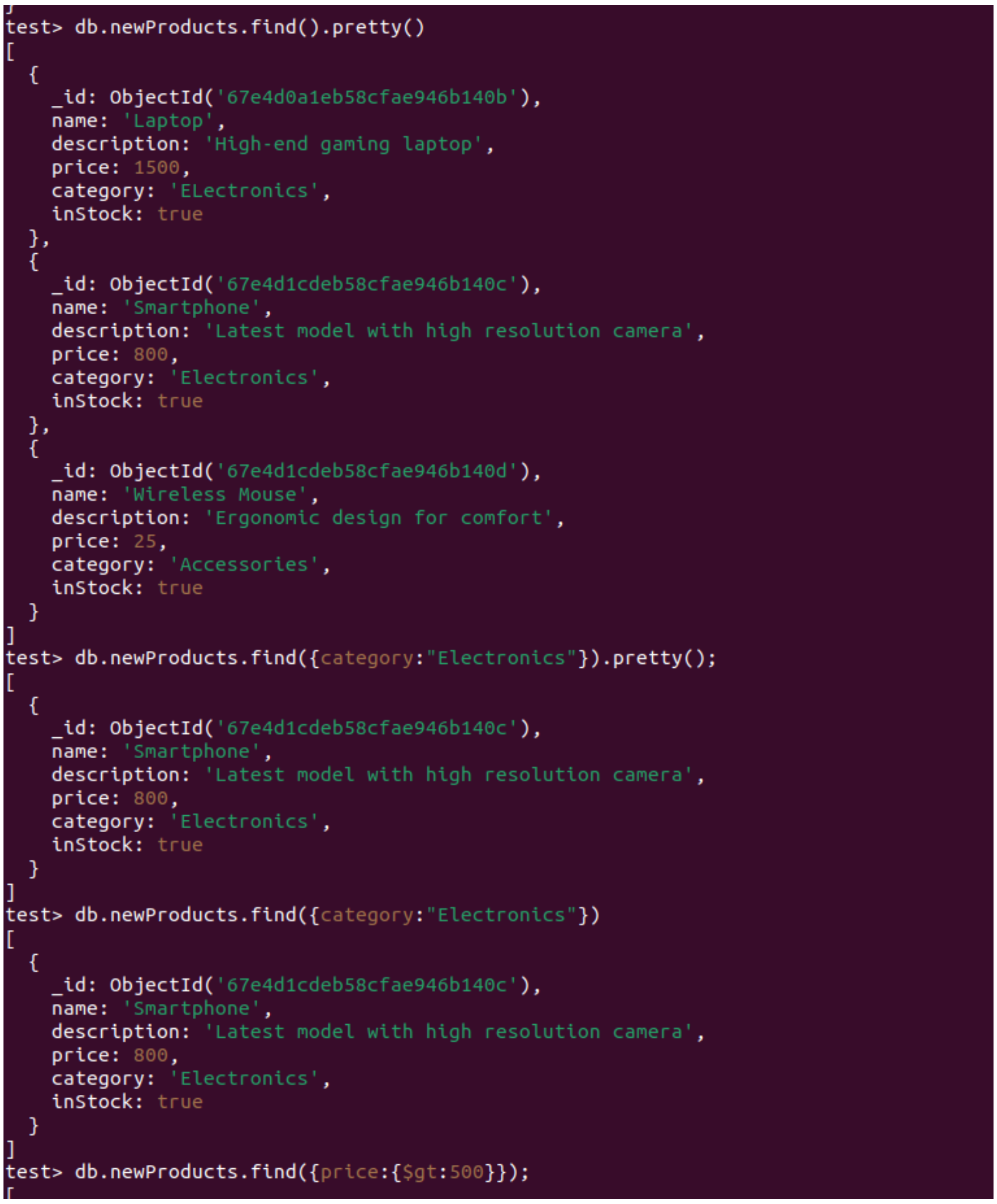
Pretty() in Mongodb:-

The. pretty() method in MongoDB is used to **format query results** in a more readable, structured JSON format.

```
test> db.newProducts.find({category:"Electronics"}).pretty();
[
  {
    _id: ObjectId('67e4d1cdeb58cfae946b140c'),
    name: 'Smartphone',
    description: 'Latest model with high resolution camera',
    price: 800,
    category: 'Electronics',
    inStock: true
  }
]
test> db.newProducts.find({category:"Electronics"}) - finds based on the category
[
  {
    _id: ObjectId('67e4d1cdeb58cfae946b140c'),
    name: 'Smartphone',
    description: 'Latest model with high resolution camera',
    price: 800,
    category: 'Electronics',
    inStock: true
  }
]
```

test> db.newProducts.find({price:{$gt:500}}) - Finds the document based on the condition

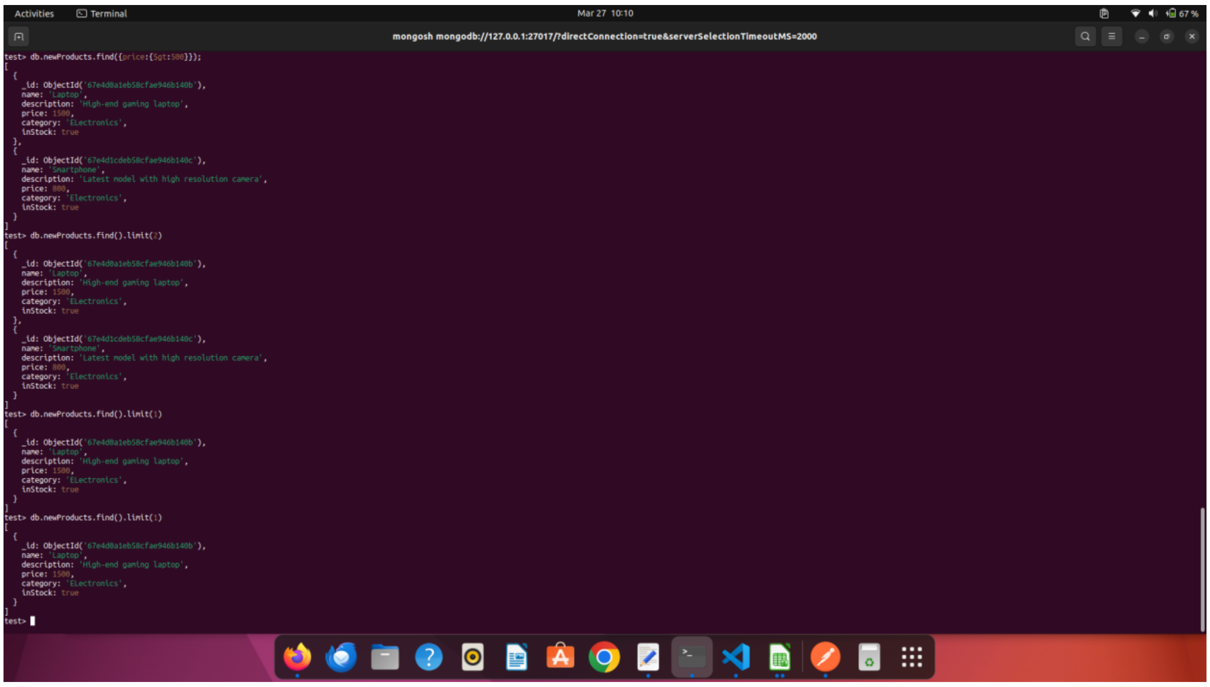
```
[
  {
   _id: ObjectId('67e4d0a1eb58cfae946b140b'),
   name: 'Laptop',
   description: 'High-end gaming laptop',
   price: 1500,
   category: 'ELectronics',
   inStock: true
  },
  {
   _id: ObjectId('67e4d1cdeb58cfae946b140c'),
   name: 'Smartphone',
   description: 'Latest model with high resolution camera',
   price: 800,
   category: 'Electronics',
   inStock: true
  }
]
```


```
test> db.newProducts.find().pretty()
[
  {
    _id: ObjectId('67e4d0a1eb58cfae946b140b'),
    name: 'Laptop',
    description: 'High-end gaming laptop',
    price: 1500,
    category: 'ELectronics',
    inStock: true
  },
  {
    _id: ObjectId('67e4d1cdeb58cfae946b140c'),
    name: 'Smartphone',
    description: 'Latest model with high resolution camera',
    price: 800,
    category: 'Electronics',
    inStock: true
  },
  {
    _id: ObjectId('67e4d1cdeb58cfae946b140d'),
    name: 'Wireless Mouse',
    description: 'Ergonomic design for comfort',
    price: 25,
    category: 'Accessories',
    inStock: true
  }
]
test> db.newProducts.find({category:"Electronics"}).pretty();
[
  {
    _id: ObjectId('67e4d1cdeb58cfae946b140c'),
    name: 'Smartphone',
    description: 'Latest model with high resolution camera',
    price: 800,
    category: 'Electronics',
    inStock: true
  }
]
test> db.newProducts.find({category:"Electronics"})
[
  {
    _id: ObjectId('67e4d1cdeb58cfae946b140c'),
    name: 'Smartphone',
    description: 'Latest model with high resolution camera',
    price: 800,
    category: 'Electronics',
    inStock: true
  }
]
test> db.newProducts.find({price:{$gt:500}});
```

27/03/2025

CRUD OPERATION: -

READ OPERATION: -



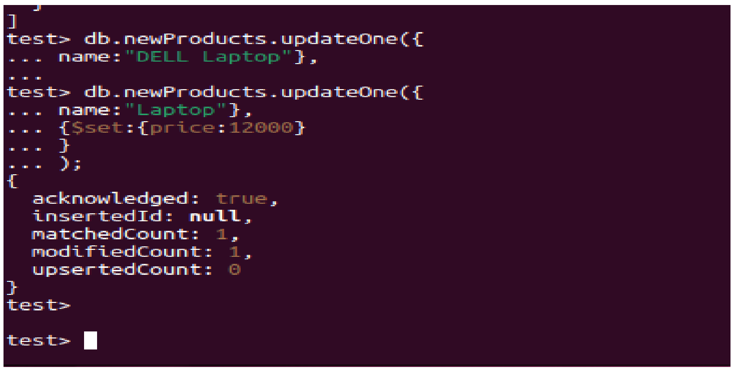
```
test> db.newProducts.find({price:{$gt:500}});
[
  {
    _id: ObjectId('67e4d0a1eb58cfae946b140b'),
    name: 'Laptop',
    description: 'High-end gaming laptop',
    price: 1500,
    category: 'ELectronics',
    inStock: true
  },
  {
    _id: ObjectId('67e4d1cdeb58cfae946b140c'),
    name: 'Smartphone',
    description: 'Latest model with high resolution camera',
    price: 800,
    category: 'Electronics',
    inStock: true
  }
]
test> db.newProducts.find().limit(2)
[
  {
    _id: ObjectId('67e4d0a1eb58cfae946b140b'),
    name: 'Laptop',
    description: 'High-end gaming laptop',
    price: 1500,
    category: 'ELectronics',
    inStock: true
  },
  {
    _id: ObjectId('67e4d1cdeb58cfae946b140c'),
    name: 'Smartphone',
    description: 'Latest model with high resolution camera',
    price: 800,
    category: 'Electronics',
    inStock: true
  }
]
test> db.newProducts.find().limit(1)
[
  {
    _id: ObjectId('67e4d0a1eb58cfae946b140b'),
    name: 'Laptop',
    description: 'High-end gaming laptop',
    price: 1500,
    category: 'ELectronics',
    inStock: true
  }
]
test> db.newProducts.find().limit(1)
[
  {
    _id: ObjectId('67e4d0a1eb58cfae946b140b'),
    name: 'Laptop',
    description: 'High-end gaming laptop',
    price: 1500,
    category: 'ELectronics',
    inStock: true
  }
]
test> 
```

Activities
Terminal
Mar 27 10:10
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
67 %

3. Update Operations

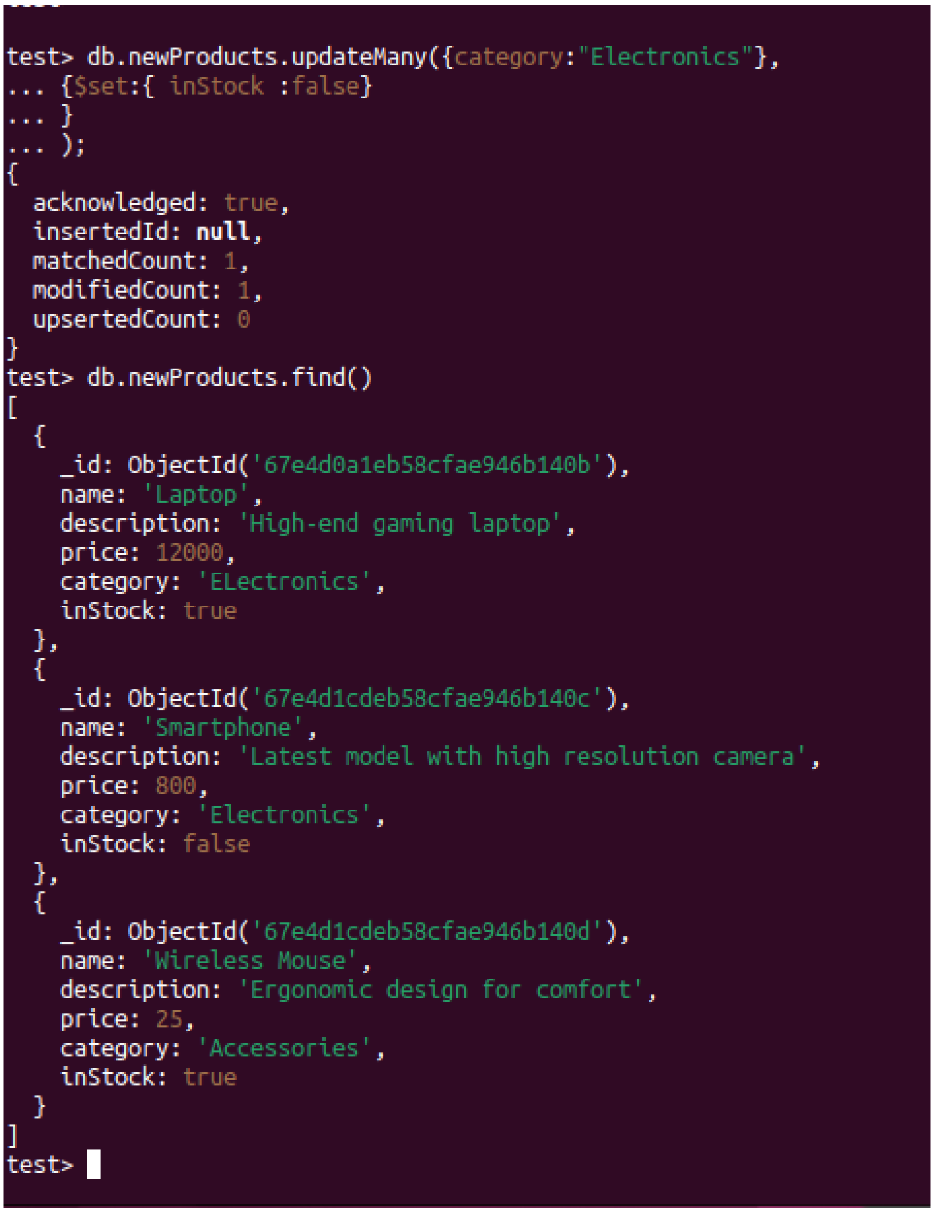
To update documents, we use the updateOne(), updateMany(), and replaceOne() methods.

UpdateOne()



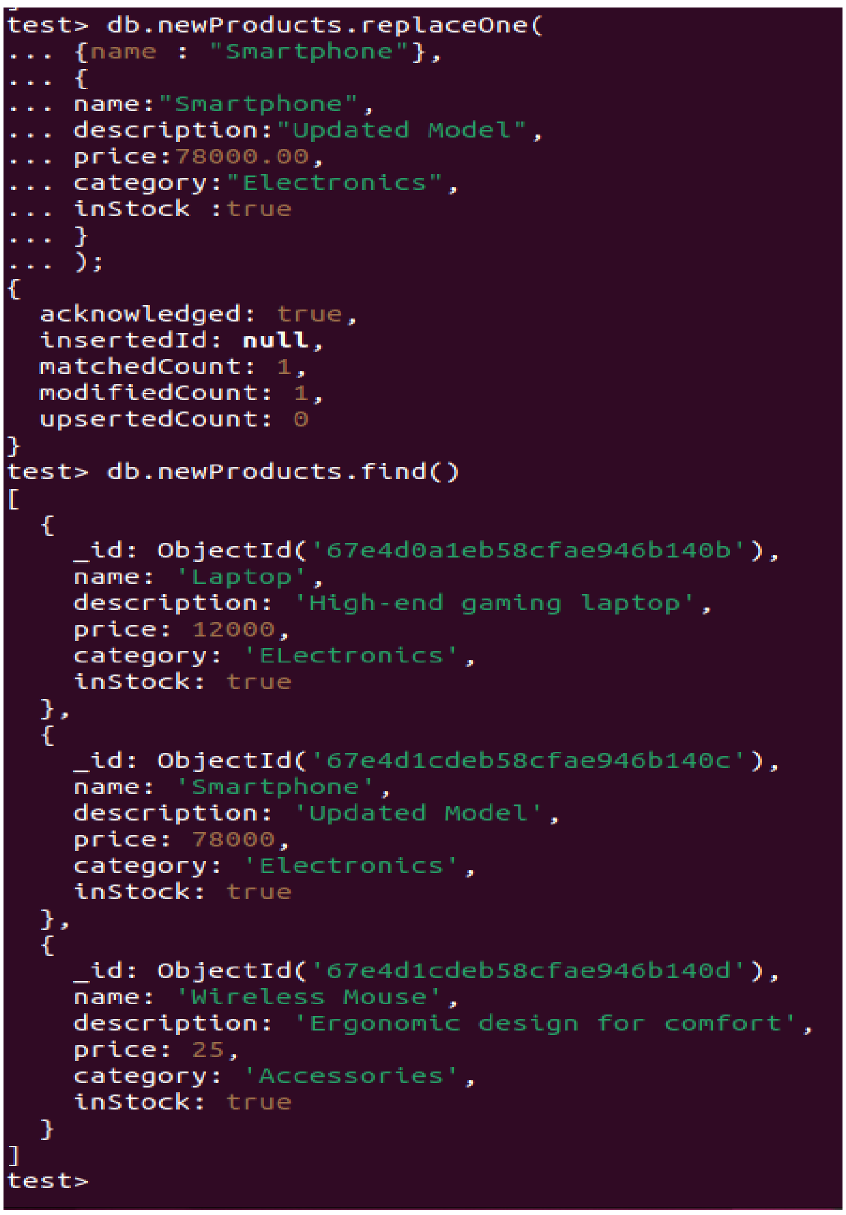
```
]
test> db.newProducts.updateOne({
... name:"DELL Laptop"},
...
test> db.newProducts.updateOne({
... name:"Laptop"},
... {$set:{price:12000}
... }
... );
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
test>

test> 
```

updateMany()

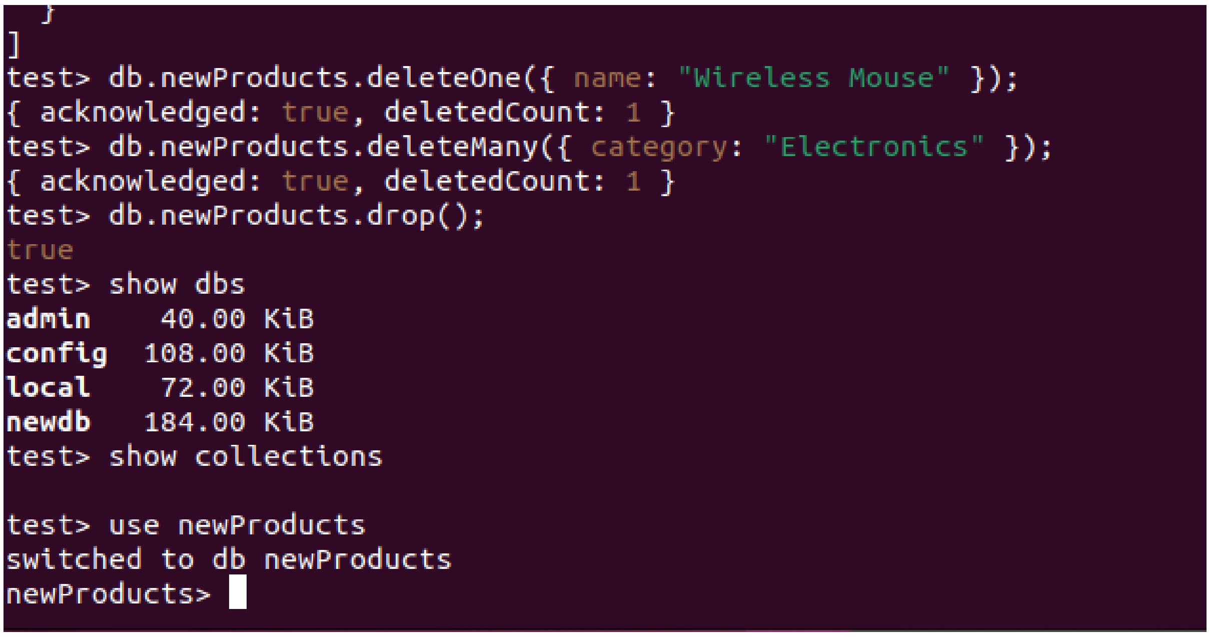


```
test> db.newProducts.updateMany({category:"Electronics"},
... {$set:{ inStock :false}
... }
... );
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
test> db.newProducts.find()
[
  {
    _id: ObjectId('67e4d0a1eb58cfae946b140b'),
    name: 'Laptop',
    description: 'High-end gaming laptop',
    price: 12000,
    category: 'ELectronics',
    inStock: true
  },
  {
    _id: ObjectId('67e4d1cdeb58cfae946b140c'),
    name: 'Smartphone',
    description: 'Latest model with high resolution camera',
    price: 800,
    category: 'Electronics',
    inStock: false
  },
  {
    _id: ObjectId('67e4d1cdeb58cfae946b140d'),
    name: 'Wireless Mouse',
    description: 'Ergonomic design for comfort',
    price: 25,
    category: 'Accessories',
    inStock: true
  }
]
test> 
```

replaceOne()

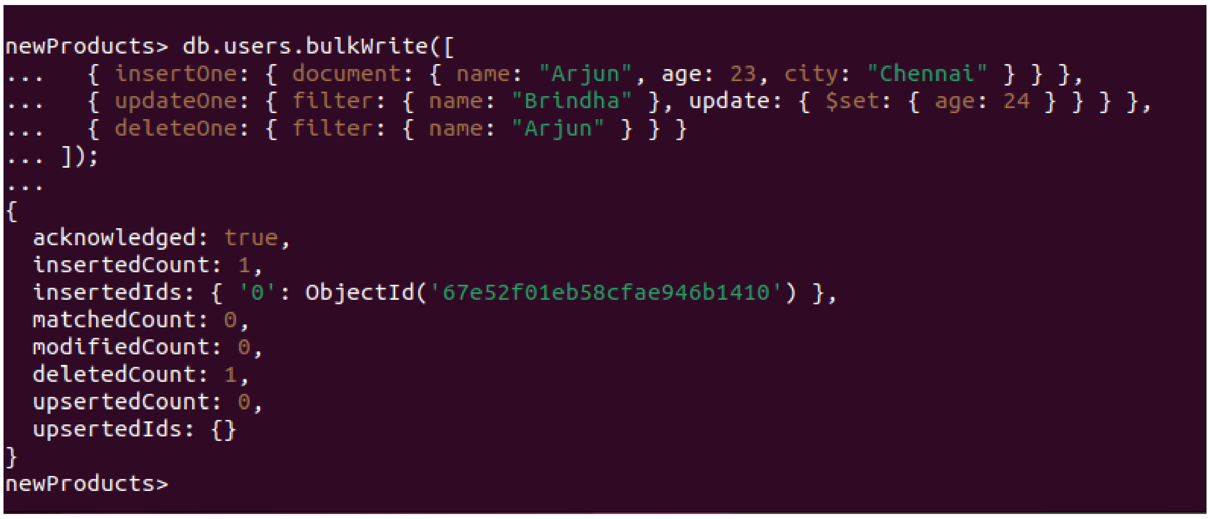

```
test> db.newProducts.replaceOne(
... {name : "Smartphone"},
... {
... name:"Smartphone",
... description:"Updated Model",
... price:78000.00,
... category:"Electronics",
... inStock :true
... }
... );
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
test> db.newProducts.find()
[
  {
    _id: ObjectId('67e4d0a1eb58cfae946b140b'),
    name: 'Laptop',
    description: 'High-end gaming laptop',
    price: 12000,
    category: 'ELectronics',
    inStock: true
  },
  {
    _id: ObjectId('67e4d1cdeb58cfae946b140c'),
    name: 'Smartphone',
    description: 'Updated Model',
    price: 78000,
    category: 'Electronics',
    inStock: true
  },
  {
    _id: ObjectId('67e4d1cdeb58cfae946b140d'),
    name: 'Wireless Mouse',
    description: 'Ergonomic design for comfort',
    price: 25,
    category: 'Accessories',
    inStock: true
  }
]
test>
```

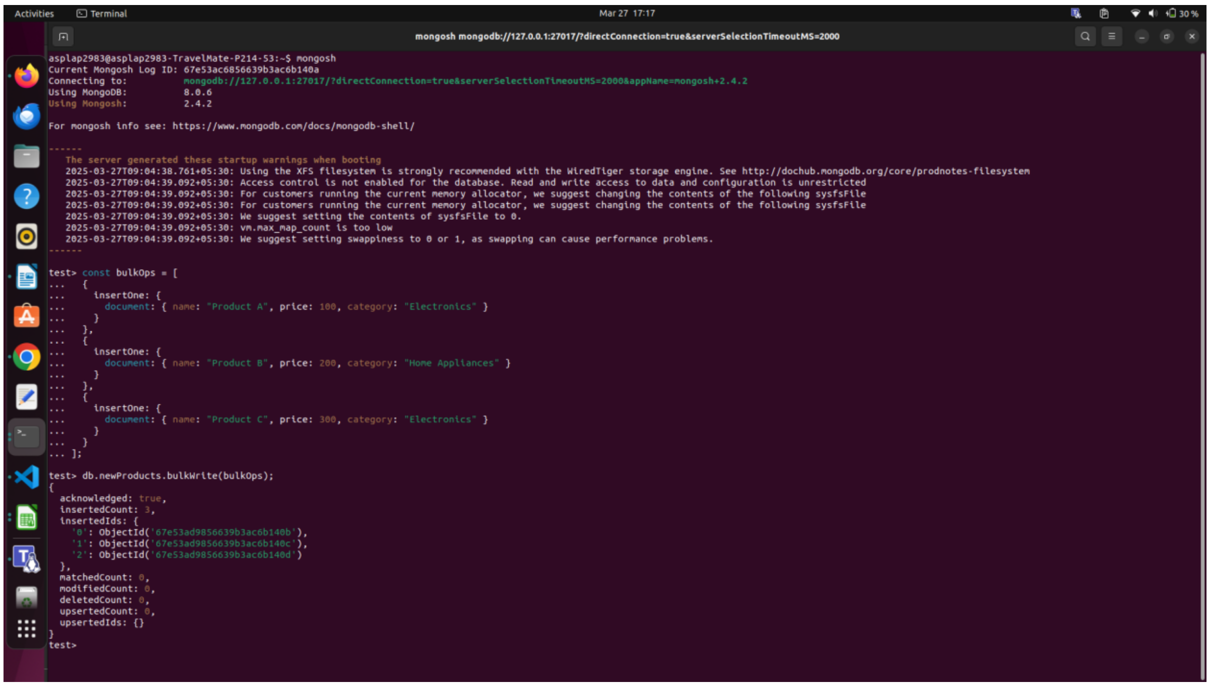
deleteOne(),deleteMany(),drop()


```
  }
]
test> db.newProducts.deleteOne({ name: "Wireless Mouse" });
{ acknowledged: true, deletedCount: 1 }
test> db.newProducts.deleteMany({ category: "Electronics" });
{ acknowledged: true, deletedCount: 1 }
test> db.newProducts.drop();
true
test> show dbs
admin     40.00 KiB
config   108.00 KiB
local     72.00 KiB
newdb    184.00 KiB
test> show collections

test> use newProducts
switched to db newProducts
newProducts> 
```

BulkWrite()

```
db.users.bulkWrite([
 { insertOne: {document: {name: "Arjun", age: 23, city: "Chennai"}}},
 { updateOne: {filter: {name: "Brindha"}, update: {$set: {age: 24}}}},
 { deleteOne: {filter: {name: "Arjun"}}}
]);
```

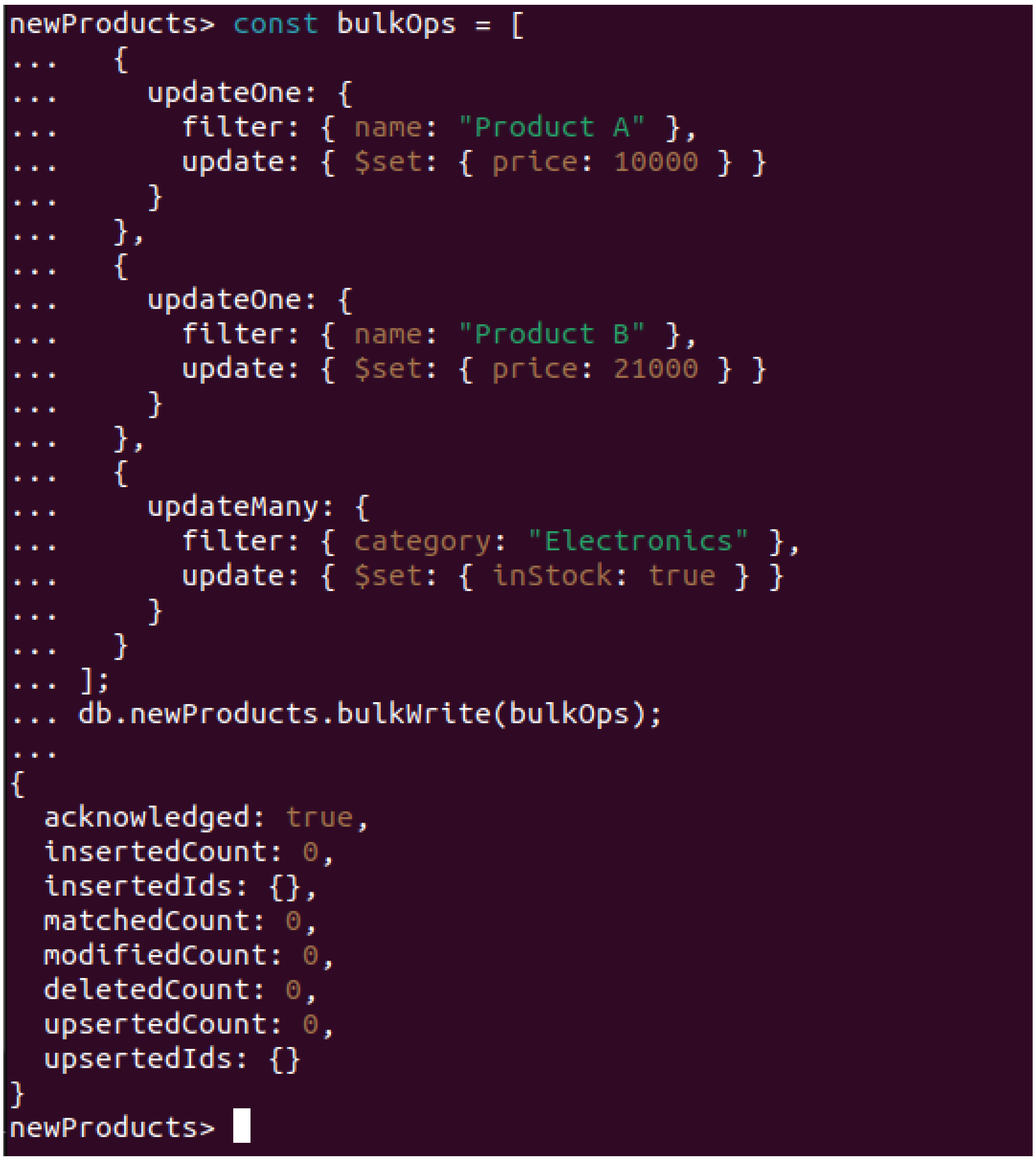

```
newProducts> db.users.bulkWrite([
...   { insertOne: { document: { name: "Arjun", age: 23, city: "Chennai" } } },
...   { updateOne: { filter: { name: "Brindha" }, update: { $set: { age: 24 } } } },
...   { deleteOne: { filter: { name: "Arjun" } } }
... ]);
...
{
  acknowledged: true,
  insertedCount: 1,
  insertedIds: { '0': ObjectId('67e52f01eb58cfae946b1410') },
  matchedCount: 0,
  modifiedCount: 0,
  deletedCount: 1,
  upsertedCount: 0,
  upsertedIds: {}
}
newProducts>
```


Activities
Terminal
Mar 27 17:17
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

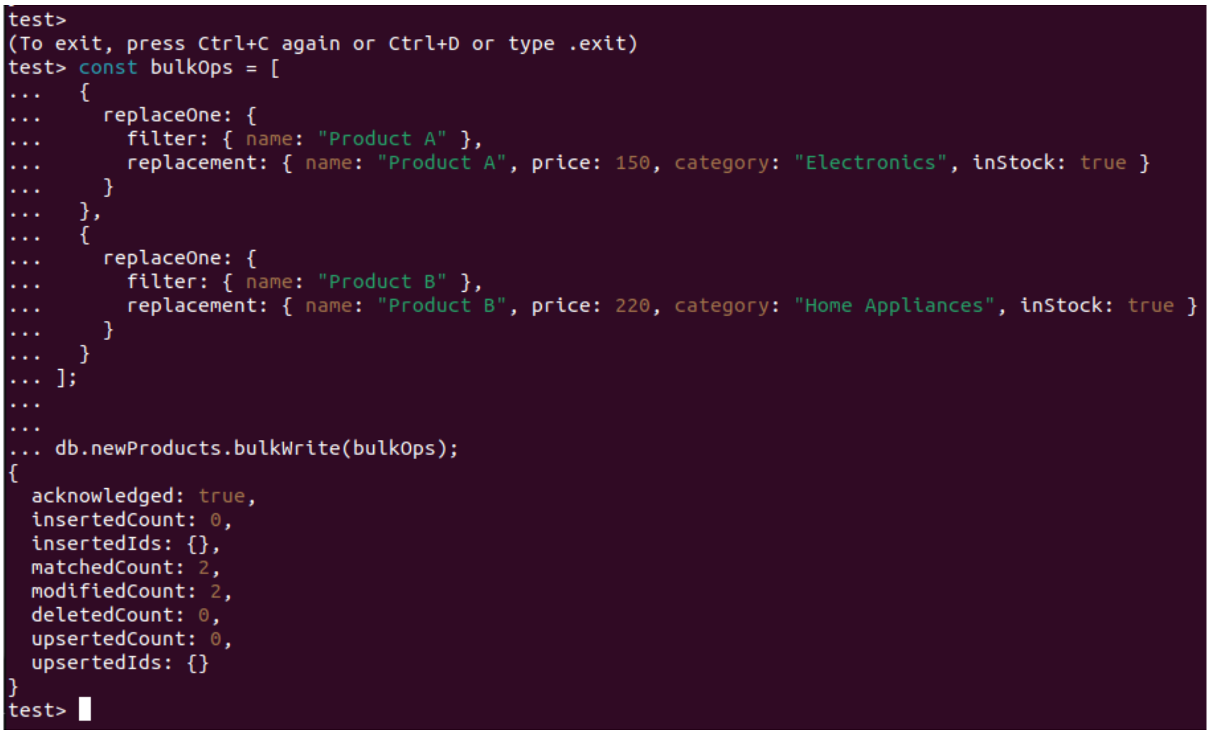
```
asplap2983@asplap2983-TravelMate-P214-53:~$ mongosh
Current Mongosh Log ID: 67e53ac6856639b3ac6b140a
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.4.2
Using MongoDB:          8.0.6
Using Mongosh:          2.4.2

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

------
   The server generated these startup warnings when booting
   2025-03-27T09:04:38.761+05:30: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
   2025-03-27T09:04:39.092+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
   2025-03-27T09:04:39.092+05:30: For customers running the current memory allocator, we suggest changing the contents of the following sysfsFile
   2025-03-27T09:04:39.092+05:30: For customers running the current memory allocator, we suggest changing the contents of the following sysfsFile
   2025-03-27T09:04:39.092+05:30: We suggest setting the contents of sysfsFile to 0.
   2025-03-27T09:04:39.092+05:30: vm.max_map_count is too low
   2025-03-27T09:04:39.092+05:30: We suggest setting swappiness to 0 or 1, as swapping can cause performance problems.
------

test> const bulkOps = [
...   {
...     insertOne: {
...       document: { name: "Product A", price: 100, category: "Electronics" }
...     }
...   },
...   {
...     insertOne: {
...       document: { name: "Product B", price: 200, category: "Home Appliances" }
...     }
...   },
...   {
...     insertOne: {
...       document: { name: "Product C", price: 300, category: "Electronics" }
...     }
...   }
... ];

test> db.newProducts.bulkWrite(bulkOps);
{
  acknowledged: true,
  insertedCount: 3,
  insertedIds: {
    '0': ObjectId('67e53ad9856639b3ac6b140b'),
    '1': ObjectId('67e53ad9856639b3ac6b140c'),
    '2': ObjectId('67e53ad9856639b3ac6b140d')
  },
  matchedCount: 0,
  modifiedCount: 0,
  deletedCount: 0,
  upsertedCount: 0,
  upsertedIds: {}
}
test>
```

```
const bulkOps = [
...   {
...     insertOne: {
...       document: {name: "Product A", price: 100, category: "Electronics" }
...     }
...   },
...   {
...     insertOne: {
...       document: {name: "Product B", price: 200, category: "Home Appliances" }
...     }
...   },
...   {
...     insertOne: {
...       document: {name: "Product C", price: 300, category: "Electronics" }
...     }
...   }
... ];

test> db.newProducts.bulkWrite(bulkOps);
{
  acknowledged: true,
  insertedCount: 3,
  insertedIds: {
    '0': ObjectId('67e53ad9856639b3ac6b140b'),
    '1': ObjectId('67e53ad9856639b3ac6b140c'),
    '2': ObjectId('67e53ad9856639b3ac6b140d')
```

```
 },
 matchedCount: 0,
 modifiedCount: 0,
 deletedCount: 0,
 upsertedCount: 0,
 upsertedIds: {}
}
```



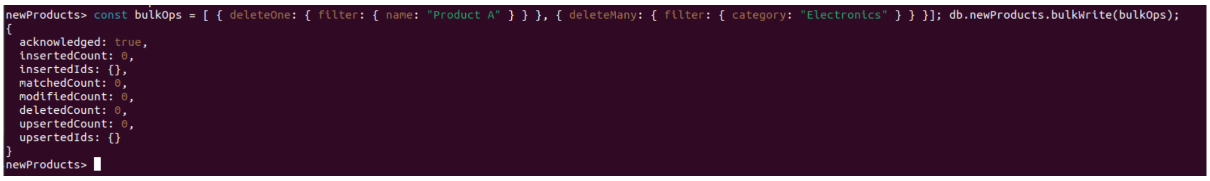
```
newProducts> const bulkOps = [
...   {
...     updateOne: {
...       filter: { name: "Product A" },
...       update: { $set: { price: 10000 } }
...     }
...   },
...   {
...     updateOne: {
...       filter: { name: "Product B" },
...       update: { $set: { price: 21000 } }
...     }
...   },
...   {
...     updateMany: {
...       filter: { category: "Electronics" },
...       update: { $set: { inStock: true } }
...     }
...   }
... ];
... db.newProducts.bulkWrite(bulkOps);
...
{
  acknowledged: true,
  insertedCount: 0,
  insertedIds: {},
  matchedCount: 0,
  modifiedCount: 0,
  deletedCount: 0,
  upsertedCount: 0,
  upsertedIds: {}
}
newProducts> 
```



```
test>
(To exit, press Ctrl+C again or Ctrl+D or type .exit)
test> const bulkOps = [
...   {
...     replaceOne: {
...       filter: { name: "Product A" },
...       replacement: { name: "Product A", price: 150, category: "Electronics", inStock: true }
...     }
...   },
...   {
...     replaceOne: {
...       filter: { name: "Product B" },
...       replacement: { name: "Product B", price: 220, category: "Home Appliances", inStock: true }
...     }
...   }
... ];
...
...
... db.newProducts.bulkWrite(bulkOps);
{
  acknowledged: true,
  insertedCount: 0,
  insertedIds: {},
  matchedCount: 2,
  modifiedCount: 2,
  deletedCount: 0,
  upsertedCount: 0,
  upsertedIds: {}
}
test> 
```

replaceOne()
newProducts>

```
... const bulkOps = [
...   {
...     replaceOne: {
...       filter: {name: "Product A"},
...       replacement: {name: "Product A", price: 15000, category: "Electronics", inStock: true}
...     }
...   },
...   {
...     replaceOne: {
...       filter: {name: "Product B"},
...       replacement: {name: "Product B", price: 22000, category: "Home Appliances", inStock: true}
...     }
...   }
... ];
...
...
... db.newProducts.bulkWrite(bulkOps);
{
  acknowledged: true,
  insertedCount: 0,
  insertedIds: {},
  matchedCount: 0,
  modifiedCount: 0,
  deletedCount: 0,
  upsertedCount: 0,
  upsertedIds: {}
}
```



```
newProducts> const bulkOps = [ { deleteOne: { filter: { name: "Product A" } } }, { deleteMany: { filter: { category: "Electronics" } } }]; db.newProducts.bulkWrite(bulkOps);
{
  acknowledged: true,
  insertedCount: 0,
  insertedIds: {},
  matchedCount: 0,
  modifiedCount: 0,
  deletedCount: 0,
  upsertedCount: 0,
  upsertedIds: {}
}
newProducts> 
```

Aggregation Pipelines

Certainly! Below is a list of each MongoDB Aggregation Pipeline stage with its one-line definition:

Aggregation Pipeline Stages with One-Line Definitions:

1. **$match**:
    a. Filters documents based on specified conditions (similar to find queries).
2. **$group**:
    a. Groups documents by a specified field and computes aggregate values (sum, count, average, etc.).

3. **$sort**:
    a. Sorts the documents based on specified fields in ascending or descending order.
4. **$project**:
    a. Reshapes each document by including, excluding, or transforming fields.
5. **$limit**:
    a. Limits the number of documents passed through the pipeline.
6. **$skip**:
    a. Skips a specified number of documents before passing the remaining documents.
7. **$unwind**:
    a. Deconstructs an array field to output a document for each element in the array.
8. **$lookup**:
    a. Joins documents from another collection based on a common field.
9. **$addFields**:
    a. Adds new fields or modifies existing fields in the documents.
10. **$count**:
    a. Counts the number of documents passing through the pipeline and outputs a count field.
11. **$facet**:
    a. Allows multiple aggregation pipelines to run in parallel and outputs the results as different fields.
12. **$geoNear**:
    a. Performs a geospatial query, returning documents sorted by proximity to a specific point.
13. **$sample**:
    a. Randomly selects a specified number of documents from the collection.
14. **$merge**:
    a. Writes the results of the aggregation pipeline to a specified collection.
15. **$replaceRoot**:
    a. Replaces the root document with a new document or sub-document.

Certainly! Below is a list of each MongoDB Aggregation Pipeline stage with one-line definitions and an example for each:

1. $match

Definition: Filters documents based on specified conditions (similar to find queries).

Example:

```
db.products.aggregate([
 { $match: { category: "Electronics" } }
]);
```

2. $group

Definition: Groups documents by a specified field and computes aggregate values (sum, count, average, etc.).

Example:

```
db.products.aggregate([
 {
   $group: {
    _id: "$category",
    totalPrice: { $sum: "$price" },
    count: { $sum: 1 }
   }
 }
]);
```

3. $sort

Definition: Sorts the documents based on specified fields in ascending or descending order.

Example:

```
db.products.aggregate([
 { $sort: { price: -1 } }  // Sort by price in descending order
]);
```

4. $project

Definition: Reshapes each document by including, excluding, or transforming fields.

Example:

```
db.products.aggregate([
 {
   $project: {
     name: 1,
     price: 1,
     discountPrice: { $multiply: ["$price", 0.9] }  // Create a new field with a 10% discount
   }
 }
]);
```

5. $limit

Definition: Limits the number of documents passed through the pipeline.

Example:

```
db.products.aggregate([
 { $limit: 5 }  // Limit the result to the top 5 documents
]);
```

6. $skip

Definition: Skips a specified number of documents before passing the remaining documents.

Example:

```
db.products.aggregate([
 { $skip: 3 }  // Skip the first 3 documents
]);
```

7. $unwind

Definition: Deconstructs an array field to output a document for each element in the array.

Example:

```
db.orders.aggregate([
 { $unwind: "$items" } // Unwind the items array in each order
]);
```

8. $lookup

Definition: Joins documents from another collection based on a common field.

Example:

```
db.orders.aggregate([
 {
   $lookup: {
     from: "products",
     localField: "productId",
     foreignField: "_id",
     as: "productDetails"
   }
 }
]);
```

9. $addFields

Definition: Adds new fields or modifies existing fields in the documents.

Example:

```
db.products.aggregate([
 {
   $addFields: {
     salePrice: { $multiply: ["$price", 0.8] } // Add a sale price field with 20% off
```

```
    }
  }
]);
```

10. $count

Definition: Counts the number of documents passing through the pipeline and outputs a count field.

Example:

```
db.products.aggregate([
  { $count: "totalProducts" }  // Count the total number of products
]);
```

11. $facet

Definition: Allows multiple aggregation pipelines to run in parallel and outputs the results as different fields.

Example:

```
db.orders.aggregate([
  {
    $facet: {
      totalSales: [
        { $group: { _id: null, totalAmount: { $sum: "$total" } } }
      ],
      totalOrders: [
        { $count: "total" }
      ]
    }
  }
]);
```

12. $geoNear

Definition: Performs a geospatial query, returning documents sorted by proximity to a specific point.

Example:

```
db.locations.aggregate([
 {
   $geoNear: {
     near: { type: "Point", coordinates: [100, 200] },
     distanceField: "distance",
     spherical: true
   }
 }
]);
```

13. $sample

Definition: Randomly selects a specified number of documents from the collection.

Example:

```
db.products.aggregate([
 { $sample: { size: 3 } }  // Randomly select 3 products
]);
```

14. $merge

Definition: Writes the results of the aggregation pipeline to a specified collection.

Example:

```
db.products.aggregate([
 {
   $group: {
     _id: "$category",
     totalPrice: { $sum: "$price" }
   }
  },
  { $merge: { into: "aggregated_results" } }  // Write the result to "aggregated_results"
collection
]);
```

15. $replaceRoot

Definition: Replaces the root document with a new document or sub-document.

Example:

```
db.orders.aggregate([
  {
    $replaceRoot: { newRoot: "$productDetails" }  // Replace root with productDetails
  }
]);
```