```
db.students.insertMany([
  { name: "Alice", subject: "Math", score: 85 },
  { name: "Bob", subject: "Math", score: 90 },
  { name: "Charlie", subject: "Science", score: 75 },
  { name: "David", subject: "Science", score: 80 }
])
```

2. Grouping students by subject and calculating the average score:

js
Copy code
```
db.students.aggregate([
  { $group: { _id: "$subject", avgScore: { $avg: "$score" } } }
])
```

Explanation:

- **Stage**: `$group`
    - Groups the students by the `subject` field (`_id: "$subject"`).
    - Computes the average score for each group using `$avg: "$score"`.

Output:

json
Copy code
```
[
  { "_id": "Math", "avgScore": 87.5 },
  { "_id": "Science", "avgScore": 77.5 }
]
```

3. Matching students with scores greater than 80:

js
Copy code
```
db.students.aggregate([
  { $match: { score: { $gt: 80 } } }
])
```

Explanation:

- **Stage**: `$match`
  - Filters out documents where `score` is greater than 80.

Output:

json
Copy code
```
[
  { "name": "Alice", "subject": "Math", "score": 85 },
  { "name": "Bob", "subject": "Math", "score": 90 }
]
```

4. Matching scores greater than 80 and then grouping by subject:

js
Copy code
```
db.students.aggregate([
  { $match: { score: { $gt: 80 } } },
  { $group: { _id: "$subject", avgScore: { $avg: "$score" } } }
])
```

Explanation:

- First, **$match** filters students with scores > 80.
- Then, **$group** computes the average score for each subject where the score is greater than 80.

Output:

json
Copy code
```
[
  { "_id": "Math", "avgScore": 87.5 }
]
```

5. Projecting only `name` and `subject` fields (hiding `_id`):

js
Copy code

```
db.students.aggregate([
  { $project: { name: 1, subject: 1, _id: 0 } }
])
```

Explanation:

- **$project** reshapes the document, including only `name` and `subject` fields and excluding the `_id`.

Output:

json
Copy code
```
[
  { "name": "Alice", "subject": "Math" },
  { "name": "Bob", "subject": "Math" },
  { "name": "Charlie", "subject": "Science" },
  { "name": "David", "subject": "Science" }
]
```

6. Sorting `students` by score in descending order:

js
Copy code
```
db.students.aggregate([
  { $sort: { score: -1 } }
])
```

Explanation:

- **$sort** arranges the students in descending order of their `score`.

Output:

json
Copy code
```
[
  { "name": "Bob", "subject": "Math", "score": 90 },
  { "name": "Alice", "subject": "Math", "score": 85 },
  { "name": "David", "subject": "Science", "score": 80 },
  { "name": "Charlie", "subject": "Science", "score": 75 }
]
```

7. Inserting and grouping `sales` data by product and summing total sales:

js
Copy code
```
db.sales.aggregate([
  { $group: { _id: "$product", totalSales: { $sum: "$amount" } } }
])
```

Explanation:

- **$group** groups by `product` and sums the total `amount` for each product.

Output:

json
Copy code
```
[
  { "_id": "Laptop", "totalSales": 3650 },
  { "_id": "Smartphone", "totalSales": 2070 },
  { "_id": "Tablet", "totalSales": 620 }
]
```

8. Joining `sales` with `products` and calculating total sales:

js
Copy code
```
db.sales.aggregate([
  {
    $lookup: {
      from: "products",
      localField: "product_id",
      foreignField: "_id",
      as: "ProductInfo"
    }
  },
  { $unwind: "$ProductInfo" },
  {
    $group: {
      _id: "$ProductInfo.name",
      totalSales: { $sum: "$amount" },
      price: { $first: "$ProductInfo.price" }
    }
  }
])
```

Explanation:

- **$lookup** performs a join with the `products` collection.
- **$unwind** breaks down the `ProductInfo` array into individual documents.
- **$group** groups the sales by product name and calculates `totalSales` and product `price`.

Output:

json
Copy code
```
[
  { "_id": "Laptop", "totalSales": 2500, "price": 1200 },
  { "_id": "Smartphone", "totalSales": 1350, "price": 700 },
  { "_id": "Tablet", "totalSales": 620, "price": 300 }
]
```

9. Joining `sales` with `products` and `customers`:

js
Copy code
```
db.sales.aggregate([
  {
    $lookup: {
      from: "products",
      localField: "product_id",
      foreignField: "_id",
      as: "productInfo"
    }
  },
  {
    $lookup: {
      from: "customers",
      localField: "customer_id",
      foreignField: "_id",
      as: "customerInfo"
    }
  },
  { $unwind: "$productInfo" },
  { $unwind: "$customerInfo" },
  {
    $group: {
      _id: "$productInfo.name",
      totalSales: { $sum: "$amount" },
      price: { $first: "$productInfo.price" },
      customers: { $addToSet: "$customerInfo.name" }
    }
  }
])
```

Explanation:

- **$lookup** joins the `sales` collection with both the `products` and `customers` collections.
- **$unwind** breaks down the `productInfo` and `customerInfo` arrays into separate documents.
- **$group** groups by product name, sums total sales, and collects unique customer names who bought that product.

Output:

json
Copy code
```
[
  {
    "_id": "Laptop",
    "totalSales": 2500,
    "price": 1200,
    "customers": ["Alice", "Bob
```

```
10. db.customers.insertMany([
  { "_id": 1, "name": "Alice", "email": "alice@example.com" },
  { "_id": 2, "name": "Bob", "email": "bob@example.com" },
  { "_id": 3, "name": "Charlie", "email": "charlie@example.com" }
])
```

11. Joining Customers and Products and Grouping Sales by Product

js
Copy code

```js
db.sales.aggregate([
  {
    $lookup: {
      from: "products",
      localField: "product_id",
      foreignField: "_id",
      as: "productInfo"
    }
  },
  {
    $lookup: {
      from: "customers",
      localField: "customer_id",
      foreignField: "_id",
      as: "customerInfo"
    }
  },
  { $unwind: "$productInfo" },
  { $unwind: "$customerInfo" },
  {
    $group: {
      _id: "$productInfo.name",
      totalSales: { $sum: "$amount" },
      price: { $first: "$productInfo.price" },
      customers: { $addToSet: "$customerInfo.name" }
    }
  }
])
```

- **Purpose**: This performs two joins:
  - **$lookup** with the products collection to get product details.
  - **$lookup** with the customers collection to get customer details.
- It groups by product name and collects customer names into an array.
- **Output**:

json
Copy code
```
[
  {
    "_id": "Laptop",
    "totalSales": 2500,
    "price": 1200,
    "customers": ["Alice", "Bob"]
  },
  {
    "_id": "Smartphone",
    "totalSales": 1350,
    "price": 700,
    "customers": ["Charlie"]
  }
]
```