Quick mongodb different queries

🧠 **Decision rule (remember this)**

Question wording	Use
how many times sold	$sum: "$items.quantity"
how many orders	$sum: 1
unique customers	$addToSet + $size

If the question says *unique / distinct / different / no duplicates* → use $addToSet inside $group.

Examples of trigger words:

- unique customers
- distinct products
- different categories
- customers who bought **both**
- count of **distinct** users

👉 Brain should auto-think: **$addToSet**

✅ **$addToSet**

- Removes **duplicates**
- Counts **entities**

uniqueProducts: { $addToSet: "$items.product" }

Use when:

- “unique customers”
- “distinct products”
- “how many different categories”

🔥 **Common interview trap**

Q: “Total number of customers per product”

- If **unique customers** → $addToSet + $size
- If **total purchases** → $sum: 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} }
]);
```

```
// 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] }
        }
    }
]);
```

```
// 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 } }
]);
```

🧠 Decision table (save this mentally)

Goal	Order
Group by array field	`$unwind → $group`
Sum / avg item values	`$unwind → $group`
Flatten grouped results	`$group → $unwind`
Create list then explode	`$group → $unwind`

VERY IMPORTANT CONCEPT (remember this forever)

When do we need this `$addFields + divide` trick?

Profit stored at	Need divide by itemCount?
Order level	✅ YES
Item level	❌ NO

If profit was already inside items (simpler case)

JavaScript

```
{
  items: [
    { product: "Pen", subCategory: "Stationery", profit: 10 },
    { product: "Book", subCategory: "Stationery", profit: 20 }
  ]
}
```

Then your query becomes 👇

```
// 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
        }
    }
]);
```

```
// 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
        }
    }
]);
```

💡 **Rule of thumb:**

To check multiple values in the same group → $addToSet + $match: { $all: [...] }

```
// 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" } },

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

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

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

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

Multiple Group by

```
// 4. For each category + region, find:
// number of orders
//     (One order counts once even if it has multiple items of same category)

// here we need to count unique orders per category+region
// so first unwind, then group by region+category+orderid -- this ensures one order is counted once
// next group again by category+region
db.orders.aggregate([
    { $unwind: "$items" },
    {
        $group: {
            _id: {
                category: "$items.category",
                region: "$customer.region",
                orderId: "$orderId"
            }
        }
    },

    // count orders per category+region
    {
        $group: {
            _id: {
                category: "$_id.category",
                region: "$_id.region"
            },
            orderCount: { $sum: 1 }
        }
    },
    {
        $project: {
            _id: 0,
            category: "$_id.category",
            region: "$_id.region",
            orderCount: 1
        }
    }
]);
```

🧠 Key rules you just learned (VERY IMPORTANT)

🔹 Rule 1: `$all` vs `$in`

Operator	Use when
`$in`	field equals **any one** value
`$all`	array contains **all values**

👉 Here you needed `$in`, not `$all`.

🔹 Rule 2: Filter before grouping

If you know what you want to include:

- `$match` → early
- `$group` → later

This avoids wrong totals and improves performance.

```
// 5. for each region+segment+category find
// total profit,only include electronics and furniture
db.orders.aggregate([
    { $unwind: "$items" },
    {
        $match: {
            "items.category": { $in:["Electronics","Furniture"]}
    }},
    {
        $group: {
            _id: {
                region: "$customer.region",
                segment: "$customer.segment",
                category: "$items.category"
            },
            totalProfit: { $sum: "$profit" }
        }
    },
    {
        $project: {
            _id: 0,
            region: "$_id.region",
            segment: "$_id.segment",
            category: "$_id.category",
            totalProfit: 1
        }
    }
]);
```

```
// 6. For each year + month + region, find:
// total orders
// Sort by year, then month.

db.orders.aggregate([
    {
        $group: {
            _id: {
                year: { $year: "$orderDate" },
                month: { $month: "$orderDate" },
                region: "$customer.region"
            },
            totalOrders: { $sum: 1 }
        }
    },
    {
        $sort: {
            "_id.year": 1,
            "_id.month":1
        },
    },
    {
        $project: {
            _id: 0, year: "$_id.year", month: "$_id.month", region: "$_id.region", totalOrders: 1
        }
    }
]);
```

🧠 Rule to remember (VERY IMPORTANT)

Requirement says	Use
"how many **rows / records**"	`$sum: 1`
"how many **items / units / quantity**"	`$sum: "$items.quantity"`

Interviewers **expect you to notice this.**

```
// Q10 For each region + month, find:
// total profit
// Then return only the top month per region.

db.orders.aggregate([
    {
        $group: {
            _id: {
                region: "$customer.region",
                month: { $month: "$orderDate" }
            },
            totalProfit: { $sum: "$profit" }
        }
    },
    // Sort months by profit (highest first) per region
    {
        $sort: {
            "_id.region": 1,
            totalProfit: -1
        }
    },
    // Pick top month per region
    {
        $group: {
            _id: "$_id.region",
            topMonth: { $first: "$_id.month" },
            maxProfit: { $first: "$totalProfit" }
        }
    },
    {
        $project: {
            _id: 0,
            region: "$_id",
            topMonth: 1,
            maxProfit: 1
        }
    }
]);
```