**QUESTION-1: Operations on the Students Collection**

**a) Retrieve all students whose age is greater than 15 and less than 18.**

db.Students.find({

age: { $gt: 15, $lt: 18 }

});

**b) Find students who are enrolled in either "Mathematics" or "Science" as one of their subjects.**

db.Students.find({

subjects: { $in: ["Mathematics", "Science"] }

});

**c) Identify students who have scored more than 90 in at least one of their grades.**

db.Students.find({

grades: { $elemMatch: { $gt: 90 } }

});

**d) Sort the students based on their age in descending order and limit the result to the top 5 records.**

db.Students.find()

.sort({ age: -1 })

.limit(5);

**e) Group students by their address city and calculate the total number of students residing in each city.**

db.Students.aggregate([

{

$group: {

\_id: "$address.city", // Grouping by the city field in address

totalStudents: { $sum: 1 } // Counting the number of students per city

}

}

]);

**QUESTION-2: Operations on the Teachers Collection**

**a) Retrieve teachers who have more than 10 years of experience but do not teach "History."**

db.Teachers.find({

experience: { $gt: 10 },

subject: { $ne: "History" }

});

**b) Find teachers who are assigned to teach both "Class A" and "Class B."**

db.Teachers.find({

classes: { $all: ["Class A", "Class B"] }

});

**c) Display the names and contact details of teachers whose contact information includes a phone number starting with "91."**

db.Teachers.find(

{ "contact.phone": { $regex: "^91" } },

{ name: 1, "contact": 1 }

);

**d) Create an index on the subject field to ensure that no two teachers can have the same subject.**

db.Teachers.createIndex({ subject: 1 }, { unique: true });

**e) Use aggregation to project only the teacher's name, subject, and experience, and then filter out teachers with less than 5 years of experience.**

db.Teachers.aggregate([

{

$project: {

name: 1,

subject: 1,

experience: 1

}

},

{

$match: {

experience: { $gte: 5 }

}

}

]);