```
// use("practice_db");

// Q1. Display total sales by region along with average profit
// Thinking
// Group needed → aggregation

// Two accumulators → $sum, $avg
db.orders.aggregate([
    {
        $group: {
            _id: "$region",
            totalSales: { $sum: "$sales" },
            avgProfit: { $avg: "$profit" }
        }
    },
    {
        $project: {
            _id: 0,
            region: "$_id",
            totalSales: 1,
            avgProfit: 1
        }
    }
]);

// Q2.Find the top 3 products by total quantity sold
// Thinking
// Sum qty per product

// Sort desc

// Limit 3
db.orders.aggregate([
    {
        $group: {
            _id: "$productName",
            totalQty: { $sum: "$qty" }
        }
    },
    { $sort: { totalQty: -1 } },
    { $limit: 3 },
    {
        $project: {
            _id: 0,
            productName: "$_id",
            totalQty: 1
        }
    }
]);

// Q3.Count number of orders per customer
//  Thinking
```

```
// Each document = 1 order line

// Group + sum(1)
db.orders.aggregate([
    {
        $group: {
            _id: "$customerName",
            totalOrders: { $sum: 1 }
        }
    }
]);

// Q4.Display customers whose total profit is negative
// Thinking

// First aggregate profit per customer

// Then filter
db.orders.aggregate([
    {
        $group: {
            _id: "$customerName",
            totalProfit: { $sum: "$profit" }
        }
    },
    {
        $match: {
            totalProfit: { $lt: 0 }
        }
    }
]);

// Q5.Add a field orderValue = sales × quantity
// Thinking

// No grouping

// Computed field → $addFields
db.orders.aggregate([
    {
        $addFields: {
            orderValue: { $multiply: ["$sales", "$qty"] }
        }
    },
    {
        $project: {
            _id: 0,
            orderId: 1,
            orderValue: 1
        }
    }
]);
```

```
// Q6.Show first and last order date for each customer
//  Thinking

// $first / $last need sorting

// Group by customer
db.orders.aggregate([
    { $sort: { orderDate: 1 } },
    {
        $group: {
            _id: "$customerName",
            firstOrder: { $first: "$orderDate" },
            lastOrder: { $last: "$orderDate" }
        }
    }
]);


// Q7.Count number of unique customers per region
// Thinking

// Unique → $addToSet

// Then $size
db.orders.aggregate([
    {
        $group: {
            _id: "$region",
            customers: { $addToSet: "$customerName" }
        }
    },
    {
        $project: {
            _id: 0,
            region: "$_id",
            customerCount: { $size: "$customers" }
        }
    }
]);

// Q8.Display number of orders where color is Red or Black
// Thinking

// Filter first

// Count documents
db.orders.aggregate([
    {
        $match: {
            color: { $in: ["Red", "Black"] }
        }
    },
    { $count: "totalOrders" }
]);
```

```
// Q9.Display customer name and cities they ordered from(no duplicates)
// Thinking

// One customer → many cities

// Avoid duplicates → $addToSet
db.orders.aggregate([
    {
        $group: {
            _id: "$customerName",
            cities: { $addToSet: "$city" }
        }
    },
    {
        $project: {
            _id: 0,
            customerName: "$_id",
            cities: 1
        }
    }
]);

// Q10.Which region–segment combination has minimum total profit
// Thinking

// Group by 2 fields

// Sort asc

// Limit 1
db.orders.aggregate([
    {
        $group: {
            _id: {
                region: "$region",
                segment: "$segment"
            },
            totalProfit: { $sum: "$profit" }
        }
    },
    { $sort: { totalProfit: 1 } },
    { $limit: 1 }
]);
```