✅ Task 1: Retrieve all products.

-> db.Products.find()

✅ Task 2: Find a product by category (e.g., "Footwear").

-> db.Products.find("Category": "Footwear"})

✅ Task 3: Find a product with a price greater than $100.

->db.Products.find({ "Price": { $gt: 100 } })

✅ Task 4: Update the stock quantity of a product (e.g., increase by 10).

-> db.Products.updateOne({ \_id: ObjectId('67ad8cb38c341e5f2475ff91') }, { $inc: { "Inventory.StockQuantity": 10 } } )

✅ Task 5: Add a new size to "Men's Running Shoes".

-> db.Products.updateOne( { Name: "Men's Running Shoes" }, { $push: { Sizes: "XXL" } })

✅ Task 6: Remove a specific size ("M") from "Laptop Backpack".

-> db.Products.updateOne({ "Name": "Laptop Backpack" }, { $pull: { Sizes: "M" } } )

✅ Task 7: Delete a product by name (e.g., "Smartwatch").

-> db.Products.deleteOne({ Name: "Smartwatch" })

✅ Task 8: Find products that have more than 3 sizes available.

-> db.Products.find({ $expr: { $gt: [ { $size: "$Sizes" }, 3 ] } })

✅ Task 9: Find products with a discount greater than 20%.

-> db.Products.find({ "Discount.DiscountPercentage": { $gt: 20 } })

✅ Task 10: Find products with stock less than 50.

-> db.Products.find({ "Inventory.StockQuantity": { $lt: 50 } })

✅ Task 11: Find products that have no discount applied.

-> db.Products.find({ $or: [{ "Discount": { $exists: false } },{ "Discount.DiscountPercentage": { $lte: 0 } } ] })

✅ Task 12: Find products that have a warehouse location starting with "A".

-> db.Products.find({ "Inventory.WarehouseLocation": { $regex: /^A/i } })

✅ Task 13: Find products that have "M" as a size.

-> db.Products.find({ "Sizes": "M" })

✅ Task 14: Find the total stock of all products.

-> db.Products.aggregate([ {$group: { \_id: null, totalStock: { $sum: "$Inventory.StockQuantity" } }}])

✅ Task 15: Find the average price of all products in the "Electronics" category.

-> db.Products.aggregate([{ $match: { "Category": "Electronics" } }, { $group: { \_id: "Electronics", averagePrice: { $avg: "$Price" } } }])

✅ Task 16: Find the most expensive product in each category.

-> db.Products.aggregate([{$group: {\_id: "$Category", MaxPrice :{$max: "$Price"} } }]);

✅ Task 17: Count the number of products in each category.

-> db.Products.aggregate([ { $group: { \_id: "$Category", productCount: { $sum: 1 } } }])

✅ Task 18: Get the top 3 most expensive products.

-> db.Products.aggregate([ { $sort: { Price: -1 } }, { $limit: 3 } ])

✅ Task 19: Get the total revenue if all products were sold (Price × StockQuantity).

-> db.Products.aggregate([{ $project: { revenue: { $multiply: ["$Price", "$Inventory.StockQuantity"] } } },

{ $group: { \_id: null, totalRevenue: { $sum: "$revenue" } }}])

✅ Task 20: Bulk insert 3 new products into the collection.

-> db.Products.insertMany([{},{},{}])

✅ Task 21: Bulk update all products to add a new field (Brand).

-> db.Products.updateMany( {}, { $set: { Brand: "Puma" } } )

✅ Task 22: Bulk delete all products in the "Accessories" category.

-> db.Products.deleteMany({ Category: "Accessories" })

✅ Task 23: Create an index on the Category field.

-> db.Products.createIndex({ Category: 1 })

✅ Task 24: Create a compound index on Price and StockQuantity.

-> db.products.createIndex({ Price: 1, "Inventory.StockQuantity": 1 })

✅ Task 25: Use .explain("executionStats") to analyze a query’s performance.

-> db.Products.find({ Price: { $gt: 100 } }).explain("executionStats")

✅ Task 26: Find products with either a high discount (> 25%) or low stock (< 10).

-> db.Products.find({$or: [ { "Discount.DiscountPercentage": { $gt: 25 } },{ "Inventory.StockQuantity": { $lt: 10 } }]})

✅ Task 27: Find products where the discount expires within the next 7 days.

->db.Products.find({"Discount.ValidUntil": { $gte: new Date(),$lte: new Date(new Date().getTime() + 7 \* 24 \* 60 \* 60 \* 1000)}})

✅ Task 28: Find products that have a discount, are in stock, and cost more than $50.

-> db.Products.find({ "Discount.DiscountPercentage": { $gt: 0 },"Inventory.StockQuantity": { $gt: 0 },"Price": { $gt: 50 }})

✅ Task 29: Find products that have at least one size and a discount.

-> db.Products.find({ "Sizes": { $ne: [] }, "Discount.DiscountPercentage": { $gt: 0 }

✅ Task 30: Find products available in multiple warehouse locations.

->db.Products.find({ "Inventory.WarehouseLocation.1": { $exists: true } })