✅ $count

Yes ✔
$count **counts documents flowing through the pipeline at that stage**

Example:

```
db.orders.aggregate([
 { $match: { region: "West" } },
 { $count: "totalOrders" }
]);
```

Equivalent to:

```
{ $group: { _id: null, totalOrders: { $sum: 1 } } }
```

Now let's move step-by-step 👇

1️⃣ $lookup — JOIN in MongoDB

🧠 What it does

Joins **two collections** (like SQL JOIN)

📌 Basic syntax

```
{
 $lookup: {
   from: "otherCollection",
   localField: "fieldInThisCollection",
   foreignField: "fieldInOtherCollection",
   as: "joinedData"
 }
}
```

🗄️ Example collections

orders

{

orderId: "O1",

customerId: 101,

sales: 200

}

customers

{

customerId: 101,

customerName: "Alice",

city: "Dallas"

}

❓ **Question**

👉 *Display orderId, sales, customerName, city*

✍️ **How I think**

1. Main collection → orders
2. Need customer details → $lookup
3. Join on customerId

✅ **Solution**

```
db.orders.aggregate([
 {
  $lookup: {
   from: "customers",
   localField: "customerId",
   foreignField: "customerId",
   as: "customerInfo"
  }
```

```
  },
  { $unwind: "$customerInfo" },
  {
    $project: {
      _id: 0,
      orderId: 1,
      sales: 1,
      customerName: "$customerInfo.customerName",
      city: "$customerInfo.city"
    }
  }
]);
```

📌 $lookup **always produces an array**, so $unwind is common.

2️⃣ $addFields — add or modify fields

🧠 What it does

- Adds **new computed fields**
- Or modifies existing ones
- Does NOT remove old fields

📌 Syntax

```
{
  $addFields: {
    newField: <expression>
  }
}
```

❓ Question

👉 *Add a field profitStatus → "Profit" or "Loss"*

✍️ **Thinking**

- Conditional logic → $cond

✅ **Solution**

```
db.orders.aggregate([
 {
  $addFields: {
   profitStatus: {
    $cond: [
     { $gt: ["$profit", 0] },
     "Profit",
     "Loss"
    ]
   }
  }
 },
 {
  $project: {
   _id: 0,
   orderId: 1,
   profit: 1,
   profitStatus: 1
  }
 }
]);
```

🆚 **$addFields vs $project**

$addFields	$project
Keeps all fields	Removes unless included
Adds/modifies	Selects/shapes

3️⃣ $first and $last (inside $group)

🧠 Important rule

They depend on **sort order before grouping**

❓ Question

👉 *For each customer, show first order date and last order date*

✍️ Thinking

1. Sort by date
2. Group by customer
3. Use $first and $last

✅ Solution

```
db.orders.aggregate([
 { $sort: { orderDate: 1 } },
 {
  $group: {
   _id: "$customerName",
   firstOrder: { $first: "$orderDate" },
   lastOrder: { $last: "$orderDate" }
  }
 },
 {
  $project: {
   _id: 0,
```

```
    customerName: "$_id",
    firstOrder: 1,
    lastOrder: 1
   }
  }
]);
```

📌 **Without $sort, $first/$last are meaningless**

4️⃣ **$out — write aggregation result to a collection**

🧠 **What it does**

- Saves aggregation result into **a new collection**
- Replaces existing collection if name already exists

📌 **Syntax**

```
{ $out: "newCollectionName" }
```

❓ **Question**

👉 *Store total sales by region into region_sales collection*

✅ **Solution**

```
db.orders.aggregate([
 {
  $group: {
   _id: "$region",
   totalSales: { $sum: "$sales" }
  }
 },
 {
  $project: {
```

```
    _id: 0,
    region: "$_id",
    totalSales: 1
  }
 },
 { $out: "region_sales" }
]);
```

📌 Used in **ETL / reporting**, rarely in exams but good to know.

5️⃣ Indexes — VERY IMPORTANT for interviews

🧠 What is an index?

Like an **index in a book** → faster search

Without index → collection scan
With index → fast lookup

📌 Create index

```
db.orders.createIndex({ customerName: 1 })
```

Descending:

```
db.orders.createIndex({ sales: -1 })
```

📌 Compound (multiple fields)

```
db.orders.createIndex({ region: 1, segment: 1 })
```

Order matters:

{ region: 1, segment: 1 } ≠ { segment: 1, region: 1 }

📌 List indexes

```
db.orders.getIndexes()
```

📌 Drop index

db.orders.dropIndex("customerName_1")

or

db.orders.dropIndex({ customerName: 1 })

📌 **When index is used**

- $match
- find
- sort (sometimes)
- NOT useful inside $group

🧠 **Mental model (remember this)**

- $lookup → joining data
- $addFields → computed fields
- $first/$last → after sort + group
- $out → save results
- $count → count pipeline docs
- Indexes → performance, not logic

🧠 **Pattern recognition (THIS is what interviewers test)**

Question asks	**Use**
filter only	$match
totals / averages	$group
reshape output	$project
new calculated field	$addFields
unique values	$addToSet
counting docs	$count
first / last	$sort + $group

Question asks	Use
performance	index

```
// Examples
// 1. join orders with customers(display orderid, sales , customerName, ci
// Thinking
// Start from orders
// Need customer data → $lookup
// Array output → $unwind
// Select fields → $project
db.orders.aggregate([
    {
        $lookup: {
            from: "customers",
            localField: "customerId",
            foreignField: "customerId",
            as: "customerInfo"
        }
    },
    { $unwind: "$customerInfo" },
    {
        $project: {
            _id: 0,
            orderId: 1,
            sales: 1,
            customerName: "$customerInfo.customerName",
            city: "$customerInfo.city"
        }
    }
]);
```

```
//2. basic filter after lookup
// show orders placedby customers from dallas
// Thinking
// Need customer city → lookup first
// Then filter
db.orders.aggregate([
    {
        $lookup: {
            from: "customers",
            localField: "customerId",
            foreignField: "customerId",
            as: "cust"
        }
    },
    { $unwind: "$cust" },
    { $match: { "cust.city": "Dallas" } },
    {
        $project: {
            _id: 0,
            orderId: 1,
            sales: 1,
            city: "$cust.city"
        }
    }
]);
```

```
// customers with or without orders
// Display all customers, even those with no orders
// Thinking
// Start from customers
// Lookup orders
// Do NOT unwind
db.customers.aggregate([
    {
        $lookup: {
            from: "orders",
            localField: "customerId",
            foreignField: "customerId",
            as: "orders"
        }
    },
    {
        $project: {
            _id: 0,
            customerName: 1,
            totalOrders: { $size: "$orders" }
        }
    }
]);
```

```
// Medium -- total sales per customer, display customerName and their total sales
// Thinking
// Customers → lookup orders
// Unwind orders
// Group by customer
db.customers.aggregate([
    {
        $lookup: {
            from: "orders",
            localField: "customerId",
            foreignField: "customerId",
            as: "orders"
        }
    },
    { $unwind: "$orders" },
    {
        $group: {
            _id: "$customerName",
            totalSales: { $sum: "$orders.sales" }
        }
    }
]);
```

```
// display orderId, productName, category, qty( orders with product details)
// Thinking
// Orders → lookup products
// Unwind
// Project
db.orders.aggregate([
    {
        $lookup: {
            from: "products",
            localField: "productId",
            foreignField: "productId",
            as: "productInfo"
        }
    },
    { $unwind: "$productInfo" },
    {
        $project: {
            _id: 0,
            orderId: 1,
            qty: 1,
            productName: "$productInfo.productName",
            category: "$productInfo.category"
        }
    }
]);
```

```
// find total qty sold for each product category
// Thinking
// Orders → lookup products
// Unwind
// Group by category
db.orders.aggregate([
    {
        $lookup: {
            from: "products",
            localField: "productId",
            foreignField: "productId",
            as: "prod"
        }
    },
    { $unwind: "$prod" },
    {
        $group: {
            _id: "$prod.category",
            totalQty: { $sum: "$qty" }
        }
    }
]);
```

```
// top customer by total sales

// Find customer who spent the most
// Thinking
// Customers → orders
// Group → sort → limit
db.customers.aggregate([
    {
        $lookup: {
            from: "orders",
            localField: "customerId",
            foreignField: "customerId",
            as: "orders"
        }
    },
    { $unwind: "$orders" },
    {
        $group: {
            _id: "$customerName",
            totalSales: { $sum: "$orders.sales" }
        }
    },
    { $sort: { totalSales: -1 } },
    { $limit: 1 }
]);
```