```
// db.collection.aggregate([
//     { stage1 },
//     { stage2 },
//     { stage3 }
// ])

// 3️⃣ MOST IMPORTANT: Order of Stages(Interview Question)
// ☑️ GENERAL & BEST ORDER
// $match → $unwind → $group → $project → $sort → $limit

// Why this order ?

//     Filter early → improves performance

// Unwind before grouping

// Group before calculating

// Project near the end

// Sort & limit last

// ⚠️ You can change order, but this is best practice

// Requirement  Use
// Filter rows  $match
// Break arrays $unwind
// Group & calculate    $group
// Rename / select fields   $project
// Order results    $sort
// Top N results    $limit
// Count    $sum: 1
// Total    $sum
// Average  $avg
// Multiply $multiply

//  ◇ $multiply

// Used inside $project or $group

// totalCost: { $multiply: ["$price", "$quantity"] }


// practice

// 1.find total profit per region,sorted descending
db.orders.aggregate([
    // for safety we can check if region is there or not
    // {$match:{"customer.region":{$exists:true}}}
    { $group: { _id: "$customer.region", totalProfit: {$sum: "$profit" } } },
    { $sort: { totalProfit: -1 } },
```

```
    {$project:{_id:0,region:"$_id",totalProfit:1}}
]);

// 2. find top 3 customers by total profit
db.orders.aggregate([
    { $group: { _id: "$customer.name", totalProfit: { $sum: "$profit" } } },
    { $project: { _id: 0, customerName: "$_id", totalProfit: 1 } },
    { $sort: { totalProfit: -1 } },
    { $limit: 3 }
]);

// 3. find average order profit per segment, but only for paid users
db.orders.find();

db.orders.aggregate([
    { $match: { "payment.status": "Paid" } },
    { $group: { _id: "$customer.segment", avgOrderProfit: { $avg: "$profit" } } },
    { $project: { _id: 0, segment: "$_id", avgOrderProfit: 1 } }
]);

// 4. find subCategory wise total quantity sold
db.orders.aggregate([
    {$unwind:"$items"},
    { $group: { _id: "$items.subCategory", totalQtySold: { $sum: "$items.quantity"
} } },
    { $project: { _id: 0, subcategory: "$_id", totalQtySold: 1 } }
]);

// 5. find products brought by more than 1 unique customer
db.orders.aggregate([
    { $unwind: "$items" },
    {
        $group: {
            _id: "$items.product",
            uniqueCustomers: { $addToSet: "$customer.id" }
        }
    },
    {
        $project: {
            _id: 0,
            product: "$_id",
            customerCount: { $size: "$uniqueCustomers" }
        }
    },
    {
        $match: {
            customerCount: { $gt: 1 }
        }
    }
]);

// Arrays (+$unwind )
// 6. List each orderid with individual products and their category
db.orders.find();
```

```
db.orders.aggregate([
    { $unwind: "$items" },
    {$project:{_id:0,orderId:1,"items.product":1,"items.category":1}}
])

// 7. find total revenue per category (sum of price*qty per category)
db.orders.aggregate([
    { $unwind: "$items" },
    {
        $group: {
            _id: "$items.category",
            totalRevenue: { $sum: { $multiply: ["$items.price", "$items.quantity"]
} }
        }
    },
    { $project: { _id: 0, category: "$_id", totalRevenue: 1 } }
]);

// 8. which subCategory contributes the least profit
db.orders.aggregate([
    { $unwind:  "$items" },
    { $group: { _id: "$items.subCategory", totalProfit: { $sum: "$profit" } } },
    { $sort: { totalProfit: 1 } },
    { $limit: 1 },
    {$project:{_id:0,subCategory:"$_id",totalProfit:1} }
]);
// this is wrong bcz here unwind splits,the items at order level becomes twice

// crct way
db.orders.aggregate([
    // here $items is converted to object after unwind
    // to before unwinding store item count
    {
        $addFields: {
        itemCount:{$size:"$items"}
    }},
    { $unwind: "$items" },
    {
        $group: {
            _id: "$items.subCategory",
            totalProfit: {
                $sum: {
                    $divide: ["$profit", "$itemCount"]
                }
            }
        }
    },
    { $sort: { totalProfit: 1 } },
    { $limit: 1 },
    {
        $project: {
            _id: 0,
            subCategory: "$_id",
```

```
            totalProfit: { $round: ["$totalProfit", 2] }
        }
    }
]);

// above can be done by using push and unwind,but here unwind should be used twice
db.orders.aggregate([
    {
        $group: {
            _id: "$items.subCategory",
            items: { $push: "$items" },
            totalProfit: { $sum: "$profit" }
        }
    },
    { $unwind: "$items" },
    { $unwind: "$items"},
    { $sort: { totalProfit: 1 } },
    { $limit: 1 },
    { $project: { _id: 0, subCategory: "$_id", totalProfit: 1 } }
])

db.orders.find();

// 9. find how many times each product was sold
// 2 meanings 1. Total units sold(w.r.t quantity)
// 2. Number of orders in which product appears
db.orders.aggregate([
    { $unwind: "$items" },
    { $group: { _id: "$items.product", totalSold: { $sum: "$items.quantity" } } },
    { $project: { _id: 0, product: "$_id", totalSold: 1 } }
]);
// other(this is secondary,but mostly 1st one is crct)
db.orders.aggregate([
    { $unwind: "$items" },
    {
        $group: {
            _id: "$items.product", orderCount: { $sum: 1 }
        }
    },
    { $project: { _id: 0, product: "$_id", orderCount: 1 } }
]);

db.orders.find();

// 10. Find orders that contain both Furniture and Electronics.

db.orders.aggregate([
    { $unwind: "$items" },
    {
        $group: {
            _id: "$orderId",
            categories: { $addToSet: "$items.category" }
        }
    },
```

```
    {
        $match: {
            categories: { $all: ["Furniture", "Electronics"] }
        }
    },
    {
        $project: {
            _id: 0,
            orderId: "$_id",
        }
    }
]);

db.orders.find();

// push vs $addToSet
//11. For each customer, push all products bought (allow duplicates).
db.orders.aggregate([
    {$unwind:"$items"}, // we should do unwind bcz it stores array in array
    { $group: { _id: "$customer.name", products: { $push: "$items.product" } } },
    { $project: { _id: 0, customerName: "$_id", products: 1 } }
]);

// 12. For each customer, addToSet categories bought.
db.orders.aggregate([
    {$unwind:"$items"}, // if we dont use array in arrays are created(also
duplicates too)
    { $group: { _id: "$customer.name", categories: { $addToSet: "$items.category"
} } },
    { $project: { _id: 0, customerName: "$_id", categories: 1 } }
]);

// 13. Find customers who bought more than 2 unique subCategories.

db.orders.aggregate([
    { $unwind: "$items" },
    { $group: { _id: "$customer.name", uniqueSubCategories: { $addToSet:
"$items.subCategory" } } },
    { $project: { _id: 0, customerName: "$_id", uniqueCount: { $size:
"$uniqueSubCategories" } } },
    { $match: { uniqueCount: { $gt: 2 } } }
]);

// DATE-BASED AGGREGATION (VERY IMPORTANT)

// 14. Find month-wise total profit.
db.orders.aggregate([
    {
        $group: {
            _id: {
                month: { $month: "$orderDate" }
            },
            totalProfit: { $sum: "$profit" }
        }
```

```
    },
    {
        $project: {
            _id: 0, month: "$_id.month", totalProfit: 1
        }
    }
]);

db.orders.find();

// 15. Find orders placed in January only, grouped by region.

db.orders.aggregate([
    // 1.match, expr allows us to use aggregation expressions in $match
    {
        $match: {
            $expr: { $eq: [{ $month: "$orderDate" }, 1] } // 1- January
        }
    },
    {
        $group: {
            _id: "$customer.region",
            totalOrders: { $sum: 1 },// optional count of orders
        }
    },
    {
        $project: {
            _id: 0, region: "$_id", totalOrders: 1
        }
    }
]);

// 16.Find average profit per day.
db.orders.aggregate([
    {
        $group: {
            // _id: {
            //     day: { $dayOfMonth: "$orderDate" } // dayofmonth - for only one
month

            // },
            // so we use dateToString
            _id:{$dateToString:{format:"%Y-%m-%d",date:"$orderDate"}},
            avgProfit: { $avg: "$profit" }
        }
    },
    {
        $project: {
            _id: 0,
            Day: "$_id",
            avgProfit: 1
        }
    }
]);
```

```
// 17. Find customers who placed orders in both Jan and Feb.
db.orders.aggregate([
    // first group and unique months per customer
    // before that extract month number
    {
        $addFields: {
            month: { $month: "$orderDate" }
        }
    },
    { $group: { _id: "$customer.name", months: { $addToSet: "$month" } } },
    {
        $match: {
            months: { $all: [1, 2] } // jan & feb
        }
    },
    {
        $project:
        {
            _id: 0,
            customerName: "$_id",
            months: 1
        }
    }
]);

// 18. Find top 2 most profitable days.
db.orders.aggregate([
    {
        $group: {
            _id: {
                $dateToString: { format: "%Y-%m-%d", date:"$orderDate" }
            },
            totalProfit: { $sum: "$profit" }
        }
    },
    { $sort: { totalProfit: -1 } },
    {
        $project: {
            _id: 0, day: "$_id", totalProfit: 1
        }
    },
    {$limit:2}
]);

// 19. From Corporate segment, find top 3 highest profit orders, but only for
Electronics category.
db.orders.aggregate([
    // 1Only Corporate segment
    { $match: { "customer.segment": "Corporate" } },

    // 2️⃣ Only orders that contain Electronics items
    { $match: { "items.category": "Electronics" } },
```

```
    //  Sort by profit descending
    { $sort: { profit: -1 } },

    // 4️⃣ Limit top 3
    { $limit: 3 },

    //  Project clean output
    {
        $project: {
            _id: 0,
            orderId: 1,
            customerName: "$customer.name",
            profit: 1
        }
    }
])
```