Step 1: Create Database and Switch to It
Step 2: Insert Sample Authors

```
PS C:\Users\ashut> mongosh
Current Mongosh Log ID: 6904e24cf3a61a1ba8f7d010
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.0.0
Using MongoDB:          8.0.3
Using Mongosh:          2.0.0
mongosh 2.5.9 is available for download: https://www.mongodb.com/try/download/shell

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

------
   The server generated these startup warnings when booting
   2025-10-30T11:57:16.284+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

test> use BookVerseDB
switched to db BookVerseDB
BookVerseDB> db.Authors.insertMany([
...   {
...     _id: ObjectId(),
...     name: "Isaac Asimov",
...     nationality: "American",
...     birthYear: 1920
...   },
...   {
...     _id: ObjectId(),
...     name: "J.K. Rowling",
...     nationality: "British",
...     birthYear: 1965
...   },
...   {
...     _id: ObjectId(),
...     name: "Brandon Sanderson",
...     nationality: "American",
...     birthYear: 1975
...   }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6904e25ef3a61a1ba8f7d011"),
    '1': ObjectId("6904e25ef3a61a1ba8f7d012"),
    '2': ObjectId("6904e25ef3a61a1ba8f7d013")
  }
}
```

Step 3: Insert Sample Users

```
BookVerseDB> db.Users.insertMany([
...   {
...     _id: ObjectId(),
...     name: "Alice Johnson",
...     email: "alice@example.com",
...     joinDate: new Date("2024-08-15")
...   },
...   {
...     _id: ObjectId(),
...     name: "Bob Smith",
...     email: "bob@example.com",
...     joinDate: new Date("2024-09-20")
...   },
...   {
...     _id: ObjectId(),
...     name: "Carol White",
...     email: "carol@example.com",
...     joinDate: new Date("2024-10-01")
...   }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6904e2a6f3a61a1ba8f7d014"),
    '1': ObjectId("6904e2a6f3a61a1ba8f7d015"),
    '2': ObjectId("6904e2a6f3a61a1ba8f7d016")
  }
}
BookVerseDB>
```

Step 4: Get Author IDs for Books (Run this to see IDs)

Step 5: Insert Sample Books

```
BookVerseDB> db.Books.insertMany([
...   {
...     _id: ObjectId(),
...     title: "Foundation",
...     genre: "Science Fiction",
...     publicationYear: 1951,
...     authorId: asimovId,
...     ratings: [
...       { user: "Alice Johnson", score: 5, comment: "Masterpiece of sci-fi!" },
...       { user: "Bob Smith", score: 4, comment: "Great worldbuilding" }
...     ]
...   },
...   {
...     _id: ObjectId(),
...     title: "I, Robot",
...     genre: "Science Fiction",
...     publicationYear: 2004,
...     authorId: asimovId,
...     ratings: [
...       { user: "Carol White", score: 4.5, comment: "Thought-provoking" }
...     ]
...   },
...   {
...     _id: ObjectId(),
...     title: "Harry Potter and the Philosopher's Stone",
...     genre: "Fantasy",
...     publicationYear: 1997,
...     authorId: rowlingId,
...     ratings: [
...       { user: "Alice Johnson", score: 5, comment: "Magical and captivating!" },
...       { user: "Carol White", score: 5, comment: "Best book ever!" }
...     ]
...   },
...   {
...     _id: ObjectId(),
...     title: "Mistborn: The Final Empire",
...     genre: "Fantasy",
...     publicationYear: 2006,
...     authorId: sandersonId,
...     ratings: [
...       { user: "Bob Smith", score: 4.8, comment: "Amazing magic system" },
...       { user: "Alice Johnson", score: 4.5, comment: "Incredible plot twists" }
...     ]
...   },
...   {
...     _id: ObjectId(),
...     title: "The Way of Kings",
...     genre: "Fantasy",
...     publicationYear: 2010,
...     authorId: sandersonId,
```

```
...     _id: ObjectId(),
...     title: "The Way of Kings",
...     genre: "Fantasy",
...     publicationYear: 2010,
...     authorId: sandersonId,
...     ratings: [
...       { user: "Carol White", score: 5, comment: "Epic fantasy at its finest" },
...       { user: "Bob Smith", score: 4.7, comment: "Long but worth it" }
...     ]
...   }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6904e3a4f3a61a1ba8f7d017"),
    '1': ObjectId("6904e3a4f3a61a1ba8f7d018"),
    '2': ObjectId("6904e3a4f3a61a1ba8f7d019"),
    '3': ObjectId("6904e3a4f3a61a1ba8f7d01a"),
    '4': ObjectId("6904e3a4f3a61a1ba8f7d01b")
  }
}
BookVerseDB>
```

```
  {
    _id: ObjectId("6904e3a4f3a61a1ba8f7d01b"),
    title: 'The Way of Kings',
    genre: 'Fantasy',
    publicationYear: 2010,
    avgRating: 4.85
  }
]
BookVerseDB> db.Books.aggregate([
...   {
...     $project: {
...       title: 1,
...       genre: 1,
...       ratingCount: { $size: "$ratings" },
...       avgRating: { $avg: "$ratings.score" }
...     }
...   },
...   {
...     $sort: { ratingCount: -1 }
...   },
...   {
...     $limit: 3
...   }
... ])
[
  {
    _id: ObjectId("6904e3a4f3a61a1ba8f7d017"),
    title: 'Foundation',
    genre: 'Science Fiction',
    ratingCount: 2,
    avgRating: 4.5
  },
  {
    _id: ObjectId("6904e3a4f3a61a1ba8f7d018"),
    title: 'I, Robot',
    genre: 'Science Fiction',
    ratingCount: 2,
    avgRating: 4.35
  },
  {
    _id: ObjectId("6904e3a4f3a61a1ba8f7d019"),
    title: "Harry Potter and the Philosopher's Stone",
    genre: 'Fantasy',
    ratingCount: 2,
    avgRating: 5
  }
]
BookVerseDB>
```

```
PS C:\Users\ashut\OneDrive\Desktop\Wipro NGA\Day_7 Mongodb 1\Day 7 Assessment> node App.js
Authors inserted: 3
Users inserted: 3
Books inserted: 2
PS C:\Users\ashut\OneDrive\Desktop\Wipro NGA\Day_7 Mongodb 1\Day 7 Assessment>
```

BookVerse MongoDB Assignment -

User Story 1: Database Setup and Data Modeling

Step 1: Create Database and Switch to It

```
use BookVerseDB
```

Step 2: Insert Sample Authors

```
db.Authors.insertMany([
  {
    _id: ObjectId(),
    name: "Isaac Asimov",
    nationality: "American",
    birthYear: 1920
  },
  {
    _id: ObjectId(),
    name: "J.K. Rowling",
    nationality: "British",
    birthYear: 1965
  },
  {
    _id: ObjectId(),
    name: "Brandon Sanderson",
    nationality: "American",
    birthYear: 1975
  }
])
```

Step 3: Insert Sample Users

```
db.Users.insertMany([
  {
    _id: ObjectId(),
    name: "Alice Johnson",
    email: "alice@example.com",
    joinDate: new Date("2024-08-15")
  },
  {
    _id: ObjectId(),
```

```
    name: "Bob Smith",
    email: "bob@example.com",
    joinDate: new Date("2024-09-20")
  },
  {
    _id: ObjectId(),
    name: "Carol White",
    email: "carol@example.com",
    joinDate: new Date("2024-10-01")
  }
])
```

Step 4: Get Author IDs for Books (Run this to see IDs)

```
db.Authors.find({}, {_id: 1, name: 1})
```

Step 5: Insert Sample Books (Replace authorId with actual IDs from above)

```
// First, store author IDs in variables for easier reference
var asimovId = db.Authors.findOne({name: "Isaac Asimov"})._id
var rowlingId = db.Authors.findOne({name: "J.K. Rowling"})._id
var sandersonId = db.Authors.findOne({name: "Brandon Sanderson"})._id

db.Books.insertMany([
  {
    _id: ObjectId(),
    title: "Foundation",
    genre: "Science Fiction",
    publicationYear: 1951,
    authorId: asimovId,
    ratings: [
      { user: "Alice Johnson", score: 5, comment: "Masterpiece of sci-fi!" },
      { user: "Bob Smith", score: 4, comment: "Great worldbuilding" }
    ]
  },
  {
    _id: ObjectId(),
    title: "I, Robot",
    genre: "Science Fiction",
    publicationYear: 2004,
    authorId: asimovId,
    ratings: [
      { user: "Carol White", score: 4.5, comment: "Thought-provoking" }
```

```
    ]
  },
  {
    _id: ObjectId(),
    title: "Harry Potter and the Philosopher's Stone",
    genre: "Fantasy",
    publicationYear: 1997,
    authorId: rowlingId,
    ratings: [
      { user: "Alice Johnson", score: 5, comment: "Magical and captivating!" },
      { user: "Carol White", score: 5, comment: "Best book ever!" }
    ]
  },
  {
    _id: ObjectId(),
    title: "Mistborn: The Final Empire",
    genre: "Fantasy",
    publicationYear: 2006,
    authorId: sandersonId,
    ratings: [
      { user: "Bob Smith", score: 4.8, comment: "Amazing magic system" },
      { user: "Alice Johnson", score: 4.5, comment: "Incredible plot twists" }
    ]
  },
  {
    _id: ObjectId(),
    title: "The Way of Kings",
    genre: "Fantasy",
    publicationYear: 2010,
    authorId: sandersonId,
    ratings: [
      { user: "Carol White", score: 5, comment: "Epic fantasy at its finest" },
      { user: "Bob Smith", score: 4.7, comment: "Long but worth it" }
    ]
  }
])
```

User Story 2: CRUD Operations

Task 1: Insert New Users and Books

// Insert a new user

```
db.Users.insertOne({
  name: "David Brown",
  email: "david@example.com",
  joinDate: new Date("2024-10-15")
})

// Insert a new book
db.Books.insertOne({
  title: "The Robots of Dawn",
  genre: "Science Fiction",
  publicationYear: 1983,
  authorId: asimovId,
  ratings: []
})
```

Task 2: Retrieve All Books of Genre "Science Fiction"

```
db.Books.find({ genre: "Science Fiction" })
```

Task 3: Update the Publication Year of One Book

```
db.Books.updateOne(
  { title: "Foundation" },
  { $set: { publicationYear: 1951 } }
)

// Verify the update
db.Books.findOne({ title: "Foundation" })
```

Task 4: Delete One User Record

```
db.Users.deleteOne({ name: "David Brown" })

// Verify deletion
db.Users.find()
```

Task 5: Add a New Rating to a Book Using $push

```
db.Books.updateOne(
  { title: "I, Robot" },
  {
    $push: {
      ratings: {
```

```
      user: "Bob Smith",
      score: 4.2,
      comment: "Interesting take on AI"
    }
  }
 }
)

// Verify the rating was added
db.Books.findOne({ title: "I, Robot" })
```

User Story 3: Querying and Filtering Data

Task 1: Retrieve All Books Published After 2015

```
db.Books.find({ publicationYear: { $gt: 2015 } })
```

Task 2: Find Authors Who Have Written Books in the "Fantasy" Genre

```
// Method 1: Using toArray() to fix the circular structure issue
var fantasyAuthorIds = db.Books.find(
 { genre: "Fantasy" },
 { authorId: 1 }
).toArray().map(book => book.authorId)

// Then find authors with those IDs
db.Authors.find({ _id: { $in: fantasyAuthorIds } })

// Method 2: Using aggregation pipeline (alternative approach)
db.Books.aggregate([
 { $match: { genre: "Fantasy" } },
 { $lookup: {
    from: "Authors",
    localField: "authorId",
    foreignField: "_id",
    as: "authorDetails"
 }},
 { $unwind: "$authorDetails" },
 { $group: {
    _id: "$authorDetails._id",
    name: { $first: "$authorDetails.name" },
    nationality: { $first: "$authorDetails.nationality" },
```

```
    birthYear: { $first: "$authorDetails.birthYear" }
  }}
])
```

Task 3: Retrieve All Users Who Joined Within the Last 6 Months

```
// Calculate date 6 months ago from today (Oct 31, 2024)
var sixMonthsAgo = new Date("2024-05-01")

db.Users.find({
  joinDate: { $gte: sixMonthsAgo }
})
```

Task 4: Find Books with an Average Rating Greater Than 4

```
db.Books.aggregate([
  {
    $project: {
      title: 1,
      genre: 1,
      publicationYear: 1,
      avgRating: { $avg: "$ratings.score" }
    }
  },
  {
    $match: {
      avgRating: { $gt: 4 }
    }
  }
])
```

Bonus Challenge

Bonus 1: Top 3 Most-Rated Books

```
db.Books.aggregate([
  {
    $project: {
      title: 1,
      genre: 1,
      ratingCount: { $size: "$ratings" },
```

```
      avgRating: { $avg: "$ratings.score" }
    }
  },
  {
    $sort: { ratingCount: -1 }
  },
  {
    $limit: 3
  }
])
```

Bonus 2: Node.js + Mongoose Script

```
// Install: npm install mongoose

const mongoose = require('mongoose');

// Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/BookVerseDB');

// Define Schemas
const authorSchema = new mongoose.Schema({
  name: String,
  nationality: String,
  birthYear: Number
});

const userSchema = new mongoose.Schema({
  name: String,
  email: String,
  joinDate: Date
});

const bookSchema = new mongoose.Schema({
  title: String,
  genre: String,
  publicationYear: Number,
  authorId: mongoose.Schema.Types.ObjectId,
  ratings: [{
    user: String,
    score: Number,
    comment: String
  }]
});
```

```
// Create Models
const Author = mongoose.model('Author', authorSchema);
const User = mongoose.model('User', userSchema);
const Book = mongoose.model('Book', bookSchema);

// Insert Data
async function insertData() {
  try {
    // Insert Authors
    const authors = await Author.insertMany([
      { name: "Isaac Asimov", nationality: "American", birthYear: 1920 },
      { name: "J.K. Rowling", nationality: "British", birthYear: 1965 },
      { name: "Brandon Sanderson", nationality: "American", birthYear: 1975 }
    ]);

    console.log('Authors inserted:', authors.length);

    // Insert Users
    const users = await User.insertMany([
      { name: "Alice Johnson", email: "alice@example.com", joinDate: new Date("2024-08-15") },
      { name: "Bob Smith", email: "bob@example.com", joinDate: new Date("2024-09-20") },
      { name: "Carol White", email: "carol@example.com", joinDate: new Date("2024-10-01") }
    ]);

    console.log('Users inserted:', users.length);

    // Insert Books
    const books = await Book.insertMany([
      {
        title: "Foundation",
        genre: "Science Fiction",
        publicationYear: 1951,
        authorId: authors[0]._id,
        ratings: [
          { user: "Alice Johnson", score: 5, comment: "Masterpiece!" }
        ]
      },
      {
        title: "Harry Potter and the Philosopher's Stone",
        genre: "Fantasy",
        publicationYear: 1997,
        authorId: authors[1]._id,
        ratings: [
```

```
        { user: "Bob Smith", score: 5, comment: "Magical!" }
      ]
    }
  ]);

  console.log('Books inserted:', books.length);

 } catch (error) {
  console.error('Error inserting data:', error);
 } finally {
  mongoose.connection.close();
 }
}

insertData();
```

Export Collections to JSON

To export collections, run these commands in your terminal (not MongoDB shell):

```
mongoexport --db=BookVerseDB --collection=Authors --out=Authors.json --jsonArray
mongoexport --db=BookVerseDB --collection=Books --out=Books.json --jsonArray
mongoexport --db=BookVerseDB --collection=Users --out=Users.json --jsonArray
```

Quick Verification Commands

```
// Check all collections
show collections

// Count documents
db.Authors.countDocuments()
db.Books.countDocuments()
db.Users.countDocuments()

// View all data
db.Authors.find().pretty()
db.Books.find().pretty()
db.Users.find().pretty()
```

Self-Evaluation Checklist

✅ Collections follow logical data modeling structure

- Authors, Books, and Users collections created with proper schema

✅ Used appropriate references and embedded documents

- Books reference Authors using authorId
- Ratings are embedded within Books

✅ CRUD operations performed successfully

- Create: insertOne, insertMany
- Read: find, findOne
- Update: updateOne with $set and $push
- Delete: deleteOne

✅ Filter and query operators applied correctly

- Used $gt, $gte, $in, $avg, $size operators
- Implemented aggregation pipeline

✅ Queries return expected results without syntax errors

- All queries tested and functional