**1. Write a MongoDB query to display at least 10 details of the documents of book id, Btitle, aut**

**Ans.**

Bookshop> db.books.find({}, { \_id: 0, book\_id: 1, B\_Title: 1, authors: 1 }).limit(10)

[

{ book\_id: 1, B\_Title: 'Book A', authors: [ 'Poet 1', 'Poet 2' ] },

{ book\_id: 2, B\_Title: 'Book B', authors: [ 'Poet 2', 'Poet 3' ] },

{ book\_id: 3, B\_Title: 'Book C', authors: [ 'Poet 1', 'Poet 4' ] },

{ book\_id: 4, B\_Title: 'coreJava Practice', authors: [ 'Poet 5' ] }

]

**2. Write a MongoDB query to display the fields Btitle, authors and price but exclude the bookid**

**Ans.**

Bookshop> db.books.find({}, { \_id: 0, B\_Title: 1, authors: 1, price: 1 })

[

{ B\_Title: 'Book A', authors: [ 'Poet 1', 'Poet 2' ], price: 500 },

{ B\_Title: 'Book B', authors: [ 'Poet 2', 'Poet 3' ], price: 1000 },

{ B\_Title: 'Book C', authors: [ 'Poet 1', 'Poet 4' ], price: 800 },

{ B\_Title: 'coreJava Practice', authors: [ 'Poet 5' ], price: 1500 }

]

**3. Write a MongoDB query to display the first 3 books based on similar authors.**

**Ans.**

Bookshop> db.books.find().sort({ authors: 1 }).limit(3)

[

{

\_id: ObjectId("653e04a64fdfed973a4ebd59"),

book\_id: 3,

B\_Title: 'Book C',

authors: [ 'Poet 1', 'Poet 4' ],

price: 800

},

{

\_id: ObjectId("653e04a64fdfed973a4ebd57"),

book\_id: 1,

B\_Title: 'Book A',

authors: [ 'Poet 1', 'Poet 2' ],

price: 500

},

{

\_id: ObjectId("653e04a64fdfed973a4ebd58"),

book\_id: 2,

B\_Title: 'Book B',

authors: [ 'Poet 2', 'Poet 3' ],

price: 1000

}

]

**4. Write a MongoDB query to find the book price more than 1500.**

**Ans.**

bookstore> db.books.find({ price: { $gt: 1500 } })

[

{

\_id: ObjectId("653ded5331f0eb623c45795f"),

book\_id: 4,

B\_Title: 'coreJava Practice',

authors: [ 'Poet 5' ],

price: 2000

},

{

\_id: ObjectId("653deea431f0eb623c457963"),

book\_id: 8,

B\_Title: 'coreJava Practice',

authors: [ 'Poet 4' ],

price: 2500

},

{

\_id: ObjectId("653def2731f0eb623c457965"),

book\_id: 10,

B\_Title: 'MySql Practice',

authors: [ 'Poet 1, Poet 9' ],

price: 2800

}

**]**

**5. Write a MongoDB query to find the book title who's price is more than 800 but less than 2000.**

**Ans**

Bookshop> db.books.find({price:{$gt:800,$lt:2000}},{B\_Title:1 })

[

{ \_id: ObjectId("653e04a64fdfed973a4ebd58"), B\_Title: 'Book B' },

{

\_id: ObjectId("653e04a64fdfed973a4ebd5a"),

B\_Title: 'coreJava Practice'

}

]

**6. Write a MongoDB query to arrange the book name of the bookstore in ascending order along with all the columns.**

**Ans.**

Bookshop> db.books.find().sort({ B\_Title: 1 })

[

{

\_id: ObjectId("653e04a64fdfed973a4ebd57"),

book\_id: 1,

B\_Title: 'Book A',

authors: [ 'Poet 1', 'Poet 2' ],

price: 500

},

{

\_id: ObjectId("653e04a64fdfed973a4ebd58"),

book\_id: 2,

B\_Title: 'Book B',

authors: [ 'Poet 2', 'Poet 3' ],

price: 1000

},

{

\_id: ObjectId("653e04a64fdfed973a4ebd59"),

book\_id: 3,

B\_Title: 'Book C',

authors: [ 'Poet 1', 'Poet 4' ],

price: 800

},

{

\_id: ObjectId("653e04a64fdfed973a4ebd5a"),

book\_id: 4,

B\_Title: 'coreJava Practice',

authors: [ 'Poet 5' ],

price: 1500

}

]

**7.Write a MongoDB query to arrange the book name of the bookstore in ascending order along with all the columns.**

**Ans.**

Bookshop> db.books.find().sort({ B\_Title: 1 })

[

{

\_id: ObjectId("653e04a64fdfed973a4ebd57"),

book\_id: 1,

B\_Title: 'Book A',

authors: [ 'Poet 1', 'Poet 2' ],

price: 500

},

{

\_id: ObjectId("653e04a64fdfed973a4ebd58"),

book\_id: 2,

B\_Title: 'Book B',

authors: [ 'Poet 2', 'Poet 3' ],

price: 1000

},

{

\_id: ObjectId("653e04a64fdfed973a4ebd59"),

book\_id: 3,

B\_Title: 'Book C',

authors: [ 'Poet 1', 'Poet 4' ],

price: 800

},

{

\_id: ObjectId("653e04a64fdfed973a4ebd5a"),

book\_id: 4,

B\_Title: 'coreJava Practice',

authors: [ 'Poet 5' ],

price: 1500

}

]

**8. Write a MongoDB query to know whether the author in the authors list.**

**Ans.**

Bookshop> db.books.find({ authors: "AuthorName" }).count() > 0

false

**9. Write a MongoDB query to update & increase the book price Rs.500 in all book name as "MongoDB Practice"**

**Ans.**

Bookshop> db.books.update({ B\_title: "MongoDB Practice" }, { $inc: { price: 500 } }, { multi: true })

DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.

{

acknowledged: true,

insertedId: null,

matchedCount: 0,

modifiedCount: 0,

upsertedCount: 0

}

**10. Write a MongoDB query to use Update modifiers ($Set, $push, $pull, $setOnInsert, $upsert) in bookstore.**

**$set**

Bookshop> db.books.update({ B\_title: "Book 1" }, { $set: { price: 1200 } })

{

acknowledged: true,

insertedId: null,

matchedCount: 0,

modifiedCount: 0,

upsertedCount: 0

}

**$push**

Bookshop> db.books.update({ B\_Title: "Book B" }, { $push: { authors: "New Author" } })

{

acknowledged: true,

insertedId: null,

matchedCount: 1,

modifiedCount: 1,

upsertedCount: 0

}

**$pull**

Bookshop> db.books.update({ B\_title: "Book C" }, { $pull: { authors: "Poet 1" } })

{

acknowledged: true,

insertedId: null,

matchedCount: 0,

modifiedCount: 0,

upsertedCount: 0

}

**$setOnInsert**

Bookshop> db.books.update({ B\_Title: "New Book" }, { $setOnInsert: { publication\_year: 2023 } }, { upsert: true })

{

acknowledged: true,

insertedId: ObjectId("653e146abe46c5904d43d471"),

matchedCount: 0,

modifiedCount: 0,

upsertedCount: 1

}

#### $Upsert

Bookshop> db.books.update({ B\_Title: "Non-Existent Book" }, { $set: { price: 900 } }, { upsert: true })

{

acknowledged: true,

insertedId: ObjectId("653e14f7be46c5904d43d493"),

matchedCount: 0,

modifiedCount: 0,

upsertedCount: 1

}