<u>Module : 04</u>

<u>MongoDB Exercise</u>

Name : Anubhav Ranjan ***Registration No : 774***

College : IIIT kalyani ***Roll No : ECE/21114***

<u>Scenario: Online Shopping Platform</u>

Background:

You are managing a MongoDB database for an online shopping platform. The database contains the following collections:

1. users: Stores user details.

2. orders: Stores order information.

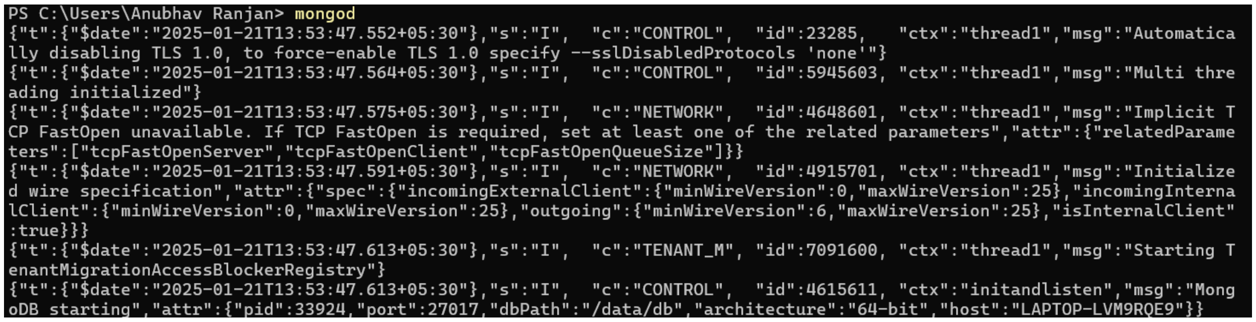
3. products: Stores product information.

SetUp:

Open the command prompt/terminal to start the server:

> mongod

Output: Server is running at the default port.

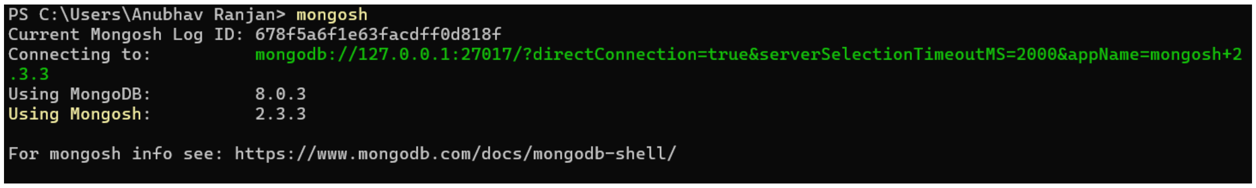


```
PS C:\Users\Anubhav Ranjan> mongod
{"t":{"$date":"2025-01-21T13:53:47.552+05:30"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"thread1","msg":"Automatica
lly disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2025-01-21T13:53:47.564+05:30"},"s":"I",  "c":"CONTROL",  "id":5945603, "ctx":"thread1","msg":"Multi thre
ading initialized"}
{"t":{"$date":"2025-01-21T13:53:47.575+05:30"},"s":"I",  "c":"NETWORK",  "id":4648601, "ctx":"thread1","msg":"Implicit T
CP FastOpen unavailable. If TCP FastOpen is required, set at least one of the related parameters","attr":{"relatedParame
ters":["tcpFastOpenServer","tcpFastOpenClient","tcpFastOpenQueueSize"]}}
{"t":{"$date":"2025-01-21T13:53:47.591+05:30"},"s":"I",  "c":"NETWORK",  "id":4915701, "ctx":"thread1","msg":"Initialize
d wire specification","attr":{"spec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":25},"incomingInterna
lClient":{"minWireVersion":0,"maxWireVersion":25},"outgoing":{"minWireVersion":6,"maxWireVersion":25},"isInternalClient"
:true}}}
{"t":{"$date":"2025-01-21T13:53:47.613+05:30"},"s":"I",  "c":"TENANT_M", "id":7091600, "ctx":"thread1","msg":"Starting T
enantMigrationAccessBlockerRegistry"}
{"t":{"$date":"2025-01-21T13:53:47.613+05:30"},"s":"I",  "c":"CONTROL",  "id":4615611, "ctx":"initandlisten","msg":"Mong
oDB starting","attr":{"pid":33924,"port":27017,"dbPath":"/data/db","architecture":"64-bit","host":"LAPTOP-LVM9RQE9"}}
```

Open another command prompt/terminal to play with data using MongoDB Shell:

> mongosh

Output: MongoDB Shell has started, Read and Write access is available.

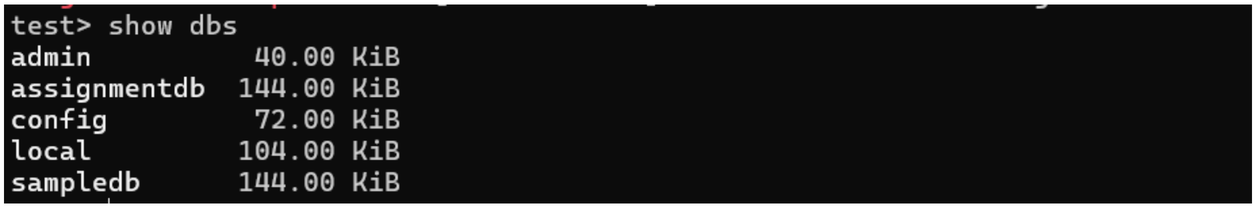


```
PS C:\Users\Anubhav Ranjan> mongosh
Current Mongosh Log ID: 678f5a6f1e63facdff0d818f
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2
.3.3
Using MongoDB:          8.0.3
Using Mongosh:          2.3.3

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/
```

> show dbs

Output:



```
test> show dbs
admin           40.00 KiB
assignmentdb   144.00 KiB
config          72.00 KiB
local          104.00 KiB
sampledb       144.00 KiB
```

-- Switch to or Create New Database:

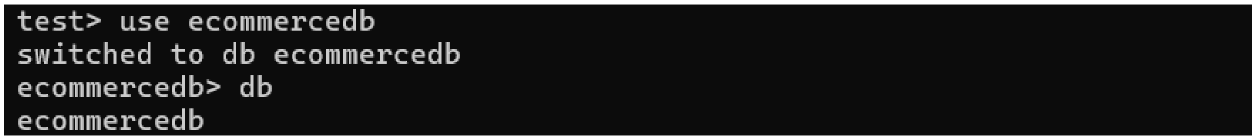
test> use ecommercedb

Output: switched to db ecommercedb

-- To check under which database I am:

ecommercedb> db

Output: ecommercedb

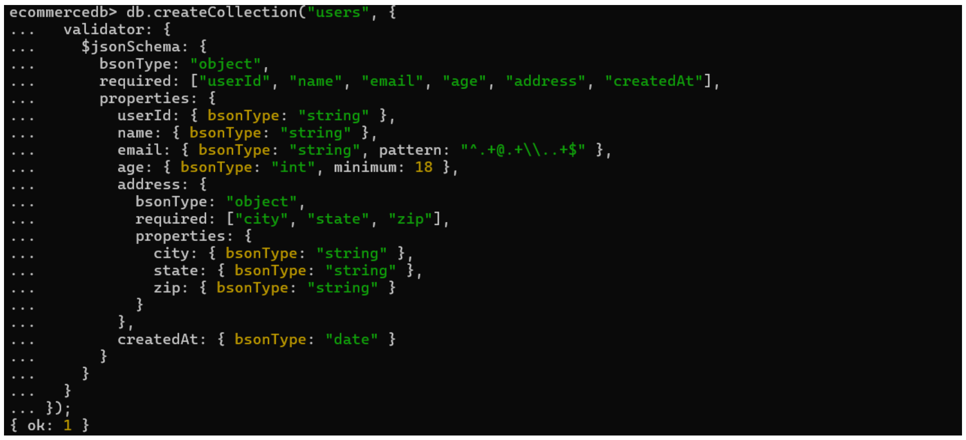


```
test> use ecommercedb
switched to db ecommercedb
ecommercedb> db
ecommercedb
```

-- Now We will Create the Collections for Users, Orders and Products..

- Cmd For Users:

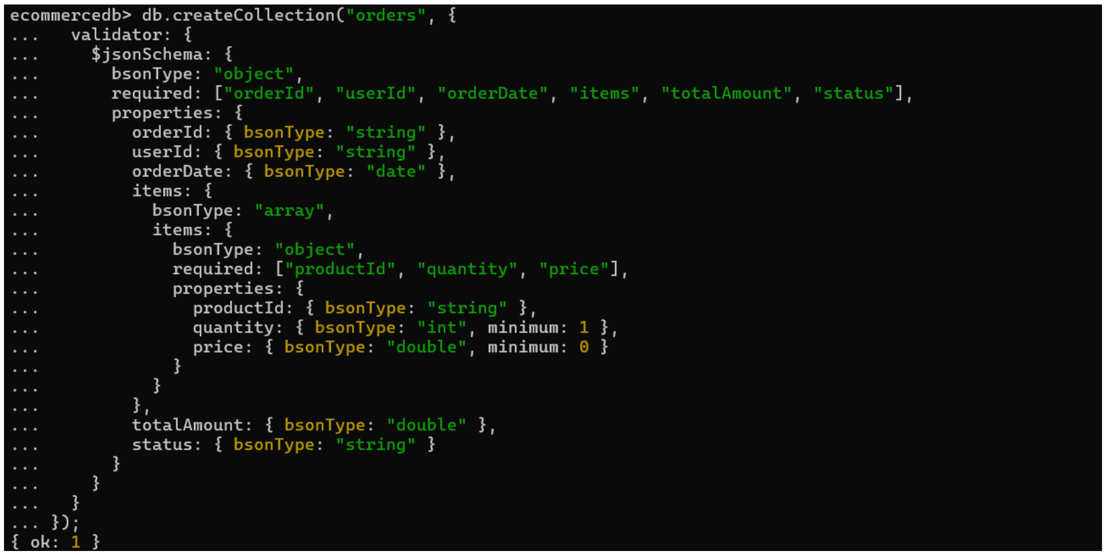
```
db.createCollection("users", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["userId", "name", "email", "age", "address", "createdAt"],
      properties: {
        userId: { bsonType: "string" },
        name: { bsonType: "string" },
        email: { bsonType: "string", pattern: "^.+@.+\\..+$" },
        age: { bsonType: "int", minimum: 18 },
        address: {
          bsonType: "object",
          required: ["city", "state", "zip"],
          properties: {
            city: { bsonType: "string" },
            state: { bsonType: "string" },
            zip: { bsonType: "string" }
          }
        },
        createdAt: { bsonType: "date" }
      }
    }
  }
});
```



```
ecommercedb> db.createCollection("users", {
...   validator: {
...     $jsonSchema: {
...       bsonType: "object",
...       required: ["userId", "name", "email", "age", "address", "createdAt"],
...       properties: {
...         userId: { bsonType: "string" },
...         name: { bsonType: "string" },
...         email: { bsonType: "string", pattern: "^.+@.+\\..+$" },
...         age: { bsonType: "int", minimum: 18 },
...         address: {
...           bsonType: "object",
...           required: ["city", "state", "zip"],
...           properties: {
...             city: { bsonType: "string" },
...             state: { bsonType: "string" },
...             zip: { bsonType: "string" }
...           }
...         },
...         createdAt: { bsonType: "date" }
...       }
...     }
...   }
... });
{ ok: 1 }
```

- Cmd For Orders:

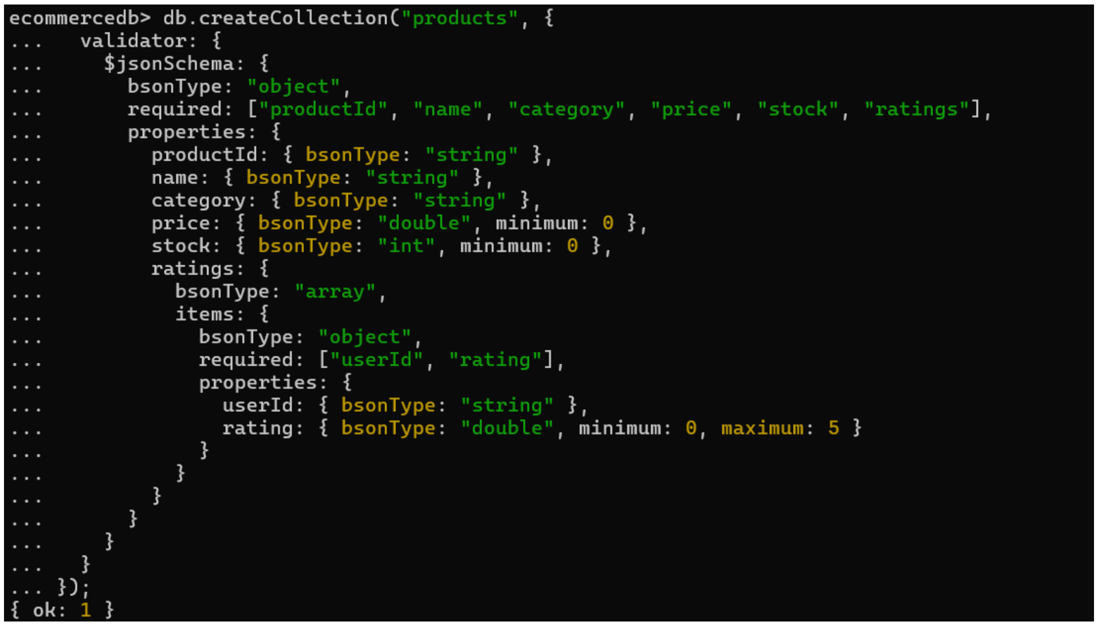
```
db.createCollection("orders", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["orderId", "userId", "orderDate", "items", "totalAmount", "status"],
      properties: {
        orderId: { bsonType: "string" },
        userId: { bsonType: "string" },
        orderDate: { bsonType: "date" },
        items: {
          bsonType: "array",
          items: {
            bsonType: "object",
            required: ["productId", "quantity", "price"],
            properties: {
              productId: { bsonType: "string" },
              quantity: { bsonType: "int", minimum: 1 },
              price: { bsonType: "double", minimum: 0 }
            }
          }
        },
        totalAmount: { bsonType: "double" },
        status: { bsonType: "string" }
      }
    }
  }
});
```



```
ecommercedb> db.createCollection("orders", {
...   validator: {
...     $jsonSchema: {
...       bsonType: "object",
...       required: ["orderId", "userId", "orderDate", "items", "totalAmount", "status"],
...       properties: {
...         orderId: { bsonType: "string" },
...         userId: { bsonType: "string" },
...         orderDate: { bsonType: "date" },
...         items: {
...           bsonType: "array",
...           items: {
...             bsonType: "object",
...             required: ["productId", "quantity", "price"],
...             properties: {
...               productId: { bsonType: "string" },
...               quantity: { bsonType: "int", minimum: 1 },
...               price: { bsonType: "double", minimum: 0 }
...             }
...           }
...         },
...         totalAmount: { bsonType: "double" },
...         status: { bsonType: "string" }
...       }
...     }
...   }
... });
{ ok: 1 }
```

- Cmd For Products:

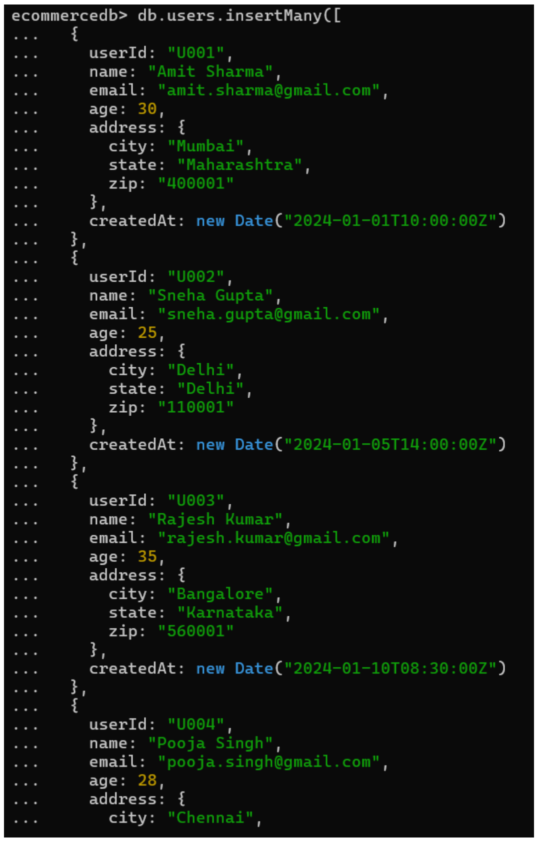
```
db.createCollection("products", {
  validator: {
    $jsonSchema: {
        bsonType: "object",
        required: ["productId", "name", "category", "price", "stock", "ratings"],
        properties: {
          productId: { bsonType: "string" },
          name: { bsonType: "string" },
          category: { bsonType: "string" },
          price: { bsonType: "double", minimum: 0 },
          stock: { bsonType: "int", minimum: 0 },
          ratings: {
            bsonType: "array",
            items: {
              bsonType: "object",
              required: ["userId", "rating"],
              properties: {
                userId: { bsonType: "string" },
                rating: { bsonType: "double", minimum: 0, maximum: 5 }
              }
            }
          }
        }
      }
    }
  });
```



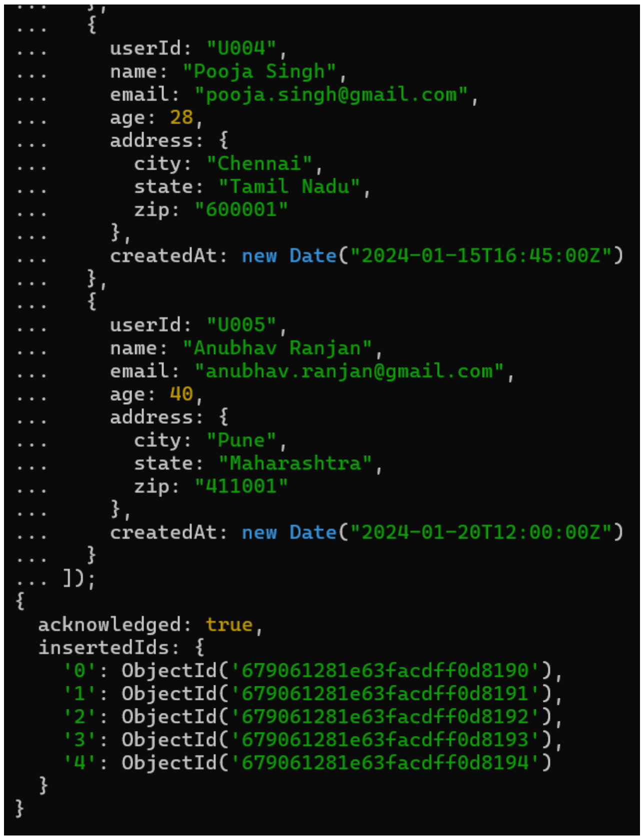
```
ecommercedb> db.createCollection("products", {
...   validator: {
...     $jsonSchema: {
...       bsonType: "object",
...       required: ["productId", "name", "category", "price", "stock", "ratings"],
...       properties: {
...         productId: { bsonType: "string" },
...         name: { bsonType: "string" },
...         category: { bsonType: "string" },
...         price: { bsonType: "double", minimum: 0 },
...         stock: { bsonType: "int", minimum: 0 },
...         ratings: {
...           bsonType: "array",
...           items: {
...             bsonType: "object",
...             required: ["userId", "rating"],
...             properties: {
...               userId: { bsonType: "string" },
...               rating: { bsonType: "double", minimum: 0, maximum: 5 }
...             }
...           }
...         }
...       }
...     }
...   }
... });
{ ok: 1 }
```

-- Now We will Populate the Collections for Users, Orders and Products with Relevant Data..

- For Users:

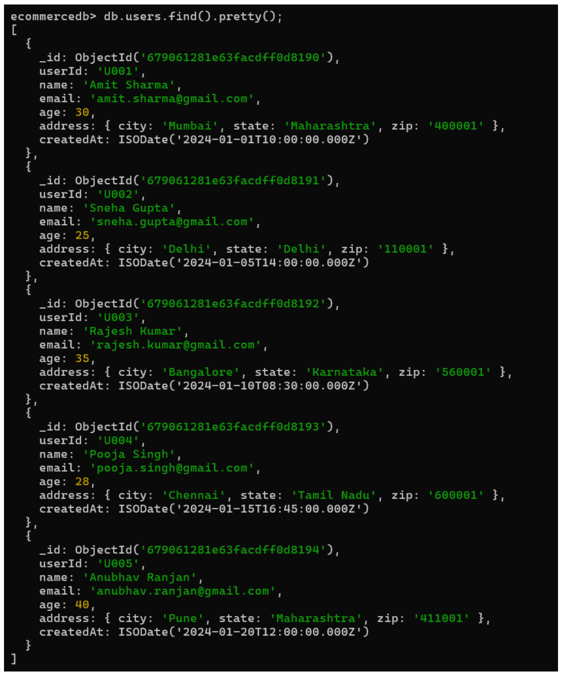


```
ecommercedb> db.users.insertMany([
...   {
...     userId: "U001",
...     name: "Amit Sharma",
...     email: "amit.sharma@gmail.com",
...     age: 30,
...     address: {
...       city: "Mumbai",
...       state: "Maharashtra",
...       zip: "400001"
...     },
...     createdAt: new Date("2024-01-01T10:00:00Z")
...   },
...   {
...     userId: "U002",
...     name: "Sneha Gupta",
...     email: "sneha.gupta@gmail.com",
...     age: 25,
...     address: {
...       city: "Delhi",
...       state: "Delhi",
...       zip: "110001"
...     },
...     createdAt: new Date("2024-01-05T14:00:00Z")
...   },
...   {
...     userId: "U003",
...     name: "Rajesh Kumar",
...     email: "rajesh.kumar@gmail.com",
...     age: 35,
...     address: {
...       city: "Bangalore",
...       state: "Karnataka",
...       zip: "560001"
...     },
...     createdAt: new Date("2024-01-10T08:30:00Z")
...   },
...   {
...     userId: "U004",
...     name: "Pooja Singh",
...     email: "pooja.singh@gmail.com",
...     age: 28,
...     address: {
...       city: "Chennai",
```



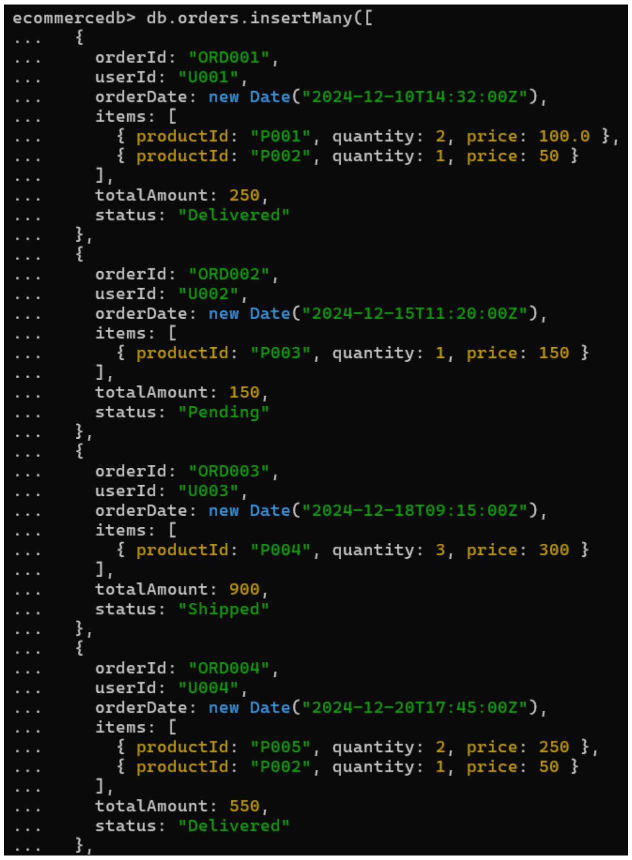
```
...   },
...   {
...     userId: "U004",
...     name: "Pooja Singh",
...     email: "pooja.singh@gmail.com",
...     age: 28,
...     address: {
...       city: "Chennai",
...       state: "Tamil Nadu",
...       zip: "600001"
...     },
...     createdAt: new Date("2024-01-15T16:45:00Z")
...   },
...   {
...     userId: "U005",
...     name: "Anubhav Ranjan",
...     email: "anubhav.ranjan@gmail.com",
...     age: 40,
...     address: {
...       city: "Pune",
...       state: "Maharashtra",
...       zip: "411001"
...     },
...     createdAt: new Date("2024-01-20T12:00:00Z")
...   }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('679061281e63facdff0d8190'),
    '1': ObjectId('679061281e63facdff0d8191'),
    '2': ObjectId('679061281e63facdff0d8192'),
    '3': ObjectId('679061281e63facdff0d8193'),
    '4': ObjectId('679061281e63facdff0d8194')
  }
}
```

- View the Users Data

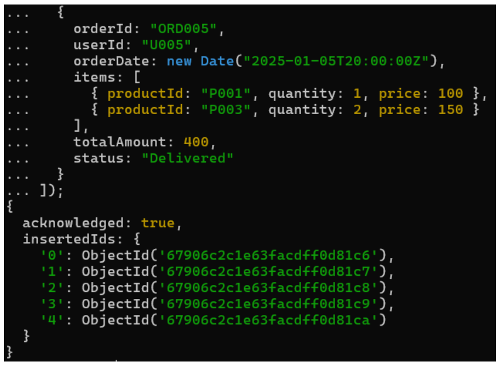


```
ecommercedb> db.users.find().pretty();
[
  {
    _id: ObjectId('679061281e63facdff0d8190'),
    userId: 'U001',
    name: 'Amit Sharma',
    email: 'amit.sharma@gmail.com',
    age: 30,
    address: { city: 'Mumbai', state: 'Maharashtra', zip: '400001' },
    createdAt: ISODate('2024-01-01T10:00:00.000Z')
  },
  {
    _id: ObjectId('679061281e63facdff0d8191'),
    userId: 'U002',
    name: 'Sneha Gupta',
    email: 'sneha.gupta@gmail.com',
    age: 25,
    address: { city: 'Delhi', state: 'Delhi', zip: '110001' },
    createdAt: ISODate('2024-01-05T14:00:00.000Z')
  },
  {
    _id: ObjectId('679061281e63facdff0d8192'),
    userId: 'U003',
    name: 'Rajesh Kumar',
    email: 'rajesh.kumar@gmail.com',
    age: 35,
    address: { city: 'Bangalore', state: 'Karnataka', zip: '560001' },
    createdAt: ISODate('2024-01-10T08:30:00.000Z')
  },
  {
    _id: ObjectId('679061281e63facdff0d8193'),
    userId: 'U004',
    name: 'Pooja Singh',
    email: 'pooja.singh@gmail.com',
    age: 28,
    address: { city: 'Chennai', state: 'Tamil Nadu', zip: '600001' },
    createdAt: ISODate('2024-01-15T16:45:00.000Z')
  },
  {
    _id: ObjectId('679061281e63facdff0d8194'),
    userId: 'U005',
    name: 'Anubhav Ranjan',
    email: 'anubhav.ranjan@gmail.com',
    age: 40,
    address: { city: 'Pune', state: 'Maharashtra', zip: '411001' },
    createdAt: ISODate('2024-01-20T12:00:00.000Z')
  }
]
```

– Similarly We will Populate Orders Data..

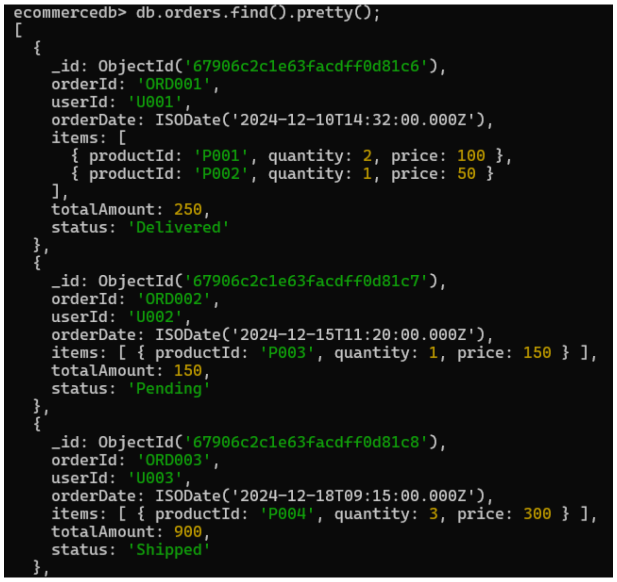


```
ecommercedb> db.orders.insertMany([
...   {
...     orderId: "ORD001",
...     userId: "U001",
...     orderDate: new Date("2024-12-10T14:32:00Z"),
...     items: [
...       { productId: "P001", quantity: 2, price: 100.0 },
...       { productId: "P002", quantity: 1, price: 50 }
...     ],
...     totalAmount: 250,
...     status: "Delivered"
...   },
...   {
...     orderId: "ORD002",
...     userId: "U002",
...     orderDate: new Date("2024-12-15T11:20:00Z"),
...     items: [
...       { productId: "P003", quantity: 1, price: 150 }
...     ],
...     totalAmount: 150,
...     status: "Pending"
...   },
...   {
...     orderId: "ORD003",
...     userId: "U003",
...     orderDate: new Date("2024-12-18T09:15:00Z"),
...     items: [
...       { productId: "P004", quantity: 3, price: 300 }
...     ],
...     totalAmount: 900,
...     status: "Shipped"
...   },
...   {
...     orderId: "ORD004",
...     userId: "U004",
...     orderDate: new Date("2024-12-20T17:45:00Z"),
...     items: [
...       { productId: "P005", quantity: 2, price: 250 },
...       { productId: "P002", quantity: 1, price: 50 }
...     ],
...     totalAmount: 550,
...     status: "Delivered"
...   },
```

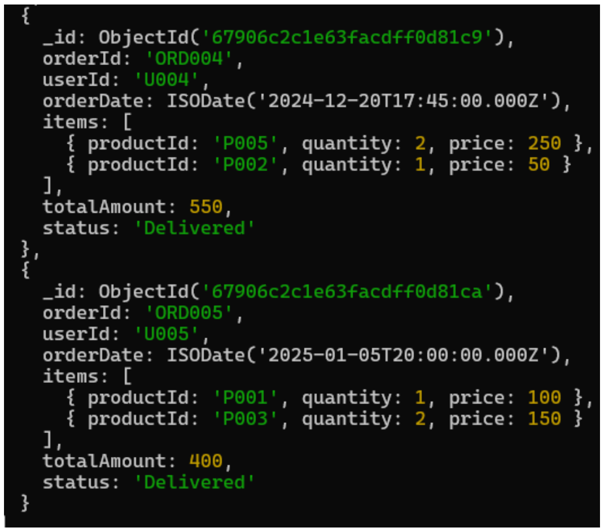


```
...   {
...     orderId: "ORD005",
...     userId: "U005",
...     orderDate: new Date("2025-01-05T20:00:00Z"),
...     items: [
...       { productId: "P001", quantity: 1, price: 100 },
...       { productId: "P003", quantity: 2, price: 150 }
...     ],
...     totalAmount: 400,
...     status: "Delivered"
...   }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67906c2c1e63facdff0d81c6'),
    '1': ObjectId('67906c2c1e63facdff0d81c7'),
    '2': ObjectId('67906c2c1e63facdff0d81c8'),
    '3': ObjectId('67906c2c1e63facdff0d81c9'),
    '4': ObjectId('67906c2c1e63facdff0d81ca')
  }
}
```

- View the Orders Data:

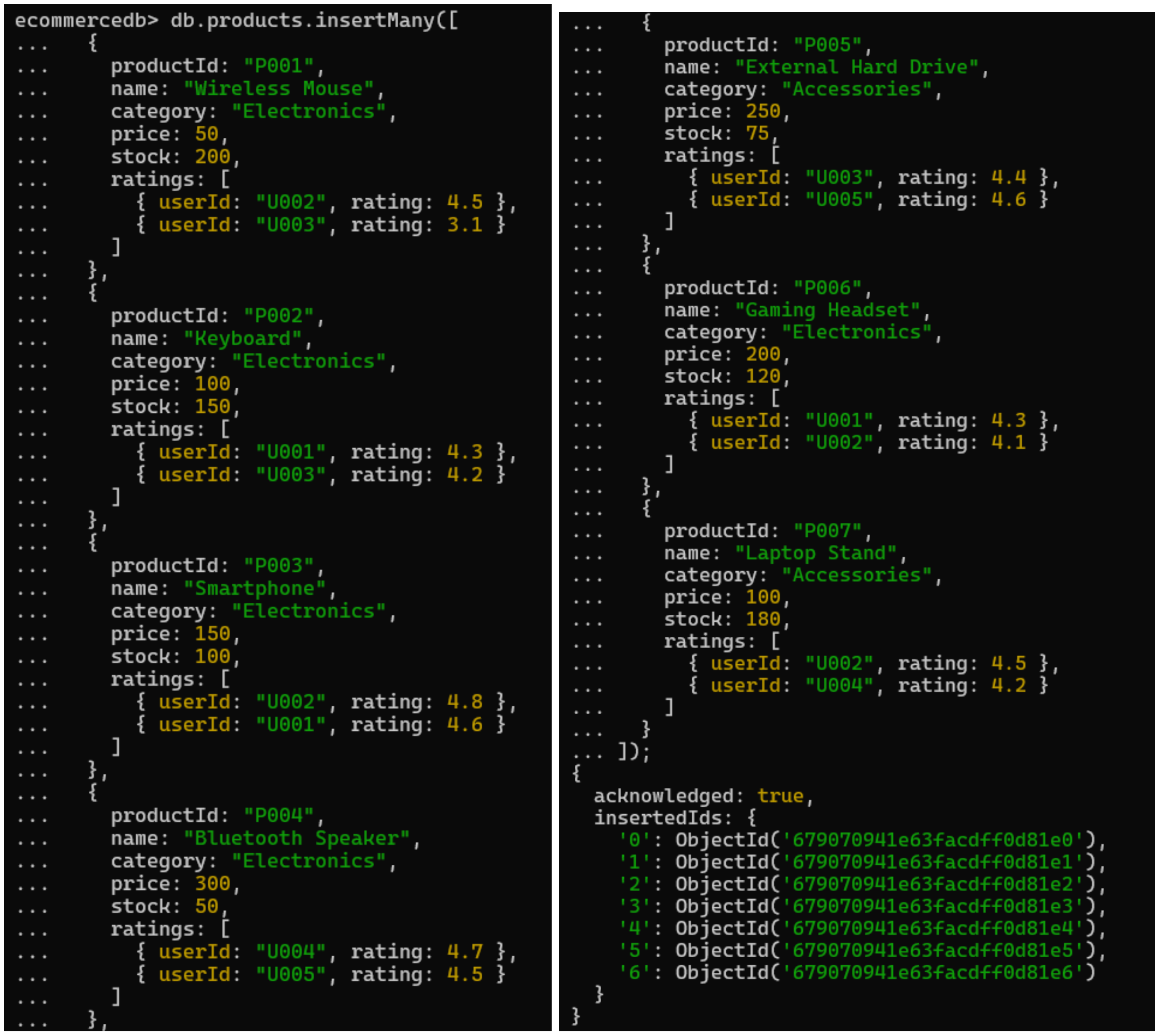


```
ecommercedb> db.orders.find().pretty();
[
  {
    _id: ObjectId('67906c2c1e63facdff0d81c6'),
    orderId: 'ORD001',
    userId: 'U001',
    orderDate: ISODate('2024-12-10T14:32:00.000Z'),
    items: [
      { productId: 'P001', quantity: 2, price: 100 },
      { productId: 'P002', quantity: 1, price: 50 }
    ],
    totalAmount: 250,
    status: 'Delivered'
  },
  {
    _id: ObjectId('67906c2c1e63facdff0d81c7'),
    orderId: 'ORD002',
    userId: 'U002',
    orderDate: ISODate('2024-12-15T11:20:00.000Z'),
    items: [ { productId: 'P003', quantity: 1, price: 150 } ],
    totalAmount: 150,
    status: 'Pending'
  },
  {
    _id: ObjectId('67906c2c1e63facdff0d81c8'),
    orderId: 'ORD003',
    userId: 'U003',
    orderDate: ISODate('2024-12-18T09:15:00.000Z'),
    items: [ { productId: 'P004', quantity: 3, price: 300 } ],
    totalAmount: 900,
    status: 'Shipped'
  },
```



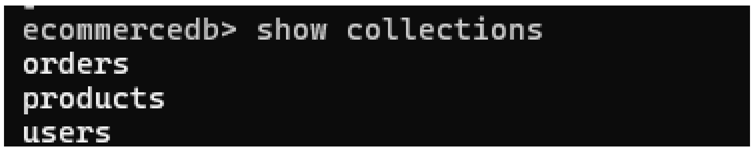
```
{
  _id: ObjectId('67906c2c1e63facdff0d81c9'),
  orderId: 'ORD004',
  userId: 'U004',
  orderDate: ISODate('2024-12-20T17:45:00.000Z'),
  items: [
    { productId: 'P005', quantity: 2, price: 250 },
    { productId: 'P002', quantity: 1, price: 50 }
  ],
  totalAmount: 550,
  status: 'Delivered'
},
{
  _id: ObjectId('67906c2c1e63facdff0d81ca'),
  orderId: 'ORD005',
  userId: 'U005',
  orderDate: ISODate('2025-01-05T20:00:00.000Z'),
  items: [
    { productId: 'P001', quantity: 1, price: 100 },
    { productId: 'P003', quantity: 2, price: 150 }
  ],
  totalAmount: 400,
  status: 'Delivered'
}
```

– Similarly We will Populate the Products Data..



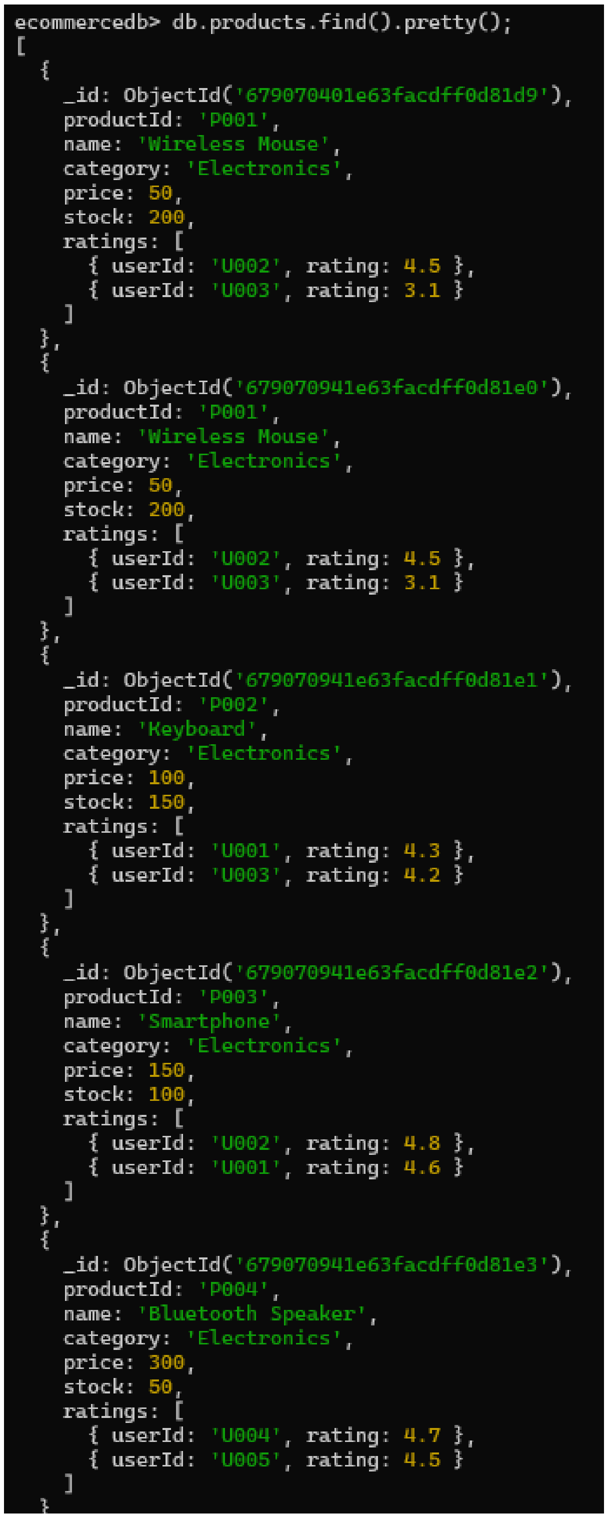
```
ecommercedb> db.products.insertMany([
...   {
...     productId: "P001",
...     name: "Wireless Mouse",
...     category: "Electronics",
...     price: 50,
...     stock: 200,
...     ratings: [
...       { userId: "U002", rating: 4.5 },
...       { userId: "U003", rating: 3.1 }
...     ]
...   },
...   {
...     productId: "P002",
...     name: "Keyboard",
...     category: "Electronics",
...     price: 100,
...     stock: 150,
...     ratings: [
...       { userId: "U001", rating: 4.3 },
...       { userId: "U003", rating: 4.2 }
...     ]
...   },
...   {
...     productId: "P003",
...     name: "Smartphone",
...     category: "Electronics",
...     price: 150,
...     stock: 100,
...     ratings: [
...       { userId: "U002", rating: 4.8 },
...       { userId: "U001", rating: 4.6 }
...     ]
...   },
...   {
...     productId: "P004",
...     name: "Bluetooth Speaker",
...     category: "Electronics",
...     price: 300,
...     stock: 50,
...     ratings: [
...       { userId: "U004", rating: 4.7 },
...       { userId: "U005", rating: 4.5 }
...     ]
...   },
...   {
...     productId: "P005",
...     name: "External Hard Drive",
...     category: "Accessories",
...     price: 250,
...     stock: 75,
...     ratings: [
...       { userId: "U003", rating: 4.4 },
...       { userId: "U005", rating: 4.6 }
...     ]
...   },
...   {
...     productId: "P006",
...     name: "Gaming Headset",
...     category: "Electronics",
...     price: 200,
...     stock: 120,
...     ratings: [
...       { userId: "U001", rating: 4.3 },
...       { userId: "U002", rating: 4.1 }
...     ]
...   },
...   {
...     productId: "P007",
...     name: "Laptop Stand",
...     category: "Accessories",
...     price: 100,
...     stock: 180,
...     ratings: [
...       { userId: "U002", rating: 4.5 },
...       { userId: "U004", rating: 4.2 }
...     ]
...   }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('679070941e63facdff0d81e0'),
    '1': ObjectId('679070941e63facdff0d81e1'),
    '2': ObjectId('679070941e63facdff0d81e2'),
    '3': ObjectId('679070941e63facdff0d81e3'),
    '4': ObjectId('679070941e63facdff0d81e4'),
    '5': ObjectId('679070941e63facdff0d81e5'),
    '6': ObjectId('679070941e63facdff0d81e6')
  }
}
```

-- To see collections:

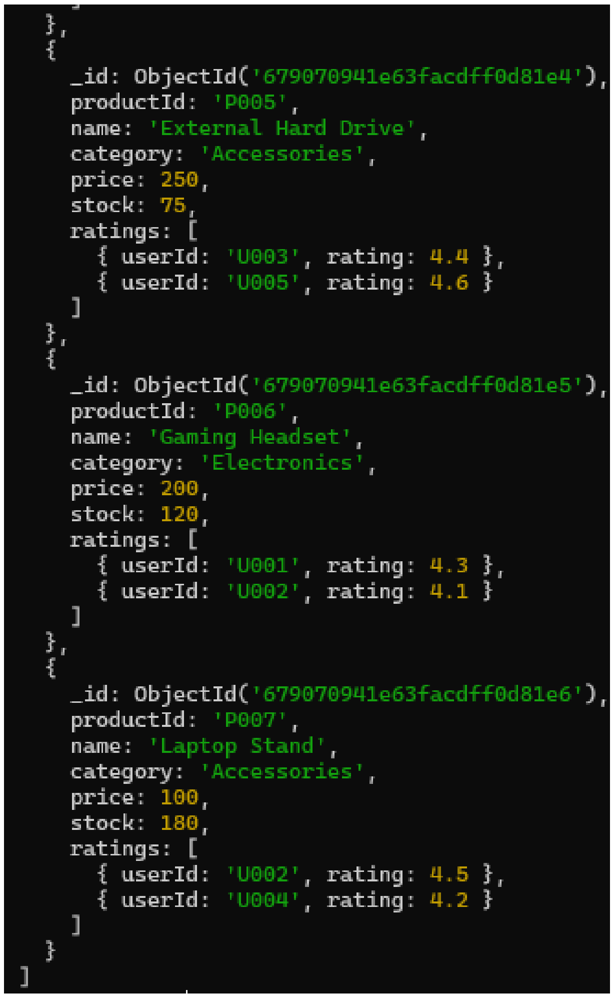


```
ecommercedb> show collections
orders
products
users
```

- **View the Products Data:**



```
ecommercedb> db.products.find().pretty();
[
  {
    _id: ObjectId('679070401e63facdff0d81d9'),
    productId: 'P001',
    name: 'Wireless Mouse',
    category: 'Electronics',
    price: 50,
    stock: 200,
    ratings: [
      { userId: 'U002', rating: 4.5 },
      { userId: 'U003', rating: 3.1 }
    ]
  },
  {
    _id: ObjectId('679070941e63facdff0d81e0'),
    productId: 'P001',
    name: 'Wireless Mouse',
    category: 'Electronics',
    price: 50,
    stock: 200,
    ratings: [
      { userId: 'U002', rating: 4.5 },
      { userId: 'U003', rating: 3.1 }
    ]
  },
  {
    _id: ObjectId('679070941e63facdff0d81e1'),
    productId: 'P002',
    name: 'Keyboard',
    category: 'Electronics',
    price: 100,
    stock: 150,
    ratings: [
      { userId: 'U001', rating: 4.3 },
      { userId: 'U003', rating: 4.2 }
    ]
  },
  {
    _id: ObjectId('679070941e63facdff0d81e2'),
    productId: 'P003',
    name: 'Smartphone',
    category: 'Electronics',
    price: 150,
    stock: 100,
    ratings: [
      { userId: 'U002', rating: 4.8 },
      { userId: 'U001', rating: 4.6 }
    ]
  },
  {
    _id: ObjectId('679070941e63facdff0d81e3'),
    productId: 'P004',
    name: 'Bluetooth Speaker',
    category: 'Electronics',
    price: 300,
    stock: 50,
    ratings: [
      { userId: 'U004', rating: 4.7 },
      { userId: 'U005', rating: 4.5 }
    ]
  }
```

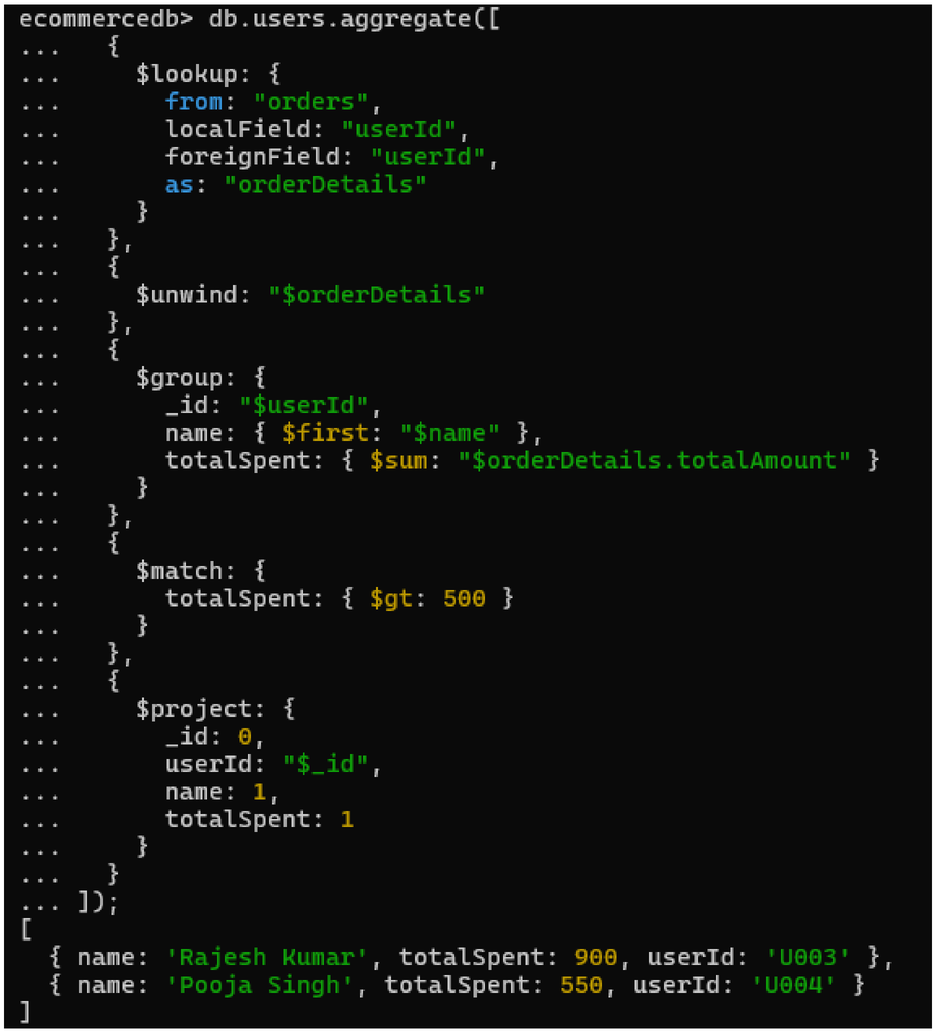


```
  },
  {
    _id: ObjectId('679070941e63facdff0d81e4'),
    productId: 'P005',
    name: 'External Hard Drive',
    category: 'Accessories',
    price: 250,
    stock: 75,
    ratings: [
      { userId: 'U003', rating: 4.4 },
      { userId: 'U005', rating: 4.6 }
    ]
  },
  {
    _id: ObjectId('679070941e63facdff0d81e5'),
    productId: 'P006',
    name: 'Gaming Headset',
    category: 'Electronics',
    price: 200,
    stock: 120,
    ratings: [
      { userId: 'U001', rating: 4.3 },
      { userId: 'U002', rating: 4.1 }
    ]
  },
  {
    _id: ObjectId('679070941e63facdff0d81e6'),
    productId: 'P007',
    name: 'Laptop Stand',
    category: 'Accessories',
    price: 100,
    stock: 180,
    ratings: [
      { userId: 'U002', rating: 4.5 },
      { userId: 'U004', rating: 4.2 }
    ]
  }
]
```

Queries:

1. Find High-Spending Users
Write a query to find users who have spent more than $500 in total across all their orders.
Hint: Use $lookup to join the users and orders collections and calculate the total Spending.

```
-> db.users.aggregate([
 {
  $lookup: {
    from: "orders",
    localField: "userId",
    foreignField: "userId",
    as: "orderDetails"
  }
 },
 {
  $unwind: "$orderDetails"
 },
 {
  $group: {
    _id: "$userId",
    name: { $first: "$name" },
    totalSpent: { $sum: "$orderDetails.totalAmount" }
  }
 },
 {
  $match: {
    totalSpent: { $gt: 500 }
  }
 },
 {
  $project: {
    _id: 0,
    userId: "$_id",
    name: 1,
    totalSpent: 1
  }
 }
]);
```



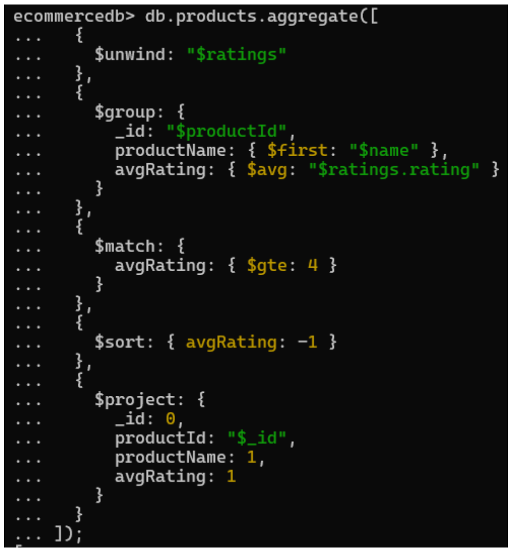
```
ecommercedb> db.users.aggregate([
...   {
...     $lookup: {
...       from: "orders",
...       localField: "userId",
...       foreignField: "userId",
...       as: "orderDetails"
...     }
...   },
...   {
...     $unwind: "$orderDetails"
...   },
...   {
...     $group: {
...       _id: "$userId",
...       name: { $first: "$name" },
...       totalSpent: { $sum: "$orderDetails.totalAmount" }
...     }
...   },
...   {
...     $match: {
...       totalSpent: { $gt: 500 }
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       userId: "$_id",
...       name: 1,
...       totalSpent: 1
...     }
...   }
... ]);
[
  { name: 'Rajesh Kumar', totalSpent: 900, userId: 'U003' },
  { name: 'Pooja Singh', totalSpent: 550, userId: 'U004' }
]
```

2. List Popular Products by Average Rating:

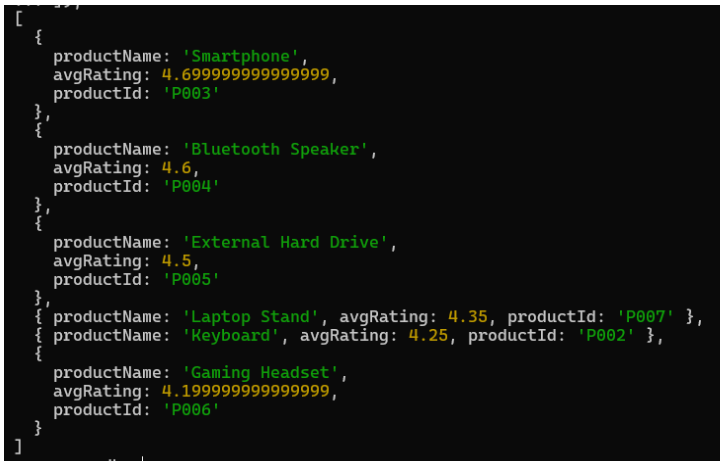
Retrieve products that have an average rating greater than or equal to 4.

Hint: Use $unwind to flatten the ratings array and $group to calculate the average rating.

```
-> db.products.aggregate([
 {
  $unwind: "$ratings"
 },
 {
  $group: {
   _id: "$productId",
   productName: { $first: "$name" },
   avgRating: { $avg: "$ratings.rating" }
  }
 },
 {
  $match: {
      avgRating: { $gte: 4 }
    }
  },
  {
    $sort: { avgRating: -1 }
  },
  {
    $project: {
      _id: 0,
      productId: "$_id",
      productName: 1,
      avgRating: 1
    }
  }
]);
```



```
ecommercedb> db.products.aggregate([
...   {
...     $unwind: "$ratings"
...   },
...   {
...     $group: {
...       _id: "$productId",
...       productName: { $first: "$name" },
...       avgRating: { $avg: "$ratings.rating" }
...     }
...   },
...   {
...     $match: {
...       avgRating: { $gte: 4 }
...     }
...   },
...   {
...     $sort: { avgRating: -1 }
...   },
...   {
...     $project: {
...       _id: 0,
...       productId: "$_id",
...       productName: 1,
...       avgRating: 1
...     }
...   }
... ]);
```



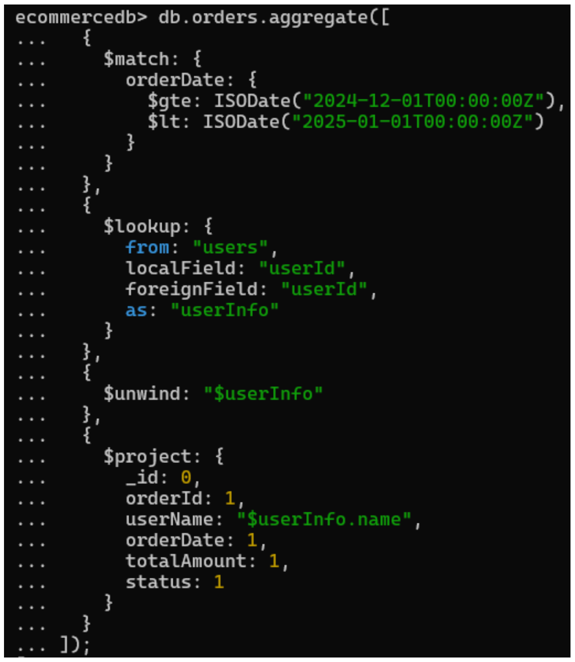
```
[
  {
    productName: 'Smartphone',
    avgRating: 4.6999999999999999,
    productId: 'P003'
  },
  {
    productName: 'Bluetooth Speaker',
    avgRating: 4.6,
    productId: 'P004'
  },
  {
    productName: 'External Hard Drive',
    avgRating: 4.5,
    productId: 'P005'
  },
  { productName: 'Laptop Stand', avgRating: 4.35, productId: 'P007' },
  { productName: 'Keyboard', avgRating: 4.25, productId: 'P002' },
  {
    productName: 'Gaming Headset',
    avgRating: 4.1999999999999999,
    productId: 'P006'
  }
]
```

3. Search for Orders in a Specific Time Range:

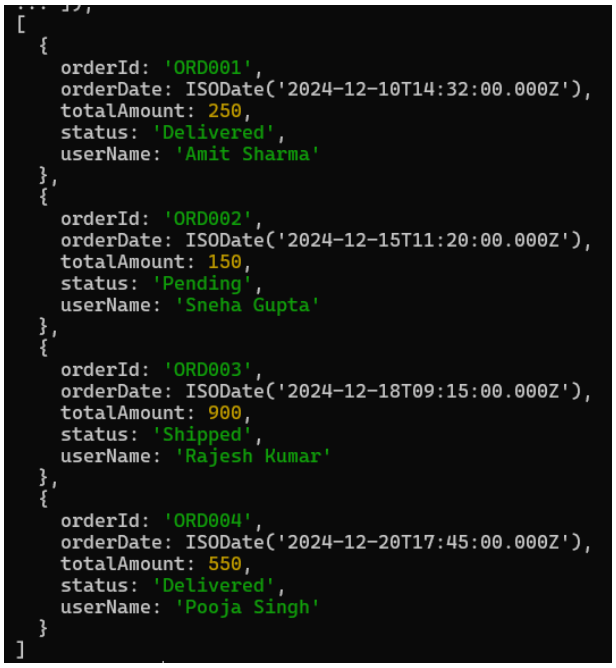
Find all orders placed between "2024-12-01" and "2024-12-31". Ensure the result includes the user name for each order.

Hint: Use $match with a date range filter and $lookup to join with the users collection.

```
-> db.orders.aggregate([
  {
    $match: {
      orderDate: {
        $gte: ISODate("2024-12-01T00:00:00Z"),
        $lt: ISODate("2025-01-01T00:00:00Z")
      }
    }
  },
 {
  $lookup: {
   from: "users",
   localField: "userId",
   foreignField: "userId",
   as: "userInfo"
  }
 },
 {
  $unwind: "$userInfo"
 },
 {
  $project: {
   _id: 0,
   orderId: 1,
   userName: "$userInfo.name",
   orderDate: 1,
   totalAmount: 1,
   status: 1
  }
 }
]);
```



```
ecommercedb> db.orders.aggregate([
...   {
...     $match: {
...       orderDate: {
...         $gte: ISODate("2024-12-01T00:00:00Z"),
...         $lt: ISODate("2025-01-01T00:00:00Z")
...       }
...     }
...   },
...   {
...     $lookup: {
...       from: "users",
...       localField: "userId",
...       foreignField: "userId",
...       as: "userInfo"
...     }
...   },
...   {
...     $unwind: "$userInfo"
...   },
...   {
...     $project: {
...       _id: 0,
...       orderId: 1,
...       userName: "$userInfo.name",
...       orderDate: 1,
...       totalAmount: 1,
...       status: 1
...     }
...   }
... ]);
```



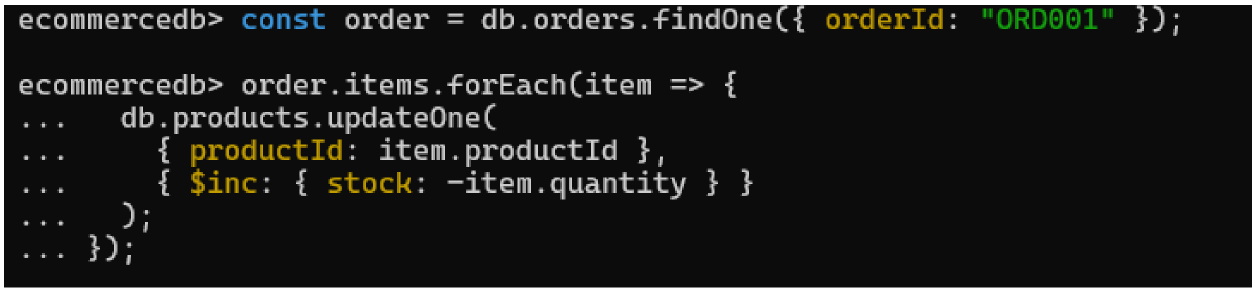
```
[
  {
    orderId: 'ORD001',
    orderDate: ISODate('2024-12-10T14:32:00.000Z'),
    totalAmount: 250,
    status: 'Delivered',
    userName: 'Amit Sharma'
  },
  {
    orderId: 'ORD002',
    orderDate: ISODate('2024-12-15T11:20:00.000Z'),
    totalAmount: 150,
    status: 'Pending',
    userName: 'Sneha Gupta'
  },
  {
    orderId: 'ORD003',
    orderDate: ISODate('2024-12-18T09:15:00.000Z'),
    totalAmount: 900,
    status: 'Shipped',
    userName: 'Rajesh Kumar'
  },
  {
    orderId: 'ORD004',
    orderDate: ISODate('2024-12-20T17:45:00.000Z'),
    totalAmount: 550,
    status: 'Delivered',
    userName: 'Pooja Singh'
  }
]
```

4. Update Stock After Order Completion :

When an order is placed, reduce the stock of each product by the quantity in the order. For example, if 2 units of P001 were purchased, decrement its stock by 2.

Hint: Use $inc with updateOne or updateMany.

```
-> const order = db.orders.findOne({ orderId: "ORD001" });
order.items.forEach(item => {
  db.products.updateOne(
    { productId: item.productId },
    { $inc: { stock: -item.quantity } }
  );
});
```



```
ecommercedb> const order = db.orders.findOne({ orderId: "ORD001" });

ecommercedb> order.items.forEach(item => {
...   db.products.updateOne(
...     { productId: item.productId },
...     { $inc: { stock: -item.quantity } }
...   );
... });
```

5. Find Nearest Warehouse :

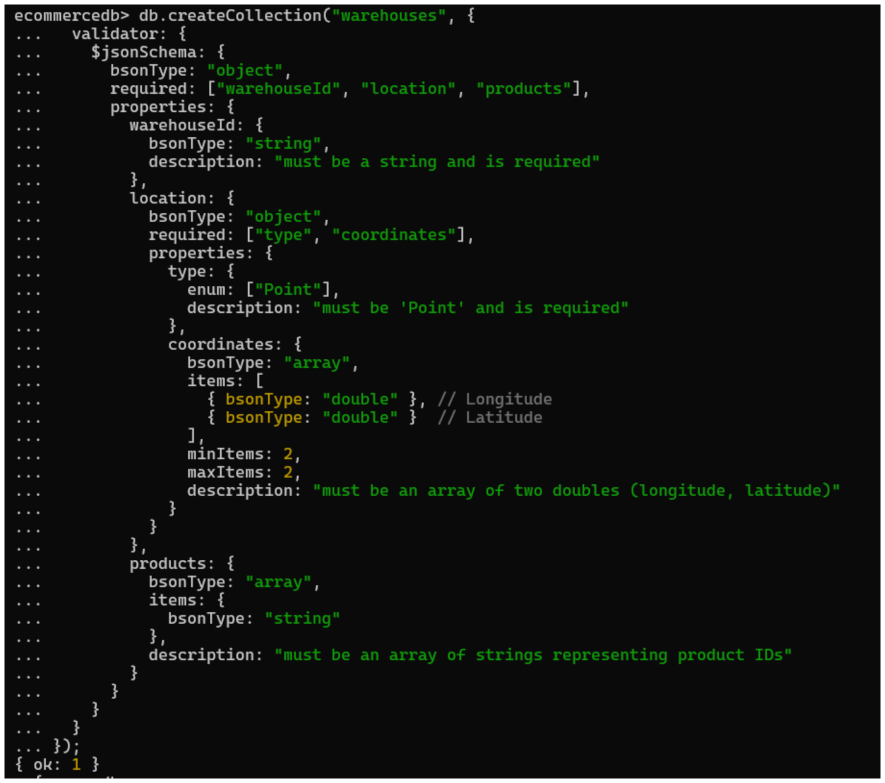
Assume there's a warehouses collection with geospatial data:

```
{
"warehouseId": "W001",
"location": { "type": "Point", "coordinates": [-74.006,
40.7128] },
"products": ["P001", "P002", "P003"]
}
```

Find the nearest warehouse within a 50-kilometer radius that stocks "P001".

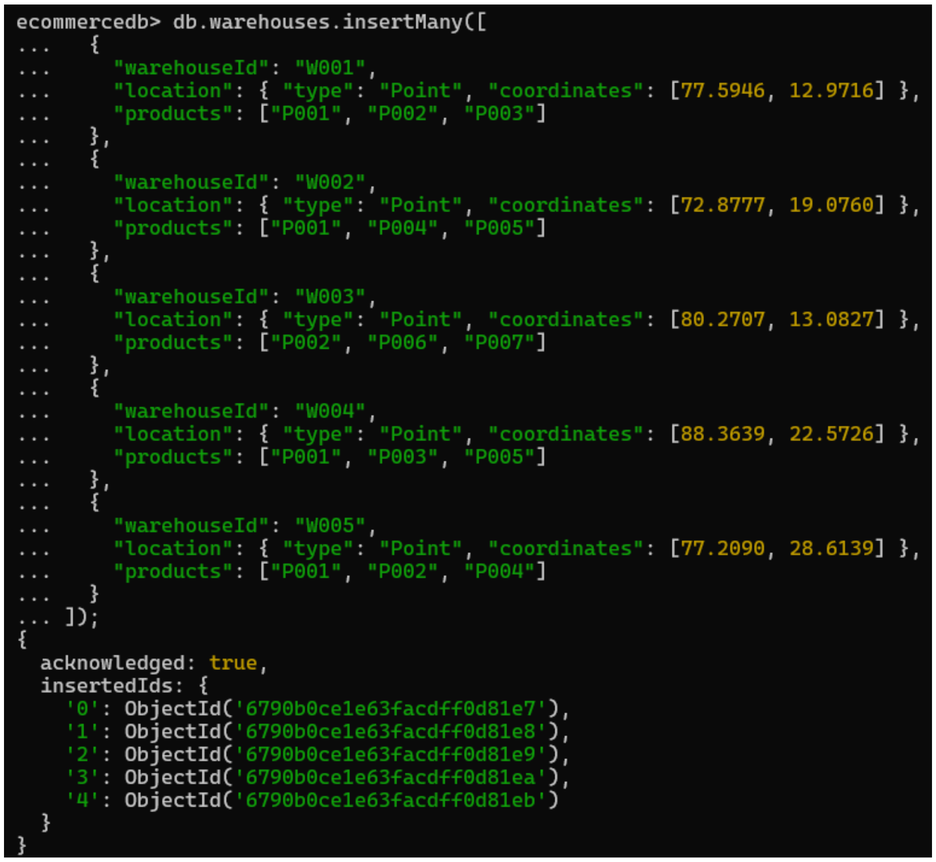
Hint: Use the $geoNear aggregation stage with a filter on the products array.

-> First we will create a Warehouse Collection:



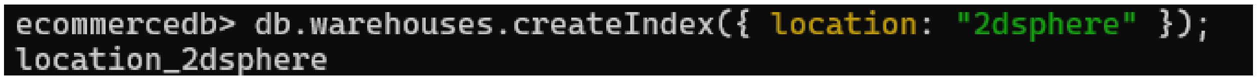
```
ecommercedb> db.createCollection("warehouses", {
...   validator: {
...     $jsonSchema: {
...       bsonType: "object",
...       required: ["warehouseId", "location", "products"],
...       properties: {
...         warehouseId: {
...           bsonType: "string",
...           description: "must be a string and is required"
...         },
...         location: {
...           bsonType: "object",
...           required: ["type", "coordinates"],
...           properties: {
...             type: {
...               enum: ["Point"],
...               description: "must be 'Point' and is required"
...             },
...             coordinates: {
...               bsonType: "array",
...               items: [
...                 { bsonType: "double" }, // Longitude
...                 { bsonType: "double" }  // Latitude
...               ],
...               minItems: 2,
...               maxItems: 2,
...               description: "must be an array of two doubles (longitude, latitude)"
...             }
...           }
...         },
...         products: {
...           bsonType: "array",
...           items: {
...             bsonType: "string"
...           },
...           description: "must be an array of strings representing product IDs"
...         }
...       }
...     }
...   }
... });
{ ok: 1 }
```

-> Now We will Populate The Warehouse Collection with Some Data:



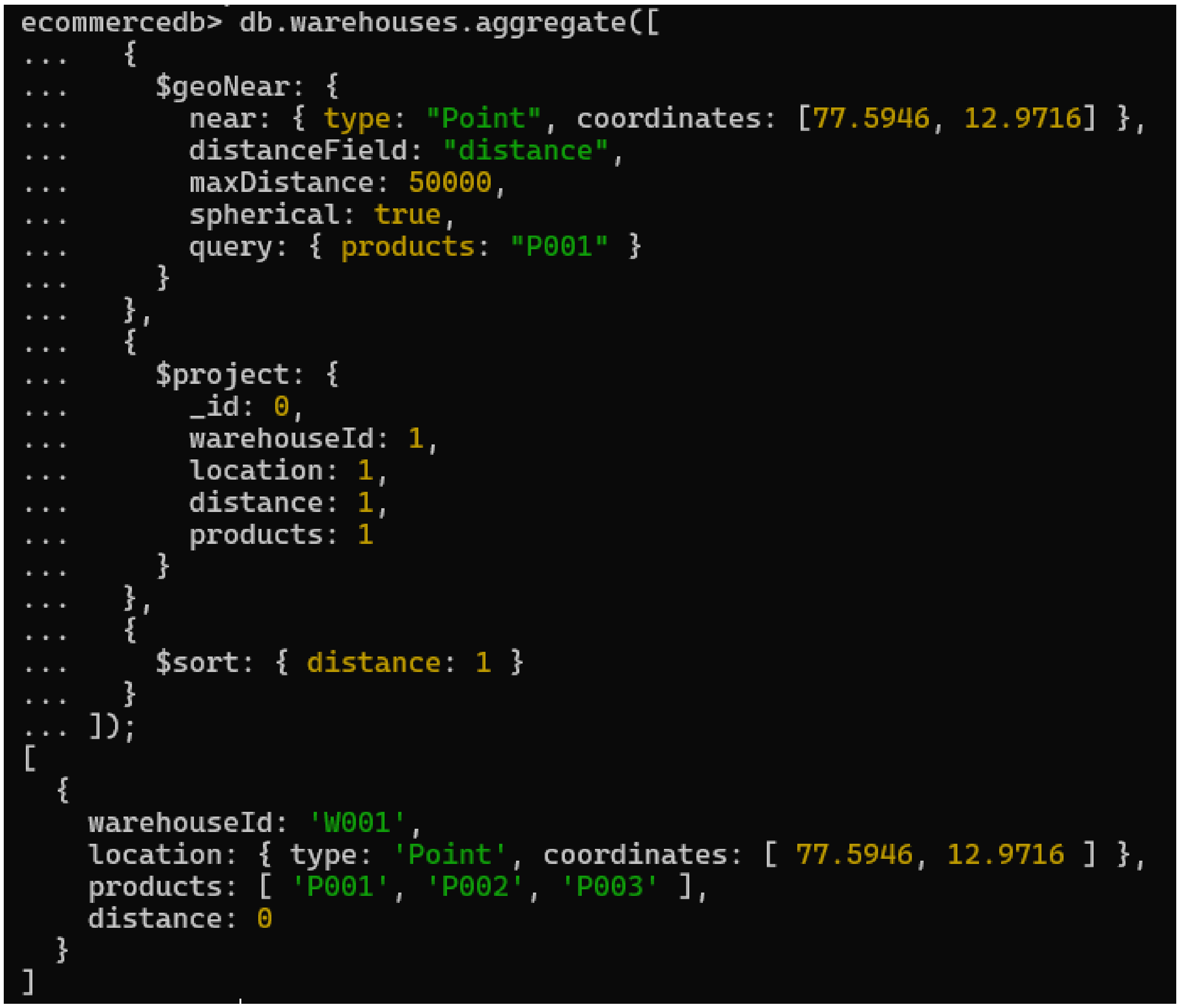
```
ecommercedb> db.warehouses.insertMany([
...   {
...     "warehouseId": "W001",
...     "location": { "type": "Point", "coordinates": [77.5946, 12.9716] },
...     "products": ["P001", "P002", "P003"]
...   },
...   {
...     "warehouseId": "W002",
...     "location": { "type": "Point", "coordinates": [72.8777, 19.0760] },
...     "products": ["P001", "P004", "P005"]
...   },
...   {
...     "warehouseId": "W003",
...     "location": { "type": "Point", "coordinates": [80.2707, 13.0827] },
...     "products": ["P002", "P006", "P007"]
...   },
...   {
...     "warehouseId": "W004",
...     "location": { "type": "Point", "coordinates": [88.3639, 22.5726] },
...     "products": ["P001", "P003", "P005"]
...   },
...   {
...     "warehouseId": "W005",
...     "location": { "type": "Point", "coordinates": [77.2090, 28.6139] },
...     "products": ["P001", "P002", "P004"]
...   }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6790b0ce1e63facdff0d81e7'),
    '1': ObjectId('6790b0ce1e63facdff0d81e8'),
    '2': ObjectId('6790b0ce1e63facdff0d81e9'),
    '3': ObjectId('6790b0ce1e63facdff0d81ea'),
    '4': ObjectId('6790b0ce1e63facdff0d81eb')
  }
}
```

-> Now, Create Index for Geospatial Queries:



```
ecommercedb> db.warehouses.createIndex({ location: "2dsphere" });
location_2dsphere
```

-> Query:



```
ecommercedb> db.warehouses.aggregate([
...   {
...     $geoNear: {
...       near: { type: "Point", coordinates: [77.5946, 12.9716] },
...       distanceField: "distance",
...       maxDistance: 50000,
...       spherical: true,
...       query: { products: "P001" }
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       warehouseId: 1,
...       location: 1,
...       distance: 1,
...       products: 1
...     }
...   },
...   {
...     $sort: { distance: 1 }
...   }
... ]);
[
  {
    warehouseId: 'W001',
    location: { type: 'Point', coordinates: [ 77.5946, 12.9716 ] },
    products: [ 'P001', 'P002', 'P003' ],
    distance: 0
  }
]
```