MongoDB Exercises

Scenario: Online Shopping Platform

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

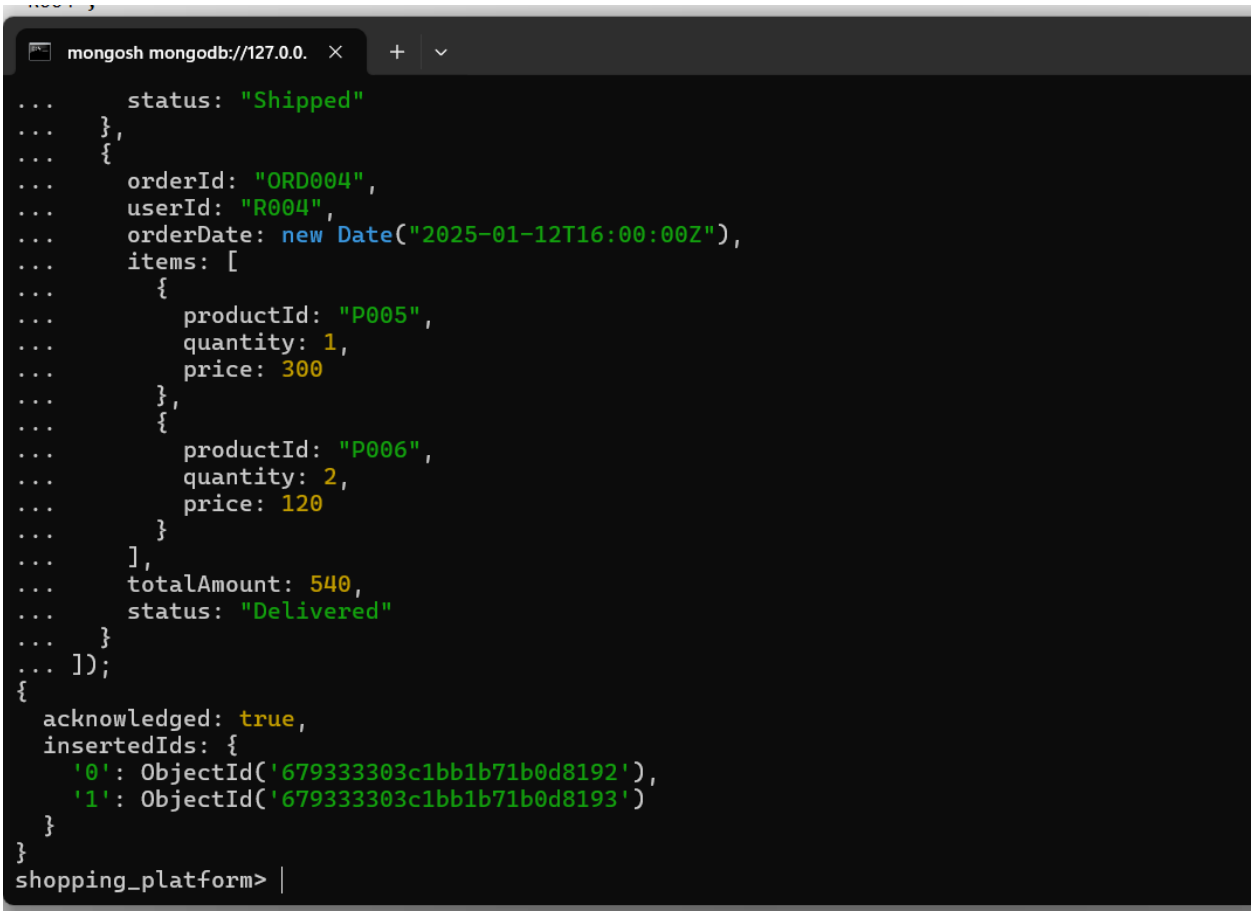
1. users: Stores user details.
2. orders: Stores order information.
3. products: Stores product information.

Creating Database:

```
// Create and use the shopping database

db.users.insertMany([

 {

  userId: "R003",

  name: "Rahul",

  email: "rahul@example.com",

  age: 28,

  address: {

   city: "Chicago",

   state: "IL",

   zip: "60601"

  },

  createdAt: new Date("2024-03-01T12:00:00Z")

 },

 {

  userId: "R004",

  name: "Emma",

  email: "emma@example.com",

  age: 29,

  address: {

   city: "San Francisco",
```

```
    state: "CA",
    zip: "94101"
   },
   createdAt: new Date("2024-03-02T14:00:00Z")
  }
]);

// 2. Create orders collection and insert sample data


db.orders.insertMany([
  {
   orderId: "ORD003",
   userId: "R003",
   orderDate: new Date("2025-01-10T10:15:00Z"),
   items: [
    {
     productId: "P003",
     quantity: 3,
     price: 200
    },
    {
     productId: "P004",
     quantity: 2,
     price: 150
    }
   ],
   totalAmount: 850,
   status: "Shipped"
```

```
  },
  {
    orderId: "ORD004",
    userId: "R004",
    orderDate: new Date("2025-01-12T16:00:00Z"),
    items: [
      {
        productId: "P005",
        quantity: 1,
        price: 300
      },
      {
        productId: "P006",
        quantity: 2,
        price: 120
      }
    ],
    totalAmount: 540,
    status: "Delivered"
  }
]);
```



// 3. Create products collection and insert sample data

```
db.products.insertMany([
 {
  productId: "P007",
  name: "Gaming Headset",
  category: "Electronics",
  price: 250,
  stock: 100,
  ratings: [
   {
    userId: "R003",
```

```
      rating: 4.8
    },
    {
      userId: "R004",
      rating: 4.2
    }
  ]
 },
 {
  productId: "P008",
  name: "Smartwatch",
  category: "Wearables",
  price: 300,
  stock: 50,
  ratings: [
   {
    userId: "R004",
    rating: 4.5
   }
  ]
 }
]);
```

// 4. Create warehouses collection with geospatial index

db.warehouses.createIndex({ location: "2dsphere" });

db.warehouses.insertMany([

{

```
  warehouseId: "W003",
  location: {
    type: "Point",
    coordinates: [41.8781, -87.6298]
  },
  products: ["P007", "P008", "P009"]
}
]);
```

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"
 },
 {
  $group: {
   _id: {
    productId: "$productId",
    name: "$name",
    category: "$category",
    price: "$price",
    stock: "$stock"

   },
   averageRating: { $avg: "$ratings.rating" }
  }
 },
 {
  $match: {
   averageRating: { $gte: 4 }
  }
 },
 {
```

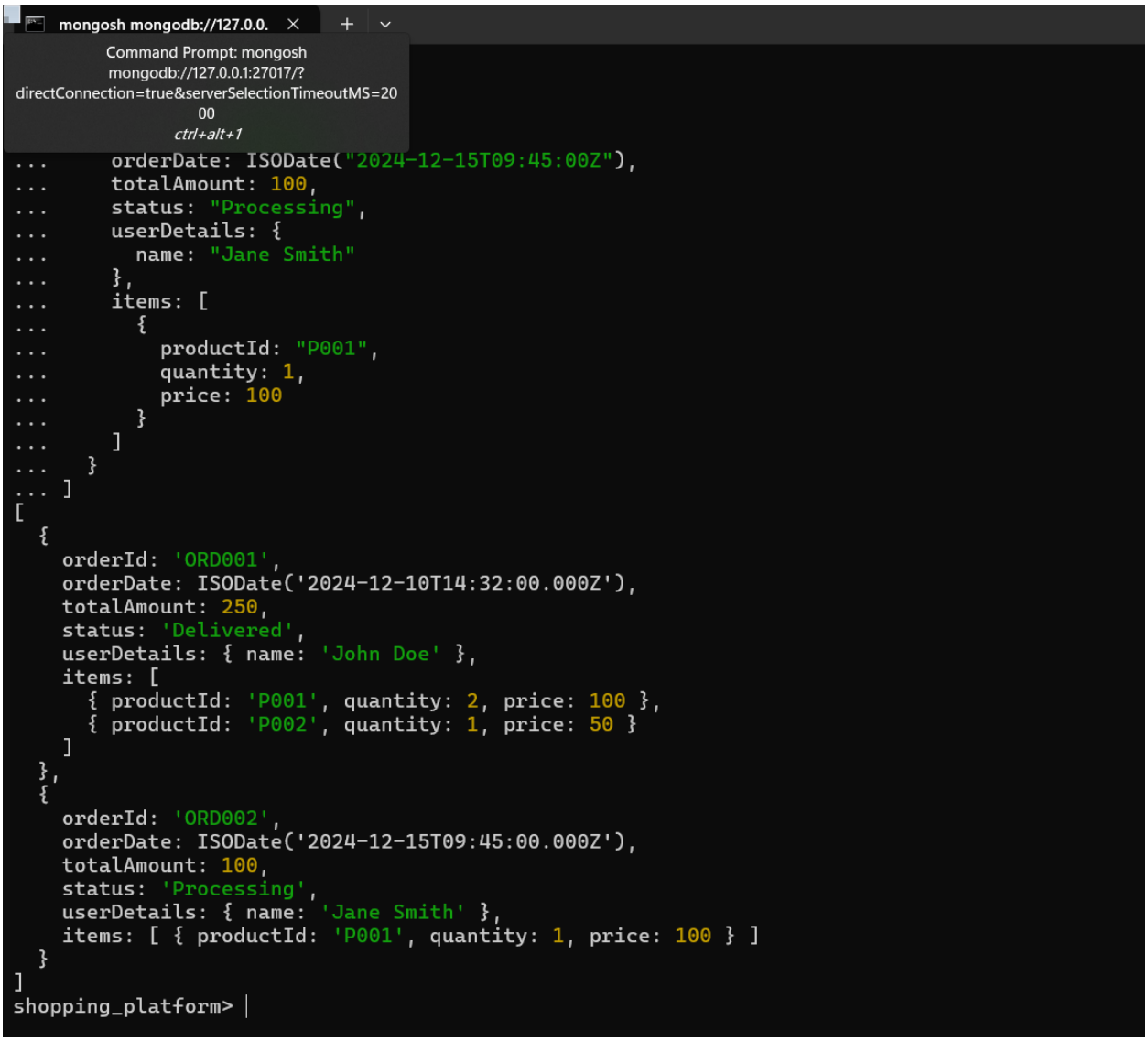
```
  $project: {
    _id: 0,

    productId: "$_id.productId",
    name: "$_id.name",
    category: "$_id.category",
    price: "$_id.price",
    stock: "$_id.stock",
    averageRating: 1
  }
 }
]);
```

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.

[

```
{
  orderId: "ORD001",
  orderDate: ISODate("2024-12-10T14:32:00Z"),
  totalAmount: 250,
  status: "Delivered",
  userDetails: {
    name: "John Doe"
  },
  items: [
    {
      productId: "P001",
      quantity: 2,
      price: 100
    },
    {
      productId: "P002",
      quantity: 1,
      price: 50
    }
  ]
},
{
  orderId: "ORD002",
  orderDate: ISODate("2024-12-15T09:45:00Z"),
  totalAmount: 100,
  status: "Processing",
```

```
  userDetails: {
    name: "Jane Smith"
  },
  items: [
    {
      productId: "P001",
      quantity: 1,
      price: 100
    }
  ]
 }
]
```



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.

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

5. Find Nearest Warehouse

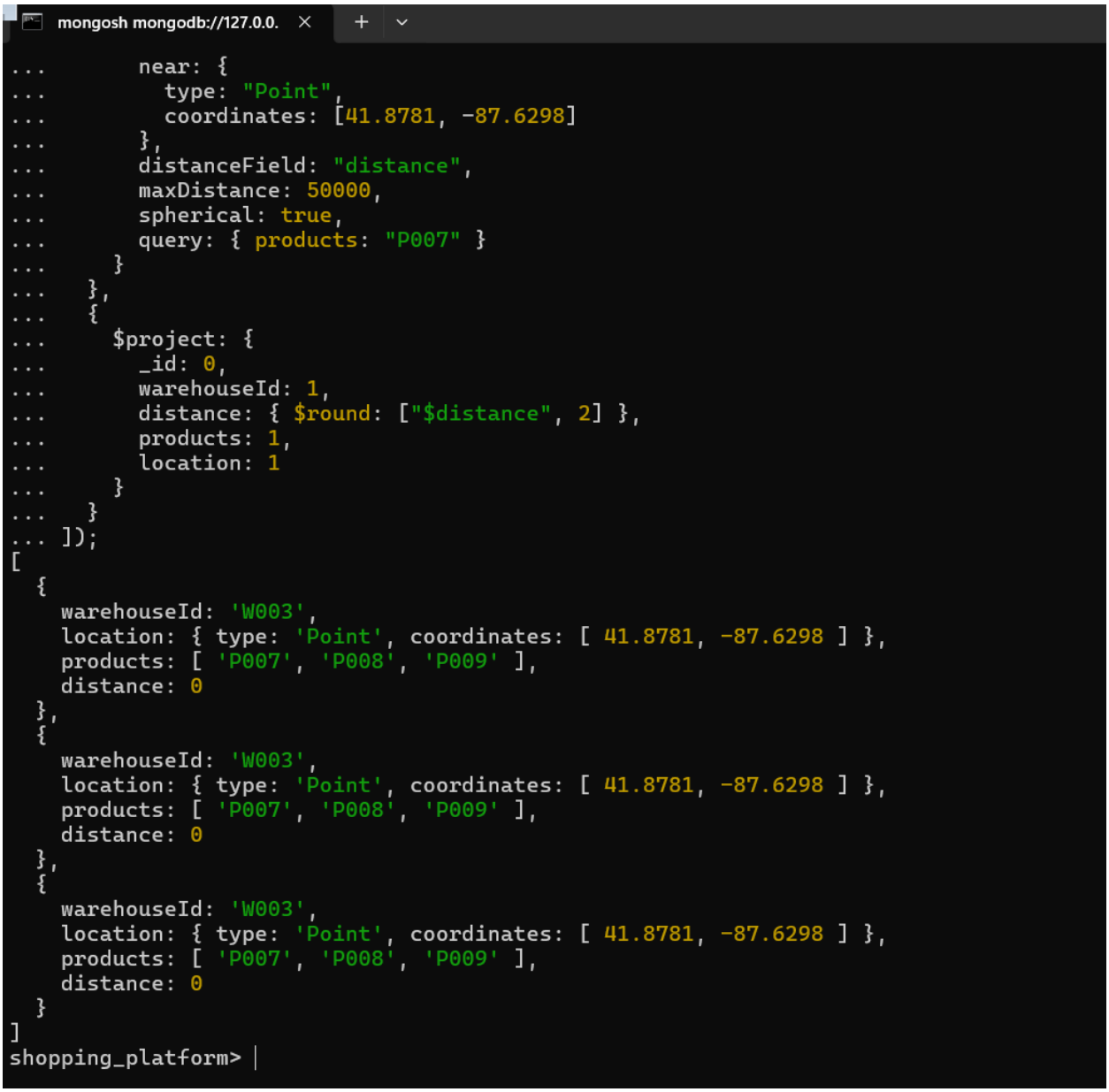
Assume there's a warehouses collection with geospatial data:
{ "warehouseId": "W001",
"location": { "type": "Point", "coordinates": [-74.006,
40.7128] },
"products": ["P001", "P002", "P003"] }
Find the nearest warehouse within a 50-kilometer radius that stocks "P001".
Hint: Use the $geoNear aggregation stage with a filter on the products array.

db.warehouses.createIndex({ location: "2dsphere" });

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

```
db.warehouses.aggregate([
 {
  $geoNear: {
   near: {
    type: "Point",
    coordinates: [41.8781, -87.6298]
   },
   distanceField: "distance",
   maxDistance: 50000,
   spherical: true,
   query: { products: "P007" }
  }
 },
 {
  $project: {
```

```
    _id: 0,
    warehouseId: 1,
    distance: { $round: ["$distance", 2] },
    products: 1,
    location: 1
   }
  }
]);
```


mongosh mongodb://127.0.0.
... near: {
... type: "Point",
... coordinates: [41.8781, -87.6298]
... },
... distanceField: "distance",
... maxDistance: 50000,
... spherical: true,
... query: { products: "P007" }
... }
... },
... {
... $project: {
... _id: 0,
... warehouseId: 1,
... distance: { $round: ["$distance", 2] },
... products: 1,
... location: 1
... }
... }
...]);
[
{
warehouseId: 'W003',
location: { type: 'Point', coordinates: [41.8781, -87.6298] },
products: ['P007', 'P008', 'P009'],
distance: 0
},
{
warehouseId: 'W003',
location: { type: 'Point', coordinates: [41.8781, -87.6298] },
products: ['P007', 'P008', 'P009'],
distance: 0
},
{
warehouseId: 'W003',
location: { type: 'Point', coordinates: [41.8781, -87.6298] },
products: ['P007', 'P008', 'P009'],
distance: 0
}
]
shopping_platform>