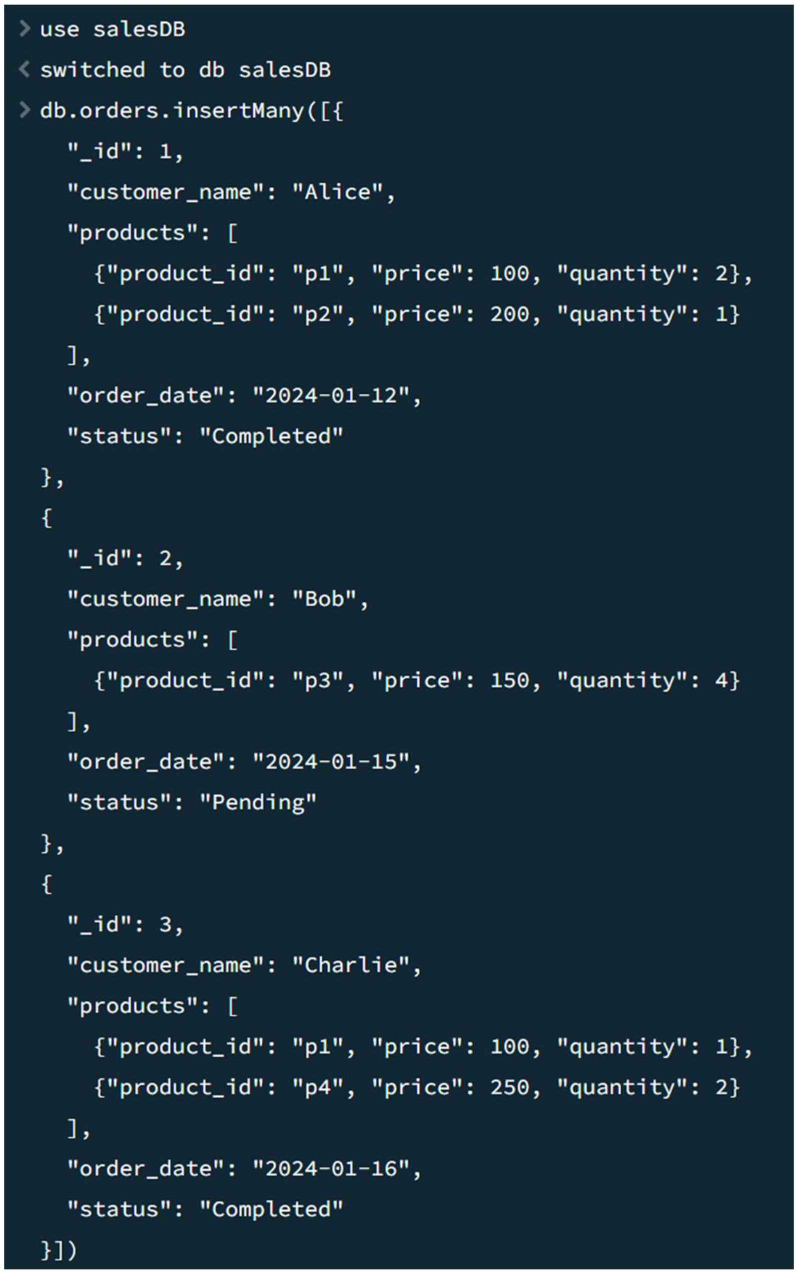
MongoDB Assignment 3

MongoDB Assignment on Aggregate Functions

Database & Collections:

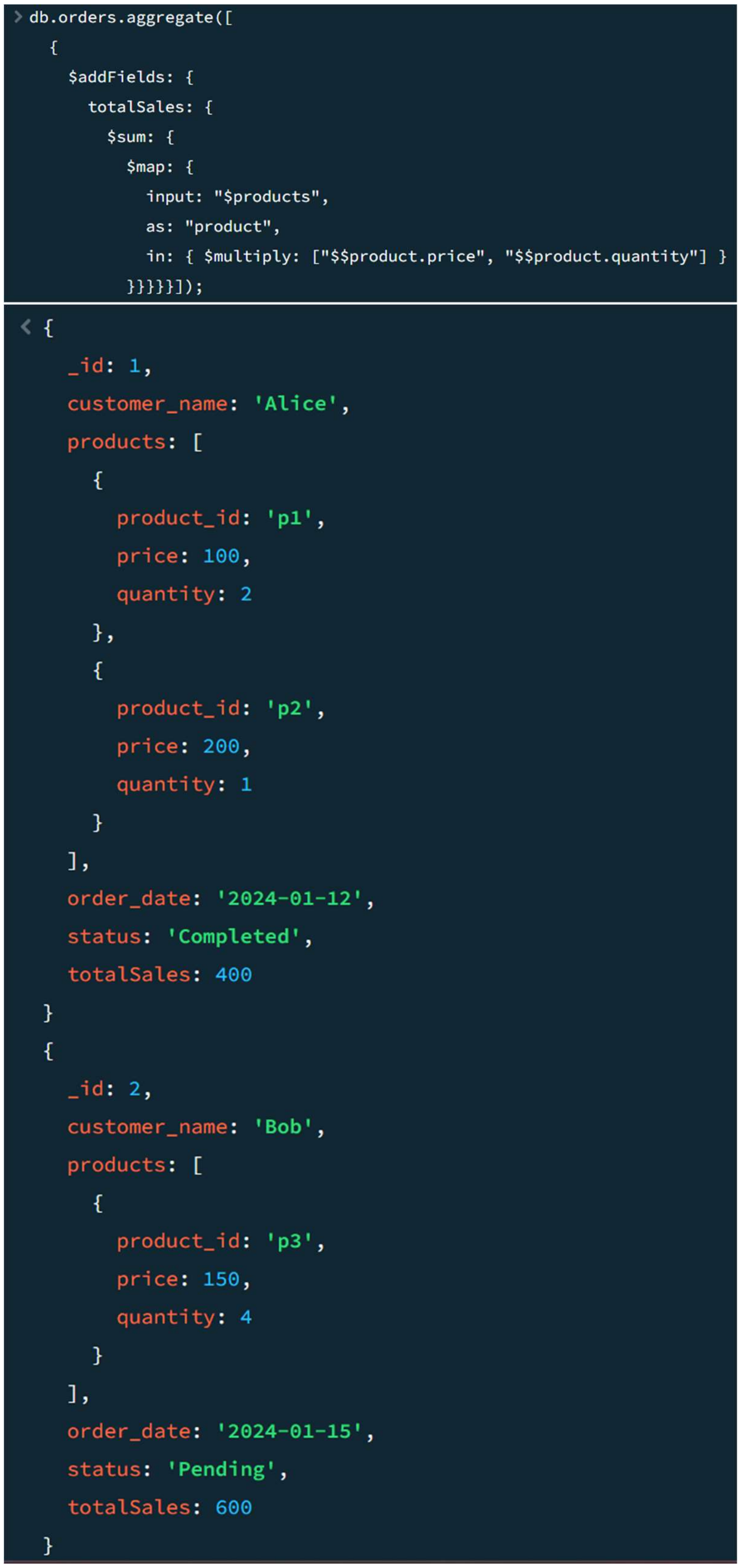
- **Database**: salesDB
- **Collection**: orders

```
{
  "_id": 1,
  "customer_name": "Alice",
  "products": [
    {"product_id": "p1", "price": 100, "quantity": 2},
    {"product_id": "p2", "price": 200, "quantity": 1}
  ],
  "order_date": "2024-01-12",
  "status": "Completed"
},
{
  "_id": 2,
  "customer_name": "Bob",
  "products": [
    {"product_id": "p3", "price": 150, "quantity": 4}
  ],
  "order_date": "2024-01-15",
  "status": "Pending"
},
{
  "_id": 3,
  "customer_name": "Charlie",
  "products": [
    {"product_id": "p1", "price": 100, "quantity": 1},
    {"product_id": "p4", "price": 250, "quantity": 2}
  ],
  "order_date": "2024-01-16",
  "status": "Completed"
}
```



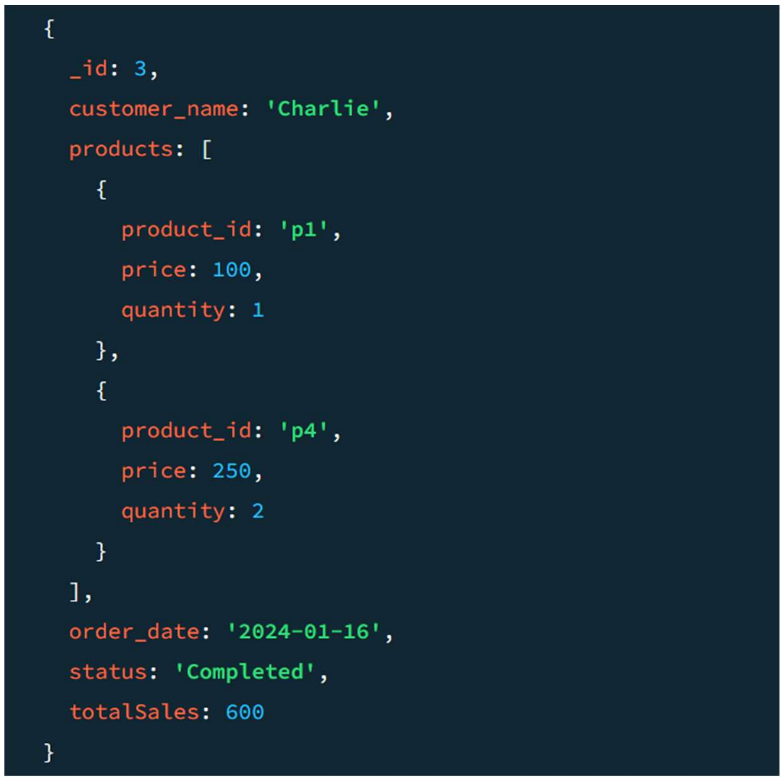
```
> use salesDB
< switched to db salesDB
> db.orders.insertMany([{
    "_id": 1,
    "customer_name": "Alice",
    "products": [
      {"product_id": "p1", "price": 100, "quantity": 2},
      {"product_id": "p2", "price": 200, "quantity": 1}
    ],
    "order_date": "2024-01-12",
    "status": "Completed"
  },
  {
    "_id": 2,
    "customer_name": "Bob",
    "products": [
      {"product_id": "p3", "price": 150, "quantity": 4}
    ],
    "order_date": "2024-01-15",
    "status": "Pending"
  },
  {
    "_id": 3,
    "customer_name": "Charlie",
    "products": [
      {"product_id": "p1", "price": 100, "quantity": 1},
      {"product_id": "p4", "price": 250, "quantity": 2}
    ],
    "order_date": "2024-01-16",
    "status": "Completed"
  }])
```

1. Calculate Total Sales for Each Order.



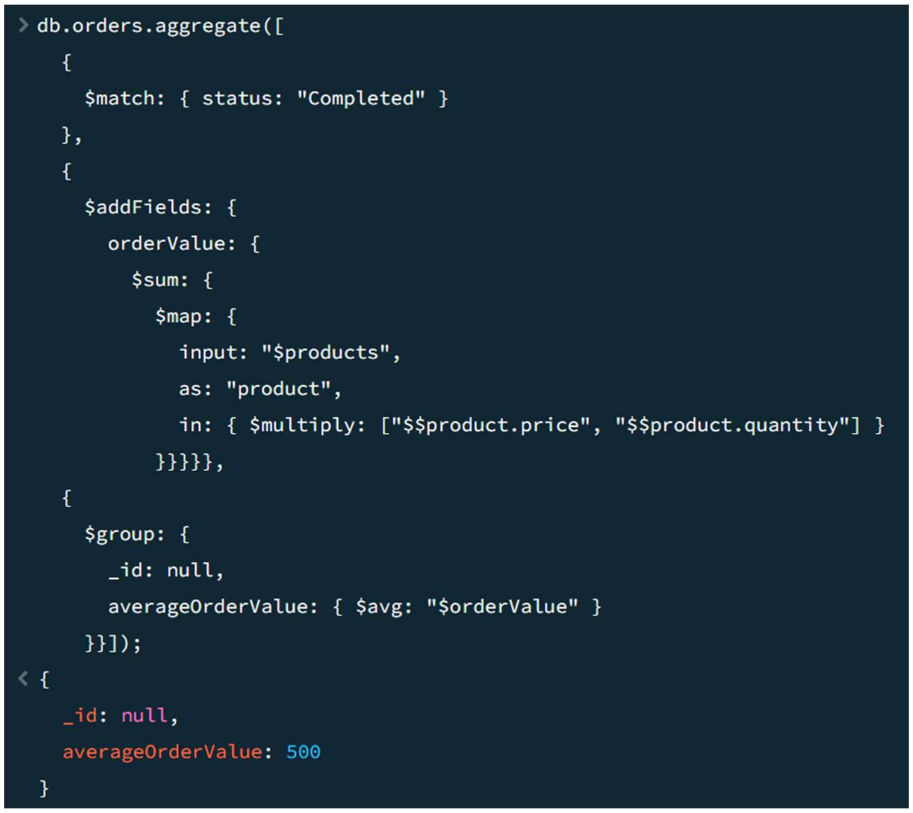
```
> db.orders.aggregate([
    {
      $addFields: {
        totalSales: {
          $sum: {
            $map: {
              input: "$products",
              as: "product",
              in: { $multiply: ["$$product.price", "$$product.quantity"] }
            }}}}}]);
```

```
< {
    _id: 1,
    customer_name: 'Alice',
    products: [
      {
        product_id: 'p1',
        price: 100,
        quantity: 2
      },
      {
        product_id: 'p2',
        price: 200,
        quantity: 1
      }
    ],
    order_date: '2024-01-12',
    status: 'Completed',
    totalSales: 400
  }
  {
    _id: 2,
    customer_name: 'Bob',
    products: [
      {
        product_id: 'p3',
        price: 150,
        quantity: 4
      }
    ],
    order_date: '2024-01-15',
    status: 'Pending',
    totalSales: 600
  }
```



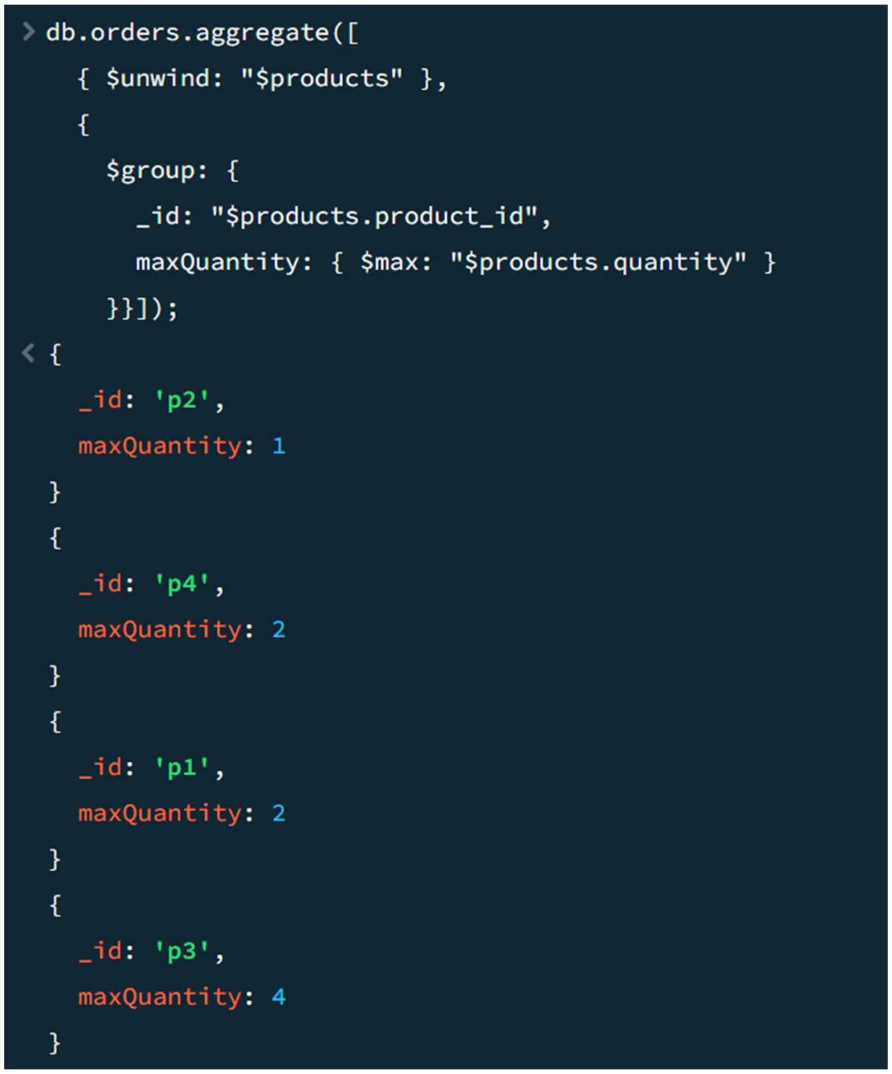
```
{
  _id: 3,
  customer_name: 'Charlie',
  products: [
    {
      product_id: 'p1',
      price: 100,
      quantity: 1
    },
    {
      product_id: 'p4',
      price: 250,
      quantity: 2
    }
  ],
  order_date: '2024-01-16',
  status: 'Completed',
  totalSales: 600
}
```

2. Calculate Average Order Value for Completed Orders.



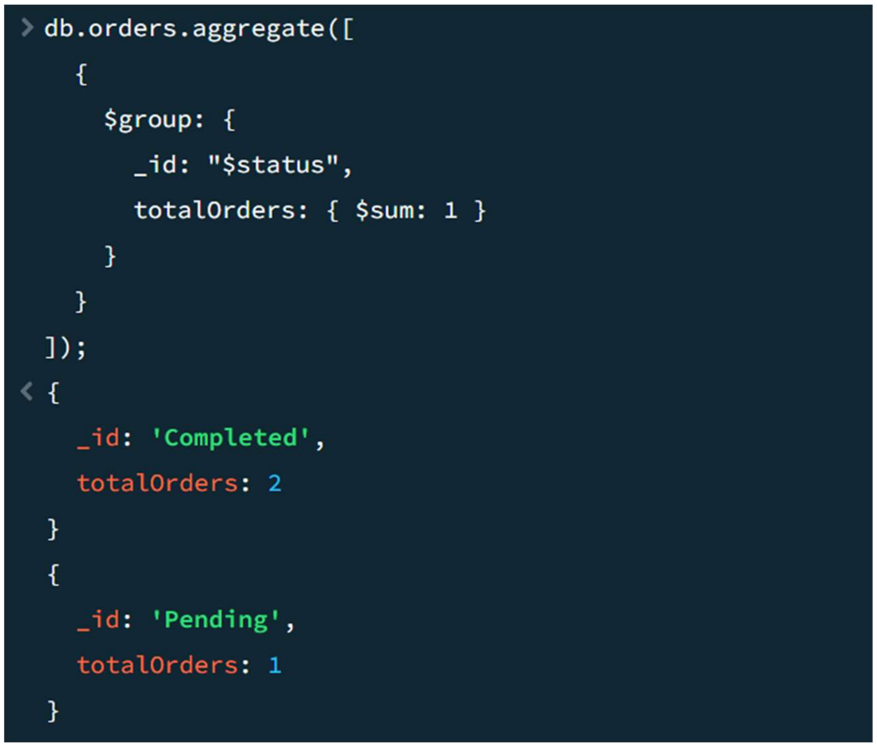
```
> db.orders.aggregate([
    {
      $match: { status: "Completed" }
    },
    {
      $addFields: {
        orderValue: {
          $sum: {
            $map: {
              input: "$products",
              as: "product",
              in: { $multiply: ["$$product.price", "$$product.quantity"] }
            }}}}},
    {
      $group: {
        _id: null,
        averageOrderValue: { $avg: "$orderValue" }
      }}]);
< {
    _id: null,
    averageOrderValue: 500
  }
```

3. Find the Maximum Quantity Sold per Product.



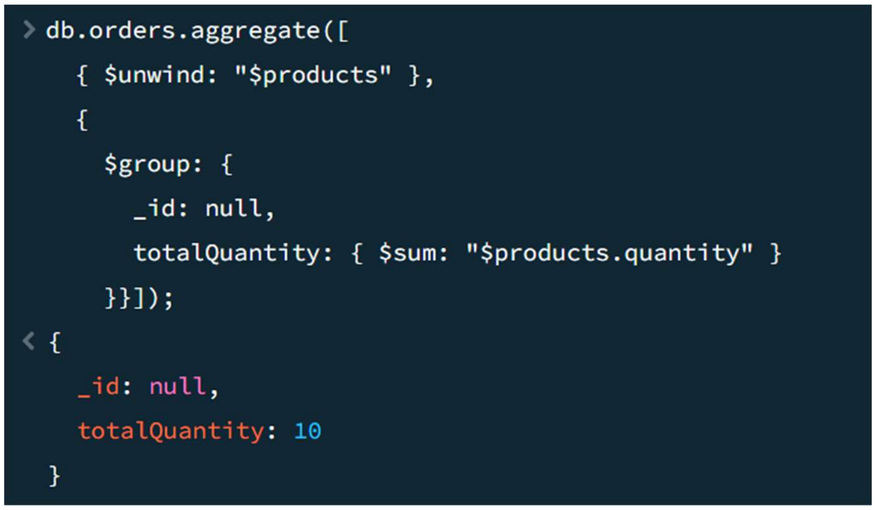
```
> db.orders.aggregate([
    { $unwind: "$products" },
    {
      $group: {
        _id: "$products.product_id",
        maxQuantity: { $max: "$products.quantity" }
      }}]);
< {
    _id: 'p2',
    maxQuantity: 1
  }
  {
    _id: 'p4',
    maxQuantity: 2
  }
  {
    _id: 'p1',
    maxQuantity: 2
  }
  {
    _id: 'p3',
    maxQuantity: 4
  }
```

4. Find Total Number of Orders for Each Status.



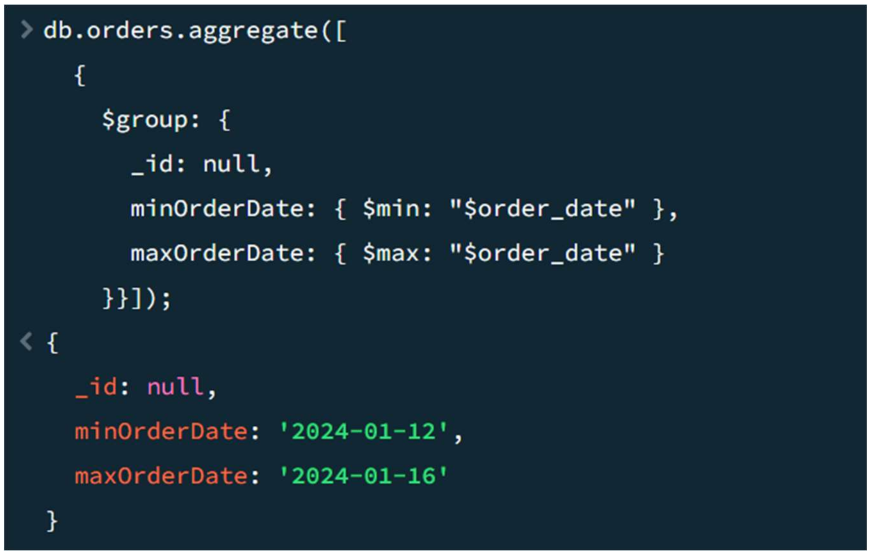
```
> db.orders.aggregate([
    {
      $group: {
        _id: "$status",
        totalOrders: { $sum: 1 }
      }
    }
  ]);
< {
    _id: 'Completed',
    totalOrders: 2
  }
  {
    _id: 'Pending',
    totalOrders: 1
  }
```

5. Calculate Total Quantity of Products Sold Across All Orders.



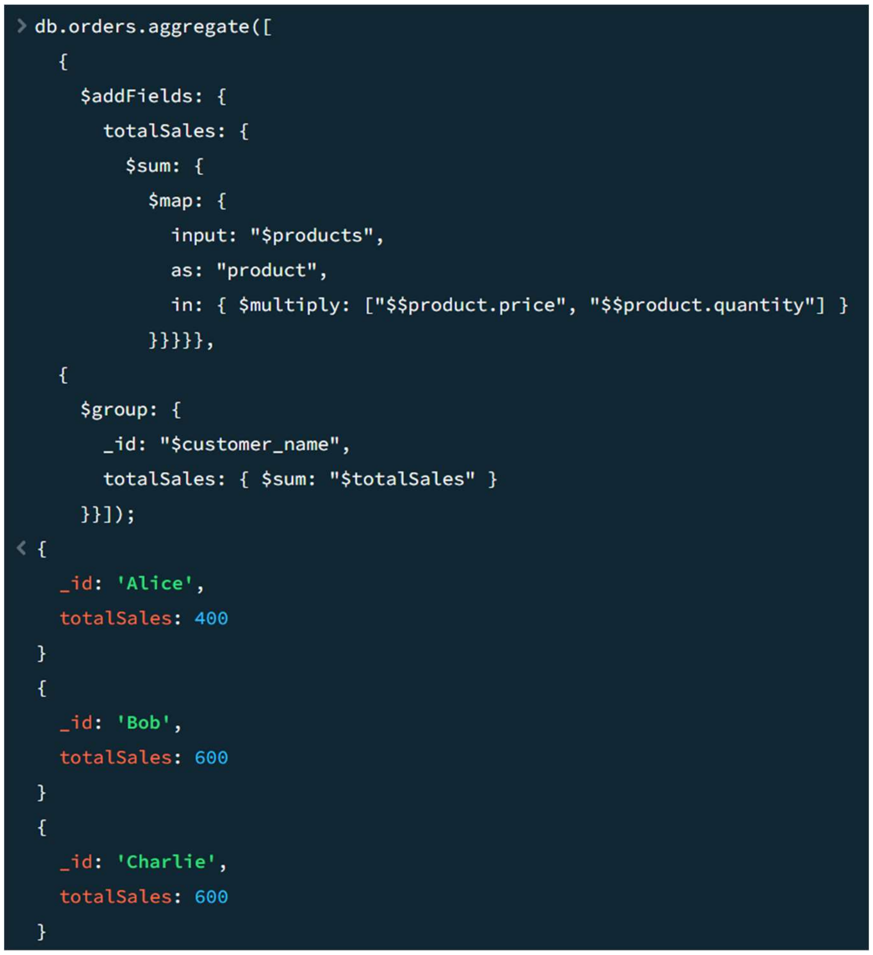
```
> db.orders.aggregate([
    { $unwind: "$products" },
    {
      $group: {
        _id: null,
        totalQuantity: { $sum: "$products.quantity" }
      }}]);
< {
    _id: null,
    totalQuantity: 10
  }
```

6. Get Minimum and Maximum Order Dates.



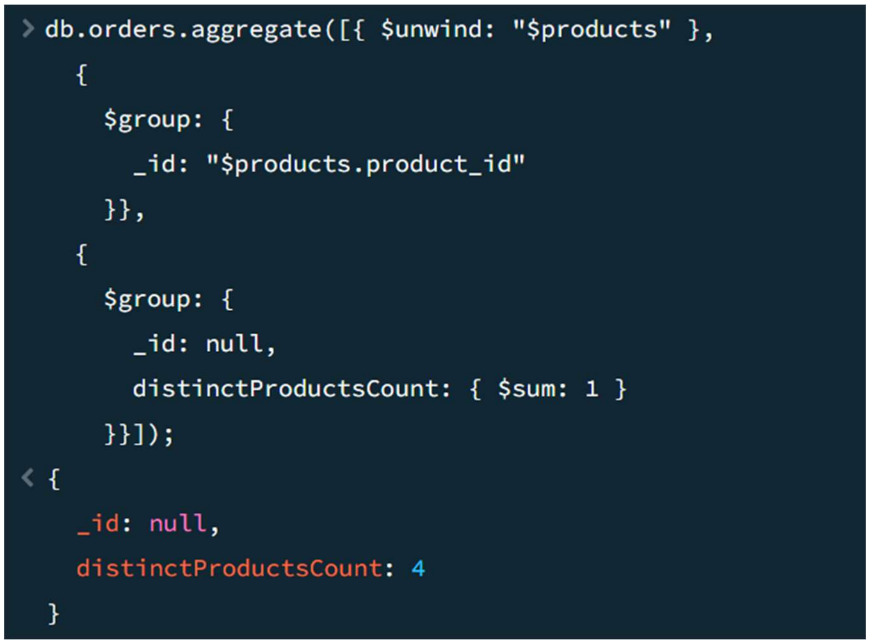
```
> db.orders.aggregate([
    {
      $group: {
        _id: null,
        minOrderDate: { $min: "$order_date" },
        maxOrderDate: { $max: "$order_date" }
      }}]);
< {
    _id: null,
    minOrderDate: '2024-01-12',
    maxOrderDate: '2024-01-16'
  }
```

7. Find Total Sales for Each Customer.



```
> db.orders.aggregate([
    {
      $addFields: {
        totalSales: {
          $sum: {
            $map: {
              input: "$products",
              as: "product",
              in: { $multiply: ["$$product.price", "$$product.quantity"] }
            }}}}},
    {
      $group: {
        _id: "$customer_name",
        totalSales: { $sum: "$totalSales" }
      }}]);
< {
    _id: 'Alice',
    totalSales: 400
  }
  {
    _id: 'Bob',
    totalSales: 600
  }
  {
    _id: 'Charlie',
    totalSales: 600
  }
```

8. Calculate the Total Number of Distinct Products Sold.



```
> db.orders.aggregate([{ $unwind: "$products" },
    {
      $group: {
        _id: "$products.product_id"
      }},
    {
      $group: {
        _id: null,
        distinctProductsCount: { $sum: 1 }
      }}]);
< {
    _id: null,
    distinctProductsCount: 4
  }
```