**QUESTION SET-2**

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

**a) Retrieve all products with a price greater than 500 and less than 1000.**

db.Products.find({

price: { $gt: 500, $lt: 1000 }

});

**b) Find products that belong to either the "Electronics" or "Home Appliances" category.**

db.Products.find({

categories: { $in: ["Electronics", "Home Appliances"] }

});

**c) Identify products that have received a rating of 4 or higher from at least one customer.**

db.Products.find({

ratings: { $elemMatch: { $gte: 4 } }

});

**d) Sort the products in ascending order of price and skip the first 10 records.**

db.Products.find()

.sort({ price: 1 })

.skip(10);

**e) Group products by their category and calculate the average price of products in each category.**

db.Products.aggregate([

{

$unwind: "$categories" // Deconstruct the categories array into separate documents

},

{

$group: {

\_id: "$categories", // Grouping by category

avgPrice: { $avg: "$price" } // Calculating the average price for each category

}

}

]);

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

**a) Retrieve customers who have placed more than 5 orders but do not have "Electronics" as a preference.**

db.Customers.find({

"orders.5": { $exists: true }, // Ensures there are more than 5 orders (at least index 5 exists)

preferences: { $ne: "Electronics" }

});

**b) Find customers who have ordered both "Laptop" and "Smartphone" in their order history.**

db.Customers.find({

orders: { $all: ["Laptop", "Smartphone"] }

});

**c) Display the names and email addresses of customers whose location includes a city named "New York."**

db.Customers.find(

{ "location.city": "New York" },

{ name: 1, email: 1 }

);

.

**d) Create a compound index on the email and location.city fields to ensure uniqueness for each customer.**

db.Customers.createIndex(

{ email: 1, "location.city": 1 },

{ unique: true }

);

**e) Use aggregation to project only the customer's name, email, and total number of orders, and then filter out customers with fewer than 3 orders.**

db.Customers.aggregate([

{

$project: {

name: 1,

email: 1,

totalOrders: { $size: "$orders" } // Counting the total number of orders

}

},

{

$match: {

totalOrders: { $gte: 3 } // Filtering customers with fewer than 3 orders

}

}

]);