Q1

A:List all products in the inventory.
Ans:-db.produts.find();

O/p: { _id: ObjectId("65f40820829dae7da0aa33f5"), name: 'Product 1', price: 100}

{_id: ObjectId("65f40820829dae7da0aa33f6"),name: 'Product 2',price: 200}

{_id: ObjectId("65f40820829dae7da0aa33f7"),name: 'Product 3',price: 300}

{_id: ObjectId("65f40820829dae7da0aa33f8"),name: 'Product 4',price: 400}

{_id: ObjectId("65f40820829dae7da0aa33f7"),name: 'Product 3',price: 300 }

B:List the details of orders with a value >20000.
{_id:ObjectId("65f40a00829dae7da0aa33ff"),customer:'Mr.'Product2'],total:30000,processed: true}

{ _id: ObjectId("65f40a00829dae7da0aa3400"),customer: 'Ms. Sarah',products: ['Product 3','Product 4'],total: 25000,processed: true}

{_id: ObjectId("65f40a00829dae7da0aa3401"),customer: 'Mr. John',products:

['Product 2','Product 5'],total: 15000,processed: false}

{_id: ObjectId("65f40a00829dae7da0aa3402"),customer: 'Ms. Emily',products: ['Product 1','Product 3'], total: 18000,processed: false }

{_id: ObjectId("65f40a00829dae7da0aa3403"),customer: 'Mr. David', products: ['Product 4', 'Product 5'],total: 22000,processed: true}

C: List all the orders which has not been processed (invoice not generated)

db.orders.find({ "status": { $ne: "processed" } })

D: List all the orders along with their invoice for “Mr. Rajiv”.

db.orders.aggregate([{$match: {"customer_id": db.customers.findOne({ "name": "Mr. Rajiv" }).customer_id } }, { $lookup: {from: "invoices",localField: "order_id", foreignField: "order_id", as: "invoice"} }])

Q2.4. Answer the following Queries.

A: List the names of product whose warranty period is one year .
db.products.find({ "warranty_period": "1 year" });

o/p: {_id: ObjectId("65f40a7051210f2732bb9083"), product_id: 1, name: 'Product A', brand: 'Brand X', price: 100, warranty_period: '1 year', ratings: [4, 5, 3,4,5]}

{_id: ObjectId("65f40f4551210f2732bb9086"), product_id: 1,name: 'Laptop', brand: 'Dell', price:1000, warranty_period: '1 year',ratings: [4, 5, 4, 3, 5]}

{_id: ObjectId("65f40f4551210f2732bb9088"), product_id: 3, name: 'Headphones', brand: 'Sony', price: 100, warranty_period: '1 year', ratings: [4, 3,5, 4, 4]}

{ _id: ObjectId("65f40f4551210f2732bb9088"),product_id: 3, name: 'Headphones', brand: 'Sony', price: 100,warranty_period: '1 year', ratings: [4, 3, 5, 4, 4]}

B: List the customers has done purchase on “15/08/2023”.
db.orders.aggregate([{ $match: { "purchase_date": "2023-08-15" } }, { $lookup: { from: "customers", localField: "customer_id", foreignField: "customer_id", as:"customer" } }, { $project: {"customer.name": 1,"_id": 0 }}])

o/p: { customer: [{ name: 'John Doe'}]} { customer: [{name: 'Jane Smith' }]}

C: Display the names of products with brand which have highest rating.
db.products.aggregate([{ $group: {_id: "$brand", avgRating: { $avg: "$ratings" } } }, { $sort: { avgRating: -1 } }, { $limit: 1 }, { $lookup: {from: "products", localField: "_id", foreignField: "brand", as: "products” } }, { $unwind: "$products" }, { $project: { "product_name": "$products.name","brand": "$products.brand", "_id": 0 } }])

o/p: { product_name: 'Smartphone', brand: 'Samsung' }

D: Display customers who stay in …… city and billamt >50000 .
db.orders.aggregate([{ $match: { "bill_amount": { $gt: 50000 } } },{ $lookup: { from: "customers", localField: "customer_id", foreignField: "customer_id", as: "customer” } },

{ $unwind: "$customer" }, { $match: {"customer.city": "Houston" // Specify the desired city } }, { $project: { "customer_name": "$customer.name", "_id": 0 } }])

O/p: { customer_name: 'Emily Brown' }

Q3. 4. Answer the following Queries.

A: List all customers whose name starts with 'D' character
db.customers.find({ "name": { $regex: "^D", $options: "i" } });

o/p: {_id: ObjectId("65f420f1c8c6af7b74ef1daa"), customer_id: 1, name: 'Doe Jhon', address: '123 Main Street', city: 'Pimpri'}

B: List the names of customer in descending order who has taken a loan from Pimpri city.
db.customers.aggregate([{$match: { "city": "Pimpri" } }, { $lookup: {from:"loans", localField: "customer_id", foreignField: "customer_id",as: "loan" } }, { $match: { "loan": { $ne: [] } // Filter customers who have taken a loan} },
{$project: { "name": 1,"_id": 0 } }, { $sort: { "name": -1 }}])

o/p: {name: 'Doe Jhon'}

C: Display customer details having maximum loan amount.
db.loans.aggregate([{$group: {_id: "$customer_id", max_loan_amount: { $max: "$loan_amount" }} }, { $sort: { max_loan_amount: -1 }},{ $limit: 1},{$lookup: from: "customers",localField: "_id", foreignField: "customer_id", as: "customer" } },{ $unwind: "$customer"},{ $project: {"customer_name": "$customer.name", "address": "$customer.address", "city": "$customer.city", "max_loan_amount": 1, "_id": 0 } }])

o/p: {max_loan_amount: 200000, customer_name: 'Doe Jhon', address: '123 Main Street', city: 'Pimpri'}

D: Update the address of customer whose name is “Mr. Patil” and loan_amt is greater than 100000.
db.customers.updateMany({"name": "Mr. Patil", "customer_id": { $in: db.loans.find({ "loan_amount": { $gt: 100000 } }).map(loan => loan.customer_id) }},{$set: {"address": "New Address"} })