Advanced Querying and Aggregation

Working with Operators and Complex Queries:

a) $and Operator:

```javascript
db.collectionName.find({
$and: [
{ condition1 },
{ condition2 },
{ condition3 }
]
})
```

Example:

```javascript
db.products.find({
$and: [
{ category: "Electronics" },
{ price: { $lt: 100 } }
]
})
```

b) $or Operator:

```javascript
db.collectionName.find({
$or: [
{ condition1 },
{ condition2 },
{ condition3 }
]
})
```

Example:

```javascript
```

```javascript
db.products.find({
$or: [
{ category: "Electronics" },
{ category: "Clothing" }
]
})
```

c) $not Operator:
```javascript
db.collectionName.find({
field: { $not: { condition } }
})
```
Example:
```javascript
db.products.find({
price: { $not: { $gt: 100 } }
})
```

d) $nor Operator:
```javascript
db.collectionName.find({
$nor: [
{ condition1 },
{ condition2 },
{ condition3 }
]
})
```
Example:
```javascript
db.products.find({
$nor: [
{ category: "Electronics" },
{ category: "Clothing" }
]
})
```

Using $in and $nin Operators for Querying Arrays:

a) $in Operator:

```javascript
db.collectionName.find({ field: { $in: [value1, value2, value3] } })
```

Example:

```javascript
db.products.find({ category: { $in: ["Electronics", "Clothing"] } })
```

b) $nin Operator:

```javascript
db.collectionName.find({ field: { $nin: [value1, value2, value3] } })
```

Example:

```javascript
db.products.find({ category: { $nin: ["Electronics", "Clothing"] } })
```

Aggregation Framework for Advanced Data Processing and Analysis:

a) Grouping Data:

```javascript
db.collectionName.aggregate([
{
$group: {
_id: "$field",
total: { $sum: "$value" }
}
}
])
```

Example:

```javascript
db.sales.aggregate([
```

```
{
$group: {
_id: "$category",
totalSales: { $sum: "$amount" }
}
}
])
```

b) Filtering Data:
```javascript
db.collectionName.aggregate([
{
$match: {
field: { $gte: value1, $lt: value2 }
}
}
])
```
Example:
```javascript
db.sales.aggregate([
{
$match: {
date: { $gte: ISODate("2023-01-01"), $lt: ISODate("2023-02-01") }
}
}
])
```

c) Transforming Data:
```javascript
db.collectionName.aggregate([
{
$project: {
newField: "$oldField"
}
}
])
```

Example:

```javascript
db.sales.aggregate([
{
$project: {
productName: "$name",
saleAmount: "$amount"
}
}
])
```

Pipeline Stages, Aggregation Operators, and Group By Operations:

Example:

```javascript
db.sales.aggregate([
{ $match: { date: { $gte: ISODate("2023-01-01"), $lt: ISODate("2023-02-01") } } },
{ $group: { _id: "$category", totalSales: { $sum: "$amount" } } },
{ $sort: { totalSales: -1 } },
{ $limit: 5 }
])
```

Ensure to adapt the collection name, field names, values, and filter criteria as per your specific MongoDB setup and data requirements. These examples serve as a starting point to understand the concepts and perform the mentioned operations in MongoDB.