VIGNAN'S
FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

Assessment-6

MongoDB Aggregation Queries

Group -5

NAME: N.BHARGAVI

REGD NO : 221FA04498

BRANCH: CSE

SECTION: 3C

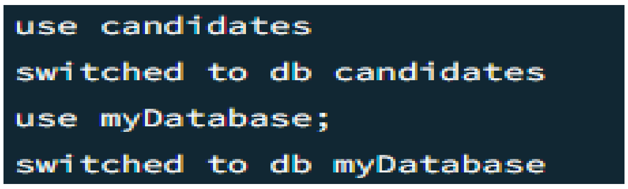
SUBJECT: MSD

Objective:

The objective of aggregation is to summarize and combine data from multiple sources or records into a single, concise representation for easier analysis and interpretation.

Step 1: Create the database

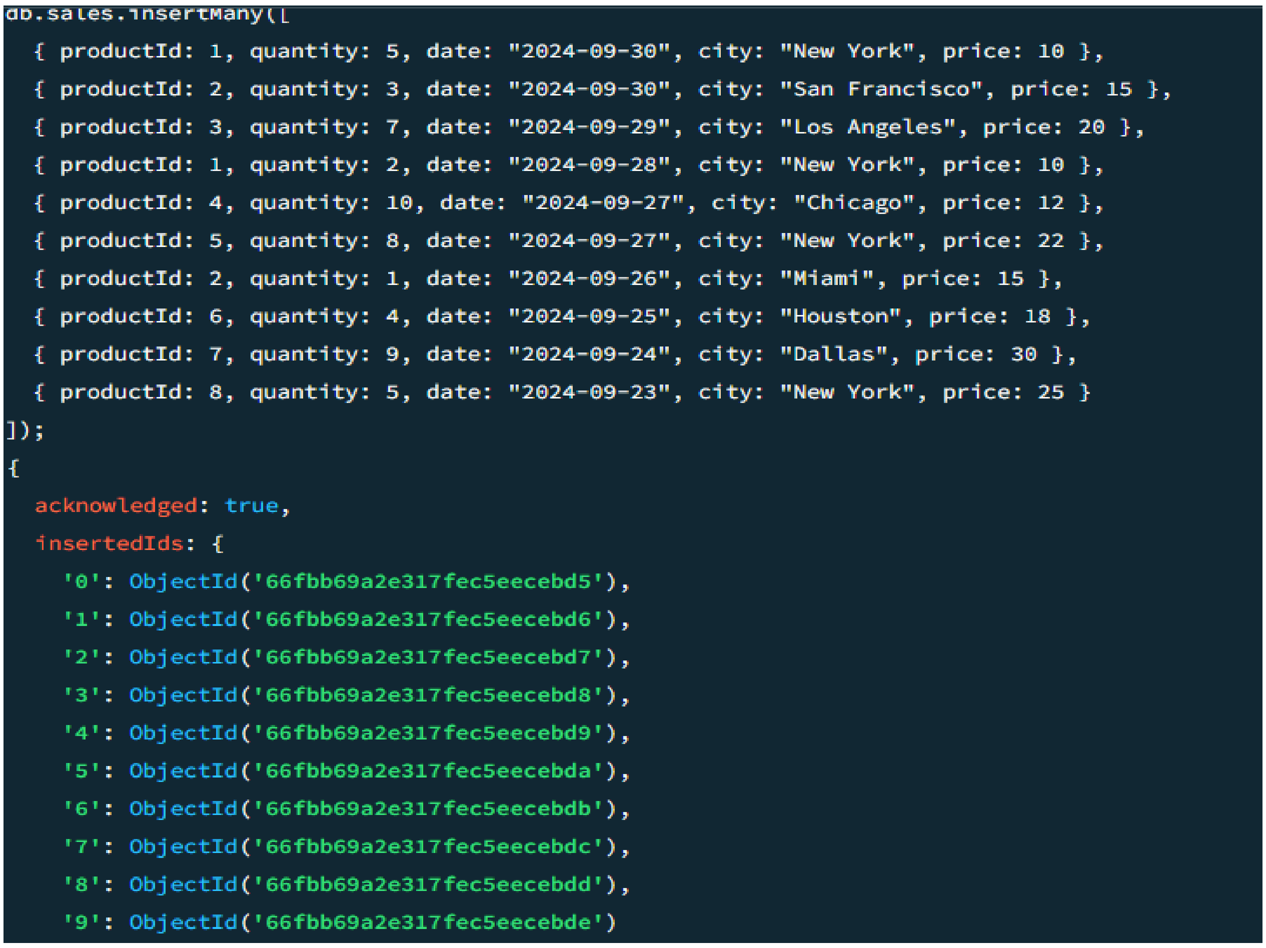
Syntax: use myDatabase;



```
use candidates
switched to db candidates
use myDatabase;
switched to db myDatabase
```

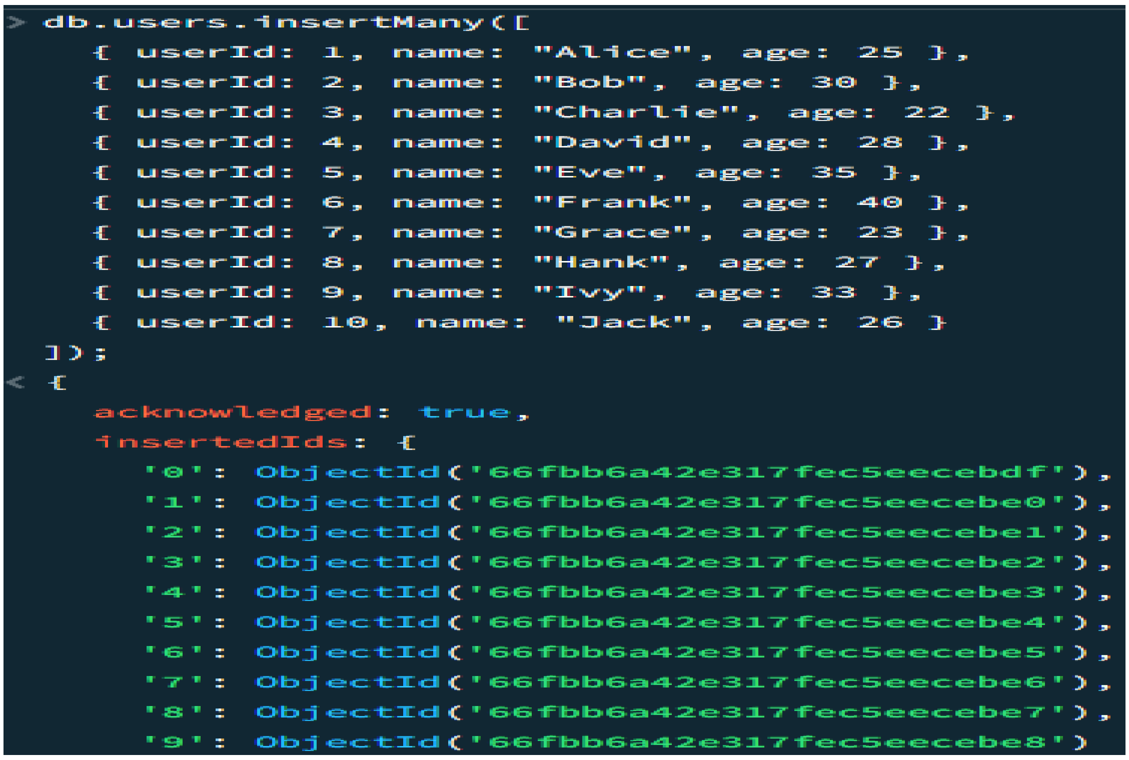
Step **2: Insert sample data**

- sales collection:



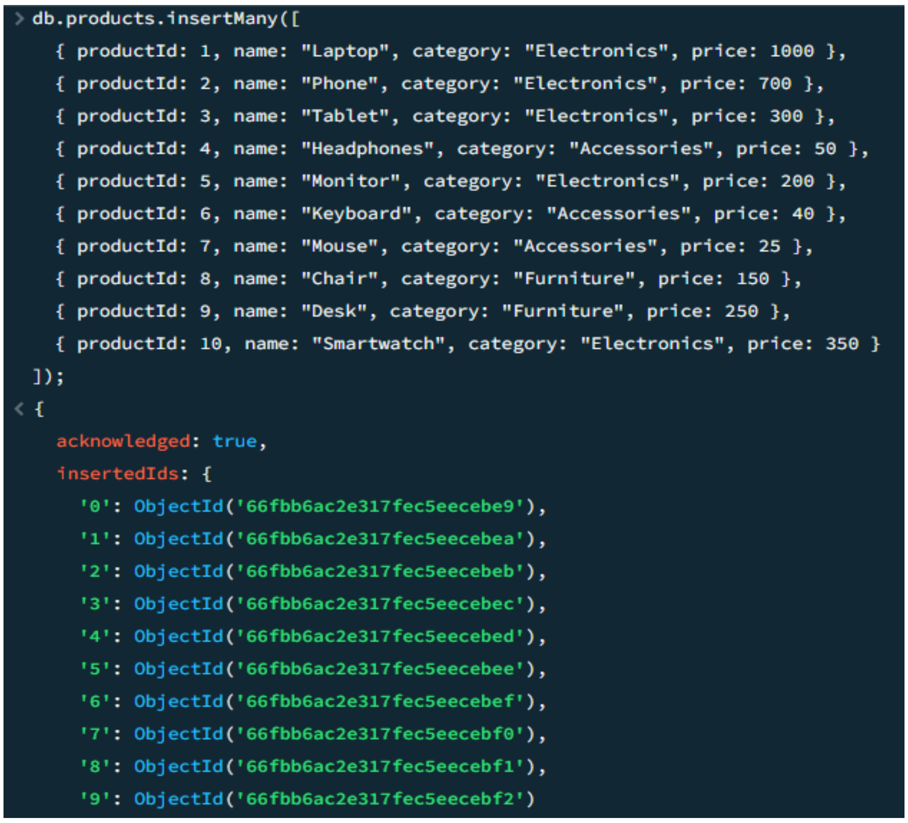
```
db.sales.insertMany([
  { productId: 1, quantity: 5, date: "2024-09-30", city: "New York", price: 10 },
  { productId: 2, quantity: 3, date: "2024-09-30", city: "San Francisco", price: 15 },
  { productId: 3, quantity: 7, date: "2024-09-29", city: "Los Angeles", price: 20 },
  { productId: 1, quantity: 2, date: "2024-09-28", city: "New York", price: 10 },
  { productId: 4, quantity: 10, date: "2024-09-27", city: "Chicago", price: 12 },
  { productId: 5, quantity: 8, date: "2024-09-27", city: "New York", price: 22 },
  { productId: 2, quantity: 1, date: "2024-09-26", city: "Miami", price: 15 },
  { productId: 6, quantity: 4, date: "2024-09-25", city: "Houston", price: 18 },
  { productId: 7, quantity: 9, date: "2024-09-24", city: "Dallas", price: 30 },
  { productId: 8, quantity: 5, date: "2024-09-23", city: "New York", price: 25 }
]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('66fbb69a2e317fec5eecebd5'),
    '1': ObjectId('66fbb69a2e317fec5eecebd6'),
    '2': ObjectId('66fbb69a2e317fec5eecebd7'),
    '3': ObjectId('66fbb69a2e317fec5eecebd8'),
    '4': ObjectId('66fbb69a2e317fec5eecebd9'),
    '5': ObjectId('66fbb69a2e317fec5eecebda'),
    '6': ObjectId('66fbb69a2e317fec5eecebdb'),
    '7': ObjectId('66fbb69a2e317fec5eecebdc'),
    '8': ObjectId('66fbb69a2e317fec5eecebdd'),
    '9': ObjectId('66fbb69a2e317fec5eecebde')
```

➢ users collection:



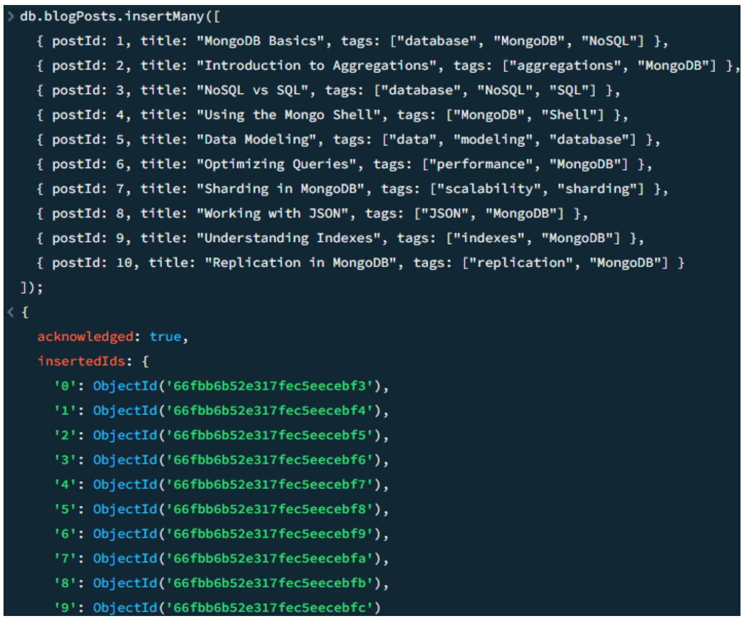
```
> db.users.insertMany([
    { userId: 1, name: "Alice", age: 25 },
    { userId: 2, name: "Bob", age: 30 },
    { userId: 3, name: "Charlie", age: 22 },
    { userId: 4, name: "David", age: 28 },
    { userId: 5, name: "Eve", age: 35 },
    { userId: 6, name: "Frank", age: 40 },
    { userId: 7, name: "Grace", age: 23 },
    { userId: 8, name: "Hank", age: 27 },
    { userId: 9, name: "Ivy", age: 33 },
    { userId: 10, name: "Jack", age: 26 }
  ]);
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('66fbb6a42e317fec5eecebdf'),
      '1': ObjectId('66fbb6a42e317fec5eecebe0'),
      '2': ObjectId('66fbb6a42e317fec5eecebe1'),
      '3': ObjectId('66fbb6a42e317fec5eecebe2'),
      '4': ObjectId('66fbb6a42e317fec5eecebe3'),
      '5': ObjectId('66fbb6a42e317fec5eecebe4'),
      '6': ObjectId('66fbb6a42e317fec5eecebe5'),
      '7': ObjectId('66fbb6a42e317fec5eecebe6'),
      '8': ObjectId('66fbb6a42e317fec5eecebe7'),
      '9': ObjectId('66fbb6a42e317fec5eecebe8')
```

➢ products collection:



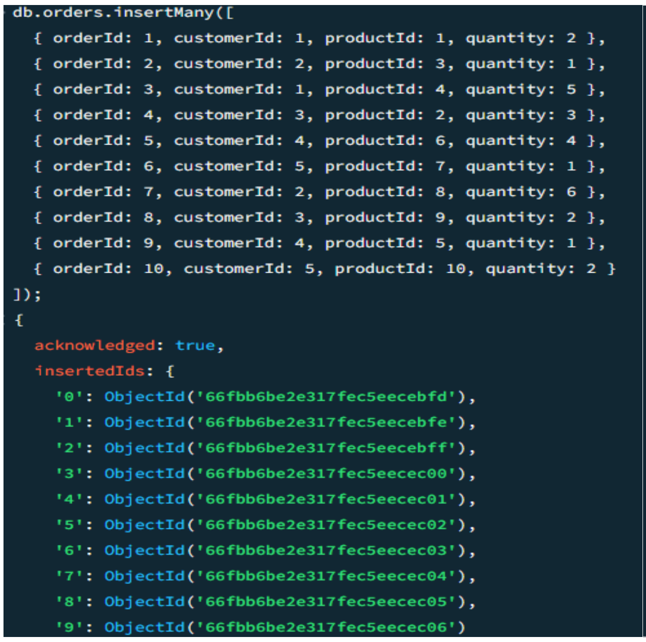
```
> db.products.insertMany([
    { productId: 1, name: "Laptop", category: "Electronics", price: 1000 },
    { productId: 2, name: "Phone", category: "Electronics", price: 700 },
    { productId: 3, name: "Tablet", category: "Electronics", price: 300 },
    { productId: 4, name: "Headphones", category: "Accessories", price: 50 },
    { productId: 5, name: "Monitor", category: "Electronics", price: 200 },
    { productId: 6, name: "Keyboard", category: "Accessories", price: 40 },
    { productId: 7, name: "Mouse", category: "Accessories", price: 25 },
    { productId: 8, name: "Chair", category: "Furniture", price: 150 },
    { productId: 9, name: "Desk", category: "Furniture", price: 250 },
    { productId: 10, name: "Smartwatch", category: "Electronics", price: 350 }
  ]);
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('66fbb6ac2e317fec5eecebe9'),
      '1': ObjectId('66fbb6ac2e317fec5eecebea'),
      '2': ObjectId('66fbb6ac2e317fec5eecebeb'),
      '3': ObjectId('66fbb6ac2e317fec5eecebec'),
      '4': ObjectId('66fbb6ac2e317fec5eecebed'),
      '5': ObjectId('66fbb6ac2e317fec5eecebee'),
      '6': ObjectId('66fbb6ac2e317fec5eecebef'),
      '7': ObjectId('66fbb6ac2e317fec5eecebf0'),
      '8': ObjectId('66fbb6ac2e317fec5eecebf1'),
      '9': ObjectId('66fbb6ac2e317fec5eecebf2')
```

➢ blogPosts collection:

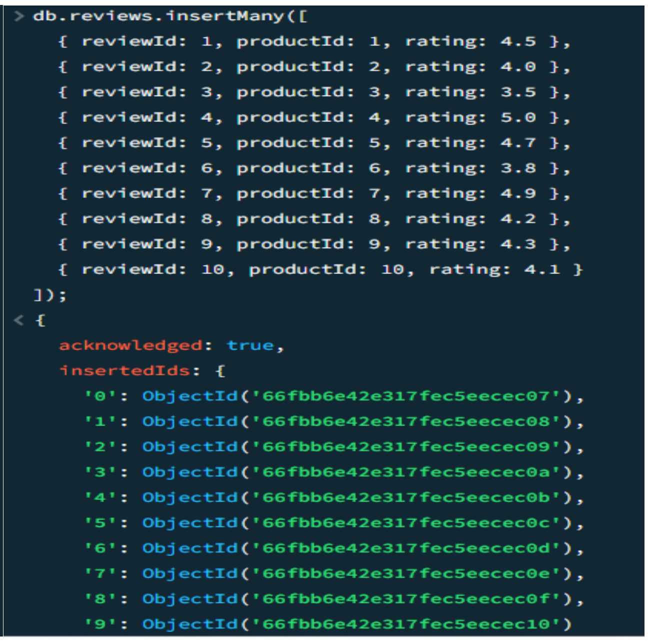


```
> db.blogPosts.insertMany([
    { postId: 1, title: "MongoDB Basics", tags: ["database", "MongoDB", "NoSQL"] },
    { postId: 2, title: "Introduction to Aggregations", tags: ["aggregations", "MongoDB"] },
    { postId: 3, title: "NoSQL vs SQL", tags: ["database", "NoSQL", "SQL"] },
    { postId: 4, title: "Using the Mongo Shell", tags: ["MongoDB", "Shell"] },
    { postId: 5, title: "Data Modeling", tags: ["data", "modeling", "database"] },
    { postId: 6, title: "Optimizing Queries", tags: ["performance", "MongoDB"] },
    { postId: 7, title: "Sharding in MongoDB", tags: ["scalability", "sharding"] },
    { postId: 8, title: "Working with JSON", tags: ["JSON", "MongoDB"] },
    { postId: 9, title: "Understanding Indexes", tags: ["indexes", "MongoDB"] },
    { postId: 10, title: "Replication in MongoDB", tags: ["replication", "MongoDB"] }
  ]);
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('66fbb6b52e317fec5eecebf3'),
      '1': ObjectId('66fbb6b52e317fec5eecebf4'),
      '2': ObjectId('66fbb6b52e317fec5eecebf5'),
      '3': ObjectId('66fbb6b52e317fec5eecebf6'),
      '4': ObjectId('66fbb6b52e317fec5eecebf7'),
      '5': ObjectId('66fbb6b52e317fec5eecebf8'),
      '6': ObjectId('66fbb6b52e317fec5eecebf9'),
      '7': ObjectId('66fbb6b52e317fec5eecebfa'),
      '8': ObjectId('66fbb6b52e317fec5eecebfb'),
      '9': ObjectId('66fbb6b52e317fec5eecebfc')
```

➢ orders collection & reviews collection:



```
db.orders.insertMany([
   { orderId: 1, customerId: 1, productId: 1, quantity: 2 },
   { orderId: 2, customerId: 2, productId: 3, quantity: 1 },
   { orderId: 3, customerId: 1, productId: 4, quantity: 5 },
   { orderId: 4, customerId: 3, productId: 2, quantity: 3 },
   { orderId: 5, customerId: 4, productId: 6, quantity: 4 },
   { orderId: 6, customerId: 5, productId: 7, quantity: 1 },
   { orderId: 7, customerId: 2, productId: 8, quantity: 6 },
   { orderId: 8, customerId: 3, productId: 9, quantity: 2 },
   { orderId: 9, customerId: 4, productId: 5, quantity: 1 },
   { orderId: 10, customerId: 5, productId: 10, quantity: 2 }
]);
{
   acknowledged: true,
   insertedIds: {
     '0': ObjectId('66fbb6be2e317fec5eecebfd'),
     '1': ObjectId('66fbb6be2e317fec5eecebfe'),
     '2': ObjectId('66fbb6be2e317fec5eecebff'),
     '3': ObjectId('66fbb6be2e317fec5eecec00'),
     '4': ObjectId('66fbb6be2e317fec5eecec01'),
     '5': ObjectId('66fbb6be2e317fec5eecec02'),
     '6': ObjectId('66fbb6be2e317fec5eecec03'),
     '7': ObjectId('66fbb6be2e317fec5eecec04'),
     '8': ObjectId('66fbb6be2e317fec5eecec05'),
     '9': ObjectId('66fbb6be2e317fec5eecec06')
```



```
> db.reviews.insertMany([
    { reviewId: 1, productId: 1, rating: 4.5 },
    { reviewId: 2, productId: 2, rating: 4.0 },
    { reviewId: 3, productId: 3, rating: 3.5 },
    { reviewId: 4, productId: 4, rating: 5.0 },
    { reviewId: 5, productId: 5, rating: 4.7 },
    { reviewId: 6, productId: 6, rating: 3.8 },
    { reviewId: 7, productId: 7, rating: 4.9 },
    { reviewId: 8, productId: 8, rating: 4.2 },
    { reviewId: 9, productId: 9, rating: 4.3 },
    { reviewId: 10, productId: 10, rating: 4.1 }
  ]);
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('66fbb6e42e317fec5eecec07'),
      '1': ObjectId('66fbb6e42e317fec5eecec08'),
      '2': ObjectId('66fbb6e42e317fec5eecec09'),
      '3': ObjectId('66fbb6e42e317fec5eecec0a'),
      '4': ObjectId('66fbb6e42e317fec5eecec0b'),
      '5': ObjectId('66fbb6e42e317fec5eecec0c'),
      '6': ObjectId('66fbb6e42e317fec5eecec0d'),
      '7': ObjectId('66fbb6e42e317fec5eecec0e'),
      '8': ObjectId('66fbb6e42e317fec5eecec0f'),
      '9': ObjectId('66fbb6e42e317fec5eecec10')
```

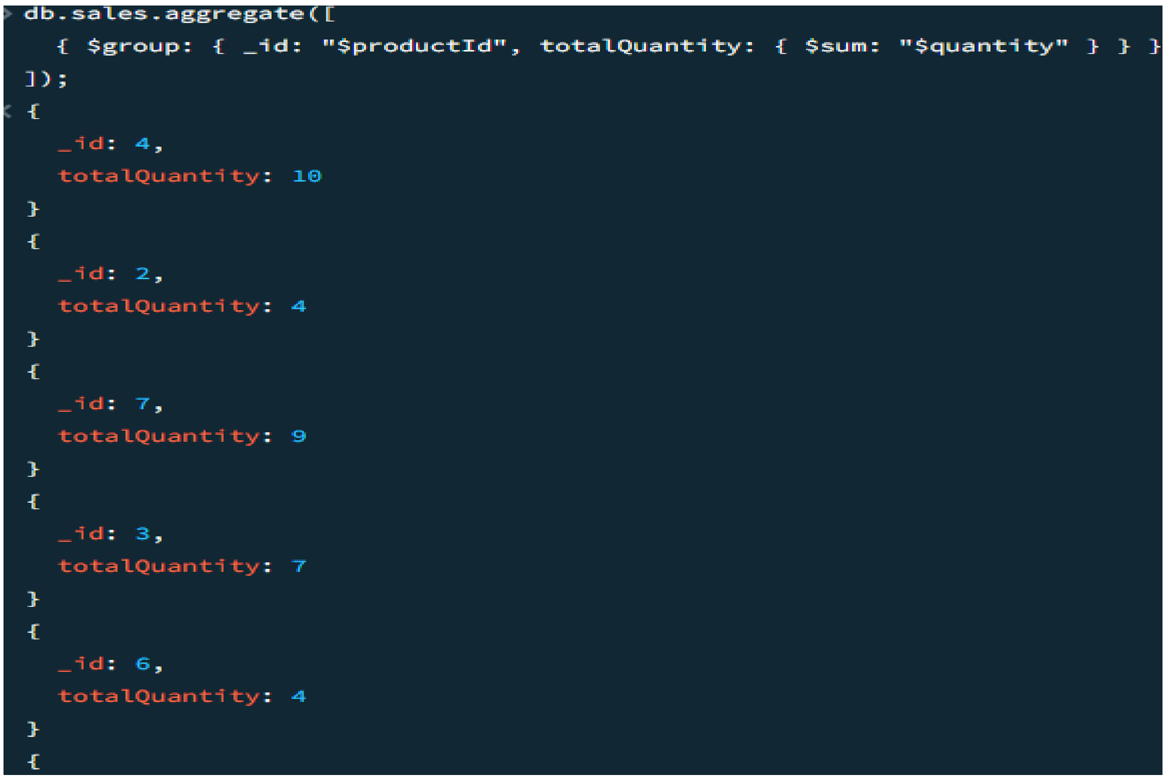
Tasks:

Question 1: Write an aggregation query to find the total quantity sold for each product in the sales collection.

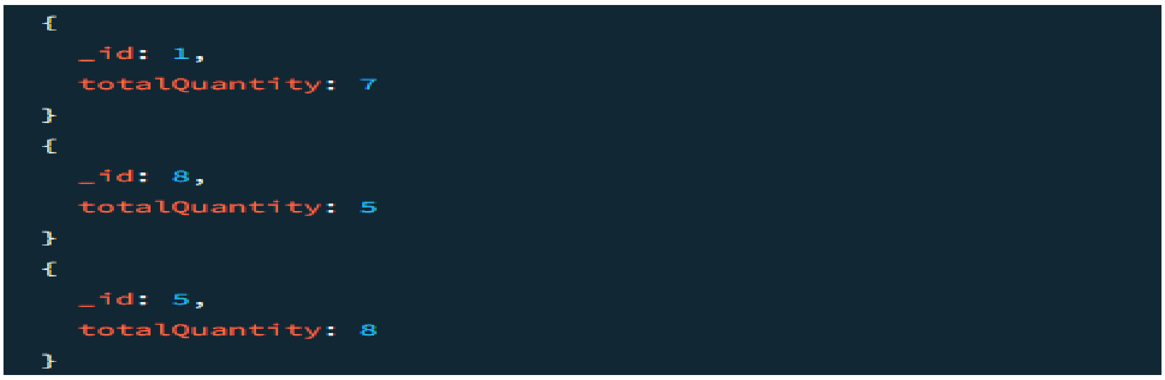
Find the total quantity sold for each product:

Querie:

```
db.sales.aggregate([
  { $group: { _id: "$productId", totalQuantity: { $sum: "$quantity" } } }
])
```



```
> db.sales.aggregate([
    { $group: { _id: "$productId", totalQuantity: { $sum: "$quantity" } } }
  ]);
< {
    _id: 4,
    totalQuantity: 10
  }
  {
    _id: 2,
    totalQuantity: 4
  }
  {
    _id: 7,
    totalQuantity: 9
  }
  {
    _id: 3,
    totalQuantity: 7
  }
  {
    _id: 6,
    totalQuantity: 4
  }
  {
```



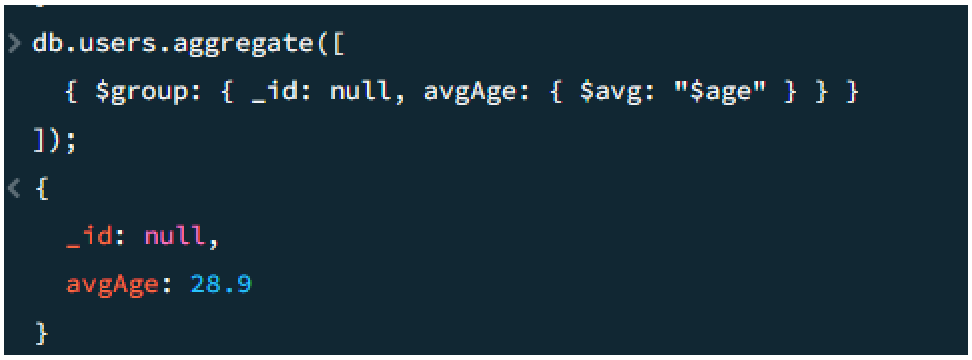
```
  {
    _id: 1,
    totalQuantity: 7
  }
  {
    _id: 8,
    totalQuantity: 5
  }
  {
    _id: 5,
    totalQuantity: 8
  }
```

Question 2: Write an aggregation query to calculate the average age of users in the users collection.

Calculate the average age of users:

Querie:

```
db.users.aggregate([
 { $group: { _id: null, avgAge: { $avg: "$age" } } }
])
```

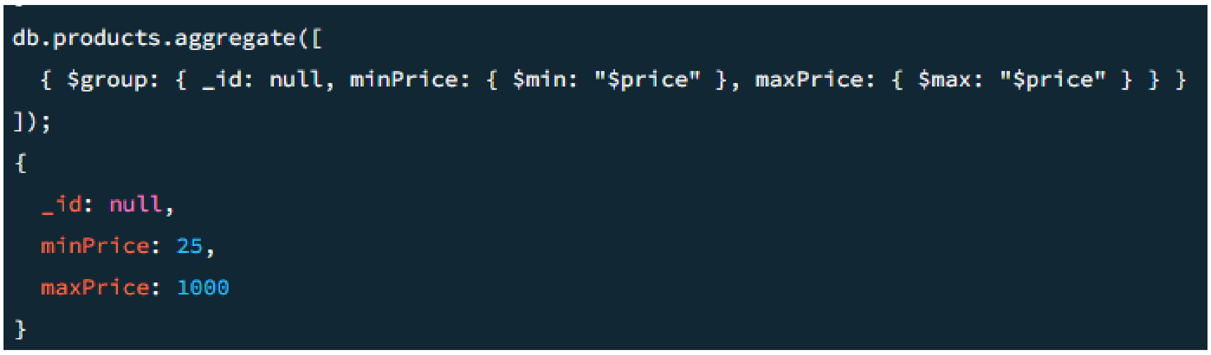

```
> db.users.aggregate([
    { $group: { _id: null, avgAge: { $avg: "$age" } } }
  ]);
< {
    _id: null,
    avgAge: 28.9
  }
```

Question 3: Write an aggregation query to find the minimum and maximum prices of products in the products collection.

Find the minimum and maximum prices of products:

<u>Querie:</u>

db.products.aggregate([
  { $group: { _id: null, minPrice: { $min: "$price" }, maxPrice: { $max: "$price" } } }
])

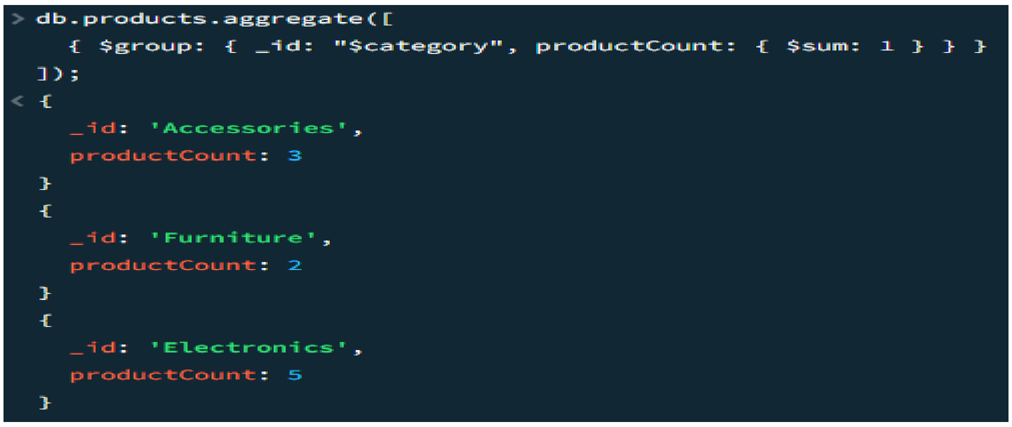

```
db.products.aggregate([
  { $group: { _id: null, minPrice: { $min: "$price" }, maxPrice: { $max: "$price" } } }
]);
{
  _id: null,
  minPrice: 25,
  maxPrice: 1000
}
```

Question 4: Write an aggregation query to count the number of products in each category in the products collection.

Count the number of products in each category:

<u>Querie:</u>

db.products.aggregate([
  { $group: { _id: "$category", count: { $sum: 1 } } }
])

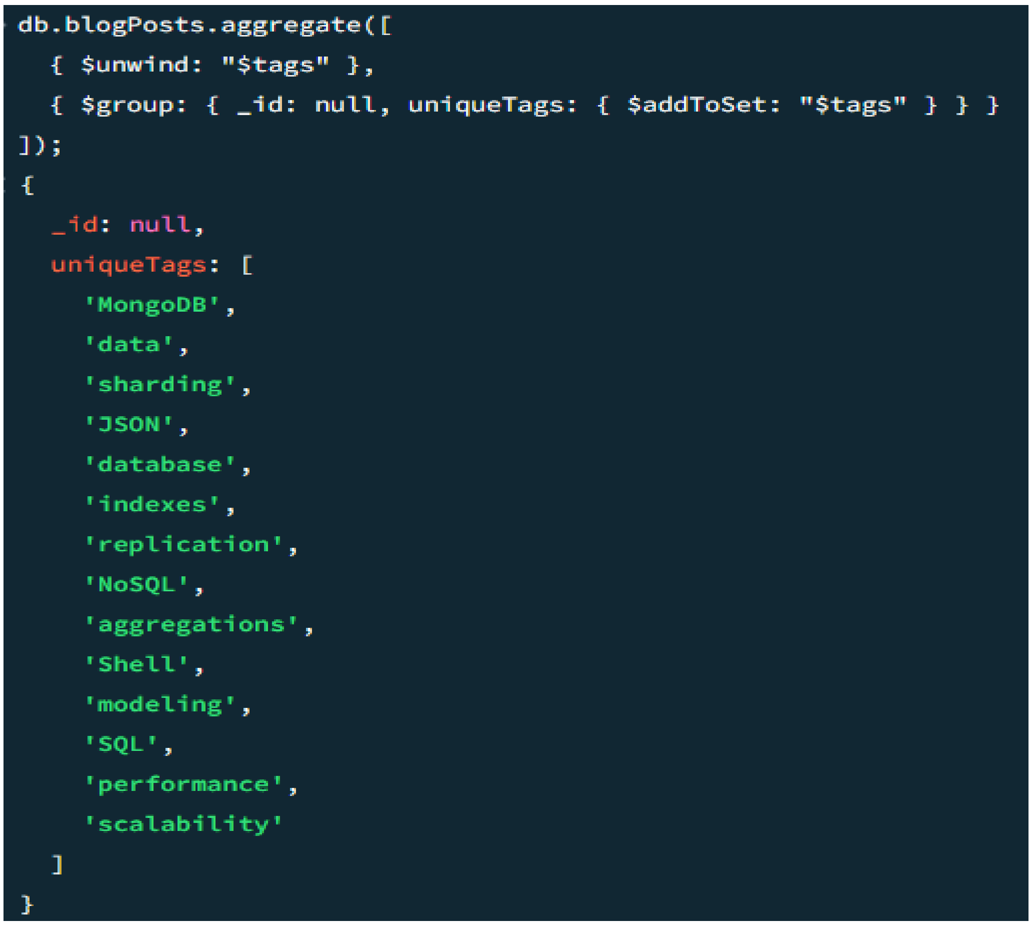

```
> db.products.aggregate([
    { $group: { _id: "$category", productCount: { $sum: 1 } } }
  ]);
< {
    _id: 'Accessories',
    productCount: 3
  }
  {
    _id: 'Furniture',
    productCount: 2
  }
  {
    _id: 'Electronics',
    productCount: 5
  }
```

Question 5: Write an aggregation query to list all unique tags used in the blogPosts collection.

List all unique tags used in the blogPosts collection:

Querie:

```
db.blogPosts.aggregate([
  { $unwind: "$tags" },
  { $group: { _id: null, uniqueTags: { $addToSet: "$tags" } } }
])
```

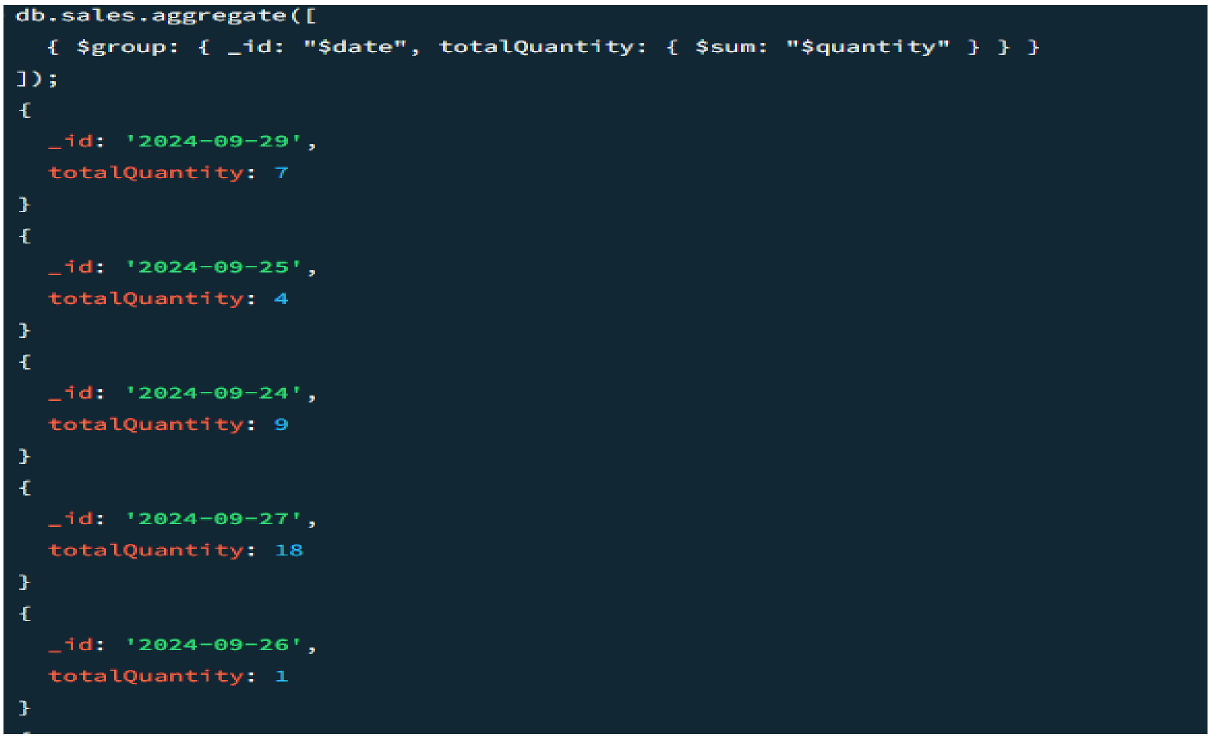

```
db.blogPosts.aggregate([
  { $unwind: "$tags" },
  { $group: { _id: null, uniqueTags: { $addToSet: "$tags" } } }
]);
{
  _id: null,
  uniqueTags: [
    'MongoDB',
    'data',
    'sharding',
    'JSON',
    'database',
    'indexes',
    'replication',
    'NoSQL',
    'aggregations',
    'Shell',
    'modeling',
    'SQL',
    'performance',
    'scalability'
  ]
}
```

Question 6: Write an aggregation query to find the total quantity sold per day from the sales collection.

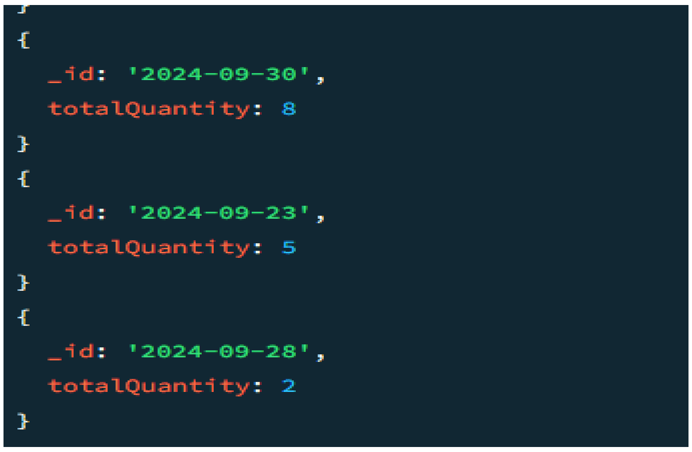
Find the total quantity sold per day:

<u>Querie:</u>

```
db.sales.aggregate([
  { $group: { _id: "$date", totalQuantity: { $sum: "$quantity" } } }
])
```



```
db.sales.aggregate([
  { $group: { _id: "$date", totalQuantity: { $sum: "$quantity" } } }
]);
{
  _id: '2024-09-29',
  totalQuantity: 7
}
{
  _id: '2024-09-25',
  totalQuantity: 4
}
{
  _id: '2024-09-24',
  totalQuantity: 9
}
{
  _id: '2024-09-27',
  totalQuantity: 18
}
{
  _id: '2024-09-26',
  totalQuantity: 1
}
```



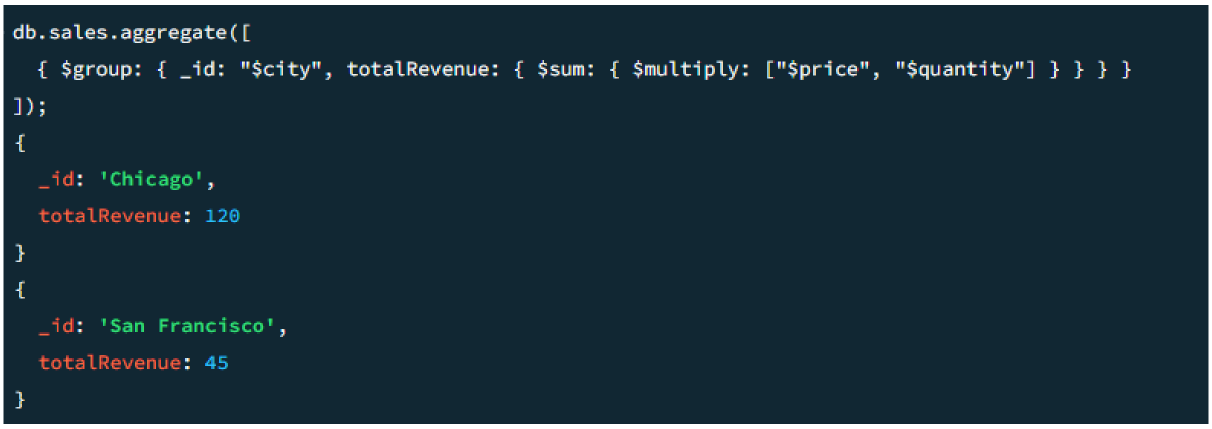
```
{
  _id: '2024-09-30',
  totalQuantity: 8
}
{
  _id: '2024-09-23',
  totalQuantity: 5
}
{
  _id: '2024-09-28',
  totalQuantity: 2
}
```

Question 7: Write an aggregation query to calculate the total revenue generated from each city in the sales collection.

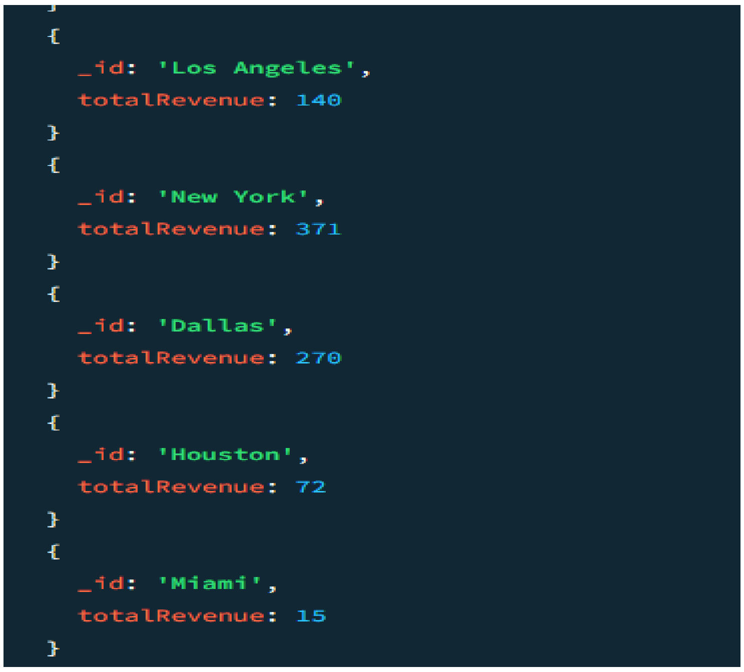
Calculate the total revenue generated from each city:

Querie:

db.sales.aggregate([
  { $group: { _id: "$city", totalRevenue: { $sum: { $multiply: ["$price", "$quantity"] } } } }
])



```
db.sales.aggregate([
  { $group: { _id: "$city", totalRevenue: { $sum: { $multiply: ["$price", "$quantity"] } } } }
]);
{
  _id: 'Chicago',
  totalRevenue: 120
}
{
  _id: 'San Francisco',
  totalRevenue: 45
}
```



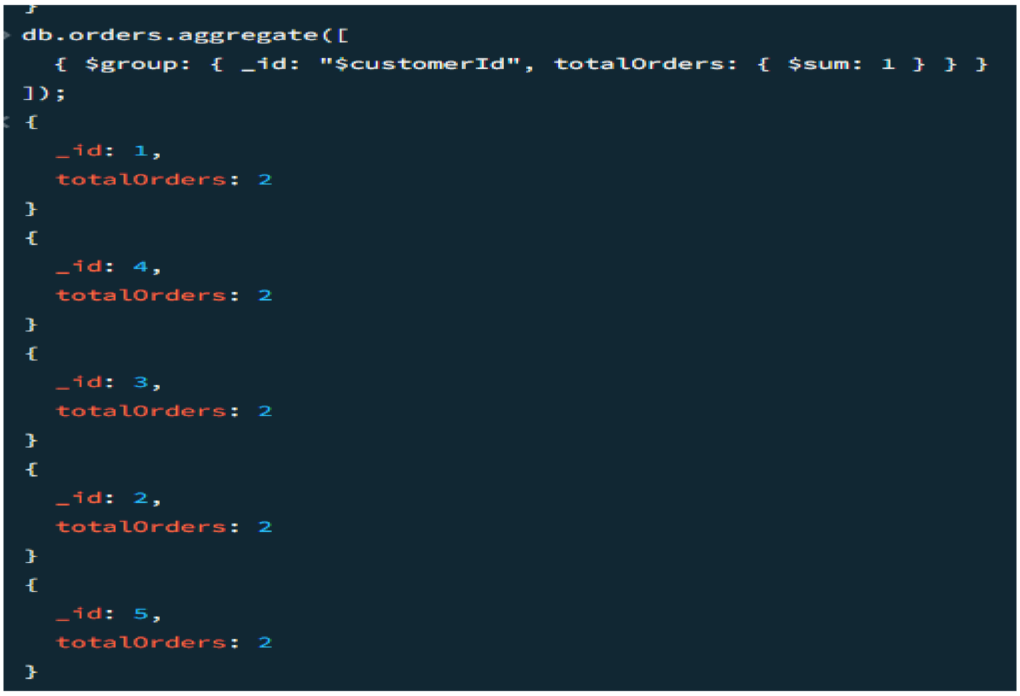
```
  {
    _id: 'Los Angeles',
    totalRevenue: 140
  }
  {
    _id: 'New York',
    totalRevenue: 371
  }
  {
    _id: 'Dallas',
    totalRevenue: 270
  }
  {
    _id: 'Houston',
    totalRevenue: 72
  }
  {
    _id: 'Miami',
    totalRevenue: 15
  }
```

Question 8: Write an aggregation query to find the total number of orders placed by each customer in the orders collection.

Find the total number of orders placed by each customer:

Querie:

db.orders.aggregate([
  { $group: { _id: "$customerId", totalOrders: { $sum: 1 } } }
])

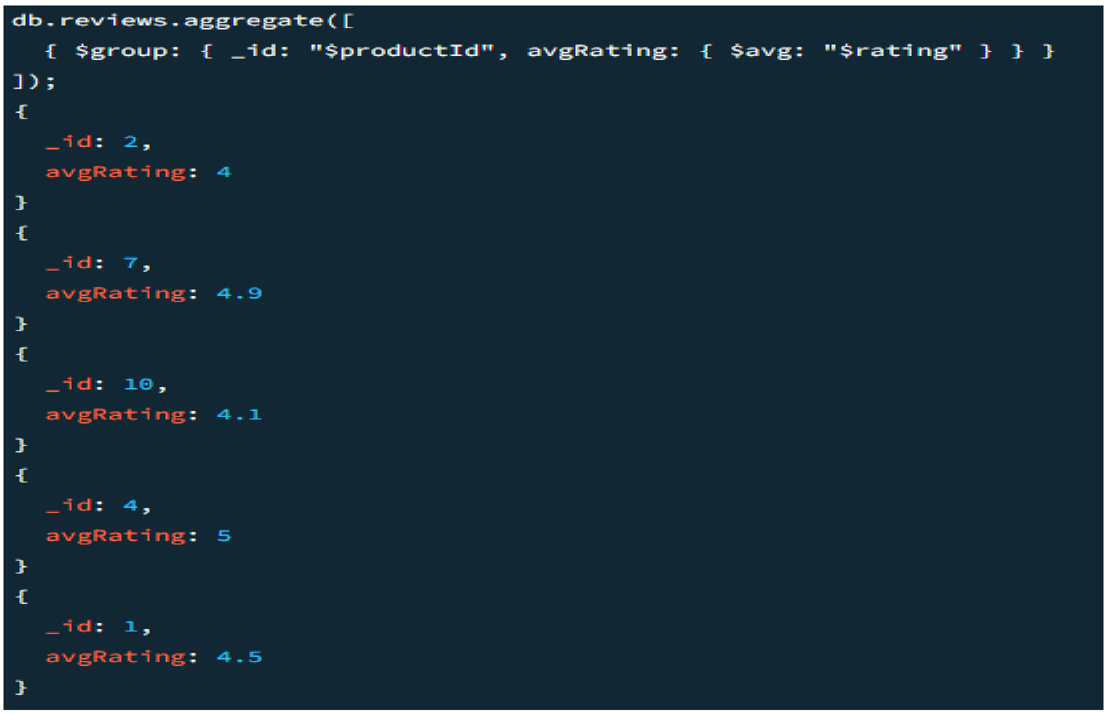

```
db.orders.aggregate([
  { $group: { _id: "$customerId", totalOrders: { $sum: 1 } } }
]);
{
  _id: 1,
  totalOrders: 2
}
{
  _id: 4,
  totalOrders: 2
}
{
  _id: 3,
  totalOrders: 2
}
{
  _id: 2,
  totalOrders: 2
}
{
  _id: 5,
  totalOrders: 2
}
```

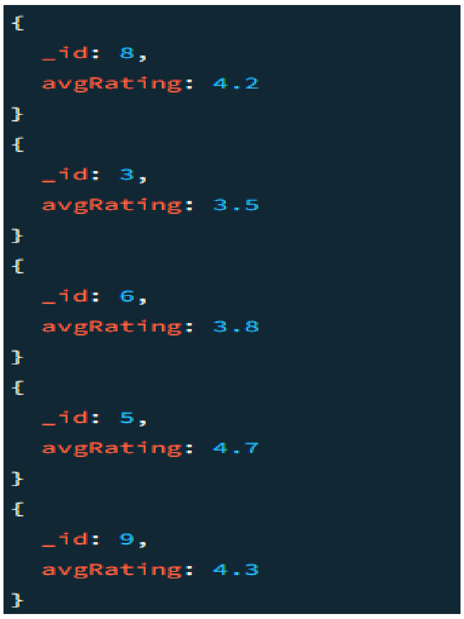
Question 9: Write an aggregation query to find the average rating for each product in the reviews collection.

Find the average rating for each product:

Querie:

```
db.reviews.aggregate([
  { $group: { _id: "$productId", avgRating: { $avg: "$rating" } } }
])
```


```
db.reviews.aggregate([
  { $group: { _id: "$productId", avgRating: { $avg: "$rating" } } }
]);
{
  _id: 2,
  avgRating: 4
}
{
  _id: 7,
  avgRating: 4.9
}
{
  _id: 10,
  avgRating: 4.1
}
{
  _id: 4,
  avgRating: 5
}
{
  _id: 1,
  avgRating: 4.5
}
```

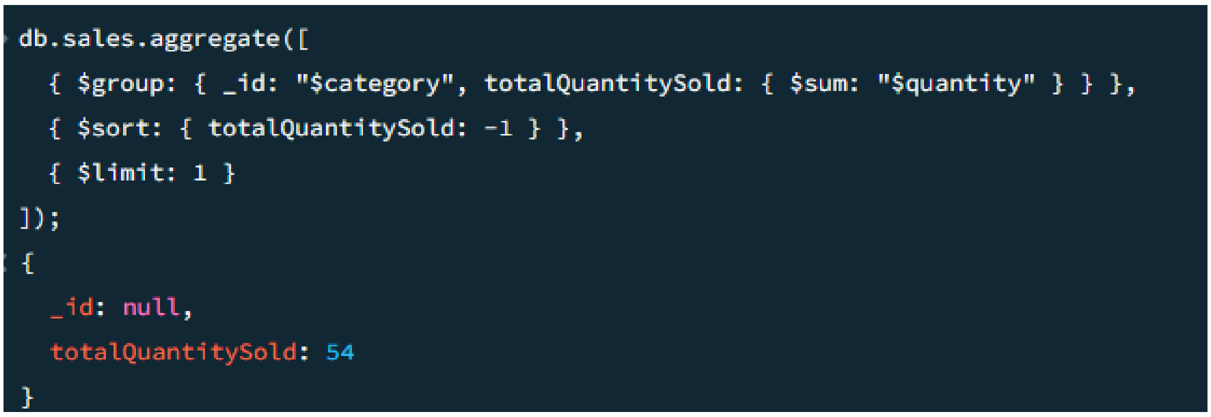

```
{
  _id: 8,
  avgRating: 4.2
}
{
  _id: 3,
  avgRating: 3.5
}
{
  _id: 6,
  avgRating: 3.8
}
{
  _id: 5,
  avgRating: 4.7
}
{
  _id: 9,
  avgRating: 4.3
}
```

Question 10: Write an aggregation query to find the most popular category based on the total number of products sold in the sales collection.

Find the most popular category based on the total number of products sold:

<u>Querie:</u>

```
db.sales.aggregate([
  { $group: { _id: "$category", totalSold: { $sum: "$quantity" } } },
  { $sort: { totalSold: -1 } },
  { $limit: 1 }
])
```


```
db.sales.aggregate([
  { $group: { _id: "$category", totalQuantitySold: { $sum: "$quantity" } } },
  { $sort: { totalQuantitySold: -1 } },
  { $limit: 1 }
]);
{
  _id: null,
  totalQuantitySold: 54
}
```