Q.1 Create a database called Inventory.

EASY QUESTIONS

• Create a collection called Products with attributes: ProductID, Name, Category, Stock, and Price. use Inventory

```
test> use Inventory
switched to db Inventory
```

• Insert 10 product documents into the Products collection.

db.Products.insertMany([{ ProductID: 1, Name: "Laptop", Category: "Electronics", Stock: 15, Price: 599.99 }, { ProductID: 2, Name: "Smartphone", Category: "Electronics", Stock: 8, Price: 399.99 }, { ProductID: 3, Name: "Tablet", Category: "Electronics", Stock: 20, Price: 299.99 }, { ProductID: 4, Name: "Headphones", Category: "Accessories", Stock: 30, Price: 49.99 }, { ProductID: 5, Name: "Keyboard", Category: "Accessories", Stock: 25, Price: 29.99 }, { ProductID: 6, Name: "Monitor", Category: "Electronics", Stock: 12, Price: 199.99 }, { ProductID: 7, Name: "Mouse", Category: "Accessories", Stock: 18, Price: 9.99 }, { ProductID: 8, Name: "Charger", Category: "Electronics", Stock: 10, Price: 15.99 }, { ProductID: 9, Name: "Desk Lamp", Category: "Home", Stock: 5, Price: 20.99 }, { ProductID: 10, Name: "Webcam", Category: "Electronics", Stock: 14, Price: 79.99 }])

```
Inventory> db.Products.insertMany([
... { ProductID: 1, Name: "Laptop", Category: "Electronics", Stock: 15, Price: 599.99 },
... { ProductID: 2, Name: "Smartphone", Category: "Electronics", Stock: 8, Price: 399.99 },
... { ProductID: 3, Name: "Tablet", Category: "Electronics", Stock: 20, Price: 299.99 },
... { ProductID: 4, Name: "Headphones", Category: "Accessories", Stock: 30, Price: 49.99 },
... { ProductID: 5, Name: "Keyboard", Category: "Accessories", Stock: 25, Price: 29.99 },
... { ProductID: 6, Name: "Monitor", Category: "Electronics", Stock: 12, Price: 199.99 },
... { ProductID: 7, Name: "Mouse", Category: "Accessories", Stock: 18, Price: 9.99 },
... { ProductID: 8, Name: "Charger", Category: "Electronics", Stock: 10, Price: 15.99 },
... { ProductID: 9, Name: "Desk Lamp", Category: "Home", Stock: 5, Price: 20.99 },
... { ProductID: 10, Name: "Webcam", Category: "Electronics", Stock: 14, Price: 79.99 }   ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('675b0c5ad50797a88a0d8190'),
    '1': ObjectId('675b0c5ad50797a88a0d8191'),
    '2': ObjectId('675b0c5ad50797a88a0d8192'),
    '3': ObjectId('675b0c5ad50797a88a0d8193'),
    '4': ObjectId('675b0c5ad50797a88a0d8194'),
    '5': ObjectId('675b0c5ad50797a88a0d8195'),
    '6': ObjectId('675b0c5ad50797a88a0d8196'),
    '7': ObjectId('675b0c5ad50797a88a0d8197'),
    '8': ObjectId('675b0c5ad50797a88a0d8198'),
    '9': ObjectId('675b0c5ad50797a88a0d8199')
  }
}
```

• Retrieve all products with stock greater than 10.

db.Products.find({ Stock: { $gt: 10 } })

• Find all products from the 'Electronics' category.

db.Products.find({ Category: "Electronics" })

```
Inventory> db.Products.find({ Category: "Electronics" })
[
  {
    _id: ObjectId('675b0c5ad50797a88a0d8190'),
    ProductID: 1,
    Name: 'Laptop',
    Category: 'Electronics',
    Stock: 15,
    Price: 599.99
  },
  {
    _id: ObjectId('675b0c5ad50797a88a0d8191'),
    ProductID: 2,
    Name: 'Smartphone',
    Category: 'Electronics',
    Stock: 8,
    Price: 399.99
  },
  {
    _id: ObjectId('675b0c5ad50797a88a0d8192'),
    ProductID: 3,
    Name: 'Tablet',
    Category: 'Electronics',
    Stock: 20,
    Price: 299.99
  },
  {
    _id: ObjectId('675b0c5ad50797a88a0d8195'),
    ProductID: 6,
    Name: 'Monitor',
    Category: 'Electronics',
    Stock: 12,
    Price: 199.99
  },
  {
    _id: ObjectId('675b0c5ad50797a88a0d8197'),
    ProductID: 8,
    Name: 'Charger',
    Category: 'Electronics',
    Stock: 10,
    Price: 15.99
  },
  {
    _id: ObjectId('675b0c5ad50797a88a0d8199'),
    ProductID: 10,
    Name: 'Webcam',
    Category: 'Electronics',
    Stock: 14,
    Price: 79.99
  }
```

• Update the price of the product with ProductID 7 to 19.99.

db.Products.updateOne({ ProductID: 7 }, { $set: { Price: 19.99 } })

```
Inventory> db.Products.updateOne({ ProductID: 7 }, { $set: { Price: 19.99 } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
{
  _id: ObjectId('675b0c5ad50797a88a0d8196'),
  ProductID: 7,
  Name: 'Mouse',
  Category: 'Accessories',
  Stock: 18,
  Price: 19.99
},
{
```

• Delete the product with ProductID 2.

db.Products.deleteOne({ ProductID: 2 })

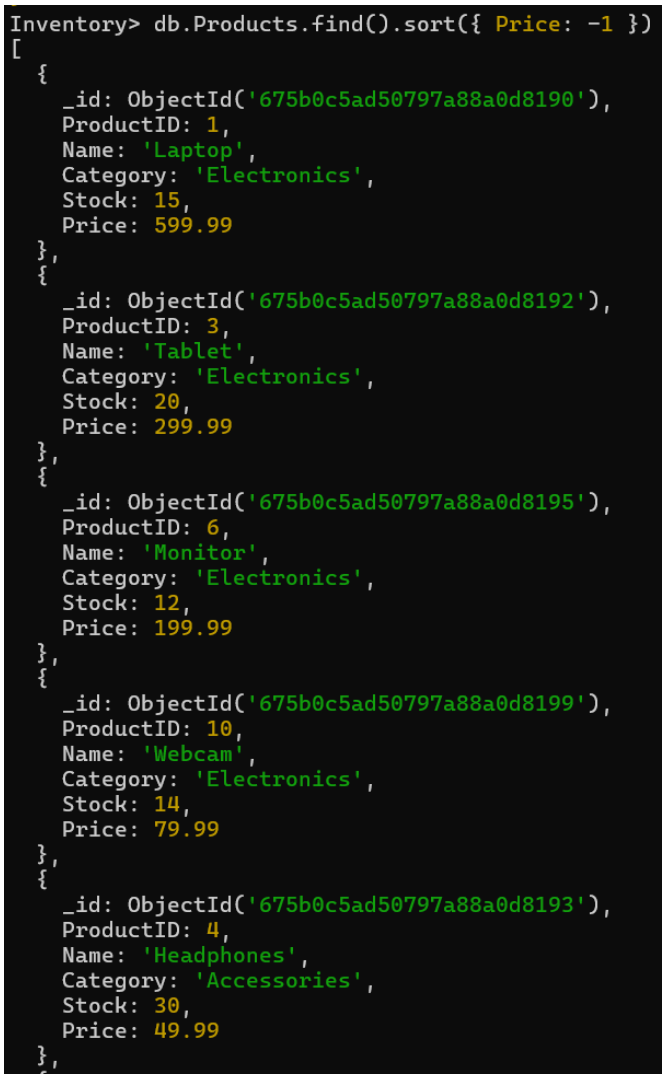
• Count how many products are in stock.

db.Products.countDocuments({ Stock: { $gt: 0 } })

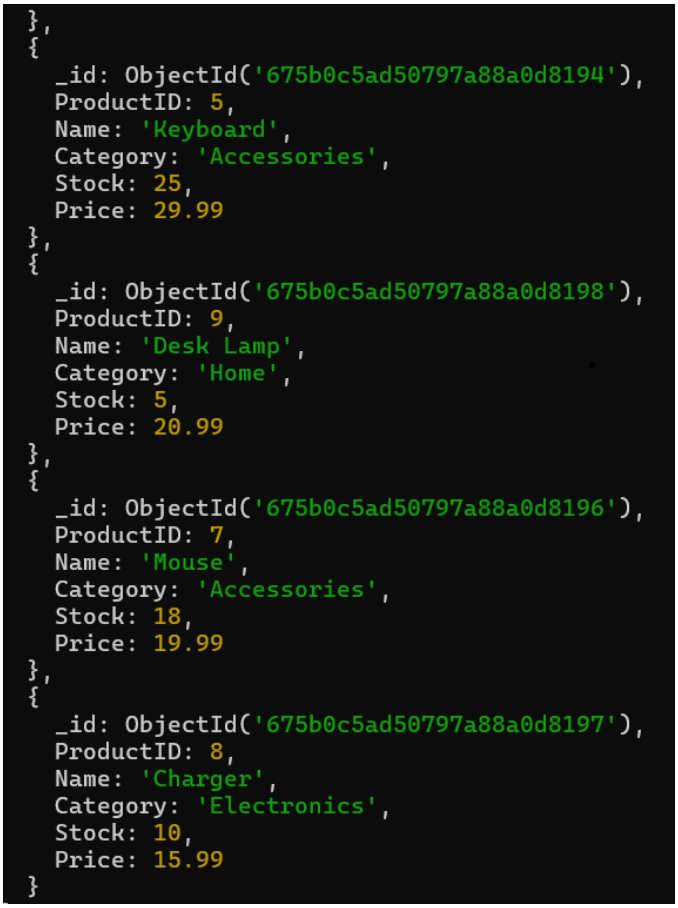
```
Inventory> db.Products.countDocuments({ Stock: { $gt: 0 } })
9
```

• Sort products by Price in descending order.

db.Products.find().sort({ Price: -1 })



```
Inventory> db.Products.find().sort({ Price: -1 })
[
  {
    _id: ObjectId('675b0c5ad50797a88a0d8190'),
    ProductID: 1,
    Name: 'Laptop',
    Category: 'Electronics',
    Stock: 15,
    Price: 599.99
  },
  {
    _id: ObjectId('675b0c5ad50797a88a0d8192'),
    ProductID: 3,
    Name: 'Tablet',
    Category: 'Electronics',
    Stock: 20,
    Price: 299.99
  },
  {
    _id: ObjectId('675b0c5ad50797a88a0d8195'),
    ProductID: 6,
    Name: 'Monitor',
    Category: 'Electronics',
    Stock: 12,
    Price: 199.99
  },
  {
    _id: ObjectId('675b0c5ad50797a88a0d8199'),
    ProductID: 10,
    Name: 'Webcam',
    Category: 'Electronics',
    Stock: 14,
    Price: 79.99
  },
  {
    _id: ObjectId('675b0c5ad50797a88a0d8193'),
    ProductID: 4,
    Name: 'Headphones',
    Category: 'Accessories',
    Stock: 30,
    Price: 49.99
  },
```



```
  },
  {
    _id: ObjectId('675b0c5ad50797a88a0d8194'),
    ProductID: 5,
    Name: 'Keyboard',
    Category: 'Accessories',
    Stock: 25,
    Price: 29.99
  },
  {
    _id: ObjectId('675b0c5ad50797a88a0d8198'),
    ProductID: 9,
    Name: 'Desk Lamp',
    Category: 'Home',
    Stock: 5,
    Price: 20.99
  },
  {
    _id: ObjectId('675b0c5ad50797a88a0d8196'),
    ProductID: 7,
    Name: 'Mouse',
    Category: 'Accessories',
    Stock: 18,
    Price: 19.99
  },
  {
    _id: ObjectId('675b0c5ad50797a88a0d8197'),
    ProductID: 8,
    Name: 'Charger',
    Category: 'Electronics',
    Stock: 10,
    Price: 15.99
  }
```

• Find products whose price is either 10 or 15.

db.Products.find({ Price: { $in: [10, 15] } })

```
Inventory> db.Products.find({ Stock: { $lt: 5 }, Price: { $gt: 10 } })
[
  {
    _id: ObjectId('675b0c5ad50797a88a0d8199'),
    ProductID: 10,
    Name: 'Webcam',
    Category: 'Electronics',
    Stock: 4,
    Price: 79.99
  }
```

HARD QUESTIONS

• Retrieve products whose Category is 'Home Appliances' and Stock is greater than 30.

db.Products.find({ Category: "Home Appliances", Stock: { $gt: 30 } })

```
Inventory> db.Products.find({ Category: "Home", Stock: { $gt: 30 } })
[
  {
    _id: ObjectId('675b0c5ad50797a88a0d8198'),
    ProductID: 9,
    Name: 'Desk Lamp',
    Category: 'Home',
    Stock: 35,
    Price: 20.99
  }
]
```

• Update the Category of the product with ProductID 4 to 'Furniture'.

db.Products.updateOne({ ProductID: 4 }, { $set: { Category: "Furniture" } })

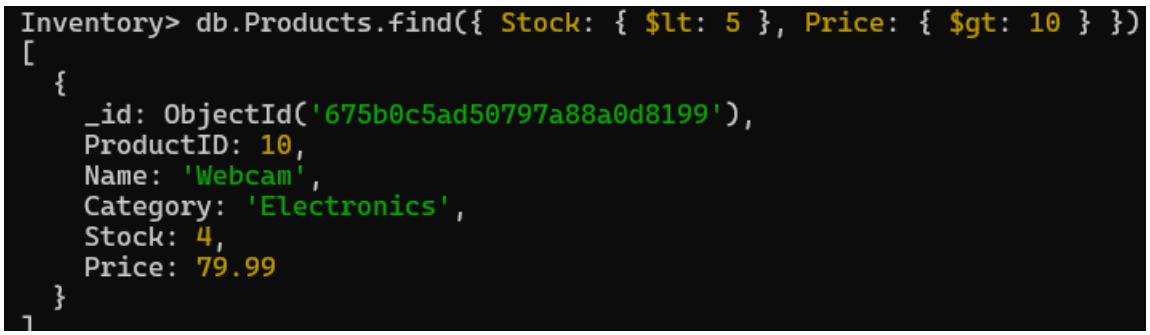
```
Inventory> db.Products.find({ ProductID: 4})
[
  {
    _id: ObjectId('675b0c5ad50797a88a0d8193'),
    ProductID: 4,
    Name: 'Headphones',
    Category: 'Furniture',
    Stock: 30,
    Price: 49.99
  }
]
```

• Use $nin to find exclude products with stock 10, 15 units.

db.Products.find({ Stock: { $nin: [10, 15] } })

• Find products with Stock less than 5 and Price greater than 10.

db.Products.find({ Stock: { $lt: 5 }, Price: { $gt: 10 } })



• Find products with a Name containing the character 'c'.

db.Products.find({ Name: { $regex: /c/i } })

```
Inventory> db.Products.find({ Name: { $regex: /c/i } })
[
  {
    _id: ObjectId('675b0c5ad50797a88a0d8197'),
    ProductID: 8,
    Name: 'Charger',
    Category: 'Electronics',
    Stock: 10,
    Price: 15.99
  },
  {
    _id: ObjectId('675b0c5ad50797a88a0d8199'),
    ProductID: 10,
    Name: 'Webcam',
    Category: 'Electronics',
    Stock: 4,
    Price: 79.99
  }
]
```

• Delete all products with stock equal to 0.

db.Products.deleteMany({ Stock: 0 })

• Retrieve the first 5 products, skipping the first 2 documents.

db.Products.find().skip(2).limit(5)

```
Inventory> db.Products.find().skip(2).limit(5)
[
  {
    _id: ObjectId('675b0c5ad50797a88a0d8193'),
    ProductID: 4,
    Name: 'Headphones',
    Category: 'Furniture',
    Stock: 30,
    Price: 49.99
  },
  {
    _id: ObjectId('675b0c5ad50797a88a0d8194'),
    ProductID: 5,
    Name: 'Keyboard',
    Category: 'Accessories',
    Stock: 25,
    Price: 29.99
  },
  {
    _id: ObjectId('675b0c5ad50797a88a0d8195'),
    ProductID: 6,
    Name: 'Monitor',
    Category: 'Electronics',
    Stock: 12,
    Price: 199.99
  },
  {
    _id: ObjectId('675b0c5ad50797a88a0d8196'),
    ProductID: 7,
    Name: 'Mouse',
    Category: 'Accessories',
    Stock: 18,
    Price: 19.99
  },
  {
    _id: ObjectId('675b0c5ad50797a88a0d8197'),
    ProductID: 8,
    Name: 'Charger',
    Category: 'Electronics',
    Stock: 10,
    Price: 15.99
  }
```

Banking System Easy Questions

• Create a database called Bank.

use Bank;

• Create a collection called Customers with attributes: CustomerID, Name, AccountBalance, and AccountType.
db.createCollection("Customers");

• Insert 10 documents representing customer information.

db.Customers.insertMany([{ CustomerID: 1, Name: "Alice", AccountBalance: 3000, AccountType: "Savings" }, { CustomerID: 2, Name: "Bob", AccountBalance: 1500, AccountType: "Checking" }, { CustomerID: 3, Name: "Charlie", AccountBalance: 500, AccountType: "Business" }, { CustomerID: 4, Name: "David", AccountBalance: 4000, AccountType: "Savings" }, { CustomerID: 5, Name: "Eve", AccountBalance: 2000, AccountType: "Checking" }, { CustomerID: 6, Name: "Frank", AccountBalance: 6000, AccountType: "Business" }, { CustomerID: 7, Name: "Grace", AccountBalance: 800, AccountType: "Savings" }, { CustomerID: 8, Name: "Hank", AccountBalance: 2500, AccountType: "Savings" }, { CustomerID: 9, Name: "Ivy", AccountBalance: 7000, AccountType: "Business" }, { CustomerID: 10, Name: "Jack", AccountBalance: 900, AccountType: "Checking" }]);

```
test> use Bank
switched to db Bank
Bank> db.createCollection("Customers");
{ ok: 1 }
Bank> db.Customers.insertMany([
... { CustomerID: 1, Name: "Alice", AccountBalance: 3000, AccountType: "Savings" },
... { CustomerID: 2, Name: "Bob", AccountBalance: 1500, AccountType: "Checking" },
... { CustomerID: 3, Name: "Charlie", AccountBalance: 500, AccountType: "Business" },
... { CustomerID: 4, Name: "David", AccountBalance: 4000, AccountType: "Savings" },
... { CustomerID: 5, Name: "Eve", AccountBalance: 2000, AccountType: "Checking" },
... { CustomerID: 6, Name: "Frank", AccountBalance: 6000, AccountType: "Business" },
... { CustomerID: 7, Name: "Grace", AccountBalance: 800, AccountType: "Savings" },
... { CustomerID: 8, Name: "Hank", AccountBalance: 2500, AccountType: "Savings" },
... { CustomerID: 9, Name: "Ivy", AccountBalance: 7000, AccountType: "Business" },
... { CustomerID: 10, Name: "Jack", AccountBalance: 900, AccountType: "Checking" }])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('675b259e28bc1d70f70d8190'),
    '1': ObjectId('675b259e28bc1d70f70d8191'),
    '2': ObjectId('675b259e28bc1d70f70d8192'),
    '3': ObjectId('675b259e28bc1d70f70d8193'),
    '4': ObjectId('675b259e28bc1d70f70d8194'),
    '5': ObjectId('675b259e28bc1d70f70d8195'),
    '6': ObjectId('675b259e28bc1d70f70d8196'),
    '7': ObjectId('675b259e28bc1d70f70d8197'),
    '8': ObjectId('675b259e28bc1d70f70d8198'),
    '9': ObjectId('675b259e28bc1d70f70d8199')
  }
```

• Find all customers with AccountType 'Savings'.

db.Customers.find({ AccountType: "Savings" });

```
Bank> db.Customers.find({AccountType: 'Savings'})
[
  {
    _id: ObjectId('675b259e28bc1d70f70d8190'),
    CustomerID: 1,
    Name: 'Alice',
    AccountBalance: 3000,
    AccountType: 'Savings'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8193'),
    CustomerID: 4,
    Name: 'David',
    AccountBalance: 4000,
    AccountType: 'Savings'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8196'),
    CustomerID: 7,
    Name: 'Grace',
    AccountBalance: 800,
    AccountType: 'Savings'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8197'),
    CustomerID: 8,
    Name: 'Hank',
    AccountBalance: 2500,
    AccountType: 'Savings'
```

• Update the AccountBalance of the customer with CustomerID 4 to 5000.

db.Customers.updateOne({ CustomerID: 4 }, { $set: { AccountBalance: 5000 } });

```
Bank> db.Customers.updateOne({ CustomerID: 4 }, { $set: { AccountBalance: 5000 } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

• Delete the customer with CustomerID 2.

db.Customers.deleteOne({ CustomerID: 2 });

```
}
Bank> db.Customers.deleteOne({ CustomerID: 2 });
{ acknowledged: true, deletedCount: 1 }
```

• Count the number of customers with an AccountBalance greater than 1000.

db.Customers.countDocuments({ AccountBalance: { $gt: 1000 } });

```
Bank> db.Customers.countDocuments({ AccountBalance: { $gt: 1000 } });
6
```

• Sort customers by AccountBalance in ascending order.

db.Customers.find().sort({ AccountBalance: 1 });

```
Bank> db.Customers.find().sort({AccountBalance:1})
[
  {
    _id: ObjectId('675b259e28bc1d70f70d8192'),
    CustomerID: 3,
    Name: 'Charlie',
    AccountBalance: 500,
    AccountType: 'Business'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8196'),
    CustomerID: 7,
    Name: 'Grace',
    AccountBalance: 800,
    AccountType: 'Savings'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8199'),
    CustomerID: 10,
    Name: 'Jack',
    AccountBalance: 900,
    AccountType: 'Checking'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8194'),
    CustomerID: 5,
    Name: 'Eve',
    AccountBalance: 2000,
    AccountType: 'Checking'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8197'),
    CustomerID: 8,
    Name: 'Hank',
    AccountBalance: 2500,
    AccountType: 'Savings'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8190'),
    CustomerID: 1,
    Name: 'Alice',
    AccountBalance: 3000,
    AccountType: 'Savings'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8193'),
    CustomerID: 4,
    Name: 'David',
    AccountBalance: 5000,
    AccountType: 'Savings'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8195'),
    CustomerID: 6,
    Name: 'Frank',
    AccountBalance: 6000,
    AccountType: 'Business'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8198'),
    CustomerID: 9,
    Name: 'Ivy',
    AccountBalance: 7000,
    AccountType: 'Business'
  }
```

• Find customers whose AccountType is either 'Checking' or 'Business'.

db.Customers.find({ AccountType: { $in: ["Checking", "Business"] } });

```
Bank> db.Customers.find({AccountType: {$in: ['Checking', 'Business']}})
[
  {
    _id: ObjectId('675b259e28bc1d70f70d8192'),
    CustomerID: 3,
    Name: 'Charlie',
    AccountBalance: 500,
    AccountType: 'Business'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8194'),
    CustomerID: 5,
    Name: 'Eve',
    AccountBalance: 2000,
    AccountType: 'Checking'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8195'),
    CustomerID: 6,
    Name: 'Frank',
    AccountBalance: 6000,
    AccountType: 'Business'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8198'),
    CustomerID: 9,
    Name: 'Ivy',
    AccountBalance: 7000,
    AccountType: 'Business'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8199'),
    CustomerID: 10,
    Name: 'Jack',
    AccountBalance: 900,
    AccountType: 'Checking'
  }
]
```

• Find customers with an AccountBalance between 2000 and 5000.

db.Customers.find({ AccountBalance: { $gte: 2000, $lte: 5000 } });

```
]
Bank> db.Customers.find({AccountBalance: {$gte:2000,$lte:5000}})
[
  {
    _id: ObjectId('675b259e28bc1d70f70d8190'),
    CustomerID: 1,
    Name: 'Alice',
    AccountBalance: 3000,
    AccountType: 'Savings'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8193'),
    CustomerID: 4,
    Name: 'David',
    AccountBalance: 5000,
    AccountType: 'Savings'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8194'),
    CustomerID: 5,
    Name: 'Eve',
    AccountBalance: 2000,
    AccountType: 'Checking'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8197'),
    CustomerID: 8,
    Name: 'Hank',
    AccountBalance: 2500,
    AccountType: 'Savings'
  }
]
```

Hard Questions

• Update the AccountBalance of all customers with AccountType 'Savings' by 1000.

db.Customers.updateMany({ AccountType: "Savings" }, { $inc: { AccountBalance: 1000 } });

```
Bank> db.Customers.updateMany({AccountType: 'Savings'},{$inc: {AccountBalance: 1000} })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 4,
  modifiedCount: 4,
  upsertedCount: 0
```

• Retrieve customers with an AccountBalance greater than 5000.

db.Customers.find({ AccountBalance: { $gt: 5000 } });

```
Bank> db.Customers.find({AccountBalance: {$gt :5000}} )
[
  {
    _id: ObjectId('675b259e28bc1d70f70d8193'),
    CustomerID: 4,
    Name: 'David',
    AccountBalance: 6000,
    AccountType: 'Savings'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8195'),
    CustomerID: 6,
    Name: 'Frank',
    AccountBalance: 6000,
    AccountType: 'Business'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8198'),
    CustomerID: 9,
    Name: 'Ivy',
    AccountBalance: 7000,
    AccountType: 'Business'
  }
```

• Find customers with AccountType 'Checking' and AccountBalance less than 1000.

db.Customers.find({ AccountType: "Checking", AccountBalance: { $lt: 1000 } });

```
Bank> db.Customers.find({AccountType: 'Checking', AccountBalance: {$lt:1000} })
[
  {
    _id: ObjectId('675b259e28bc1d70f70d8199'),
    CustomerID: 10,
    Name: 'Jack',
    AccountBalance: 900,
    AccountType: 'Checking'
  }
```

• Use $in to find customers with AccountType either 'Business' or 'Checking'.

db.Customers.find({ AccountType: { $in: ["Business", "Checking"] } });

```
]
Bank> db.Customers.find({ AccountType: { $in: ["Business", "Checking"] } })
[
  {
    _id: ObjectId('675b259e28bc1d70f70d8192'),
    CustomerID: 3,
    Name: 'Charlie',
    AccountBalance: 500,
    AccountType: 'Business'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8194'),
    CustomerID: 5,
    Name: 'Eve',
    AccountBalance: 2000,
    AccountType: 'Checking'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8195'),
    CustomerID: 6,
    Name: 'Frank',
    AccountBalance: 6000,
    AccountType: 'Business'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8198'),
    CustomerID: 9,
    Name: 'Ivy',
    AccountBalance: 7000,
    AccountType: 'Business'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8199'),
    CustomerID: 10,
    Name: 'Jack',
    AccountBalance: 900,
    AccountType: 'Checking'
  }
```

• Find customers whose AccountBalance is between 3000 and 7000.

db.Customers.find({ AccountBalance: { $gte: 3000, $lte: 7000 } });

```
]
Bank> db.Customers.find({ AccountBalance: { $gte: 3000, $lte: 7000 } });
[
  {
    _id: ObjectId('675b259e28bc1d70f70d8190'),
    CustomerID: 1,
    Name: 'Alice',
    AccountBalance: 4000,
    AccountType: 'Savings'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8193'),
    CustomerID: 4,
    Name: 'David',
    AccountBalance: 6000,
    AccountType: 'Savings'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8195'),
    CustomerID: 6,
    Name: 'Frank',
    AccountBalance: 6000,
    AccountType: 'Business'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8197'),
    CustomerID: 8,
    Name: 'Hank',
    AccountBalance: 3500,
    AccountType: 'Savings'
  },
  {
    _id: ObjectId('675b259e28bc1d70f70d8198'),
    CustomerID: 9,
    Name: 'Ivy',
    AccountBalance: 7000,
    AccountType: 'Business'
  }
]
```

• Find all customers with AccountBalance less than 1000 and update it to 1500.

db.Customers.updateMany({ AccountBalance: { $lt: 1000 } }, { $set: { AccountBalance: 1500 } });

```
Bank> db.Customers.updateMany( { AccountBalance: { $lt: 1000 } }, { $set: { AccountBaAccountBalance: 1500 } } )
.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
```

• Delete all the customers .

db.Customers.deleteMany({});

```
}
Bank> db.Customers.deleteMany({});
{ acknowledged: true, deletedCount: 9 }
Bank>
```

Book and Author Database

Easy Questions

• Create a database called "Library".

use Library;

```
test> use Library
switched to db Library
```

• Create a collection called "Books" with attributes: BookID, Title, AuthorID, Genre, PublishedYear, and Pages.
db.createCollection("Books");

```
Library> db.createCollection("Books");
{ ok: 1 }
Library> db.createCollection("Authors");
{ ok: 1 }
```

• Create a collection called "Authors" with attributes: AuthorID, Name, DateOfBirth, Nationality.
db.createCollection("Authors");

```
Library> db.createCollection("Books");
{ ok: 1 }
Library> db.createCollection("Authors");
{ ok: 1 }
```

• Insert 5 books into the "Books" collection. 9

db.Books.insertMany([{ BookID: 1, Title: "Harry Potter and the Sorcerer's Stone", AuthorID: 1, Genre: "Fantasy", PublishedYear: 1997, Pages: 309 }, { BookID: 2, Title: "The Hobbit", AuthorID: 2, Genre: "Fantasy", PublishedYear: 1937, Pages: 310 }, { BookID: 3, Title: "The Catcher in the Rye", AuthorID: 3, Genre: "Fiction", PublishedYear: 1951, Pages: 277 }, { BookID: 4, Title: "1984", AuthorID: 4, Genre: "Dystopian", PublishedYear: 1949, Pages: 328 }, { BookID: 5, Title: "The Da Vinci Code", AuthorID: 5, Genre: "Thriller", PublishedYear: 2003, Pages: 689 }]);

```
Library> db.Books.insertMany([
... { BookID: 1, Title: "Harry Potter and the Sorcerer's Stone", AuthorID: 1, Genre: "Fantasy",
... PublishedYear: 1997, Pages: 309 },
... { BookID: 2, Title: "The Hobbit", AuthorID: 2, Genre: "Fantasy", PublishedYear: 1937, PaPages: 310 },
... { BookID: 3, Title: "The Catcher in the Rye", AuthorID: 3, Genre: "Fiction", PublishedYear: 1951,
... Pages: 277 },
... { BookID: 4, Title: "1984", AuthorID: 4, Genre: "Dystopian", PublishedYear: 1949, Pages: 328 },
... { BookID: 5, Title: "The Da Vinci Code", AuthorID: 5, Genre: "Thriller", PublishedYear: 2003, Pages:
... 689 }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('675bb38dc8816913450d8190'),
    '1': ObjectId('675bb38dc8816913450d8191'),
    '2': ObjectId('675bb38dc8816913450d8192'),
    '3': ObjectId('675bb38dc8816913450d8193'),
    '4': ObjectId('675bb38dc8816913450d8194')
  }
```

• Insert 3 authors into the "Authors" collection.

db.Authors.insertMany([{ AuthorID: 1, Name: "J.K. Rowling", DateOfBirth: "1965-07-31", Nationality: "British" }, { AuthorID: 2, Name: "J.R.R. Tolkien", DateOfBirth: "1892-01-03", Nationality: "British" }, { AuthorID: 3, Name: "J.D. Salinger", DateOfBirth: "1919-01-01", Nationality: "American" }]);

```
Library> db.Authors.insertMany([
... { AuthorID: 1, Name: "J.K. Rowling", DateOfBirth: "1965-07-31", Nationality: "British" },
... { AuthorID: 2, Name: "J.R.R. Tolkien", DateOfBirth: "1892-01-03", Nationality: "British" },

... { AuthorID: 3, Name: "J.D. Salinger", DateOfBirth: "1919-01-01", Nationality: "American" }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('675bb514c8816913450d819a'),
    '1': ObjectId('675bb514c8816913450d819b'),
    '2': ObjectId('675bb514c8816913450d819c')
  }
}
```

• Find all books with the Genre "Fiction".

db.Books.find({ Genre: "Fiction" });

```
Library> db.Books.find({ Genre: "Fiction" });
[
  {
    _id: ObjectId('675bb38dc8816913450d8192'),
    BookID: 3,
    Title: 'The Catcher in the Rye',
    AuthorID: 3,
    Genre: 'Fiction',
    PublishedYear: 1951,
    Pages: 277
  },
  {
    _id: ObjectId('675bb39dc8816913450d8197'),
    BookID: 3,
    Title: 'The Catcher in the Rye',
    AuthorID: 3,
    Genre: 'Fiction',
    PublishedYear: 1951,
    Pages: 277
  }
]
```

• Find all books authored by an author with the Name "J.K. Rowling".

db.Books.find({ AuthorID: db.Authors.findOne({ Name: "J.K. Rowling" }).AuthorID });

```
Library> db.Books.find({ AuthorID: db.Authors.findOne({ Name: "J.K. Rowling" }).AuthorID })
[
  {
    _id: ObjectId('675bb38dc8816913450d8190'),
    BookID: 1,
    Title: "Harry Potter and the Sorcerer's Stone",
    AuthorID: 1,
    Genre: 'Fantasy',
    PublishedYear: 1997,
    Pages: 309
  },
  {
    _id: ObjectId('675bb39dc8816913450d8195'),
    BookID: 1,
    Title: "Harry Potter and the Sorcerer's Stone",
    AuthorID: 1,
    Genre: 'Fantasy',
    PublishedYear: 1997,
    Pages: 309
  }
]
```

• Update the PublishedYear of the book with BookID 3 to 2022.

db.Books.updateOne({ BookID: 3 }, { $set: { PublishedYear: 2022 } });

```
Library> db.Books.updateOne({ BookID: 3 }, { $set: { PublishedYear: 2022 } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

• Delete the book with BookID 2.

db.Books.deleteOne({ BookID: 2 });

```
Library> db.Books.deleteOne({ BookID: 2 });
{ acknowledged: true, deletedCount: 1 }
```

• Count how many books belong to the "Science Fiction" genre.

db.Books.countDocuments({ Genre: "Science Fiction" });

```
Library>  db.Books.countDocuments({ Genre: "Fiction" });
2
Library>
```

Hard Questions

• Find books published between the years 2000 and 2010.

db.Books.find({ PublishedYear: { $gte: 2000, $lte: 2010 } });

```
Library> db.Books.find({ PublishedYear: { $gte: 2000, $lte: 2010 } });
[
  {
    _id: ObjectId('675bb38dc8816913450d8194'),
    BookID: 5,
    Title: 'The Da Vinci Code',
    AuthorID: 5,
    Genre: 'Thriller',
    PublishedYear: 2003,
    Pages: 689
  },
  {
    _id: ObjectId('675bb39dc8816913450d8199'),
    BookID: 5,
    Title: 'The Da Vinci Code',
    AuthorID: 5,
    Genre: 'Thriller',
    PublishedYear: 2003,
    Pages: 689
  }
]
```

• Use $in to find books that belong to the genres "Thriller", "Romance", or "Adventure".

db.Books.find({ Genre: { $in: ["Thriller", "Romance", "Adventure"] } });

```
Library> db.Books.find({ Genre: { $in: ["Thriller", "Romance", "Adventure"] } });
[
  {
    _id: ObjectId('675bb38dc8816913450d8194'),
    BookID: 5,
    Title: 'The Da Vinci Code',
    AuthorID: 5,
    Genre: 'Thriller',
    PublishedYear: 2003,
    Pages: 689
  },
  {
    _id: ObjectId('675bb39dc8816913450d8199'),
    BookID: 5,
    Title: 'The Da Vinci Code',
    AuthorID: 5,
    Genre: 'Thriller',
    PublishedYear: 2003,
    Pages: 689
  }
]
```

• Retrieve the first 3 books, skipping the first 2 documents.

db.Books.find().skip(2).limit(3);

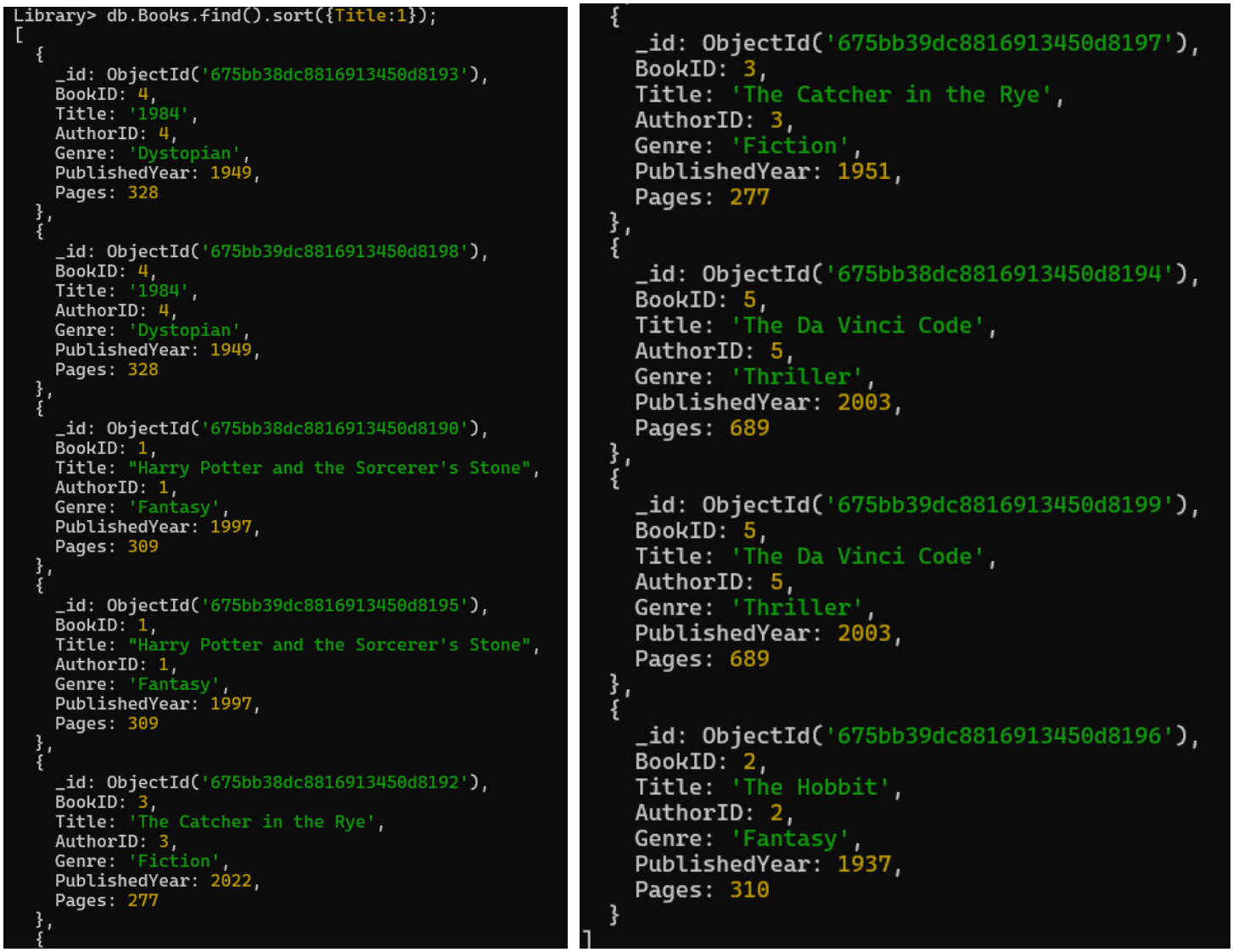
```
Library> db.Books.find().skip(2).limit(3);
[
  {
    _id: ObjectId('675bb38dc8816913450d8193'),
    BookID: 4,
    Title: '1984',
    AuthorID: 4,
    Genre: 'Dystopian',
    PublishedYear: 1949,
    Pages: 328
  },
  {
    _id: ObjectId('675bb38dc8816913450d8194'),
    BookID: 5,
    Title: 'The Da Vinci Code',
    AuthorID: 5,
    Genre: 'Thriller',
    PublishedYear: 2003,
    Pages: 689
  },
  {
    _id: ObjectId('675bb39dc8816913450d8195'),
    BookID: 1,
    Title: "Harry Potter and the Sorcerer's Stone",
    AuthorID: 1,
    Genre: 'Fantasy',
    PublishedYear: 1997,
    Pages: 309
  }
]
```

- Find all books written by authors born after 1980.

```
Library> db.Books.find({
...     AuthorID: {
...         $in: db.Authors.find({ DateOfBirth: { $gt: new Date("01-01-198
0") } }).toArray().map(function(a) { return a.AuthorID; })
...     }
... });
[
  {
    _id: ObjectId('675abe3f33e271f2cd0d8192'),
    BookID: 3,
    Title: 'Book3',
    AuthorID: 3,
    Genre: 'Fiction',
    PublishedYear: 2022,
    Pages: 350
  },
  {
    _id: ObjectId('675bb01e772c8ceaeb0d8192'),
    BookID: 3,
    Title: 'Book Three',
    AuthorID: 3,
    Genre: 'Science Fiction',
    PublishedYear: 2010,
    Pages: 400
  }
]
```

• Sort books by Title in alphabetical order.

db.Books.find().sort({ Title: 1 });



```
Library> db.Books.find().sort({Title:1});
[
  {
    _id: ObjectId('675bb38dc8816913450d8193'),
    BookID: 4,
    Title: '1984',
    AuthorID: 4,
    Genre: 'Dystopian',
    PublishedYear: 1949,
    Pages: 328
  },
  {
    _id: ObjectId('675bb39dc8816913450d8198'),
    BookID: 4,
    Title: '1984',
    AuthorID: 4,
    Genre: 'Dystopian',
    PublishedYear: 1949,
    Pages: 328
  },
  {
    _id: ObjectId('675bb38dc8816913450d8190'),
    BookID: 1,
    Title: "Harry Potter and the Sorcerer's Stone",
    AuthorID: 1,
    Genre: 'Fantasy',
    PublishedYear: 1997,
    Pages: 309
  },
  {
    _id: ObjectId('675bb39dc8816913450d8195'),
    BookID: 1,
    Title: "Harry Potter and the Sorcerer's Stone",
    AuthorID: 1,
    Genre: 'Fantasy',
    PublishedYear: 1997,
    Pages: 309
  },
  {
    _id: ObjectId('675bb38dc8816913450d8192'),
    BookID: 3,
    Title: 'The Catcher in the Rye',
    AuthorID: 3,
    Genre: 'Fiction',
    PublishedYear: 2022,
    Pages: 277
  },
  {
    _id: ObjectId('675bb39dc8816913450d8197'),
    BookID: 3,
    Title: 'The Catcher in the Rye',
    AuthorID: 3,
    Genre: 'Fiction',
    PublishedYear: 1951,
    Pages: 277
  },
  {
    _id: ObjectId('675bb38dc8816913450d8194'),
    BookID: 5,
    Title: 'The Da Vinci Code',
    AuthorID: 5,
    Genre: 'Thriller',
    PublishedYear: 2003,
    Pages: 689
  },
  {
    _id: ObjectId('675bb39dc8816913450d8199'),
    BookID: 5,
    Title: 'The Da Vinci Code',
    AuthorID: 5,
    Genre: 'Thriller',
    PublishedYear: 2003,
    Pages: 689
  },
  {
    _id: ObjectId('675bb39dc8816913450d8196'),
    BookID: 2,
    Title: 'The Hobbit',
    AuthorID: 2,
    Genre: 'Fantasy',
    PublishedYear: 1937,
    Pages: 310
  }
]
```

• Group books by Genre and find the average PublishedYear for each genre.

db.Books.aggregate([{ $group: { _id: "$Genre", avgPublishedYear: { $avg: "$PublishedYear" } } }]);

```
Library> db.Books.aggregate([ { $group: { _id: "$Genre", avgPublishedYear: { $avg: "$PublishedYear" } } } ]);
[
  { _id: 'Fiction', avgPublishedYear: 1986.5 },
  { _id: 'Thriller', avgPublishedYear: 2003 },
  { _id: 'Dystopian', avgPublishedYear: 1949 },
  { _id: 'Fantasy', avgPublishedYear: 1977 }
]
Library>
```

• Find all books whose Title contains the word "Harry Potter".

db.Books.find({ Title: /Harry Potter/i });

```
Library> db.Books.find({ Title: /Harry Potter/i });
[
  {
    _id: ObjectId('675bb38dc8816913450d8190'),
    BookID: 1,
    Title: "Harry Potter and the Sorcerer's Stone",
    AuthorID: 1,
    Genre: 'Fantasy',
    PublishedYear: 1997,
    Pages: 309
  },
  {
    _id: ObjectId('675bb39dc8816913450d8195'),
    BookID: 1,
    Title: "Harry Potter and the Sorcerer's Stone",
    AuthorID: 1,
    Genre: 'Fantasy',
    PublishedYear: 1997,
    Pages: 309
  }
]
```

• Find all authors who have written books in the "Horror" genre.

db.Authors.find({ AuthorID: { $in: db.Books.find({ Genre: "Horror" }).map(book => book.AuthorID) });