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.

Below is the structure of each collection:

Collection: users

```
{
  "userId": "U001",
  "name": "John Doe",
  "email": "john.doe@example.com",
  "age": 28,
  "address": {
    "city": "New York",
    "state": "NY",
    "zip": "10001"
  },
  "createdAt": "2024-01-01T10:00:00Z"
}
```

Collection: orders

```
{
  "orderId": "ORD001",
  "userId": "U001",
  "orderDate": "2024-12-10T14:32:00Z",
  "items": [
```

```
    { "productId": "P001", "quantity": 2, "price": 100 },
    { "productId": "P002", "quantity": 1, "price": 50 }
  ],
  "totalAmount": 250,
  "status": "Delivered"
}
```

Collection: products

```
{
  "productId": "P001",
  "name": "Wireless Mouse",
  "category": "Electronics",
  "price": 50,
  "stock": 200,
  "ratings": [
    { "userId": "U002", "rating": 4.5 },
    { "userId": "U003", "rating": 3.0 }
  ]
}
```

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.

---

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.

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.

---

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.

---

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.