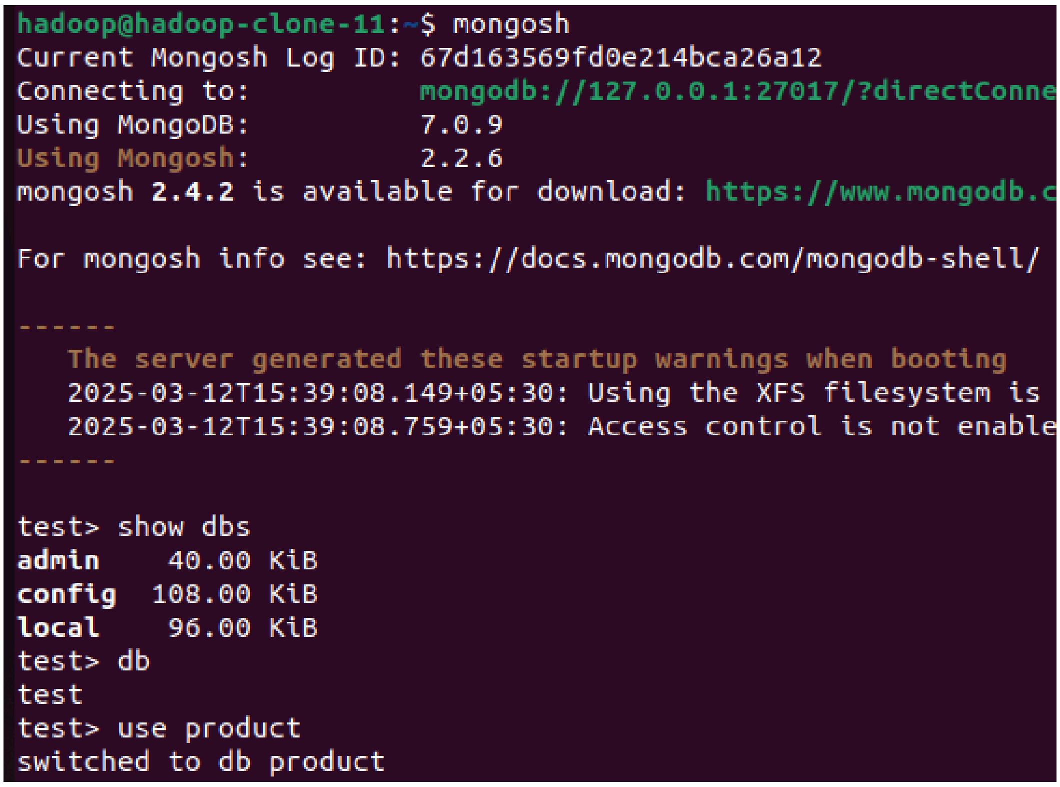
LAB - 05

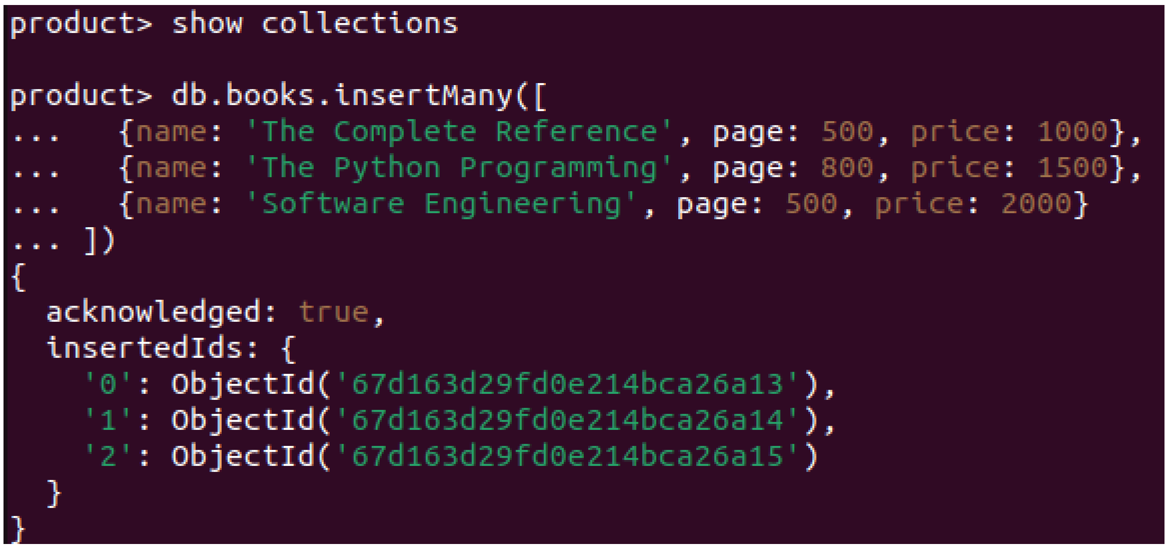
Aim:- To learn and understand the use of mongodb to create and analyze the data.

Connect to a Local Deployment on the Default Port:
mongosh
To show list of data bases available in the system use following command:
> show dbs
Create a New Database:
> use product:
To display the name of current working Database:
> db



```
hadoop@hadoop-clone-11:~$ mongosh
Current Mongosh Log ID: 67d163569fd0e214bca26a12
Connecting to:          mongodb://127.0.0.1:27017/?directConne
Using MongoDB:          7.0.9
Using Mongosh:          2.2.6
mongosh 2.4.2 is available for download: https://www.mongodb.c

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

------
   The server generated these startup warnings when booting
   2025-03-12T15:39:08.149+05:30: Using the XFS filesystem is
   2025-03-12T15:39:08.759+05:30: Access control is not enable
------

test> show dbs
admin     40.00 KiB
config   108.00 KiB
local     96.00 KiB
test> db
test
test> use product
switched to db product
```

To Create a Collection
To Display a Collections
show collections

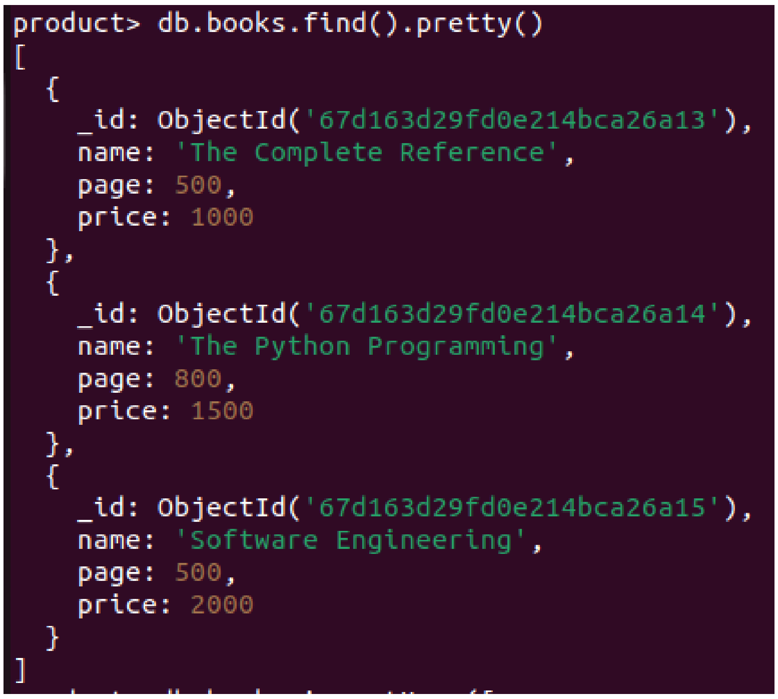


```
product> show collections

product> db.books.insertMany([
...    {name: 'The Complete Reference', page: 500, price: 1000},
...    {name: 'The Python Programming', page: 800, price: 1500},
...    {name: 'Software Engineering', page: 500, price: 2000}
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67d163d29fd0e214bca26a13'),
    '1': ObjectId('67d163d29fd0e214bca26a14'),
    '2': ObjectId('67d163d29fd0e214bca26a15')
  }
}
```

To retrieve collection

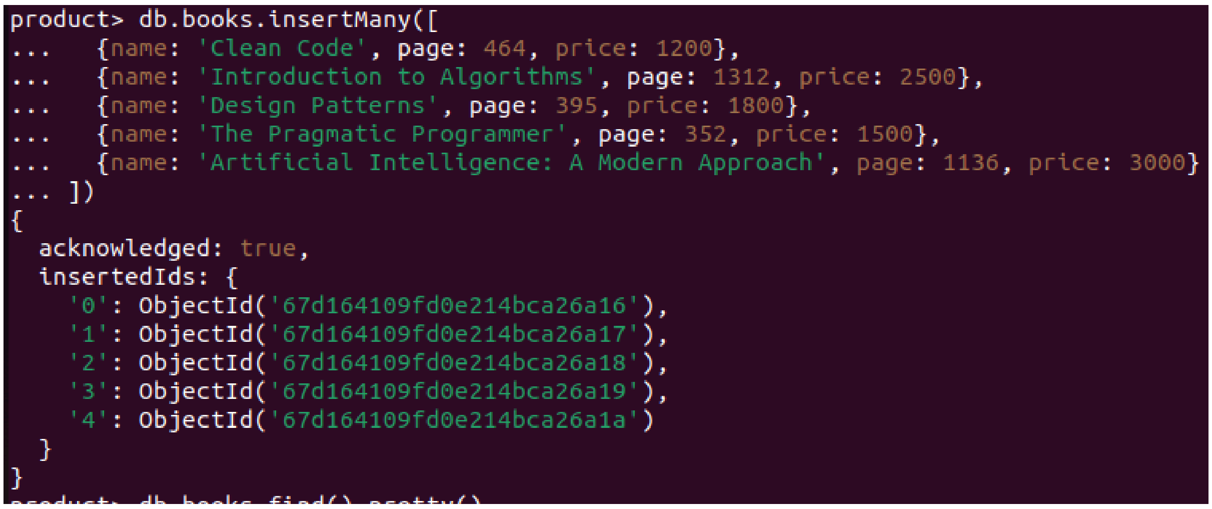
1. db.collection.find(): >db.books.find()
2. db.collection.find().pretty(): To pretify the output

>db.books.find().pretty()



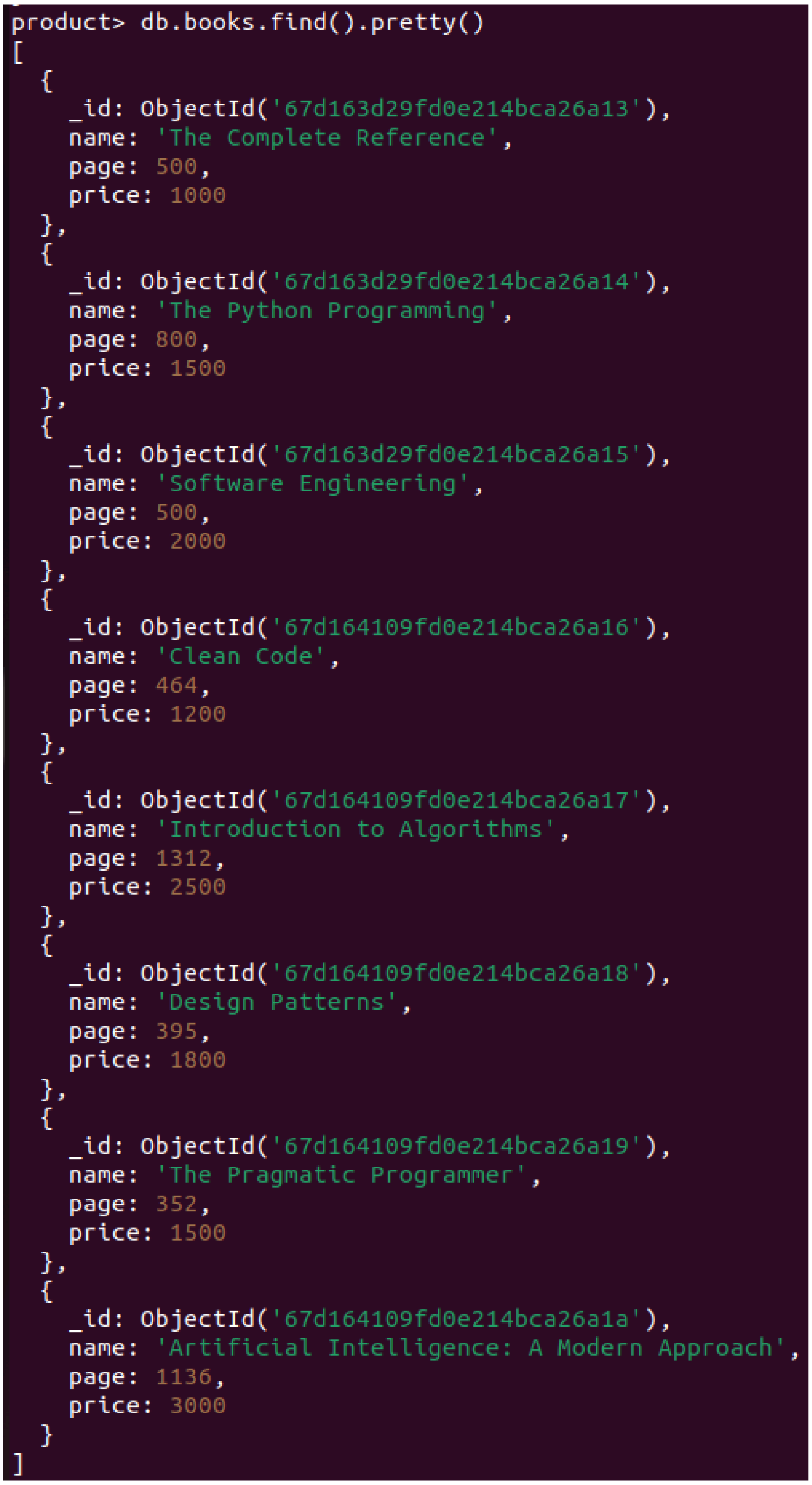
```
product> db.books.find().pretty()
[
  {
    _id: ObjectId('67d163d29fd0e214bca26a13'),
    name: 'The Complete Reference',
    page: 500,
    price: 1000
  },
  {
    _id: ObjectId('67d163d29fd0e214bca26a14'),
    name: 'The Python Programming',
    page: 800,
    price: 1500
  },
  {
    _id: ObjectId('67d163d29fd0e214bca26a15'),
    name: 'Software Engineering',
    page: 500,
    price: 2000
  }
]
```

Insert Documents(insert command):



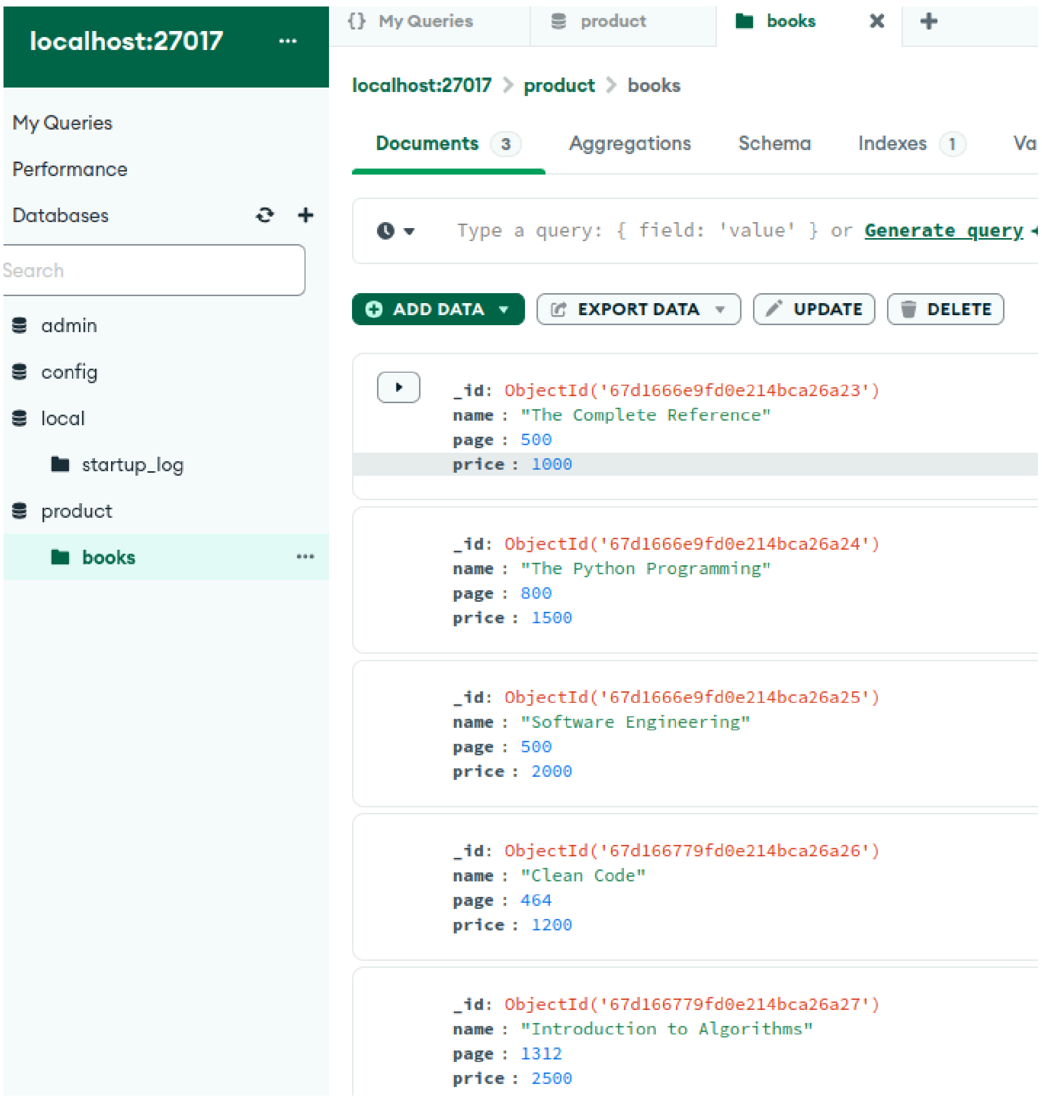
```
product> db.books.insertMany([
...   {name: 'Clean Code', page: 464, price: 1200},
...   {name: 'Introduction to Algorithms', page: 1312, price: 2500},
...   {name: 'Design Patterns', page: 395, price: 1800},
...   {name: 'The Pragmatic Programmer', page: 352, price: 1500},
...   {name: 'Artificial Intelligence: A Modern Approach', page: 1136, price: 3000}
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67d164109fd0e214bca26a16'),
    '1': ObjectId('67d164109fd0e214bca26a17'),
    '2': ObjectId('67d164109fd0e214bca26a18'),
    '3': ObjectId('67d164109fd0e214bca26a19'),
    '4': ObjectId('67d164109fd0e214bca26a1a')
  }
}
```

To retrieve this collection:

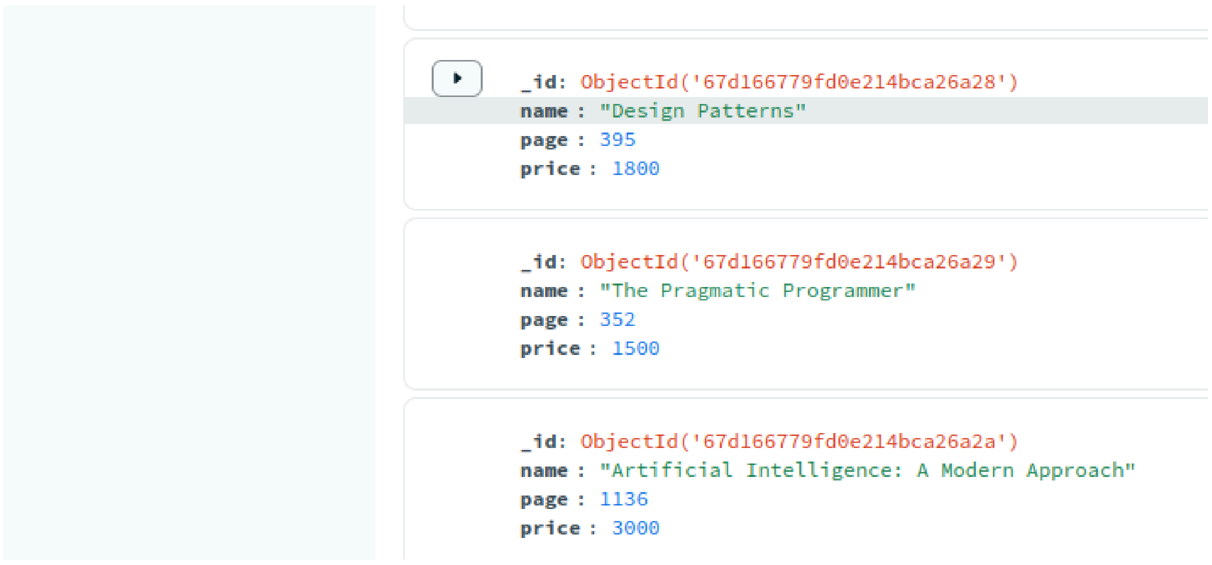


```
product> db.books.find().pretty()
[
  {
    _id: ObjectId('67d163d29fd0e214bca26a13'),
    name: 'The Complete Reference',
    page: 500,
    price: 1000
  },
  {
    _id: ObjectId('67d163d29fd0e214bca26a14'),
    name: 'The Python Programming',
    page: 800,
    price: 1500
  },
  {
    _id: ObjectId('67d163d29fd0e214bca26a15'),
    name: 'Software Engineering',
    page: 500,
    price: 2000
  },
  {
    _id: ObjectId('67d164109fd0e214bca26a16'),
    name: 'Clean Code',
    page: 464,
    price: 1200
  },
  {
    _id: ObjectId('67d164109fd0e214bca26a17'),
    name: 'Introduction to Algorithms',
    page: 1312,
    price: 2500
  },
  {
    _id: ObjectId('67d164109fd0e214bca26a18'),
    name: 'Design Patterns',
    page: 395,
    price: 1800
  },
  {
    _id: ObjectId('67d164109fd0e214bca26a19'),
    name: 'The Pragmatic Programmer',
    page: 352,
    price: 1500
  },
  {
    _id: ObjectId('67d164109fd0e214bca26a1a'),
    name: 'Artificial Intelligence: A Modern Approach',
    page: 1136,
    price: 3000
  }
]
```

The database output looks like this:

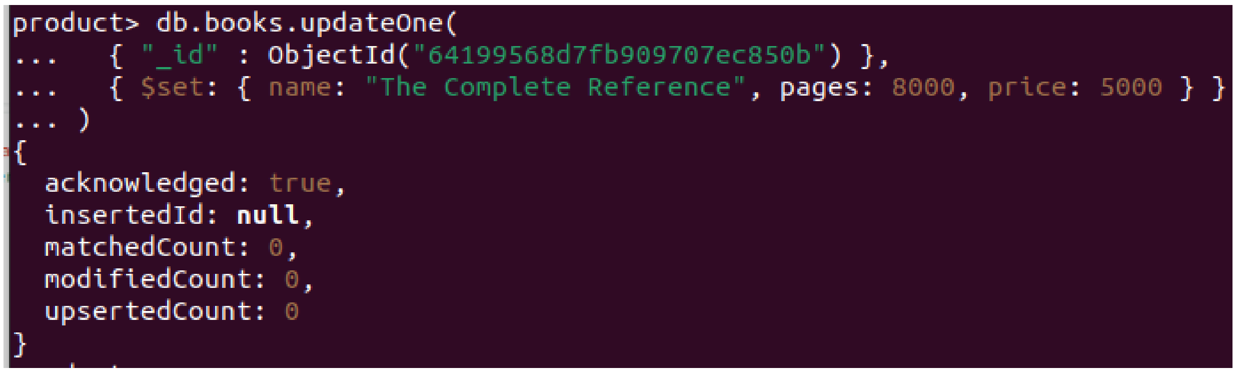


```
_id: ObjectId('67d1666e9fd0e214bca26a23')
name : "The Complete Reference"
page : 500
price : 1000

_id: ObjectId('67d1666e9fd0e214bca26a24')
name : "The Python Programming"
page : 800
price : 1500

_id: ObjectId('67d1666e9fd0e214bca26a25')
name : "Software Engineering"
page : 500
price : 2000

_id: ObjectId('67d166779fd0e214bca26a26')
name : "Clean Code"
page : 464
price : 1200

_id: ObjectId('67d166779fd0e214bca26a27')
name : "Introduction to Algorithms"
page : 1312
price : 2500
```



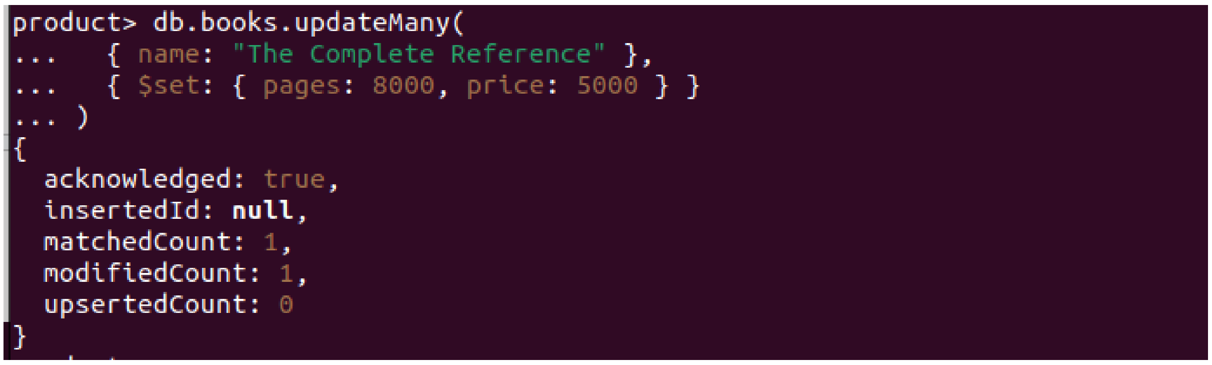
```
_id: ObjectId('67d166779fd0e214bca26a28')
name : "Design Patterns"
page : 395
price : 1800

_id: ObjectId('67d166779fd0e214bca26a29')
name : "The Pragmatic Programmer"
page : 352
price : 1500

_id: ObjectId('67d166779fd0e214bca26a2a')
name : "Artificial Intelligence: A Modern Approach"
page : 1136
price : 3000
```

Update a Single Document

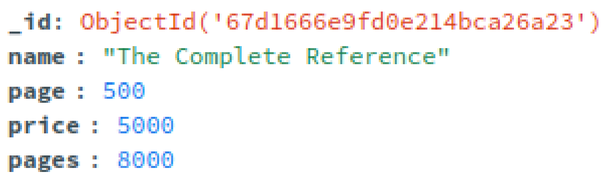


```
product> db.books.updateOne(
...    { "_id" : ObjectId("64199568d7fb909707ec850b") },
...    { $set: { name: "The Complete Reference", pages: 8000, price: 5000 } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
```

Update Multiple Documents

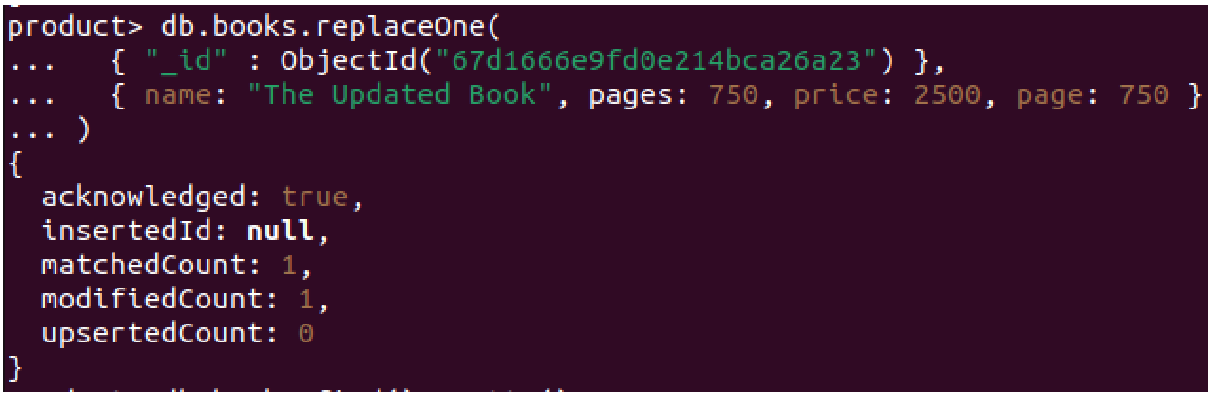


```
product> db.books.updateMany(
...    { name: "The Complete Reference" },
...    { $set: { pages: 8000, price: 5000 } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

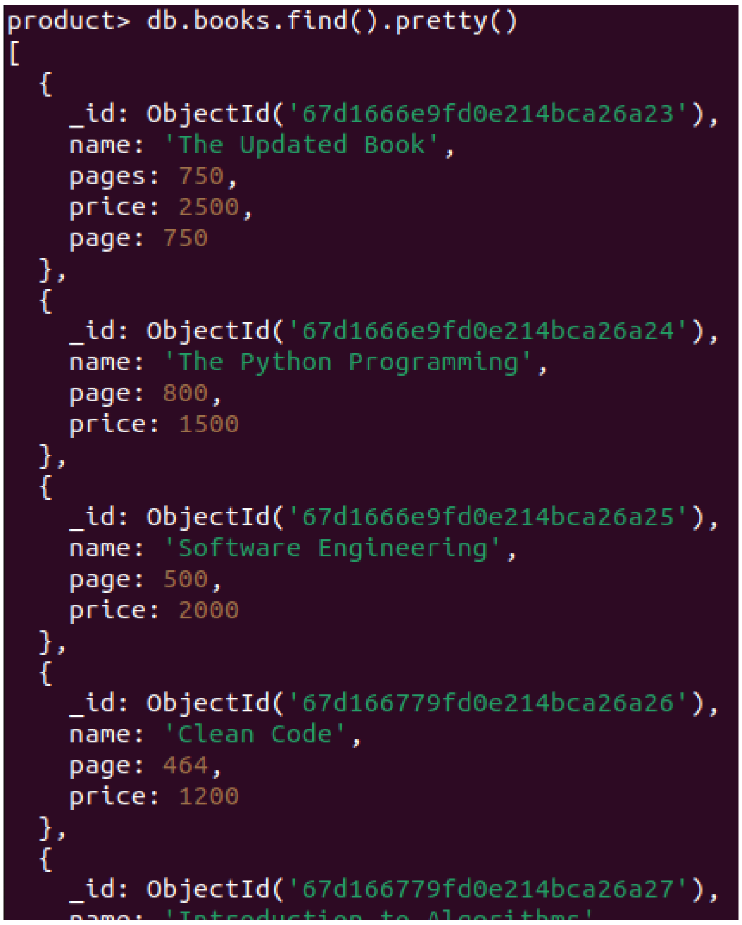


```
_id: ObjectId('67d1666e9fd0e214bca26a23')
name : "The Complete Reference"
page : 500
price : 5000
pages : 8000
```

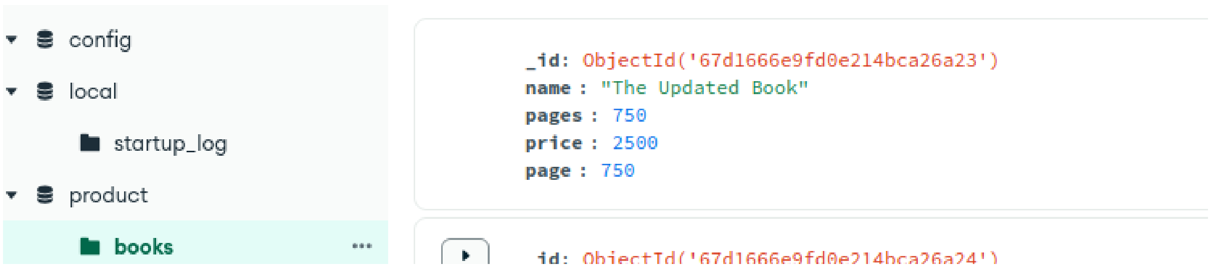
Replace a Document



```
product> db.books.replaceOne(
...    { "_id" : ObjectId("67d1666e9fd0e214bca26a23") },
...    { name: "The Updated Book", pages: 750, price: 2500, page: 750 }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```



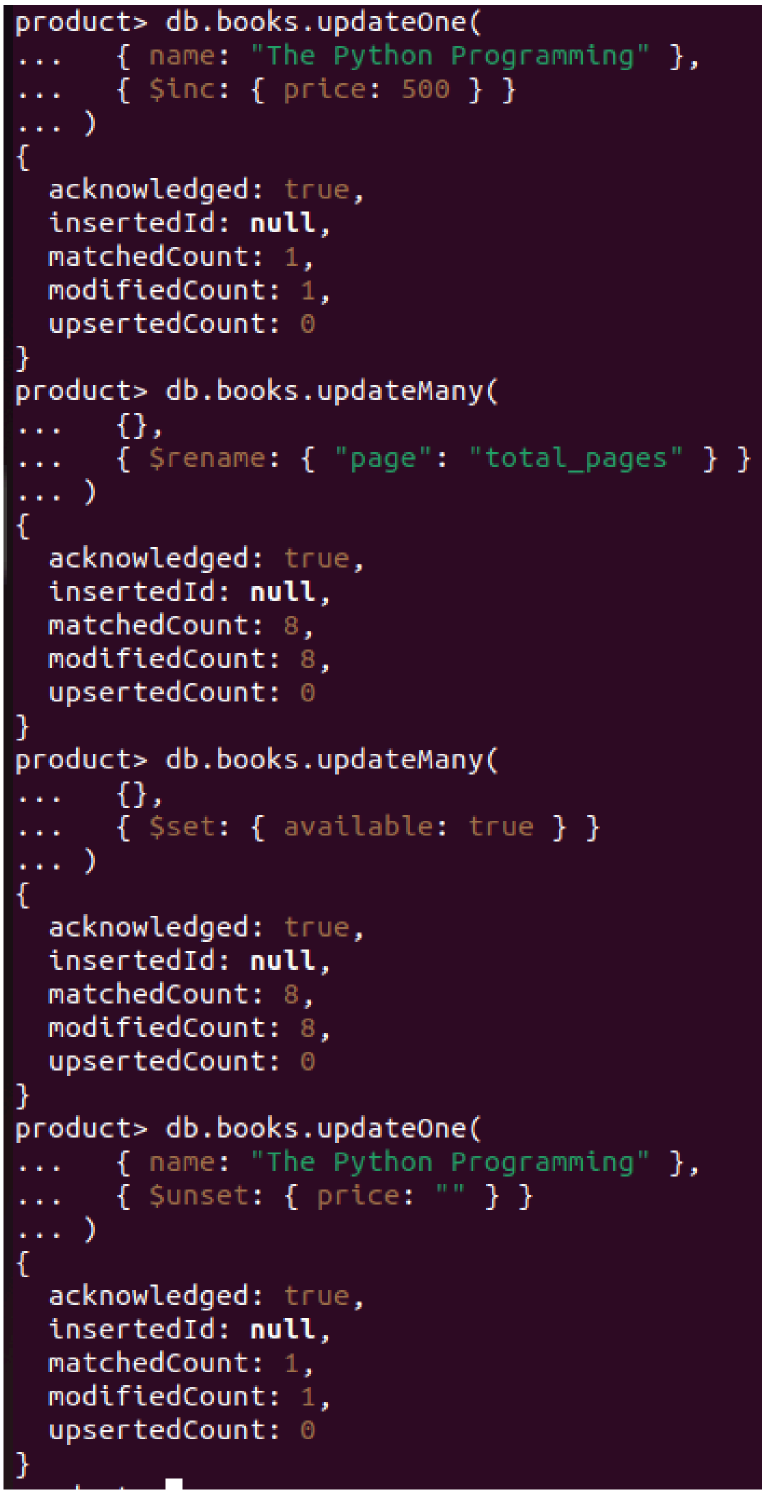
```
product> db.books.find().pretty()
[
  {
    _id: ObjectId('67d1666e9fd0e214bca26a23'),
    name: 'The Updated Book',
    pages: 750,
    price: 2500,
    page: 750
  },
  {
    _id: ObjectId('67d1666e9fd0e214bca26a24'),
    name: 'The Python Programming',
    page: 800,
    price: 1500
  },
  {
    _id: ObjectId('67d1666e9fd0e214bca26a25'),
    name: 'Software Engineering',
    page: 500,
    price: 2000
  },
  {
    _id: ObjectId('67d166779fd0e214bca26a26'),
    name: 'Clean Code',
    page: 464,
    price: 1200
  },
  {
    _id: ObjectId('67d166779fd0e214bca26a27'),
```



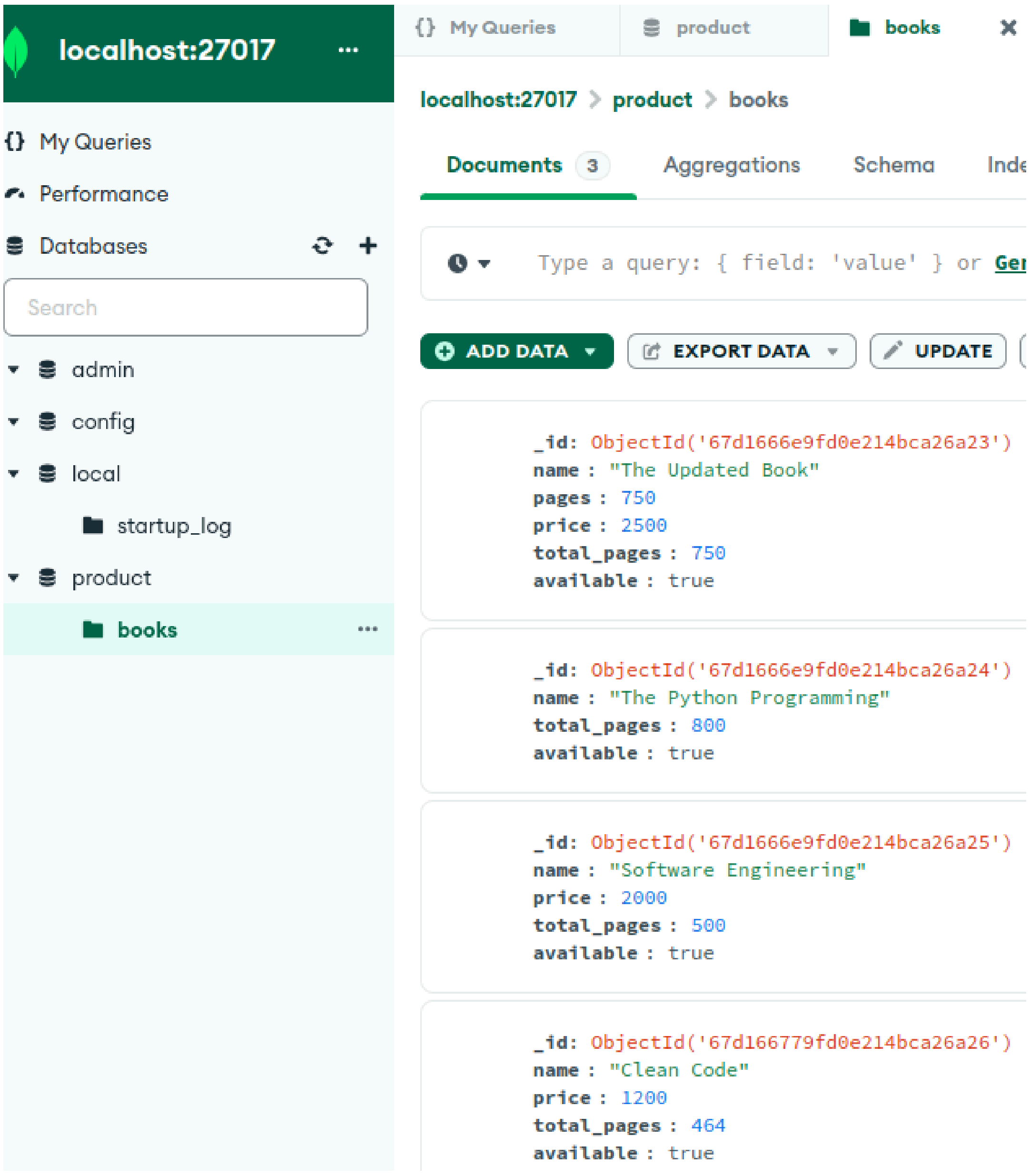
```
_id: ObjectId('67d1666e9fd0e214bca26a23')
name : "The Updated Book"
pages : 750
price : 2500
page : 750
```

Update Operators

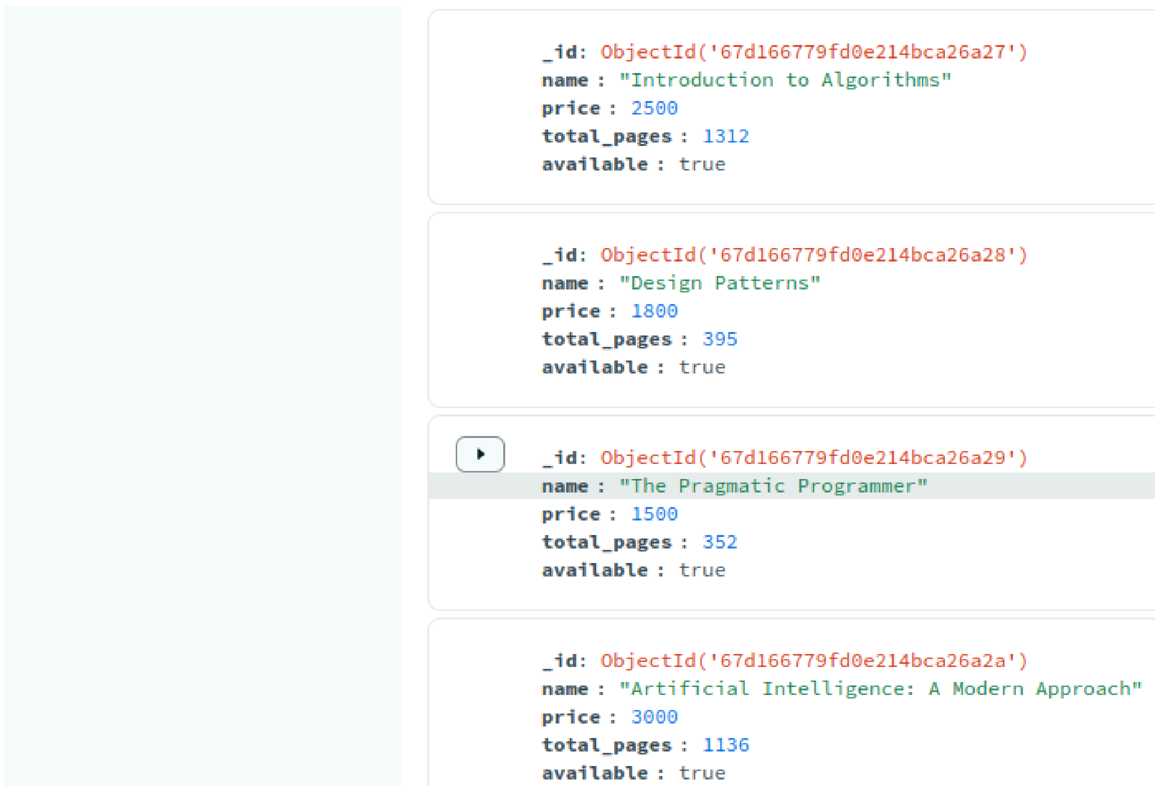
Increase Price ($inc), Rename Field ($rename), Set a New Field ($set), Remove a Field ($unset)



```
product> db.books.updateOne(
...   { name: "The Python Programming" },
...   { $inc: { price: 500 } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
product> db.books.updateMany(
...   {},
...   { $rename: { "page": "total_pages" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 8,
  modifiedCount: 8,
  upsertedCount: 0
}
product> db.books.updateMany(
...   {},
...   { $set: { available: true } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 8,
  modifiedCount: 8,
  upsertedCount: 0
}
product> db.books.updateOne(
...   { name: "The Python Programming" },
...   { $unset: { price: "" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```



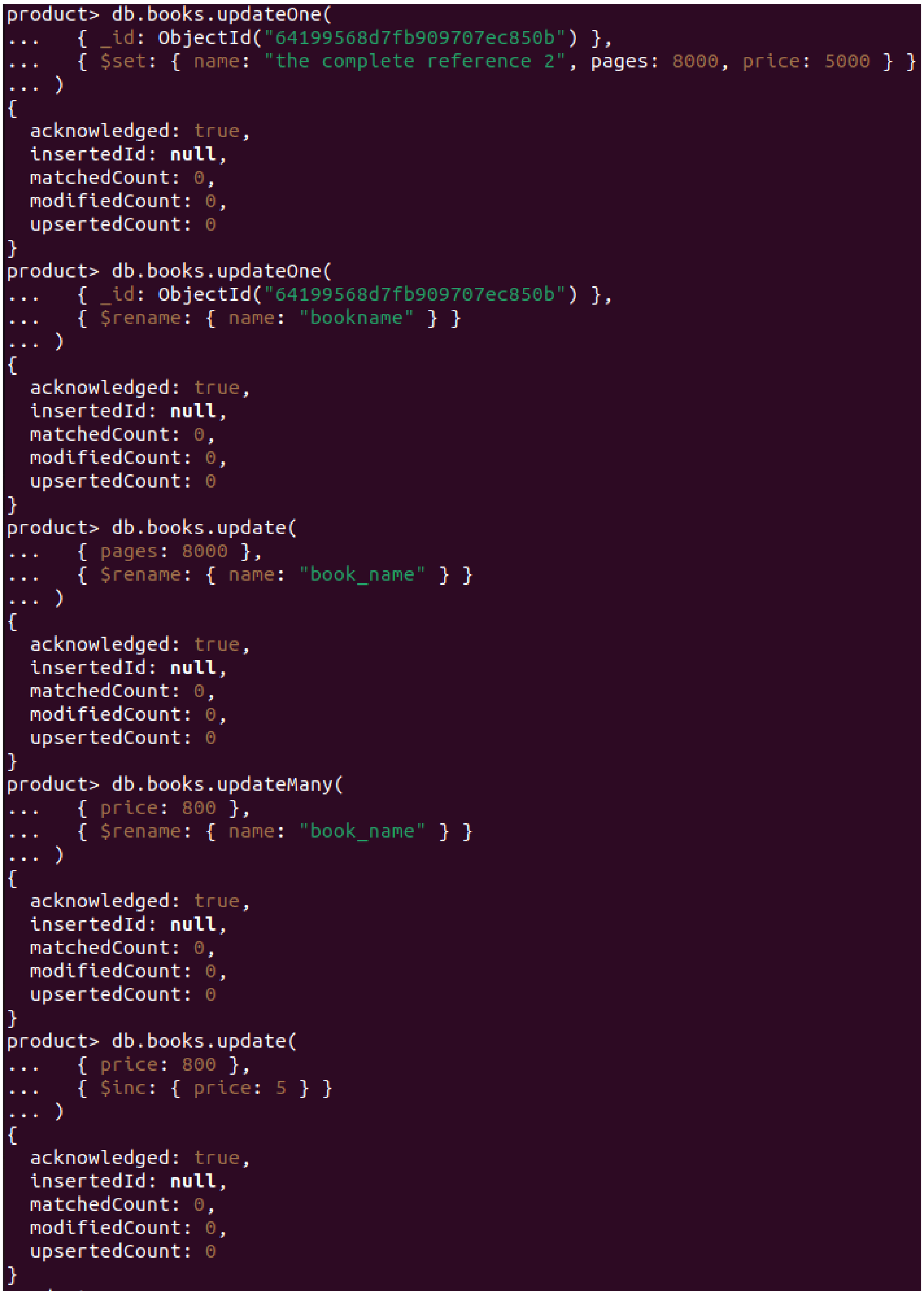
```
_id: ObjectId('67d1666e9fd0e214bca26a23')
name : "The Updated Book"
pages : 750
price : 2500
total_pages : 750
available : true

_id: ObjectId('67d1666e9fd0e214bca26a24')
name : "The Python Programming"
total_pages : 800
available : true

_id: ObjectId('67d1666e9fd0e214bca26a25')
name : "Software Engineering"
price : 2000
total_pages : 500
available : true

_id: ObjectId('67d166779fd0e214bca26a26')
name : "Clean Code"
price : 1200
total_pages : 464
available : true
```



```
_id: ObjectId('67d166779fd0e214bca26a27')
name : "Introduction to Algorithms"
price : 2500
total_pages : 1312
available : true

_id: ObjectId('67d166779fd0e214bca26a28')
name : "Design Patterns"
price : 1800
total_pages : 395
available : true

_id: ObjectId('67d166779fd0e214bca26a29')
name : "The Pragmatic Programmer"
price : 1500
total_pages : 352
available : true

_id: ObjectId('67d166779fd0e214bca26a2a')
name : "Artificial Intelligence: A Modern Approach"
price : 3000
total_pages : 1136
available : true
```

Updating Documents

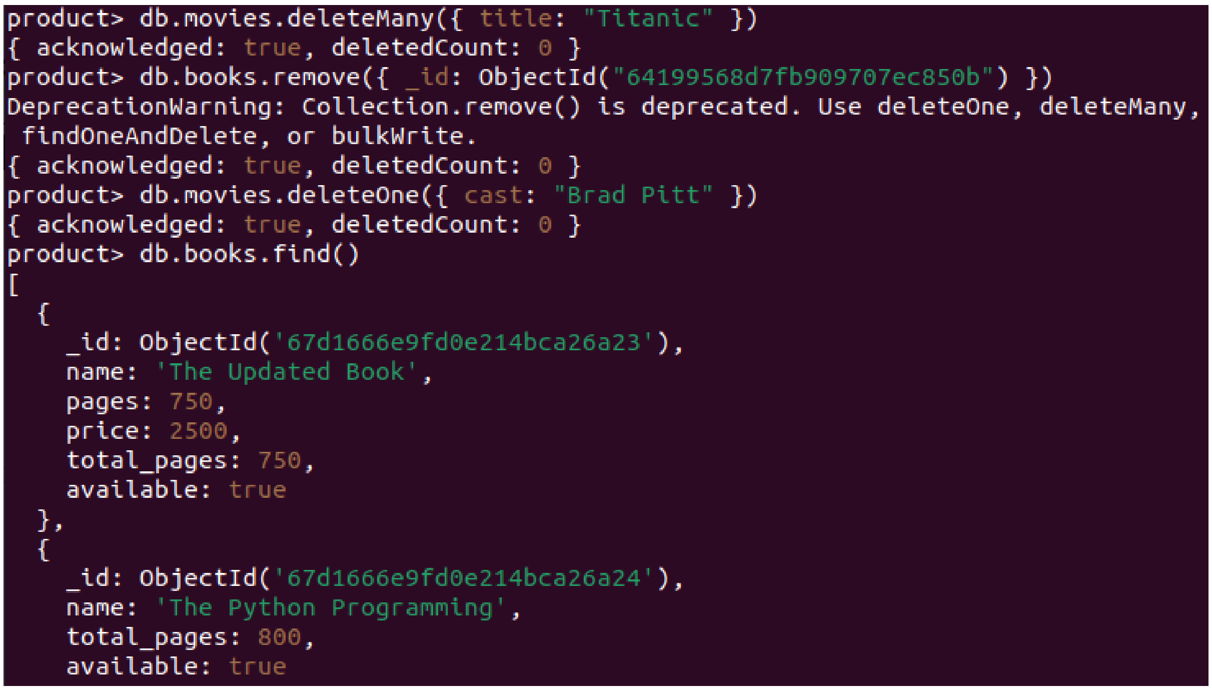
Update the Document with a Specific ID,
Update the Name to bookname for Specific ID,
Update the Name to book_name Where Pages = 8000,,
Update All Documents with Price = 800 to Rename Name to book_name,
Increase Price by 5 Where Price = 800



```
product> db.books.updateOne(
...     { _id: ObjectId("64199568d7fb909707ec850b") },
...     { $set: { name: "the complete reference 2", pages: 8000, price: 5000 } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
product> db.books.updateOne(
...     { _id: ObjectId("64199568d7fb909707ec850b") },
...     { $rename: { name: "bookname" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
product> db.books.update(
...     { pages: 8000 },
...     { $rename: { name: "book_name" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
product> db.books.updateMany(
...     { price: 800 },
...     { $rename: { name: "book_name" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
product> db.books.update(
...     { price: 800 },
...     { $inc: { price: 5 } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
```

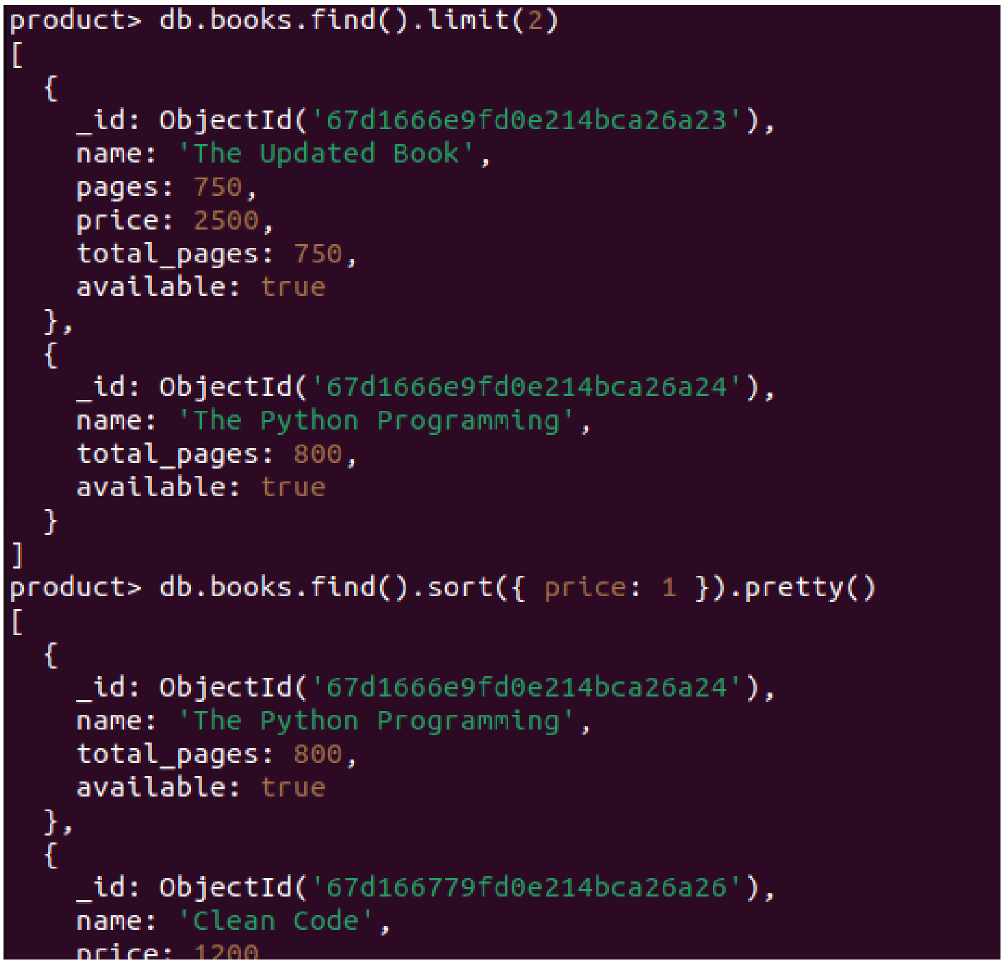
Deleting Documents

Delete Multiple Documents Where Title = Titanic,
Remove Document by ID,
Delete One Document Where Cast = Brad Pitt

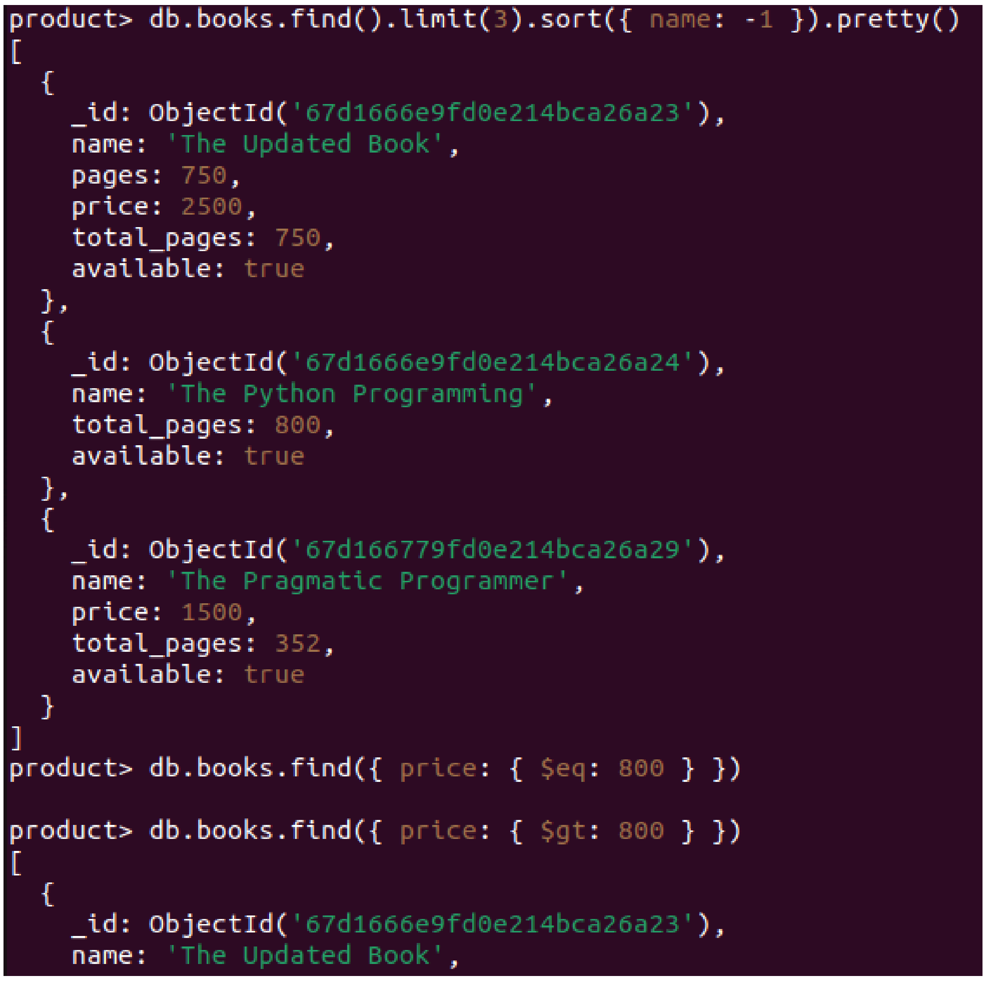


```
product> db.movies.deleteMany({ title: "Titanic" })
{ acknowledged: true, deletedCount: 0 }
product> db.books.remove({ _id: ObjectId("64199568d7fb909707ec850b") })
DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany,
 findOneAndDelete, or bulkWrite.
{ acknowledged: true, deletedCount: 0 }
product> db.movies.deleteOne({ cast: "Brad Pitt" })
{ acknowledged: true, deletedCount: 0 }
product> db.books.find()
[
  {
    _id: ObjectId('67d1666e9fd0e214bca26a23'),
    name: 'The Updated Book',
    pages: 750,
    price: 2500,
    total_pages: 750,
    available: true
  },
  {
    _id: ObjectId('67d1666e9fd0e214bca26a24'),
    name: 'The Python Programming',
    total_pages: 800,
    available: true
```

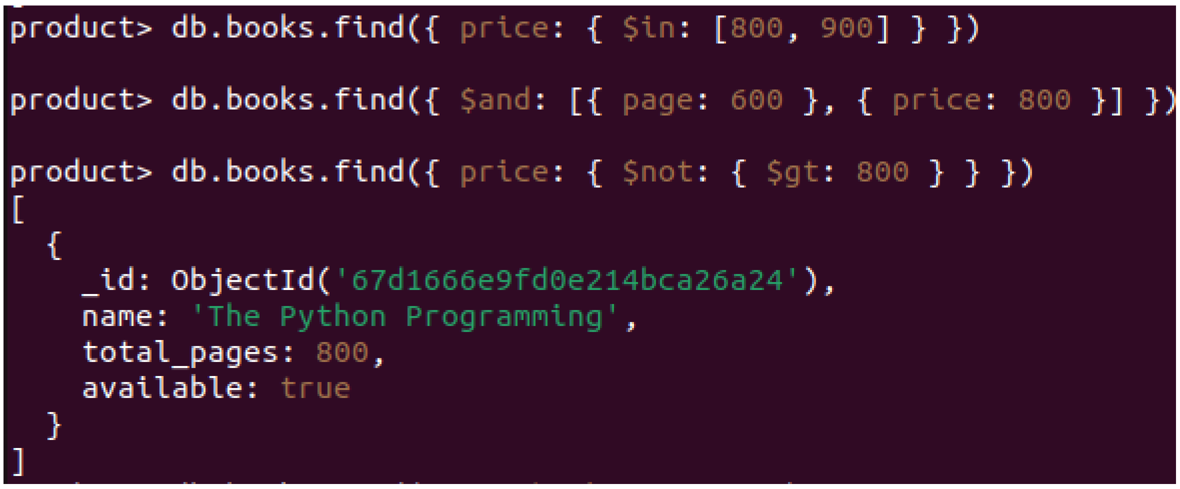
Querying Documents



```
product> db.books.find().limit(2)
[
  {
    _id: ObjectId('67d1666e9fd0e214bca26a23'),
    name: 'The Updated Book',
    pages: 750,
    price: 2500,
    total_pages: 750,
    available: true
  },
  {
    _id: ObjectId('67d1666e9fd0e214bca26a24'),
    name: 'The Python Programming',
    total_pages: 800,
    available: true
  }
]
product> db.books.find().sort({ price: 1 }).pretty()
[
  {
    _id: ObjectId('67d1666e9fd0e214bca26a24'),
    name: 'The Python Programming',
    total_pages: 800,
    available: true
  },
  {
    _id: ObjectId('67d166779fd0e214bca26a26'),
    name: 'Clean Code',
    price: 1200
```



```
product> db.books.find().limit(3).sort({ name: -1 }).pretty()
[
  {
    _id: ObjectId('67d1666e9fd0e214bca26a23'),
    name: 'The Updated Book',
    pages: 750,
    price: 2500,
    total_pages: 750,
    available: true
  },
  {
    _id: ObjectId('67d1666e9fd0e214bca26a24'),
    name: 'The Python Programming',
    total_pages: 800,
    available: true
  },
  {
    _id: ObjectId('67d166779fd0e214bca26a29'),
    name: 'The Pragmatic Programmer',
    price: 1500,
    total_pages: 352,
    available: true
  }
]
product> db.books.find({ price: { $eq: 800 } })

product> db.books.find({ price: { $gt: 800 } })
[
  {
    _id: ObjectId('67d1666e9fd0e214bca26a23'),
    name: 'The Updated Book',
```



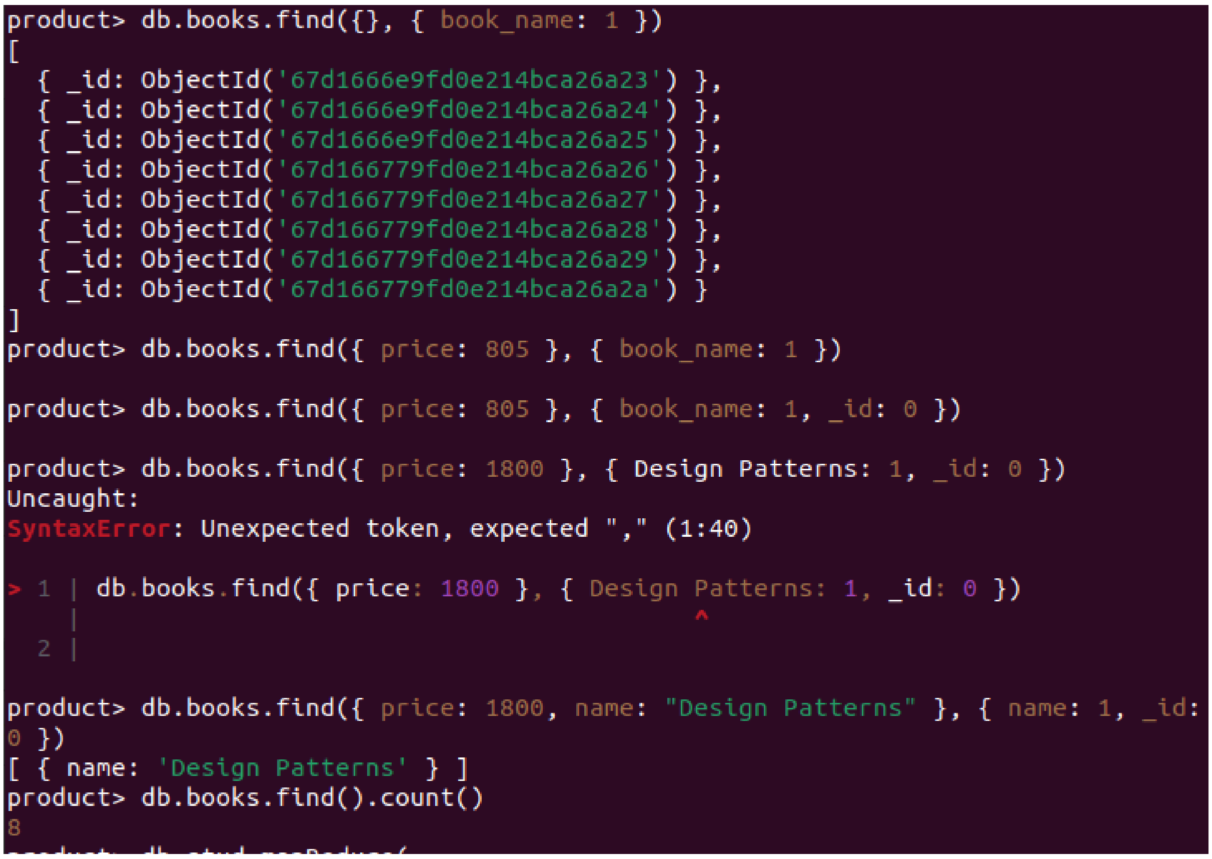
```
product> db.books.find({ price: { $in: [800, 900] } })

product> db.books.find({ $and: [{ page: 600 }, { price: 800 }] })

product> db.books.find({ price: { $not: { $gt: 800 } } })
[
  {
    _id: ObjectId('67d1666e9fd0e214bca26a24'),
    name: 'The Python Programming',
    total_pages: 800,
    available: true
  }
]
```

Projection

Projection is used to project the number of columns or attributes from a collection

Get The total number of collections

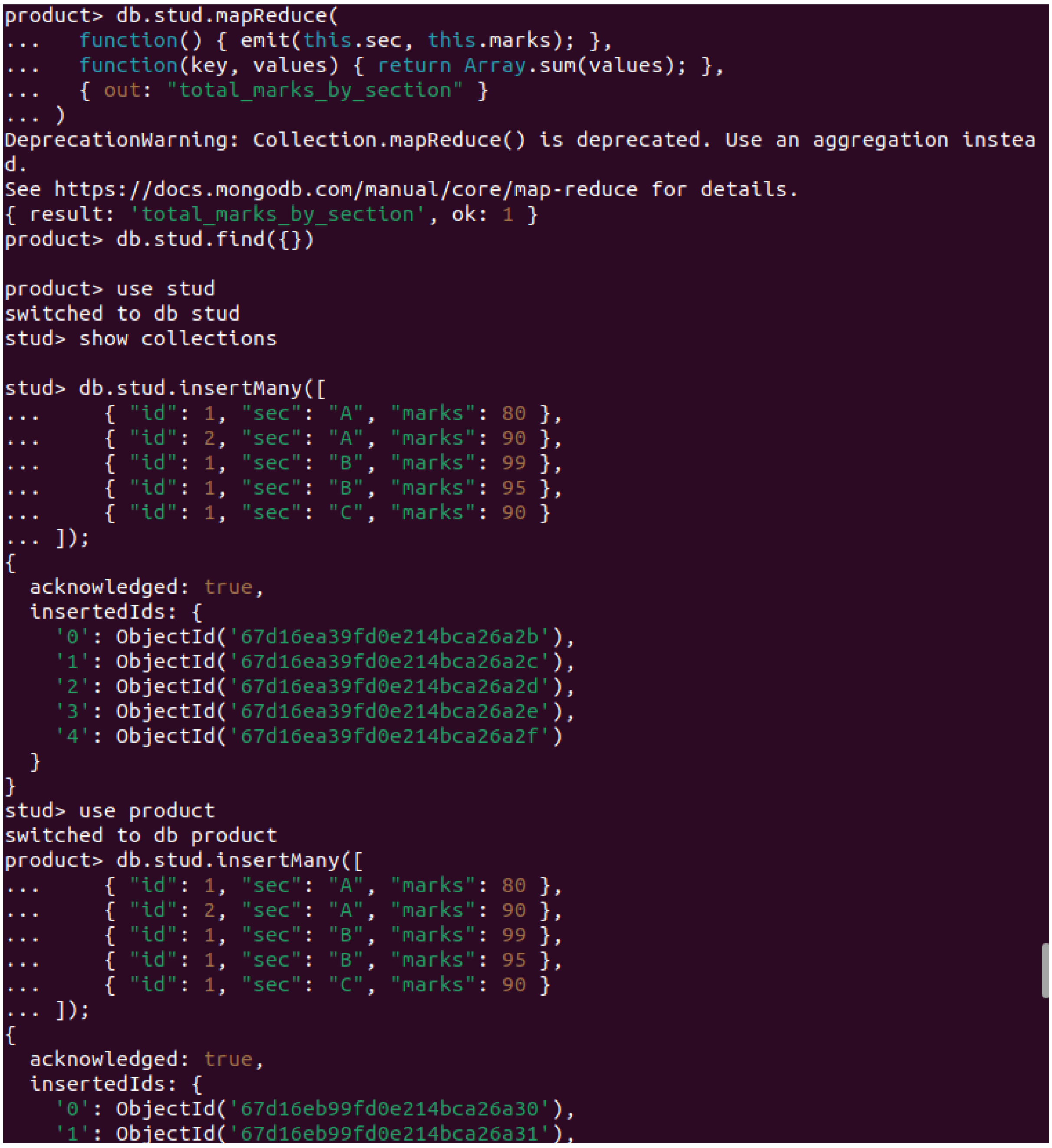
db.books.find().count()



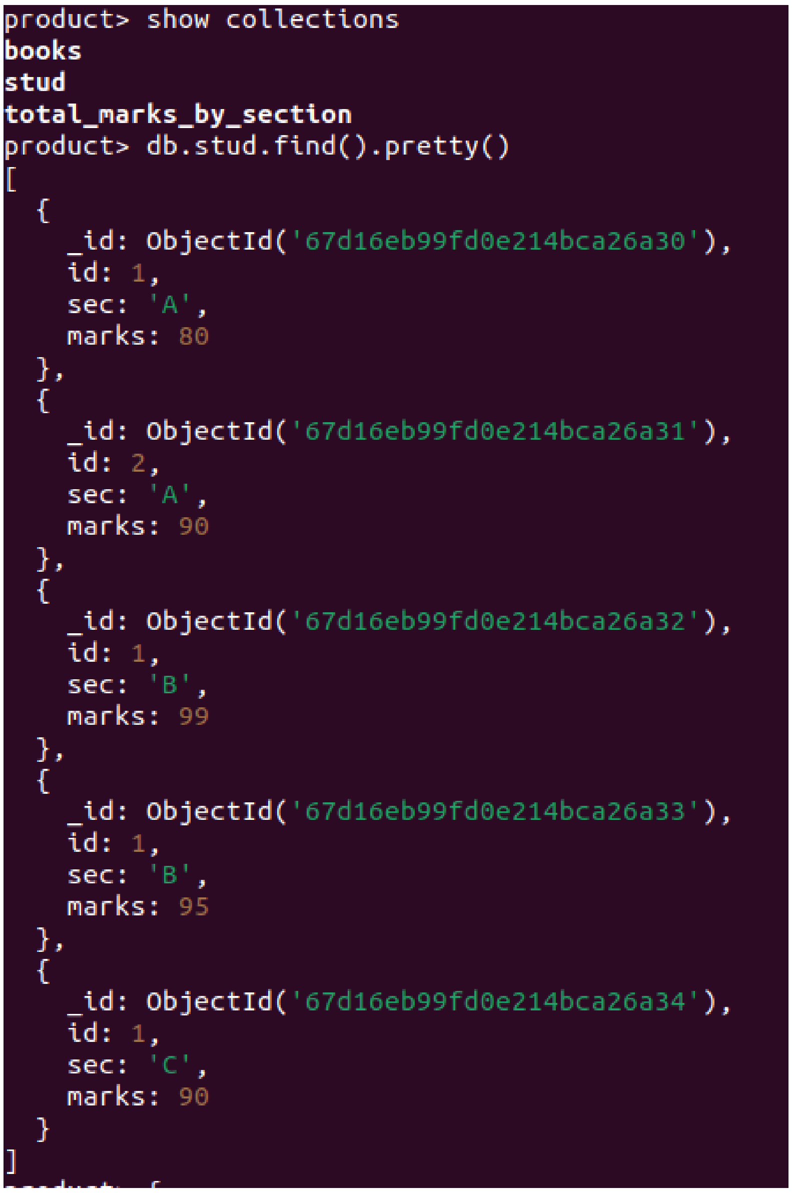
```
product> db.books.find({}, { book_name: 1 })
[
  { _id: ObjectId('67d1666e9fd0e214bca26a23') },
  { _id: ObjectId('67d1666e9fd0e214bca26a24') },
  { _id: ObjectId('67d1666e9fd0e214bca26a25') },
  { _id: ObjectId('67d166779fd0e214bca26a26') },
  { _id: ObjectId('67d166779fd0e214bca26a27') },
  { _id: ObjectId('67d166779fd0e214bca26a28') },
  { _id: ObjectId('67d166779fd0e214bca26a29') },
  { _id: ObjectId('67d166779fd0e214bca26a2a') }
]
product> db.books.find({ price: 805 }, { book_name: 1 })

product> db.books.find({ price: 805 }, { book_name: 1, _id: 0 })

product> db.books.find({ price: 1800 }, { Design Patterns: 1, _id: 0 })
Uncaught:
SyntaxError: Unexpected token, expected "," (1:40)

> 1 | db.books.find({ price: 1800 }, { Design Patterns: 1, _id: 0 })
    |                                        ^
  2 |

product> db.books.find({ price: 1800, name: "Design Patterns" }, { name: 1, _id:
0 })
[ { name: 'Design Patterns' } ]
product> db.books.find().count()
8
```

MongoDb with mapreduce

Consider a stud collection with following data

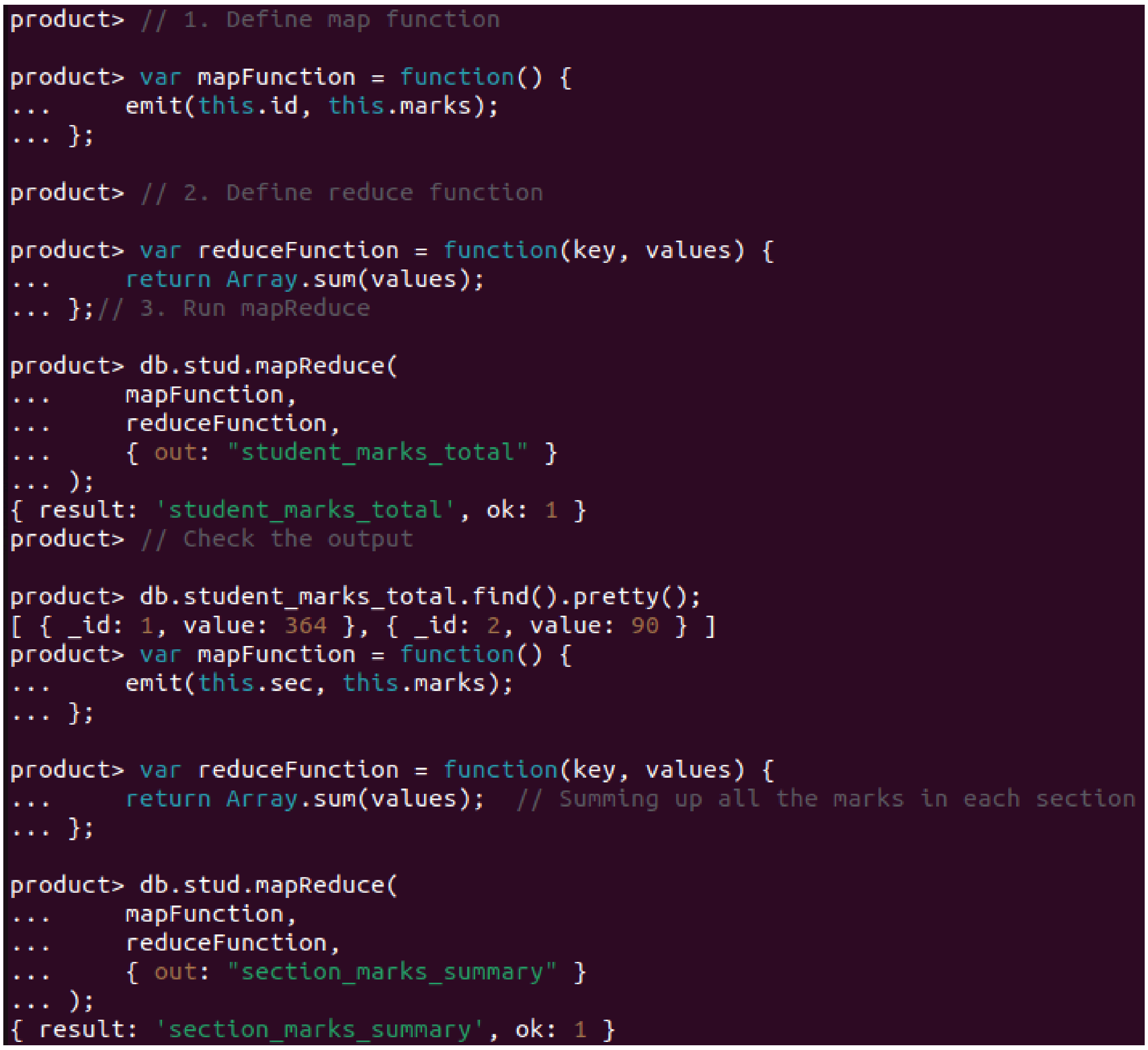


```
product> db.stud.mapReduce(
...   function() { emit(this.sec, this.marks); },
...   function(key, values) { return Array.sum(values); },
...   { out: "total_marks_by_section" }
... )
DeprecationWarning: Collection.mapReduce() is deprecated. Use an aggregation instea
d.
See https://docs.mongodb.com/manual/core/map-reduce for details.
{ result: 'total_marks_by_section', ok: 1 }
product> db.stud.find({})

product> use stud
switched to db stud
stud> show collections

stud> db.stud.insertMany([
...       { "id": 1, "sec": "A", "marks": 80 },
...       { "id": 2, "sec": "A", "marks": 90 },
...       { "id": 1, "sec": "B", "marks": 99 },
...       { "id": 1, "sec": "B", "marks": 95 },
...       { "id": 1, "sec": "C", "marks": 90 }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67d16ea39fd0e214bca26a2b'),
    '1': ObjectId('67d16ea39fd0e214bca26a2c'),
    '2': ObjectId('67d16ea39fd0e214bca26a2d'),
    '3': ObjectId('67d16ea39fd0e214bca26a2e'),
    '4': ObjectId('67d16ea39fd0e214bca26a2f')
  }
}
stud> use product
switched to db product
product> db.stud.insertMany([
...       { "id": 1, "sec": "A", "marks": 80 },
...       { "id": 2, "sec": "A", "marks": 90 },
...       { "id": 1, "sec": "B", "marks": 99 },
...       { "id": 1, "sec": "B", "marks": 95 },
...       { "id": 1, "sec": "C", "marks": 90 }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67d16eb99fd0e214bca26a30'),
    '1': ObjectId('67d16eb99fd0e214bca26a31'),
```

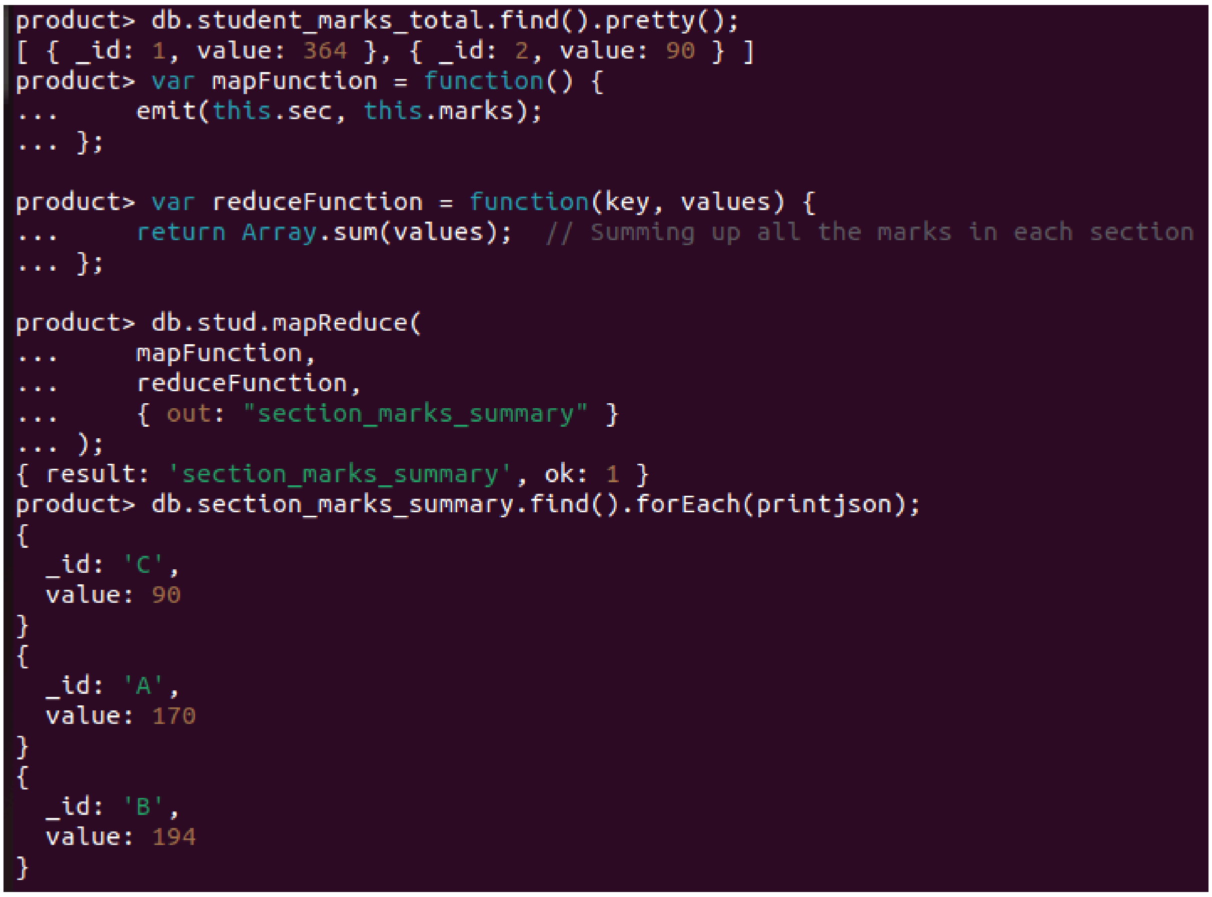


```
product> show collections
books
stud
total_marks_by_section
product> db.stud.find().pretty()
[
  {
    _id: ObjectId('67d16eb99fd0e214bca26a30'),
    id: 1,
    sec: 'A',
    marks: 80
  },
  {
    _id: ObjectId('67d16eb99fd0e214bca26a31'),
    id: 2,
    sec: 'A',
    marks: 90
  },
  {
    _id: ObjectId('67d16eb99fd0e214bca26a32'),
    id: 1,
    sec: 'B',
    marks: 99
  },
  {
    _id: ObjectId('67d16eb99fd0e214bca26a33'),
    id: 1,
    sec: 'B',
    marks: 95
  },
  {
    _id: ObjectId('67d16eb99fd0e214bca26a34'),
    id: 1,
    sec: 'C',
    marks: 90
  }
]
```

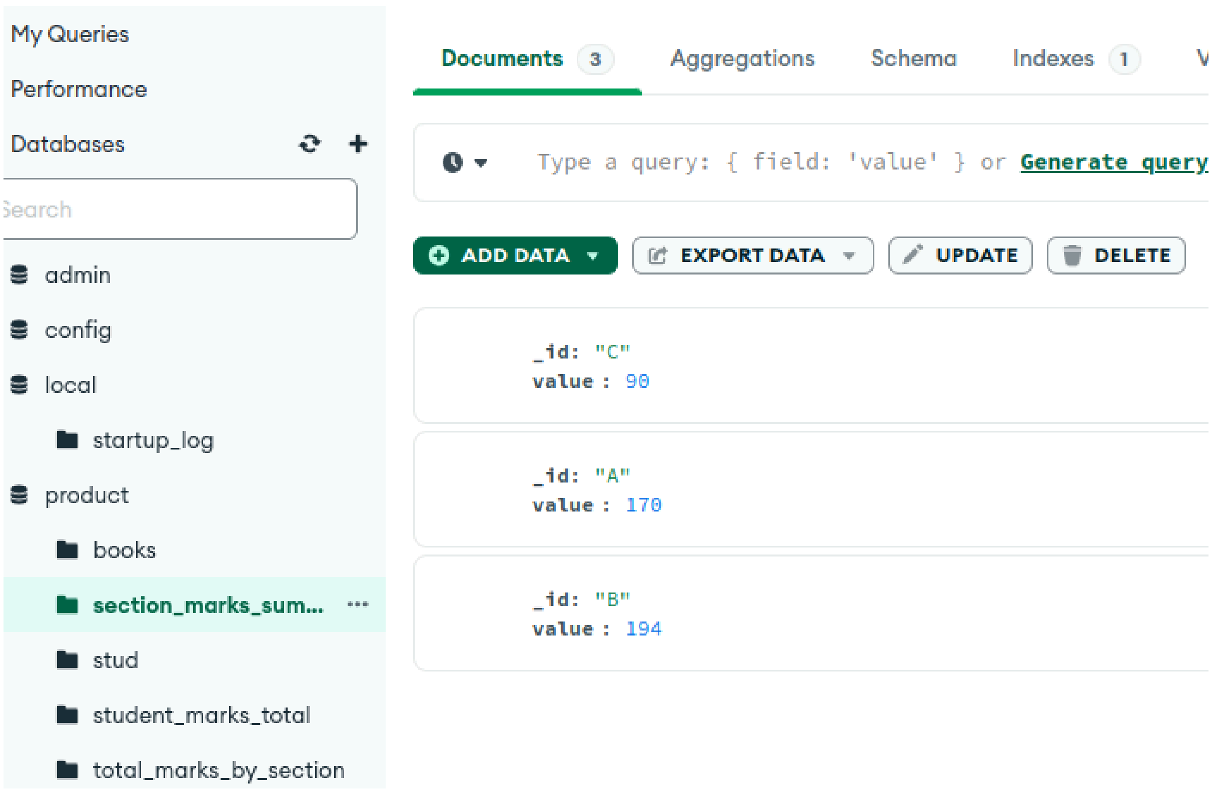
1. Define the map function
2. Define the reduce function
3. Use the mapReduce function by passing mapper function, reducer function and the output collection name



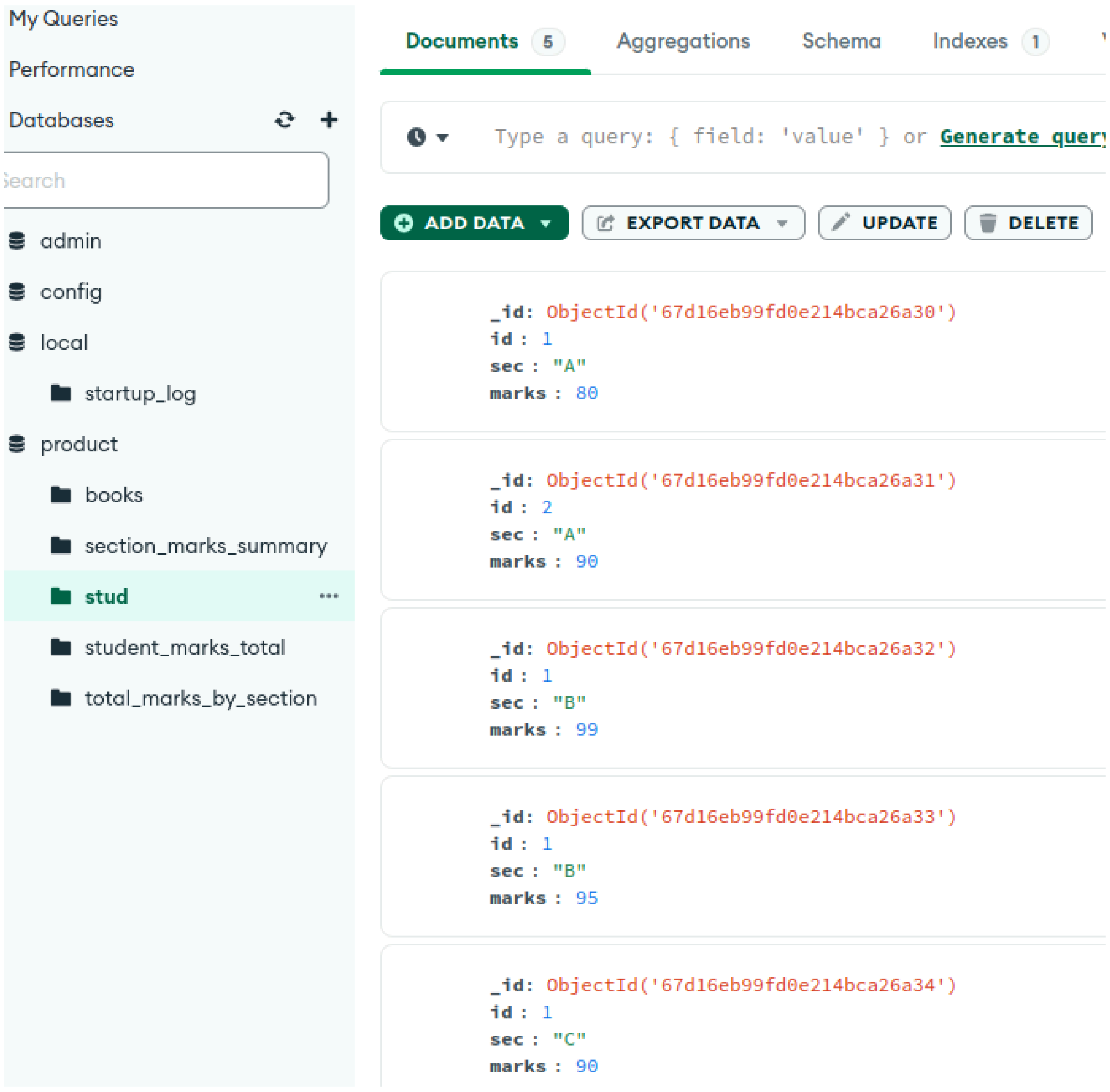
```
product> // 1. Define map function

product> var mapFunction = function() {
...     emit(this.id, this.marks);
... };

product> // 2. Define reduce function

product> var reduceFunction = function(key, values) {
...     return Array.sum(values);
... };// 3. Run mapReduce

product> db.stud.mapReduce(
...     mapFunction,
...     reduceFunction,
...     { out: "student_marks_total" }
... );
{ result: 'student_marks_total', ok: 1 }
product> // Check the output

product> db.student_marks_total.find().pretty();
[ { _id: 1, value: 364 }, { _id: 2, value: 90 } ]
product> var mapFunction = function() {
...     emit(this.sec, this.marks);
... };

product> var reduceFunction = function(key, values) {
...     return Array.sum(values);  // Summing up all the marks in each section
... };

product> db.stud.mapReduce(
...     mapFunction,
...     reduceFunction,
...     { out: "section_marks_summary" }
... );
{ result: 'section_marks_summary', ok: 1 }
```



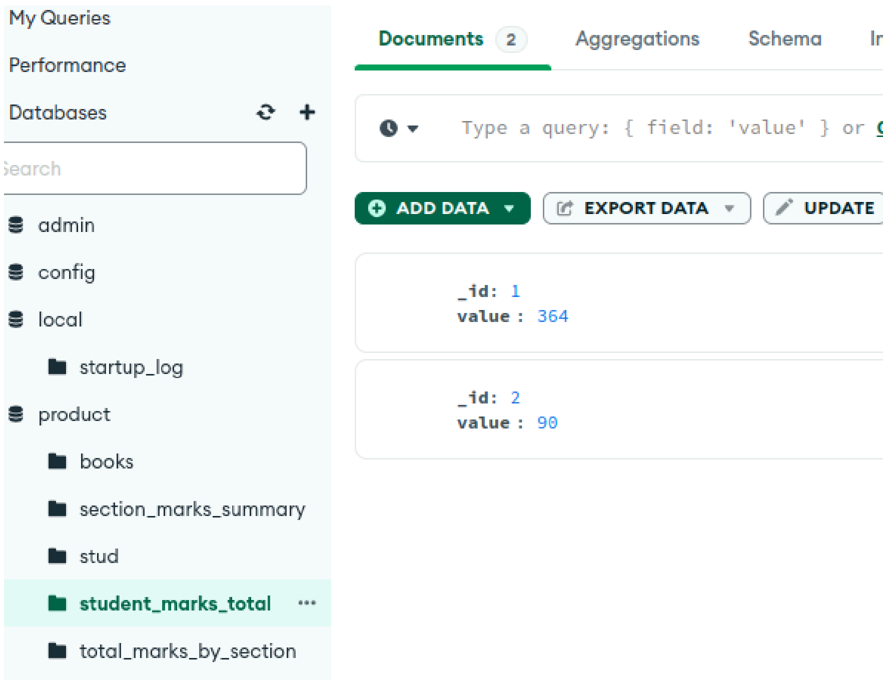
```
product> db.student_marks_total.find().pretty();
[ { _id: 1, value: 364 }, { _id: 2, value: 90 } ]
product> var mapFunction = function() {
...     emit(this.sec, this.marks);
... };

product> var reduceFunction = function(key, values) {
...     return Array.sum(values);  // Summing up all the marks in each section
... };

product> db.stud.mapReduce(
...     mapFunction,
...     reduceFunction,
...     { out: "section_marks_summary" }
... );
{ result: 'section_marks_summary', ok: 1 }
product> db.section_marks_summary.find().forEach(printjson);
{
  _id: 'C',
  value: 90
}
{
  _id: 'A',
  value: 170
}
{
  _id: 'B',
  value: 194
}
```



```
_id: "C"
value : 90

_id: "A"
value : 170

_id: "B"
value : 194
```



```
_id: ObjectId('67d16eb99fd0e214bca26a30')
id : 1
sec : "A"
marks : 80

_id: ObjectId('67d16eb99fd0e214bca26a31')
id : 2
sec : "A"
marks : 90

_id: ObjectId('67d16eb99fd0e214bca26a32')
id : 1
sec : "B"
marks : 99

_id: ObjectId('67d16eb99fd0e214bca26a33')
id : 1
sec : "B"
marks : 95

_id: ObjectId('67d16eb99fd0e214bca26a34')
id : 1
sec : "C"
marks : 90
```

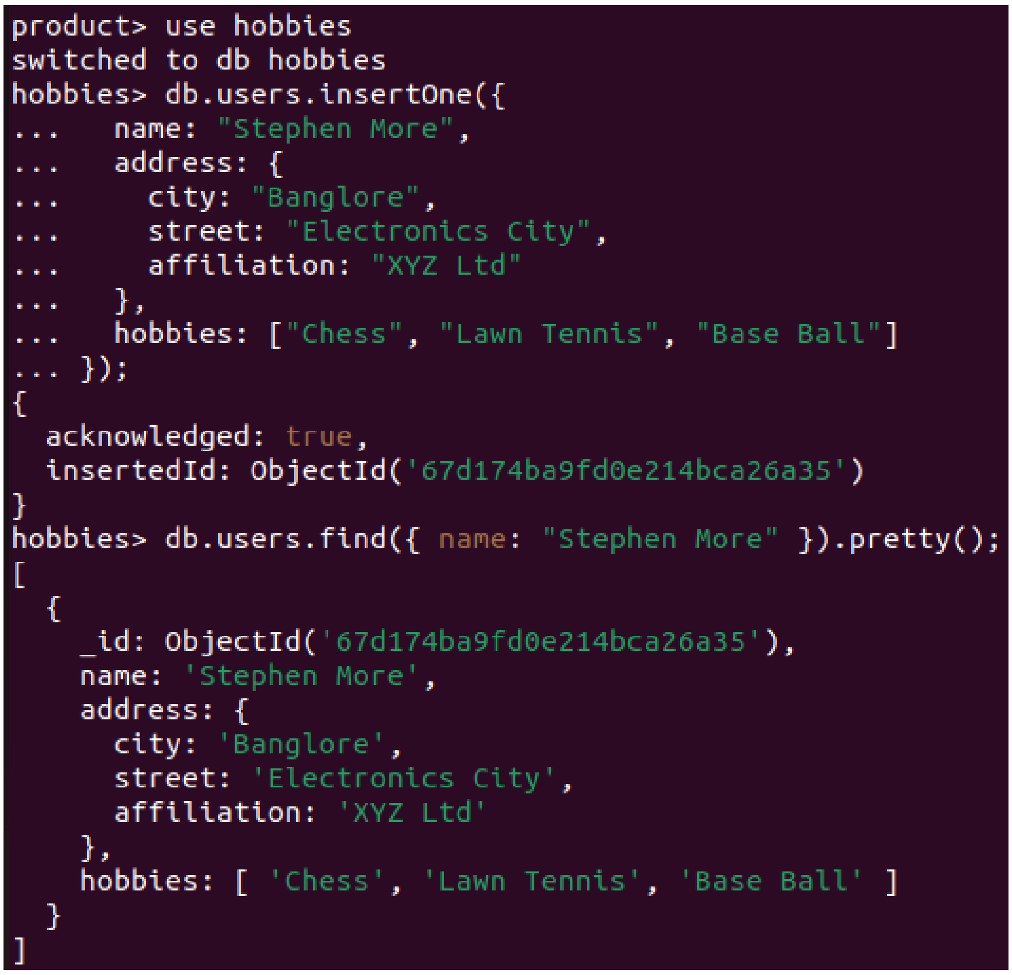


```
Type a query: { field: 'value' } or

_id: 1
value : 364

_id: 2
value : 90
```

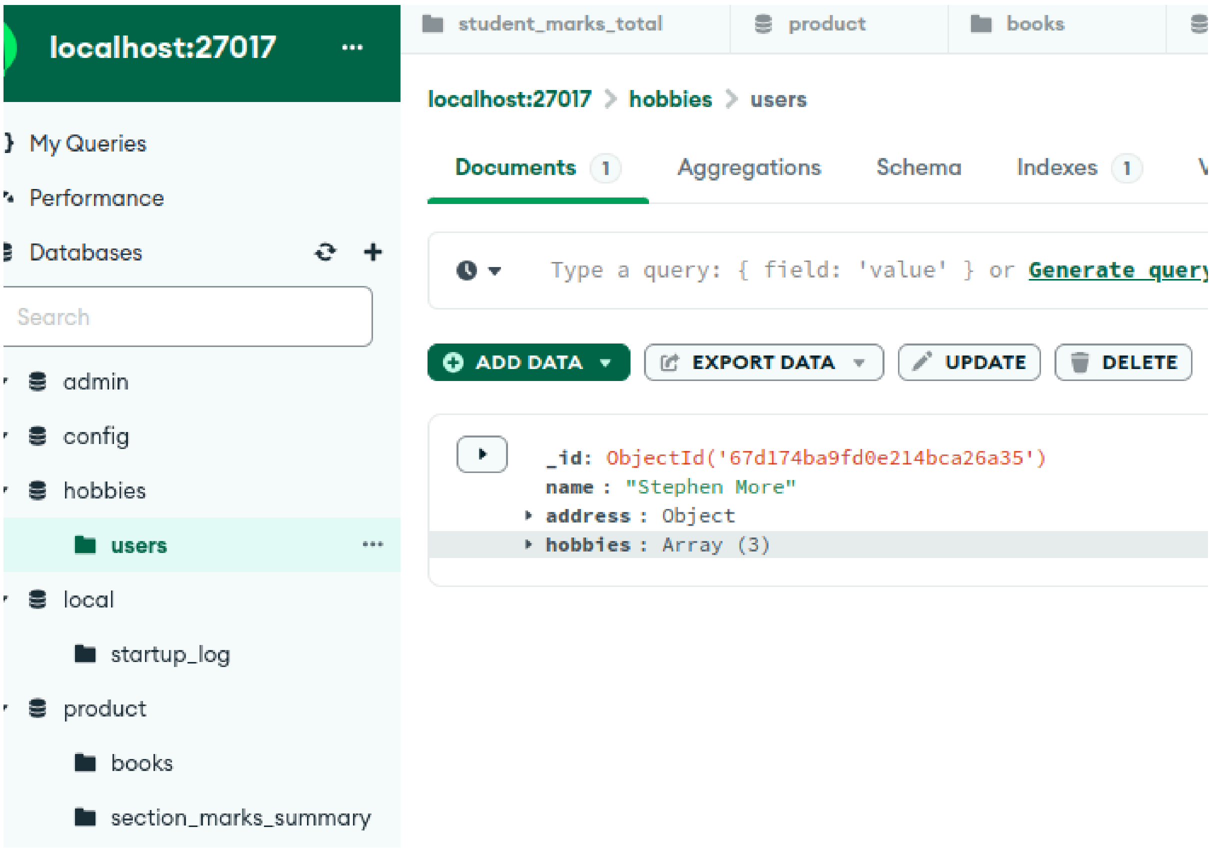
EXERCISE:

1. Write the insert method to store the following document in MongoDB and verify. May practice crud operations.

Name : "Stephen More"
Address : {
"City" : "Banglore",
"Street" : "Electronics City",
"Affiliation" : "XYZ Ltd"
}
Hobbies : Chess, Lawn Tennis, Base Ball

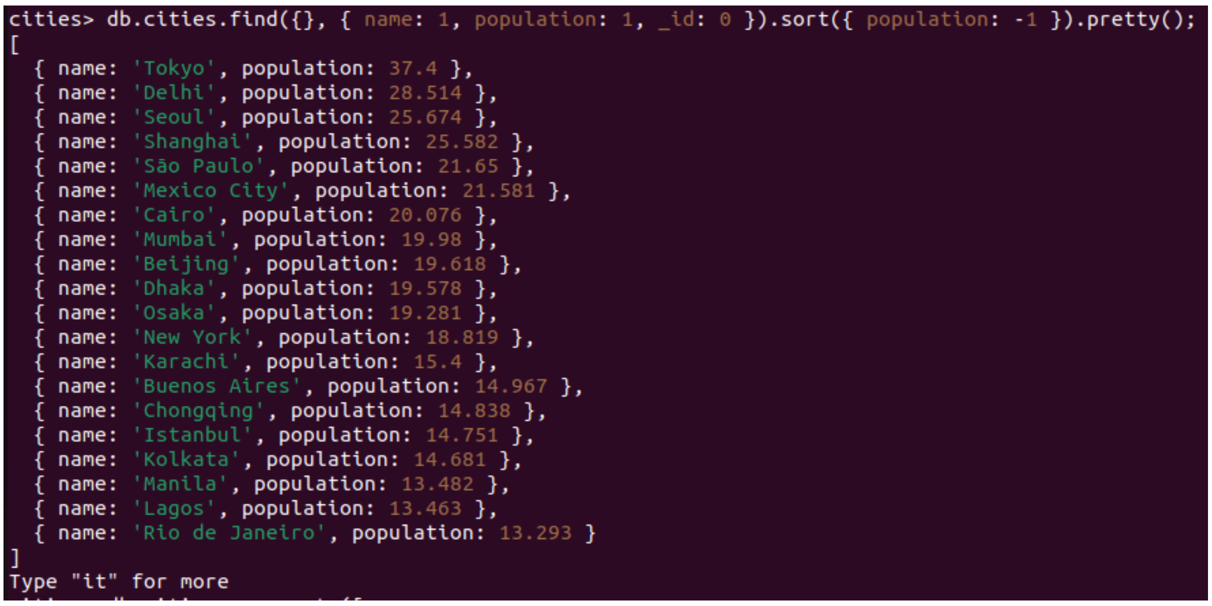


```
product> use hobbies
switched to db hobbies
hobbies> db.users.insertOne({
...   name: "Stephen More",
...   address: {
...     city: "Banglore",
...     street: "Electronics City",
...     affiliation: "XYZ Ltd"
...   },
...   hobbies: ["Chess", "Lawn Tennis", "Base Ball"]
... });
{
  acknowledged: true,
  insertedId: ObjectId('67d174ba9fd0e214bca26a35')
}
hobbies> db.users.find({ name: "Stephen More" }).pretty();
[
  {
    _id: ObjectId('67d174ba9fd0e214bca26a35'),
    name: 'Stephen More',
    address: {
      city: 'Banglore',
      street: 'Electronics City',
      affiliation: 'XYZ Ltd'
    },
    hobbies: [ 'Chess', 'Lawn Tennis', 'Base Ball' ]
  }
]
```



```
_id: ObjectId('67d174ba9fd0e214bca26a35')
name : "Stephen More"
address : Object
hobbies : Array (3)
```

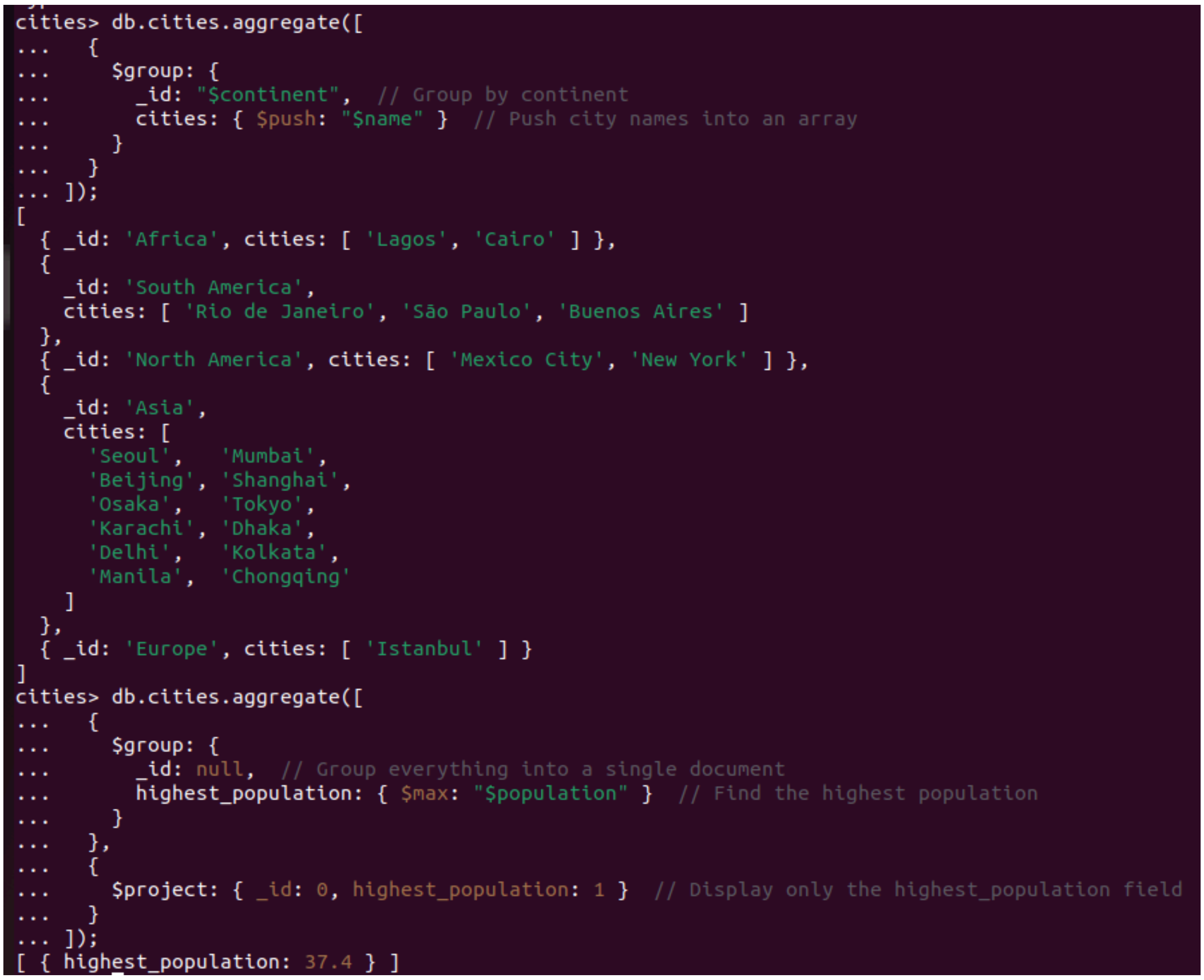
2. Create a collection 'cities' as follows



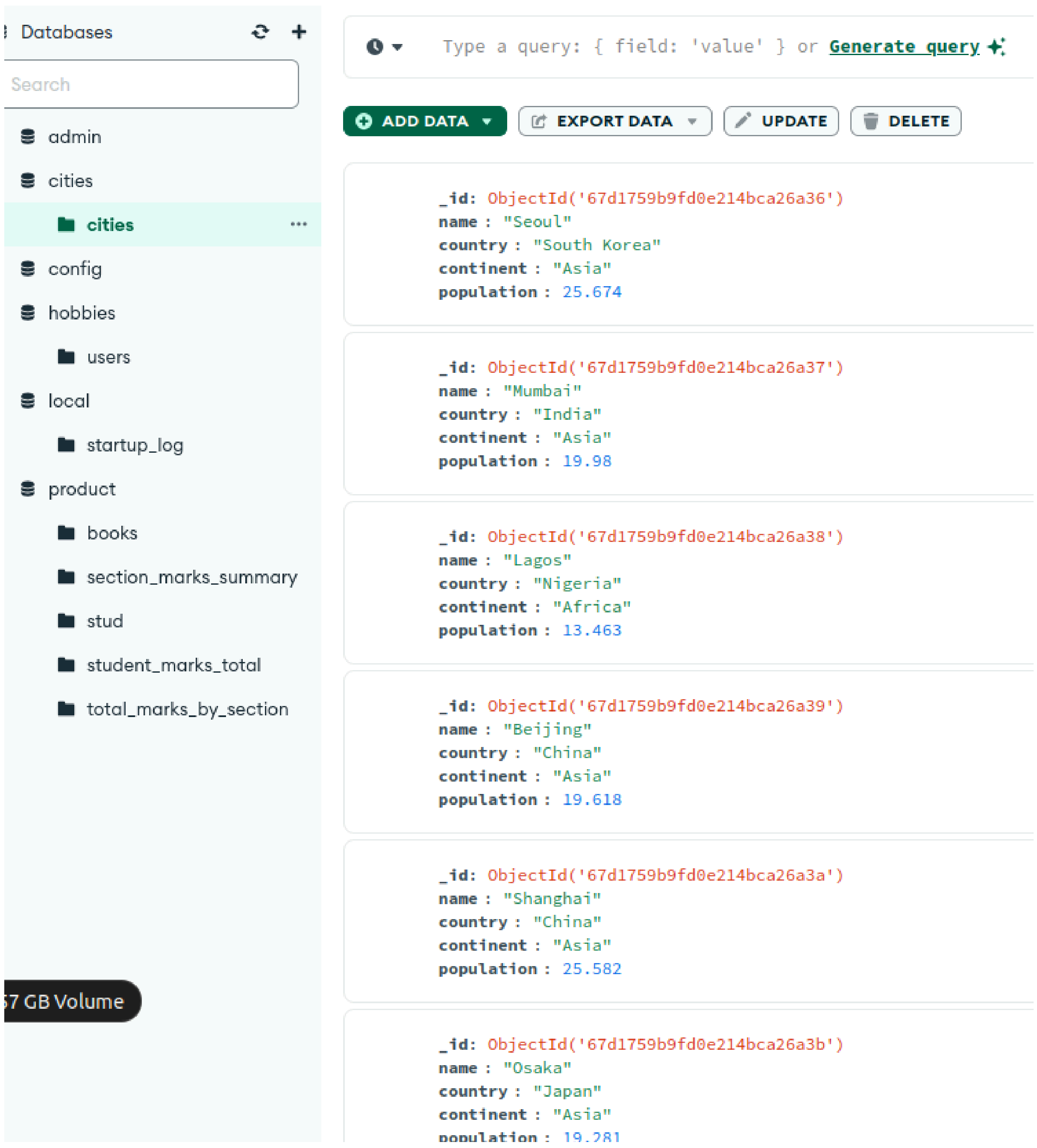
```
cities> db.cities.find({}, { name: 1, population: 1, _id: 0 }).sort({ population: -1 }).pretty();
[
  { name: 'Tokyo', population: 37.4 },
  { name: 'Delhi', population: 28.514 },
  { name: 'Seoul', population: 25.674 },
  { name: 'Shanghai', population: 25.582 },
  { name: 'São Paulo', population: 21.65 },
  { name: 'Mexico City', population: 21.581 },
  { name: 'Cairo', population: 20.076 },
  { name: 'Mumbai', population: 19.98 },
  { name: 'Beijing', population: 19.618 },
  { name: 'Dhaka', population: 19.578 },
  { name: 'Osaka', population: 19.281 },
  { name: 'New York', population: 18.819 },
  { name: 'Karachi', population: 15.4 },
  { name: 'Buenos Aires', population: 14.967 },
  { name: 'Chongqing', population: 14.838 },
  { name: 'Istanbul', population: 14.751 },
  { name: 'Kolkata', population: 14.681 },
  { name: 'Manila', population: 13.482 },
  { name: 'Lagos', population: 13.463 },
  { name: 'Rio de Janeiro', population: 13.293 }
]
Type "it" for more
```

● Write a Mongodb query to find all the cities of North America.
● Write a mongodb query to retrieve the cities names in descending order of their population.
● Write a mongodb query to display all the cities grouped by their continent.

● Write a mongodb query to add the field “highest population”.



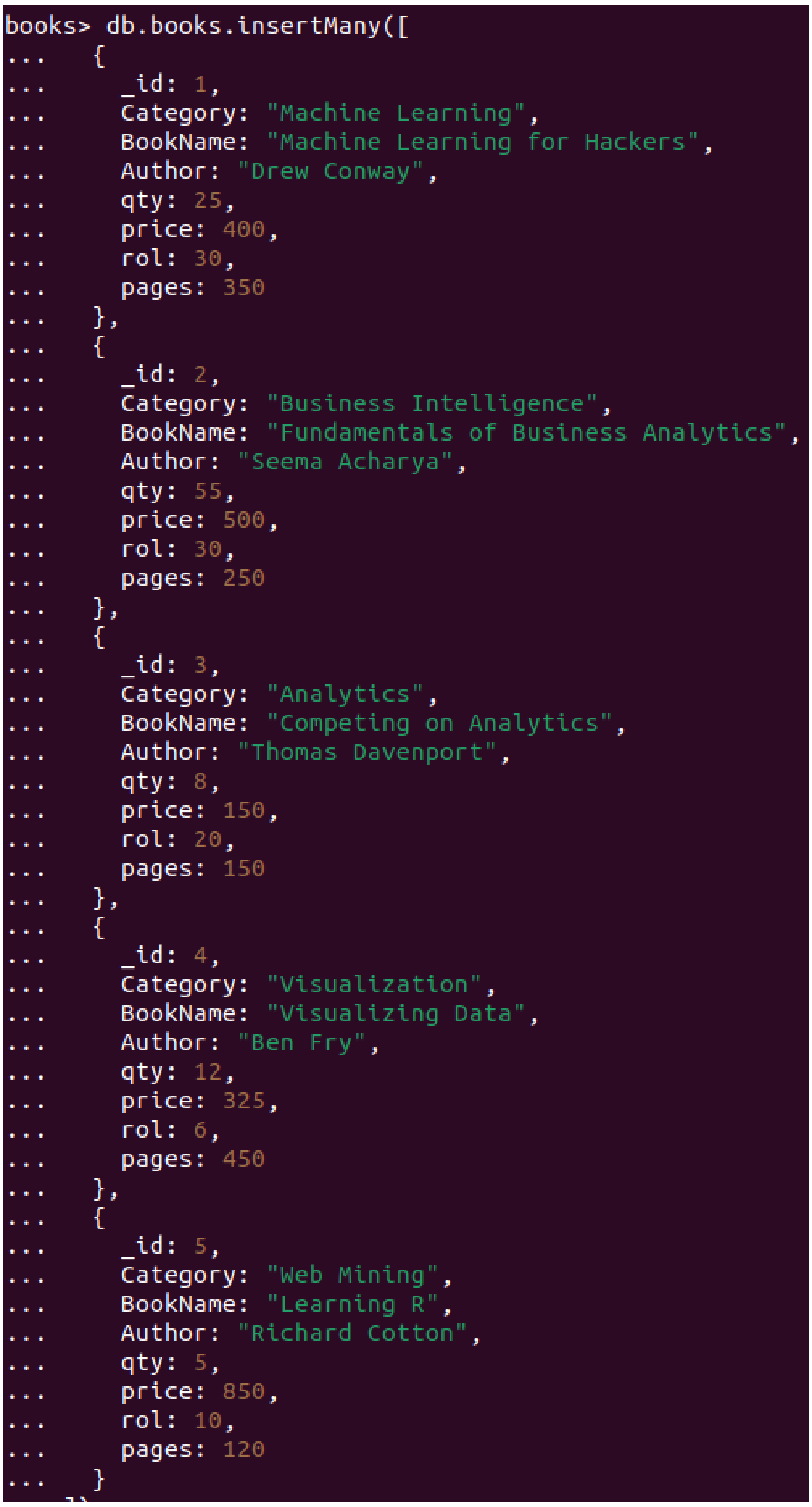
```
cities> db.cities.aggregate([
...   {
...     $group: {
...       _id: "$continent",   // Group by continent
...       cities: { $push: "$name" }  // Push city names into an array
...     }
...   }
... ]);
[
  { _id: 'Africa', cities: [ 'Lagos', 'Cairo' ] },
  {
    _id: 'South America',
    cities: [ 'Rio de Janeiro', 'São Paulo', 'Buenos Aires' ]
  },
  { _id: 'North America', cities: [ 'Mexico City', 'New York' ] },
  {
    _id: 'Asia',
    cities: [
      'Seoul',    'Mumbai',
      'Beijing', 'Shanghai',
      'Osaka',    'Tokyo',
      'Karachi', 'Dhaka',
      'Delhi',    'Kolkata',
      'Manila',   'Chongqing'
    ]
  },
  { _id: 'Europe', cities: [ 'Istanbul' ] }
]
cities> db.cities.aggregate([
...   {
...     $group: {
...       _id: null,  // Group everything into a single document
...       highest_population: { $max: "$population" }  // Find the highest population
...     }
...   },
...   {
...     $project: { _id: 0, highest_population: 1 }  // Display only the highest_population field
...   }
... ]);
[ { highest_population: 37.4 } ]
```



```
_id: ObjectId('67d1759b9fd0e214bca26a36')
name : "Seoul"
country : "South Korea"
continent : "Asia"
population : 25.674

_id: ObjectId('67d1759b9fd0e214bca26a37')
name : "Mumbai"
country : "India"
continent : "Asia"
population : 19.98

_id: ObjectId('67d1759b9fd0e214bca26a38')
name : "Lagos"
country : "Nigeria"
continent : "Africa"
population : 13.463

_id: ObjectId('67d1759b9fd0e214bca26a39')
name : "Beijing"
country : "China"
continent : "Asia"
population : 19.618

_id: ObjectId('67d1759b9fd0e214bca26a3a')
name : "Shanghai"
country : "China"
continent : "Asia"
population : 25.582

_id: ObjectId('67d1759b9fd0e214bca26a3b')
name : "Osaka"
country : "Japan"
continent : "Asia"
population : 19.281
```

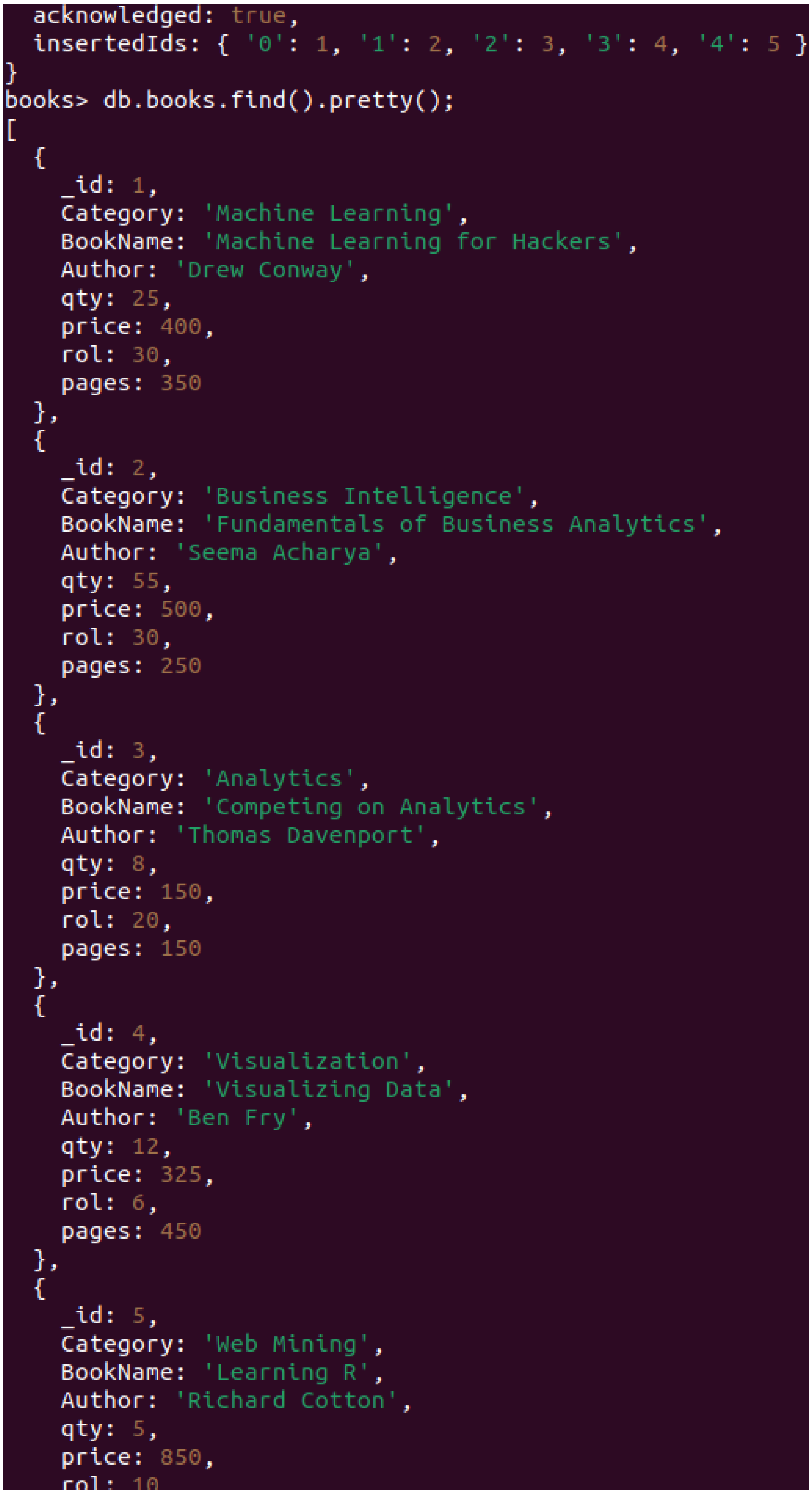
3. To practice MapReduce programming in MongoDB.

Step. 3.1: Insert 5 documents as shown below in the collection named 'books'.



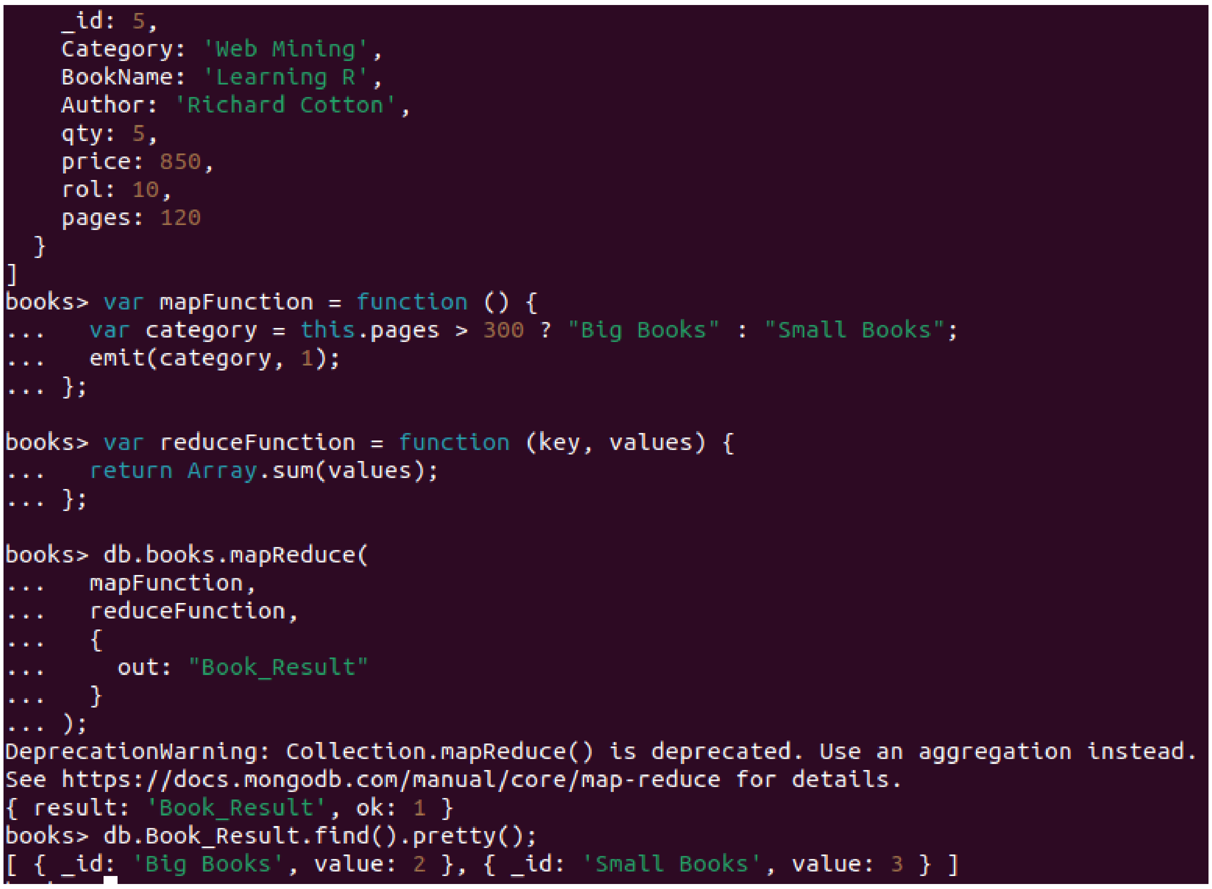
```
books> db.books.insertMany([
...   {
...     _id: 1,
...     Category: "Machine Learning",
...     BookName: "Machine Learning for Hackers",
...     Author: "Drew Conway",
...     qty: 25,
...     price: 400,
...     rol: 30,
...     pages: 350
...   },
...   {
...     _id: 2,
...     Category: "Business Intelligence",
...     BookName: "Fundamentals of Business Analytics",
...     Author: "Seema Acharya",
...     qty: 55,
...     price: 500,
...     rol: 30,
...     pages: 250
...   },
...   {
...     _id: 3,
...     Category: "Analytics",
...     BookName: "Competing on Analytics",
...     Author: "Thomas Davenport",
...     qty: 8,
...     price: 150,
...     rol: 20,
...     pages: 150
...   },
...   {
...     _id: 4,
...     Category: "Visualization",
...     BookName: "Visualizing Data",
...     Author: "Ben Fry",
...     qty: 12,
...     price: 325,
...     rol: 6,
...     pages: 450
...   },
...   {
...     _id: 5,
...     Category: "Web Mining",
...     BookName: "Learning R",
...     Author: "Richard Cotton",
...     qty: 5,
...     price: 850,
...     rol: 10,
...     pages: 120
...   }
```

Step. 3.2: Confirm the presence of the above documents in the “books” collection.



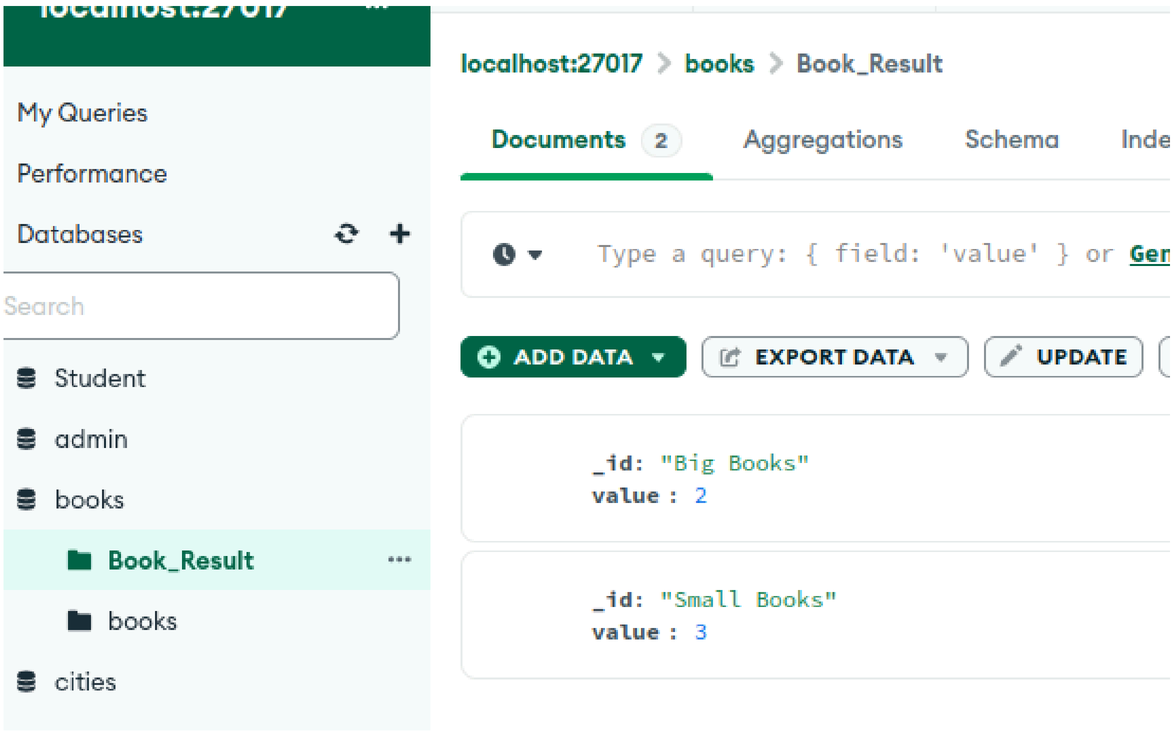
```
  acknowledged: true,
  insertedIds: { '0': 1, '1': 2, '2': 3, '3': 4, '4': 5 }
}
books> db.books.find().pretty();
[
  {
    _id: 1,
    Category: 'Machine Learning',
    BookName: 'Machine Learning for Hackers',
    Author: 'Drew Conway',
    qty: 25,
    price: 400,
    rol: 30,
    pages: 350
  },
  {
    _id: 2,
    Category: 'Business Intelligence',
    BookName: 'Fundamentals of Business Analytics',
    Author: 'Seema Acharya',
    qty: 55,
    price: 500,
    rol: 30,
    pages: 250
  },
  {
    _id: 3,
    Category: 'Analytics',
    BookName: 'Competing on Analytics',
    Author: 'Thomas Davenport',
    qty: 8,
    price: 150,
    rol: 20,
    pages: 150
  },
  {
    _id: 4,
    Category: 'Visualization',
    BookName: 'Visualizing Data',
    Author: 'Ben Fry',
    qty: 12,
    price: 325,
    rol: 6,
    pages: 450
  },
  {
    _id: 5,
    Category: 'Web Mining',
    BookName: 'Learning R',
    Author: 'Richard Cotton',
    qty: 5,
    price: 850,
    rol: 10
```

Step. 3.3: Write map and reduce functions to split the books into the following two categories:
(a) Big Books
(b) Small Books
Books which have more than 300 pages should be in the big book category. Books which have less than 300 pages should be in the small book category.
Step. 3.4: Count the number of books in each category.
Step 3.5: Store the output as follows as documents in a new collection, called "Book_Result".

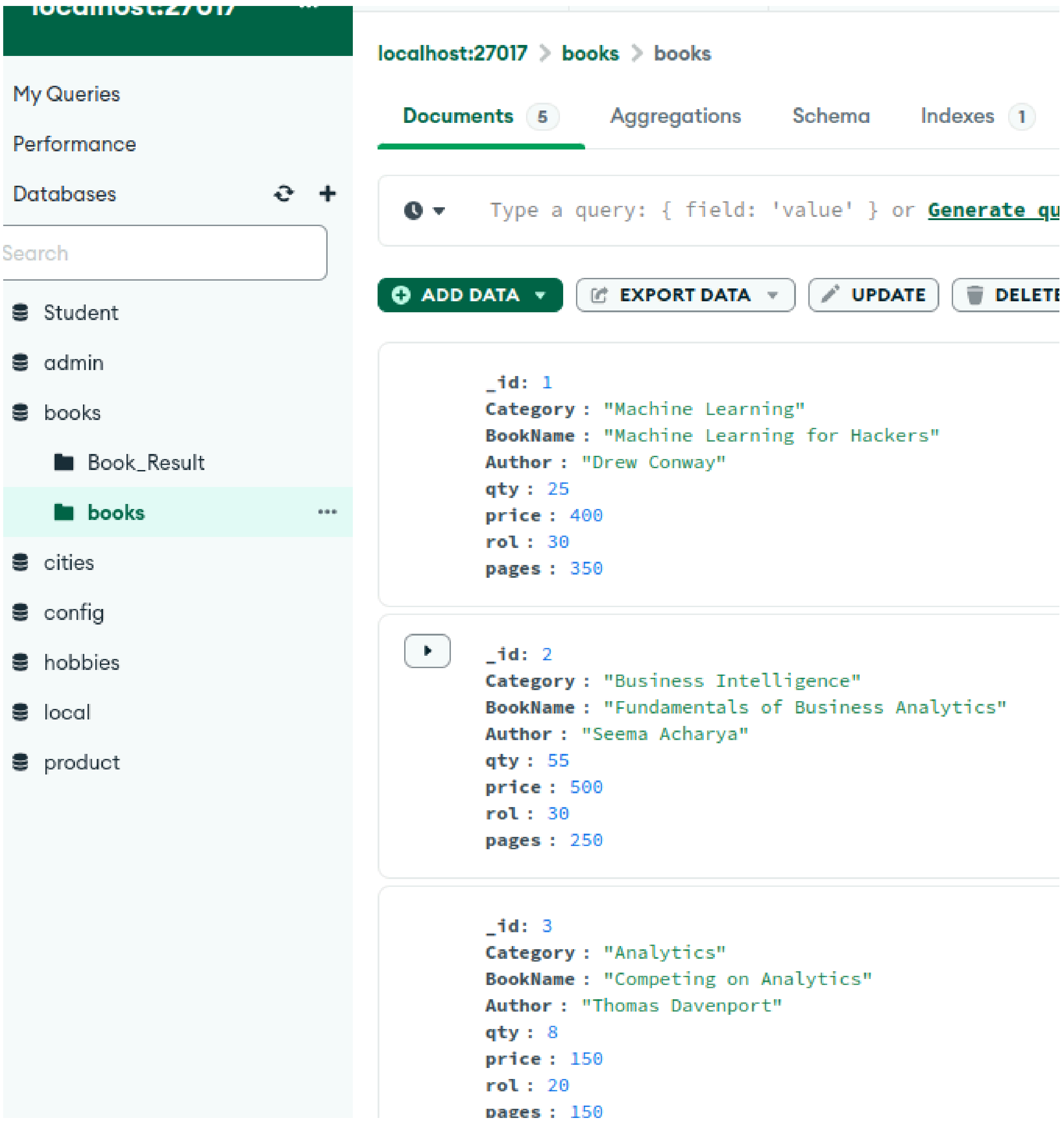


```
    _id: 5,
    Category: 'Web Mining',
    BookName: 'Learning R',
    Author: 'Richard Cotton',
    qty: 5,
    price: 850,
    rol: 10,
    pages: 120
  }
]
books> var mapFunction = function () {
...   var category = this.pages > 300 ? "Big Books" : "Small Books";
...   emit(category, 1);
... };

books> var reduceFunction = function (key, values) {
...   return Array.sum(values);
... };

books> db.books.mapReduce(
...   mapFunction,
...   reduceFunction,
...   {
...     out: "Book_Result"
...   }
... );
DeprecationWarning: Collection.mapReduce() is deprecated. Use an aggregation instead.
See https://docs.mongodb.com/manual/core/map-reduce for details.
{ result: 'Book_Result', ok: 1 }
books> db.Book_Result.find().pretty();
[ { _id: 'Big Books', value: 2 }, { _id: 'Small Books', value: 3 } ]
```

Output:



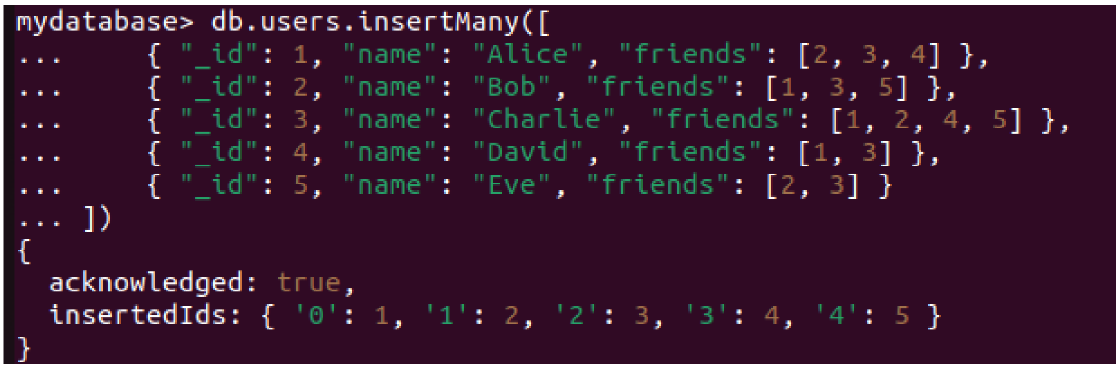
```
_id: "Big Books"
value : 2

_id: "Small Books"
value : 3
```



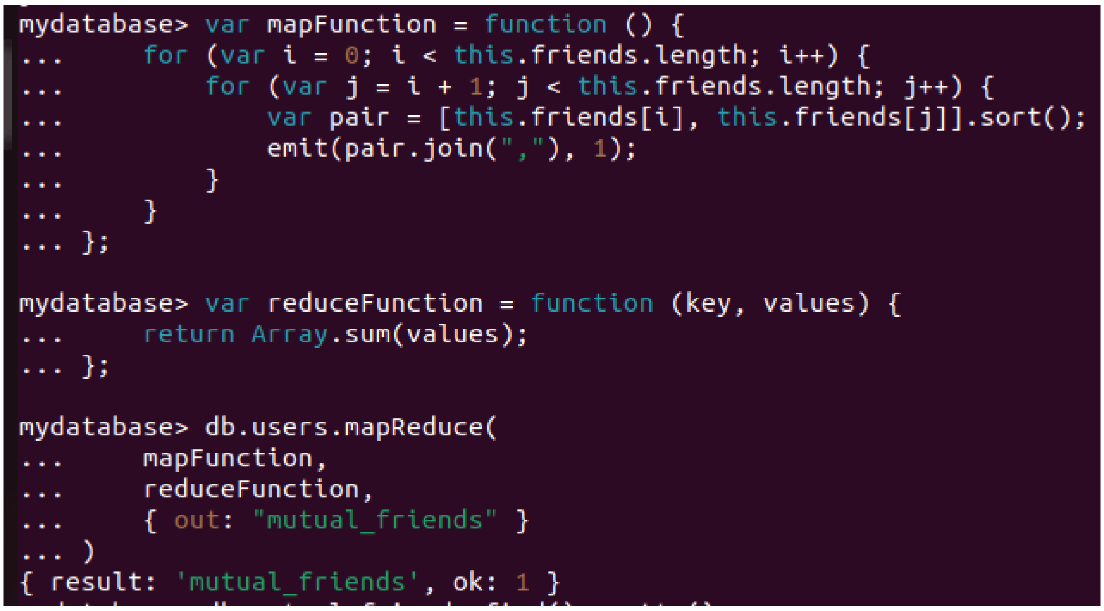
```
_id: 1
Category : "Machine Learning"
BookName : "Machine Learning for Hackers"
Author : "Drew Conway"
qty : 25
price : 400
rol : 30
pages : 350

_id: 2
Category : "Business Intelligence"
BookName : "Fundamentals of Business Analytics"
Author : "Seema Acharya"
qty : 55
price : 500
rol : 30
pages : 250

_id: 3
Category : "Analytics"
BookName : "Competing on Analytics"
Author : "Thomas Davenport"
qty : 8
price : 150
rol : 20
pages : 150
```

3.Write a mongodb query using mapreduce to find mutual friends.

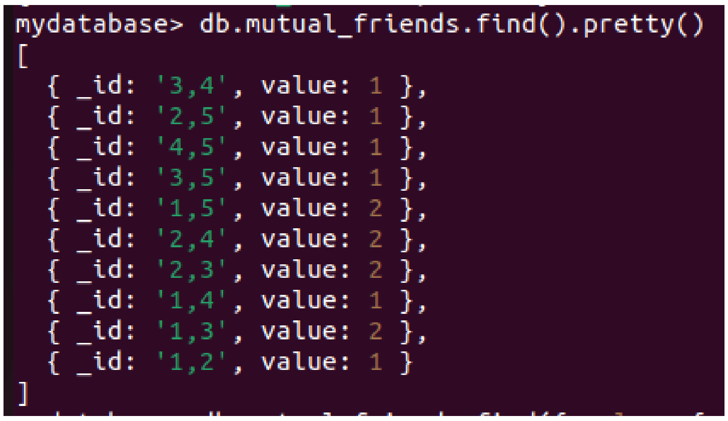
-you may consider the required collections.



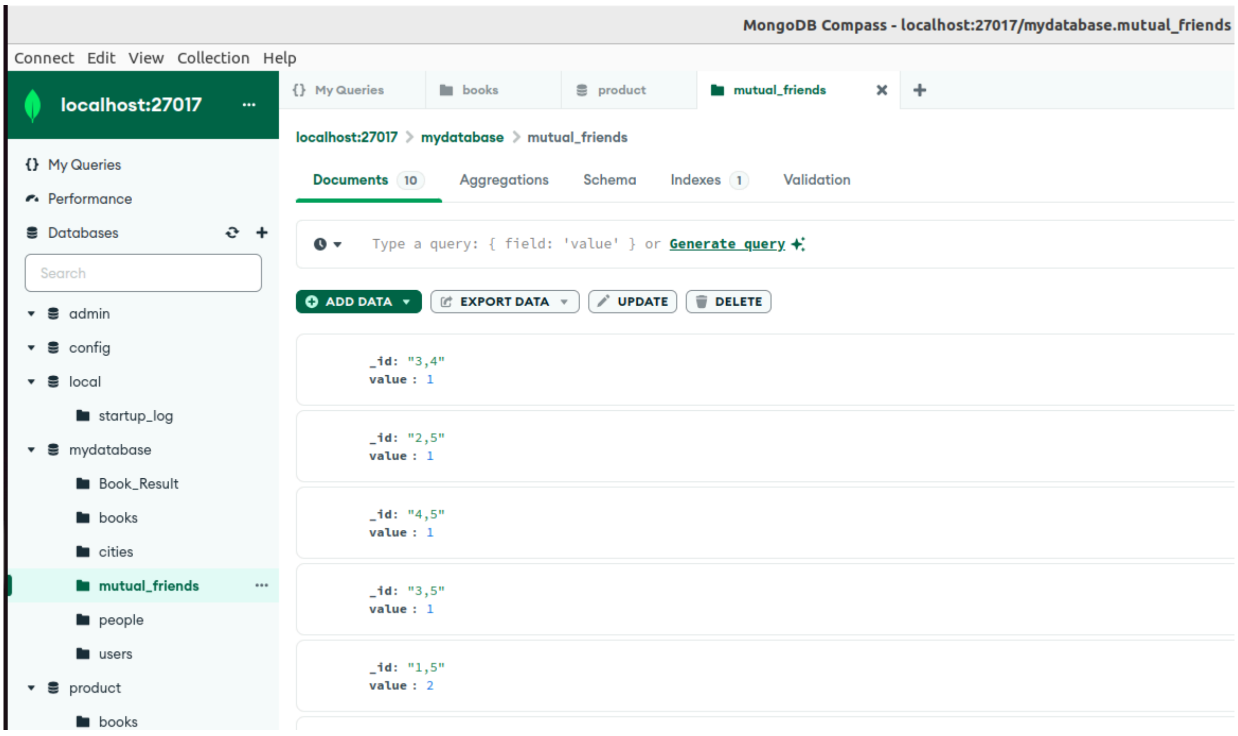
```
mydatabase> db.users.insertMany([
...     { "_id": 1, "name": "Alice", "friends": [2, 3, 4] },
...     { "_id": 2, "name": "Bob", "friends": [1, 3, 5] },
...     { "_id": 3, "name": "Charlie", "friends": [1, 2, 4, 5] },
...     { "_id": 4, "name": "David", "friends": [1, 3] },
...     { "_id": 5, "name": "Eve", "friends": [2, 3] }
... ])
{
  acknowledged: true,
  insertedIds: { '0': 1, '1': 2, '2': 3, '3': 4, '4': 5 }
}
```



```
mydatabase> var mapFunction = function () {
...     for (var i = 0; i < this.friends.length; i++) {
...         for (var j = i + 1; j < this.friends.length; j++) {
...             var pair = [this.friends[i], this.friends[j]].sort();
...             emit(pair.join(","), 1);
...         }
...     }
... };

mydatabase> var reduceFunction = function (key, values) {
...     return Array.sum(values);
... };

mydatabase> db.users.mapReduce(
...     mapFunction,
...     reduceFunction,
...     { out: "mutual_friends" }
... )
{ result: 'mutual_friends', ok: 1 }
```



```
mydatabase> db.mutual_friends.find().pretty()
[
  { _id: '3,4', value: 1 },
  { _id: '2,5', value: 1 },
  { _id: '4,5', value: 1 },
  { _id: '3,5', value: 1 },
  { _id: '1,5', value: 2 },
  { _id: '2,4', value: 2 },
  { _id: '2,3', value: 2 },
  { _id: '1,4', value: 1 },
  { _id: '1,3', value: 2 },
  { _id: '1,2', value: 1 }
]
```



MongoDB Compass - localhost:27017/mydatabase.mutual_friends

```
_id: "3,4"
value : 1

_id: "2,5"
value : 1

_id: "4,5"
value : 1

_id: "3,5"
value : 1

_id: "1,5"
value : 2
```

Summarised learning:

This lab, we focused on learning MongoDB, a document-oriented NoSQL database, through hands-on exercises involving CRUD operations, queries, and advanced features like MapReduce. Key topics include connecting to MongoDB, creating and manipulating databases and collections, inserting, updating, and deleting documents, and performing queries using operators for filtering and sorting data. Additionally, the lab introduced us to MapReduce functionality for data aggregation, such as categorizing books by size and counting them. Also practical exercises for database management, data analysis, and applying MongoDB's powerful querying capabilities for real-world scenarios like mutual friend finding were implemented.