**Managing an Online Store Database PART 1**

You are hired as a junior database manager for a small online store called “**SmartMart**”. Your job is to create and manage the store’s database to track products, customers, and orders. Your manager wants you to insert realistic data and ensure it is easy to query and maintain.

**A. Set up the Database**

* Create a database called SmartMart
* Create three collections:
  1. Products - for all items sold
  2. Customers - for registered users
  3. Orders - for purchase transactions

**B. Insert Data**

* Products: Add at least 5 items with fields like:
  + Product name
  + Category (e.g., laptops, accessories)
  + Price
  + Stock quantity
  + Tags or features (array field, e.g., “wireless”, “gaming”)
* Customers: Add at least 5 customers with:
  + Name
  + Email
  + Address
  + Membership level or loyalty points (optional)
* Orders: Add at least 5 orders linking a customer to products:
  + Include multiple products in some orders
  + Add quantity purchased
  + Status (e.g., pending, shipped, delivered)
  + Optional: discounts or special notes

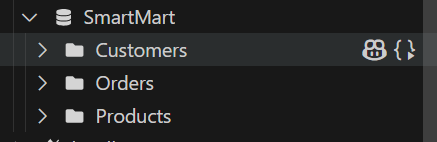
**C. Add Business Scenarios**

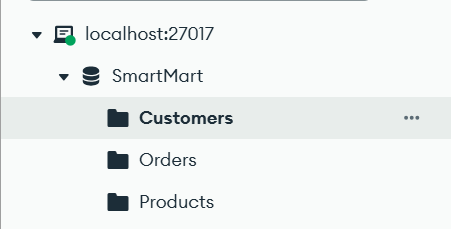
* Students should simulate real business situations while populating the database:

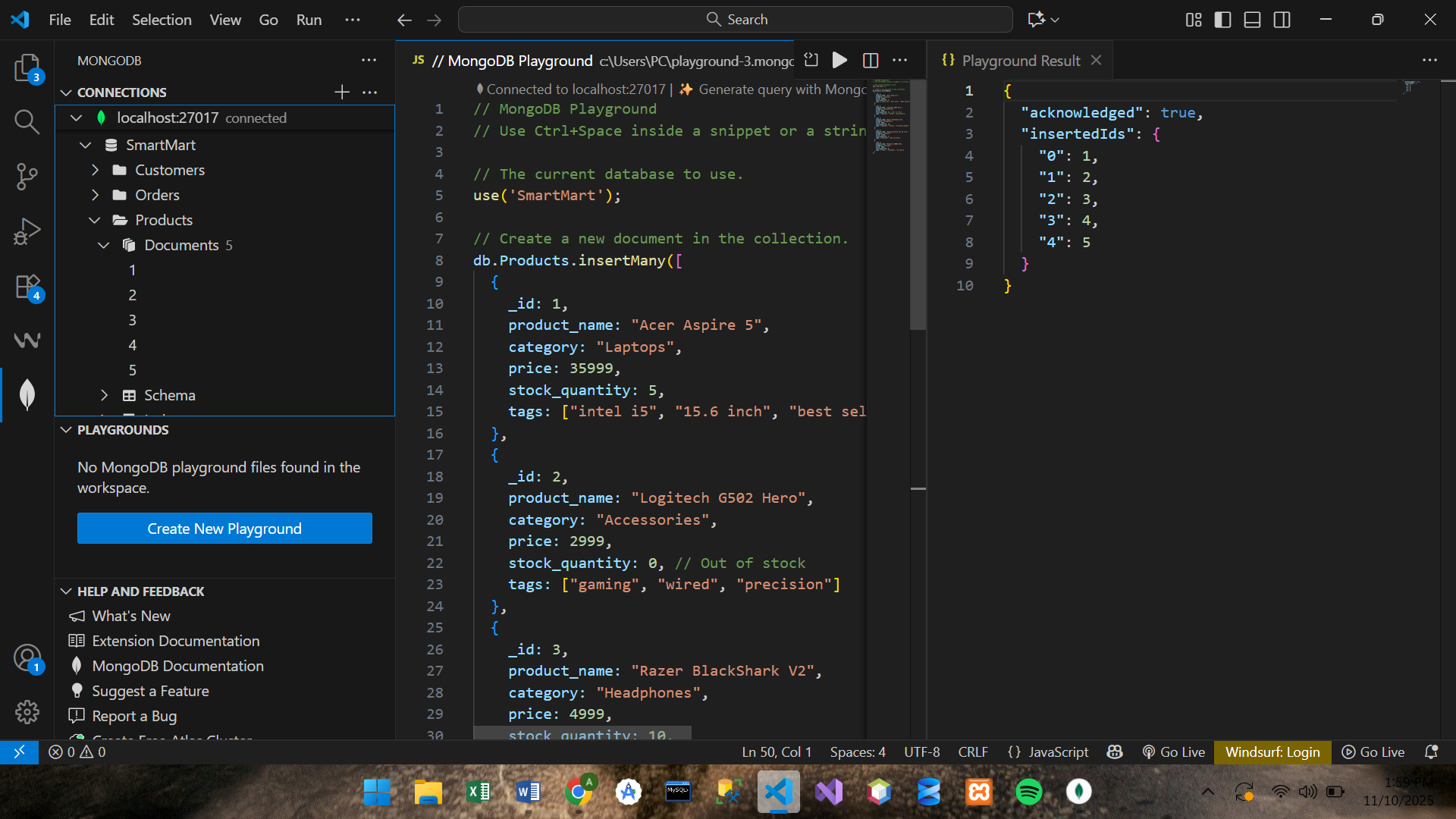
1. A customer buys multiple products in one order.
2. A product runs out of stock; mark it accordingly.
3. Some customers have loyalty points or membership levels.
4. Some products are on sale or have special tags (like “new arrival” or “best seller”).
5. One order is returned or canceled; reflect it in the order status.

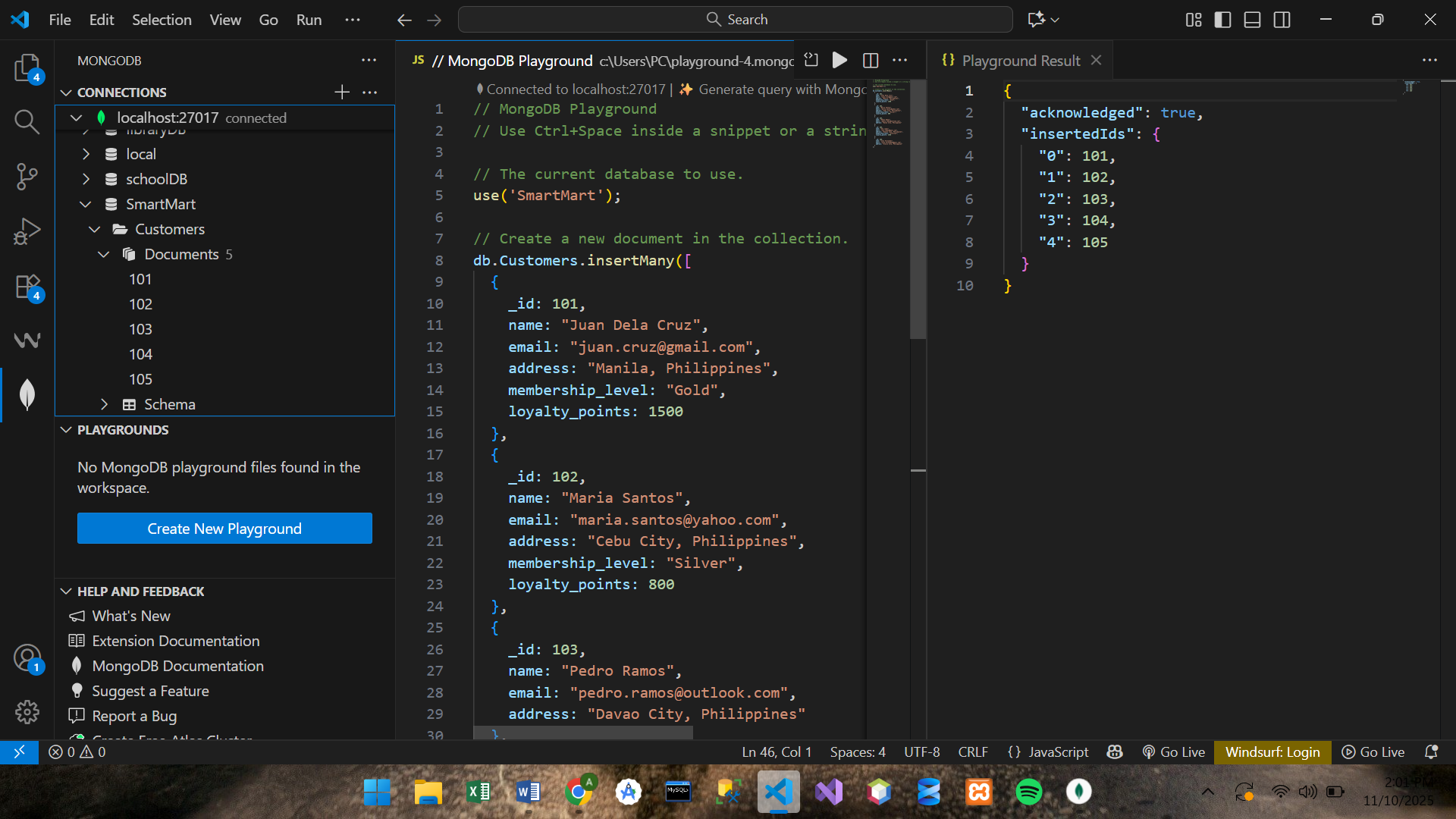
* Take screenshots from VS Code, Compass, or mongosh showing sample documents in each collection.
* Include a short paragraph or bullet points explaining how each scenario is reflected in the database. Example:
  + “Order #102 shows a customer purchasing multiple products in one order.”
  + “Product ‘Gaming Headset’ has stock 0, indicating it’s out of stock.”
  + “Customer Maria Santos has 200 loyalty points and is a Gold member.”
  + “Laptop has tags ‘gaming’ and ‘best seller’.”
  + “Order #105 has status ‘returned’.”

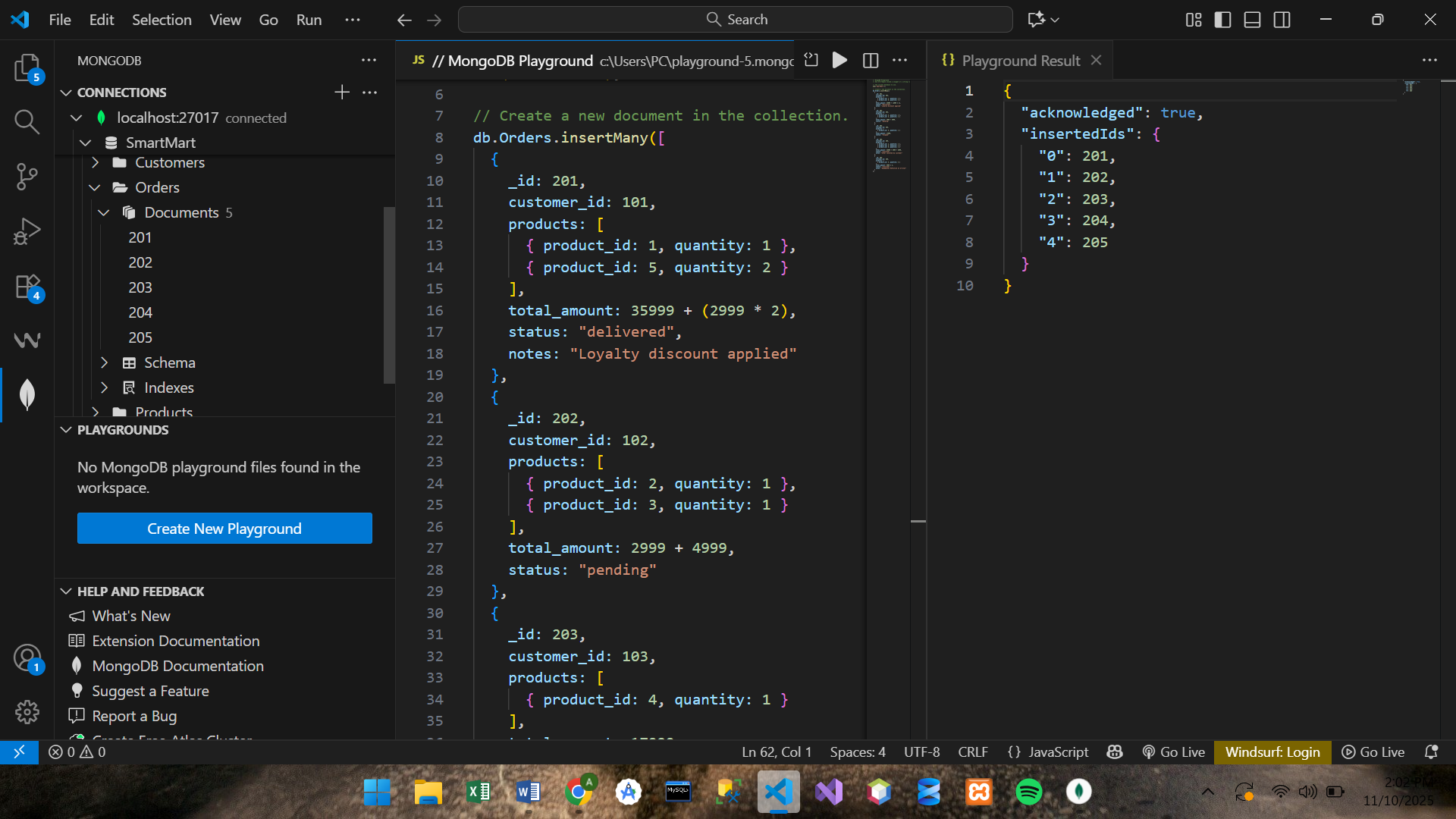
**SCREENSHOTS**

* + 1. 

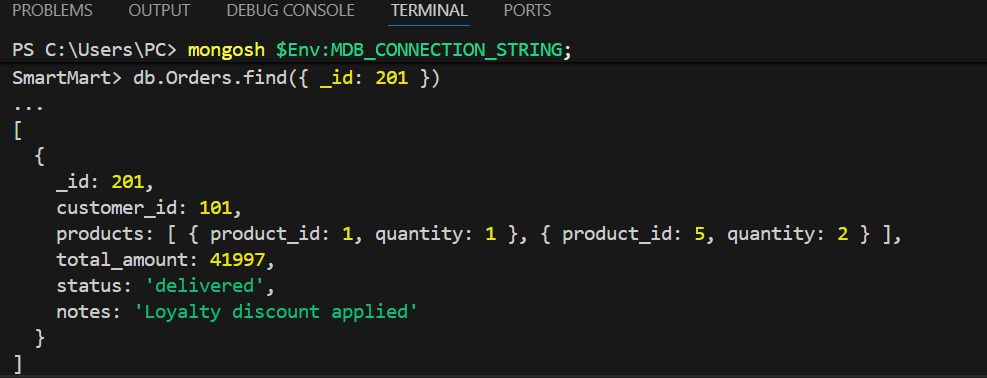


* + 1. 

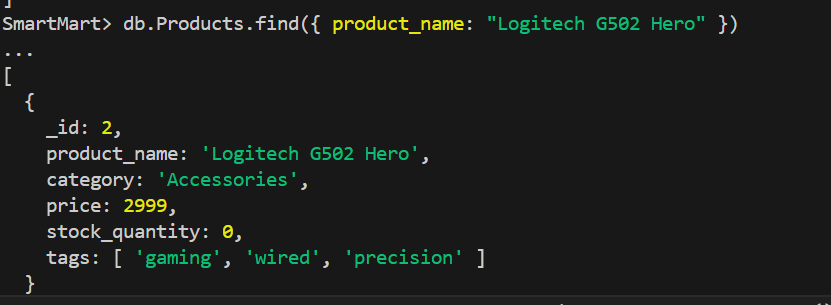




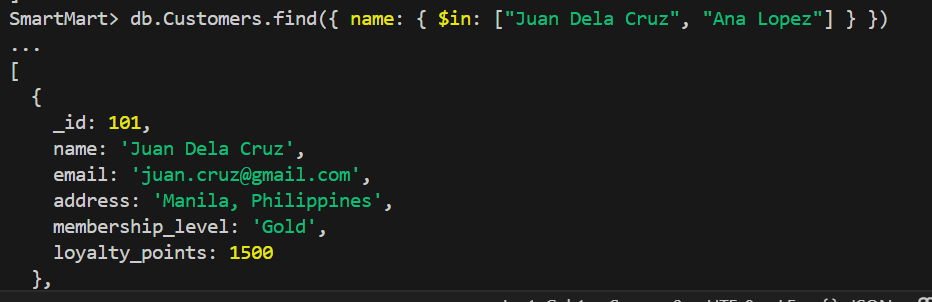
* + 1. Scenario 1: Order #201 shows that customer Juan Dela Cruz purchased multiple products in one order — an Acer Aspire 5 laptop and two Kingston SSD drives. This demonstrates the system’s ability to handle orders with more than one product.



Scenario 2: The product ‘Logitech G502 Hero’ has a stock quantity of 0, indicating it is currently out of stock. This reflects real-world inventory tracking.

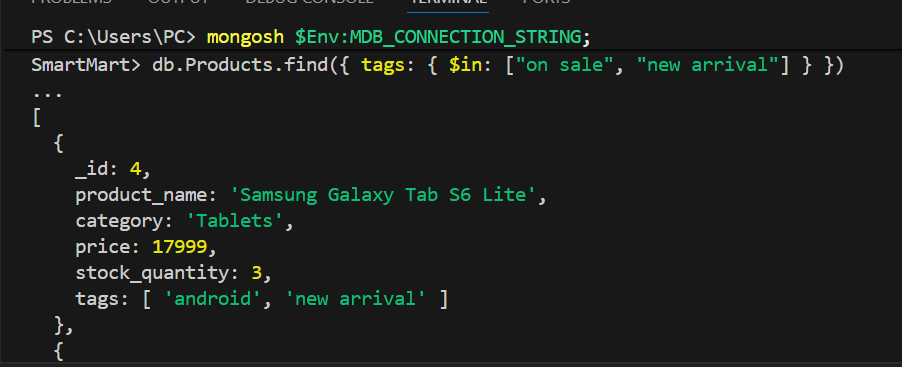


Scenario 3: Customers such as Juan Dela Cruz and Ana Lopez have loyalty points and membership levels (Gold and Platinum). This information can be used for discounts or exclusive rewards.



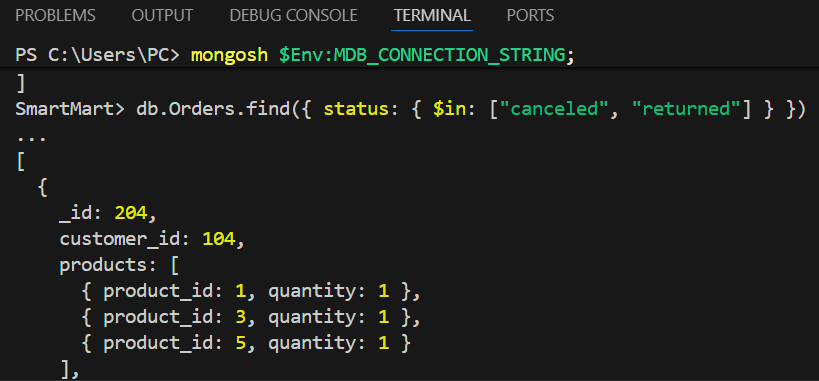


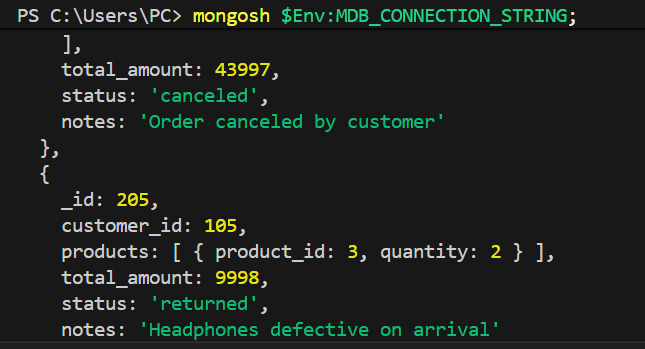
Scenario 4: The Kingston SSD includes the tag ‘on sale’ and the Samsung Galaxy Tab S6 Lite is tagged as a ‘new arrival’. Tags allow the store to highlight special products or promotions.





Scenario 5: Order #204 has the status ‘canceled’, and Order #205 has the status ‘returned’. This demonstrates how the database records different outcomes for customer transactions.





**Querying Documents in the SmartMart Database PART 2**

Your manager wants answers to everyday business questions, like which products are low in stock, which customers have made purchases, and which orders are pending. You need to query the database to retrieve this information.

**A. Prepare**

* Make sure your SmartMart has the three collections populated: products, customers, orders.

**B. Retrieving Documents by applying filters and conditions**

1. Products
   1. Find all products in a specific category (e.g., laptops or accessories).
   2. Find products with low stock (e.g., less than 5 items).
   3. Find products with a specific tag (e.g., “sale” or “best seller”).
2. Customers
   1. Find customers in a specific city.
   2. Find customers with loyalty points above a certain threshold.
3. Orders
   1. Find orders with status “pending” or “shipped”.
   2. Find all orders made by a specific customer.
   3. Find orders containing a specific product.

**C. Using Operators**

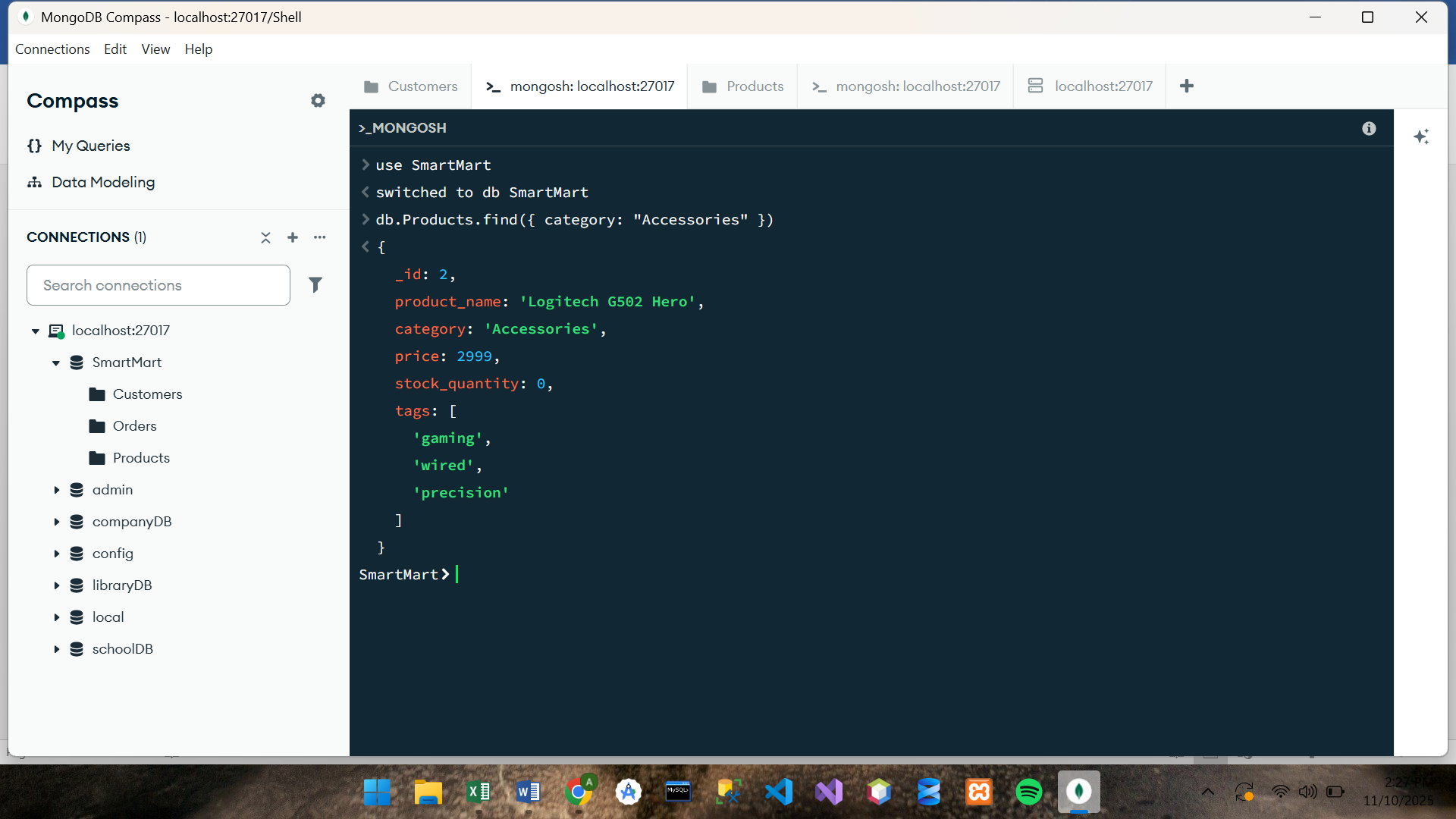
1. Finding products that are low in stock and on sale.
2. Finding customers from a certain city or with a premium membership.
3. Finding orders that are either pending or canceled.

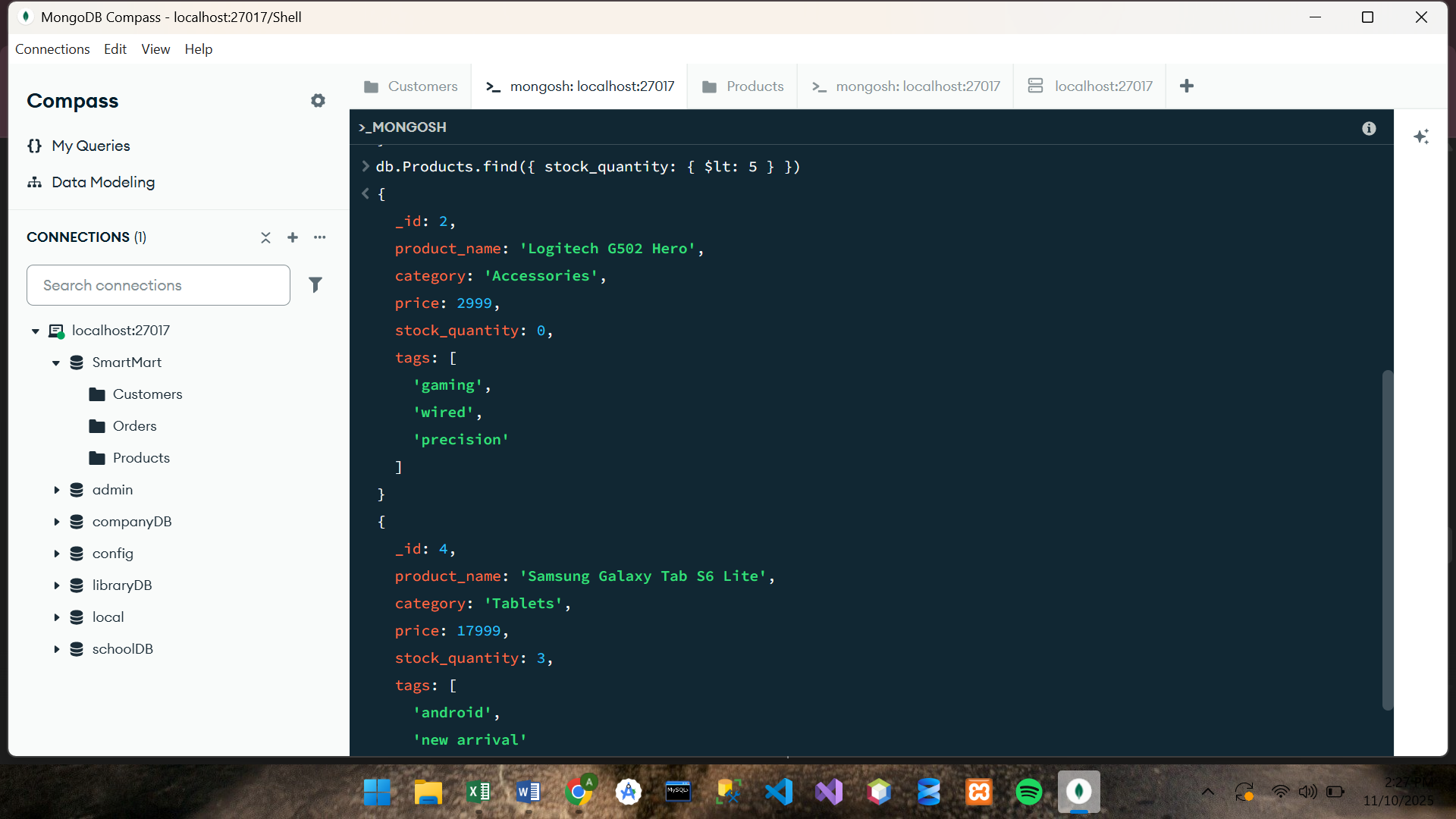
**For each query or requirement, take a clear screenshot that shows the relevant data in Mongosh, Compass, or VS Code. Label each screenshot clearly.**

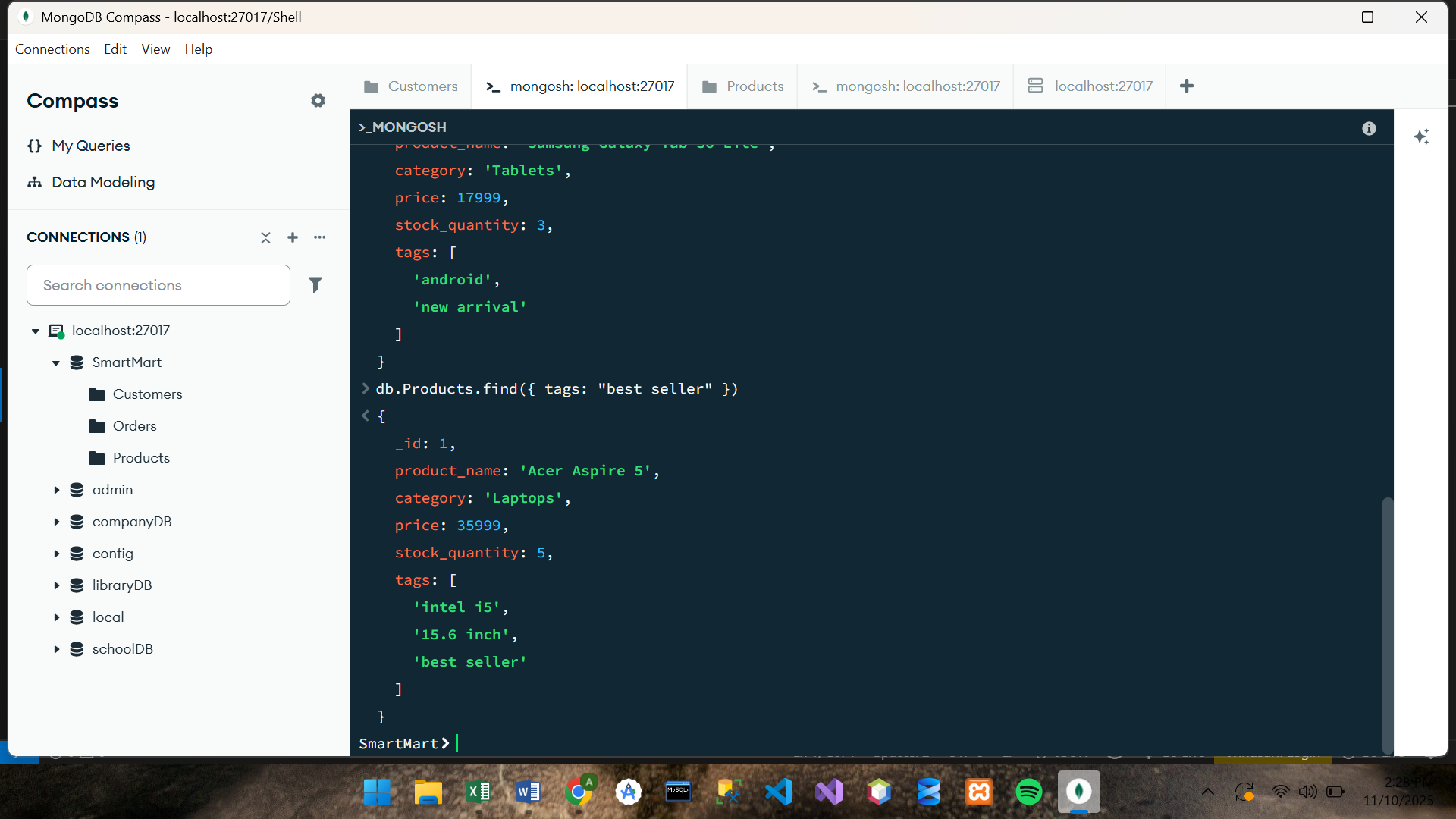
**SCREENSHOTS**

**B. Retrieving Documents by applying filters and conditions**

1. Products

****

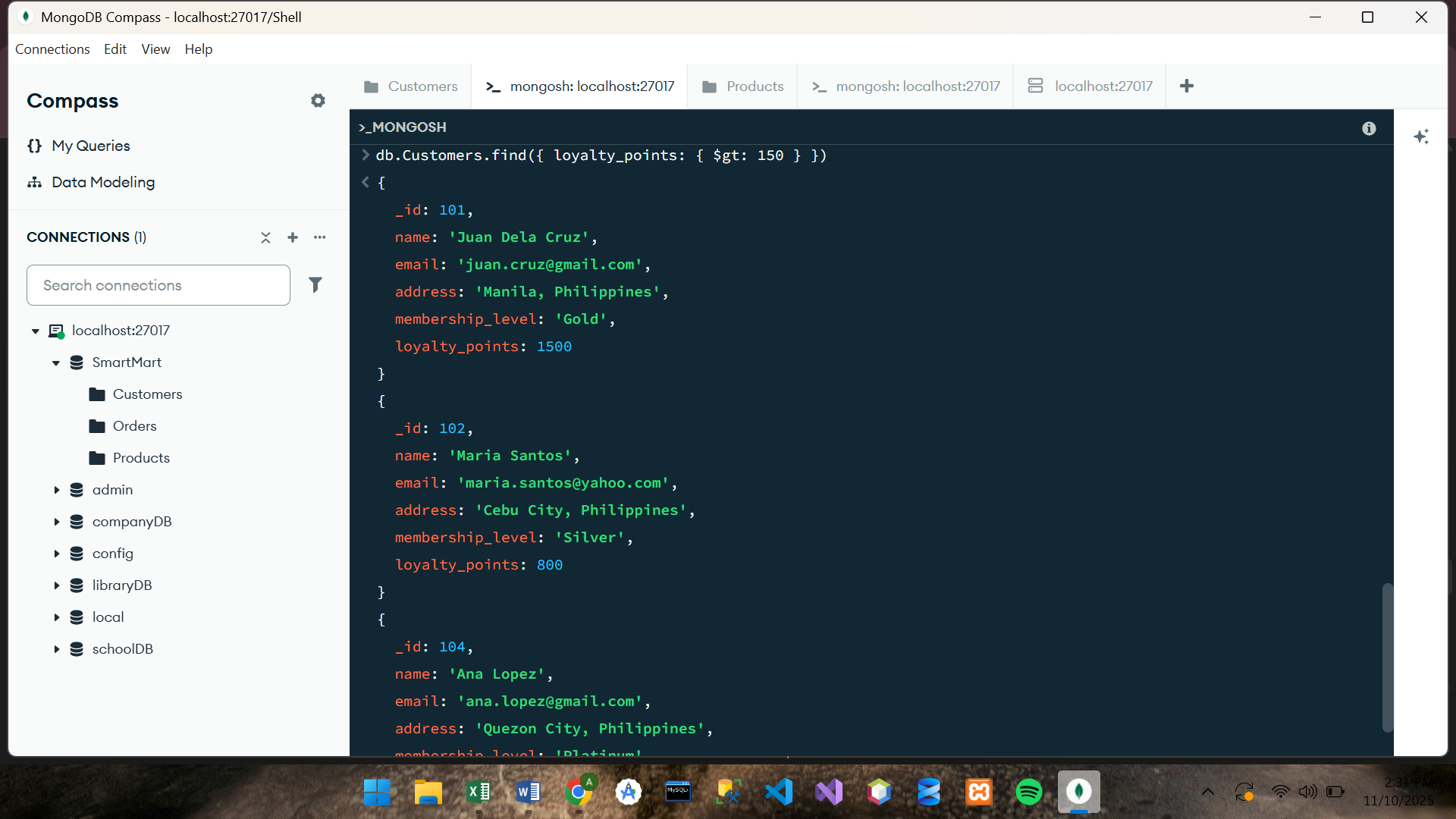
****

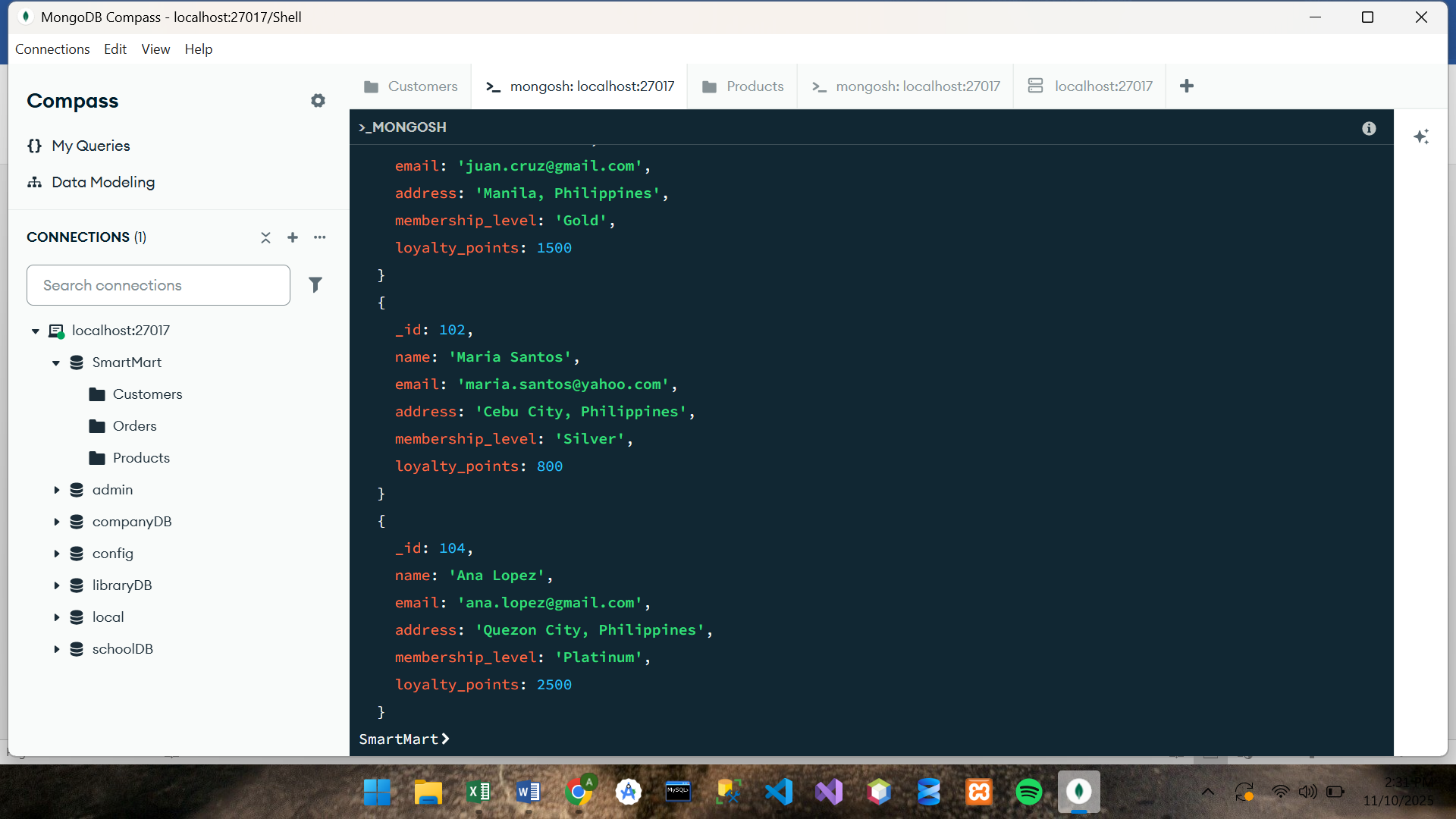
****

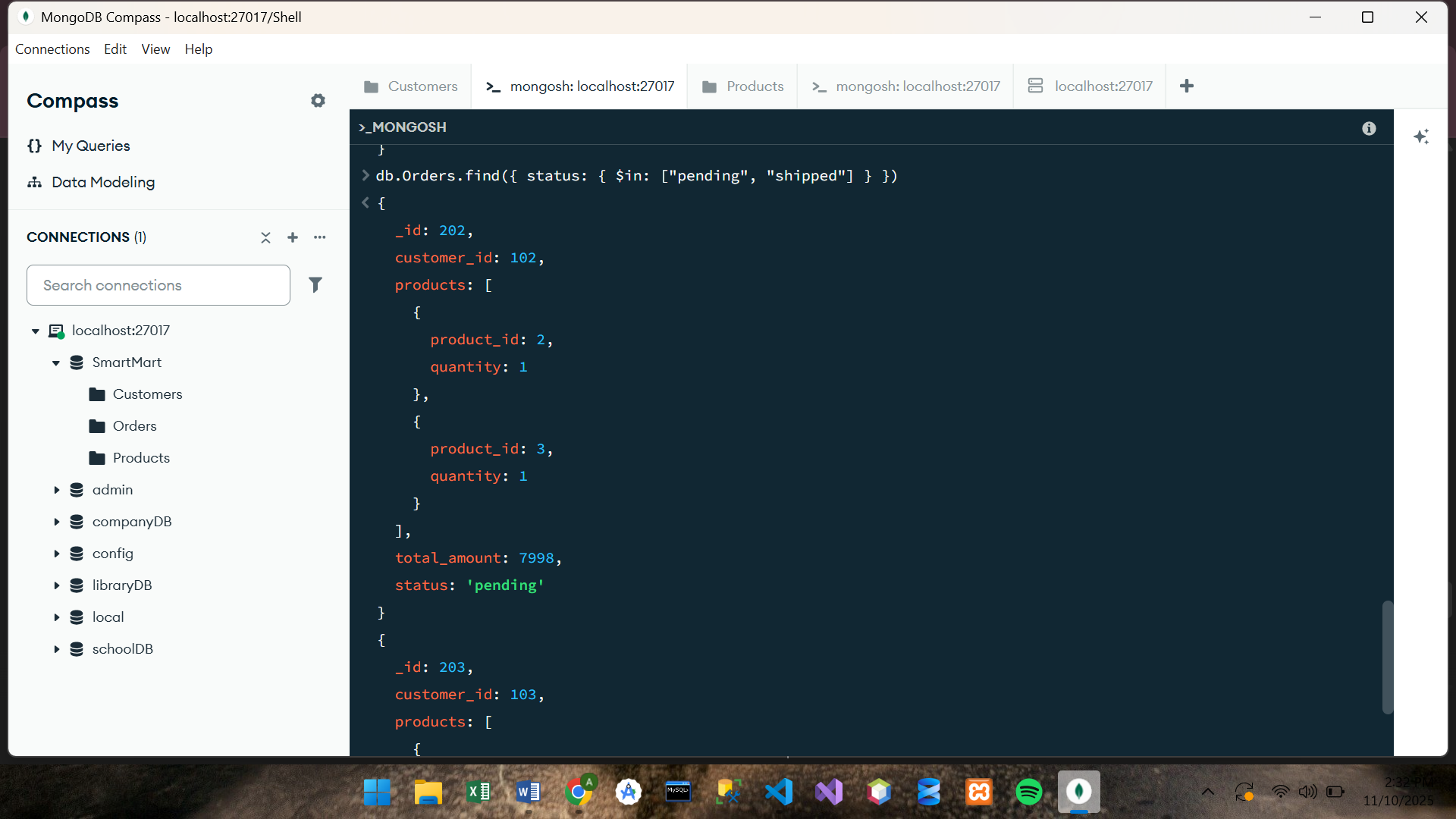
1. Customers

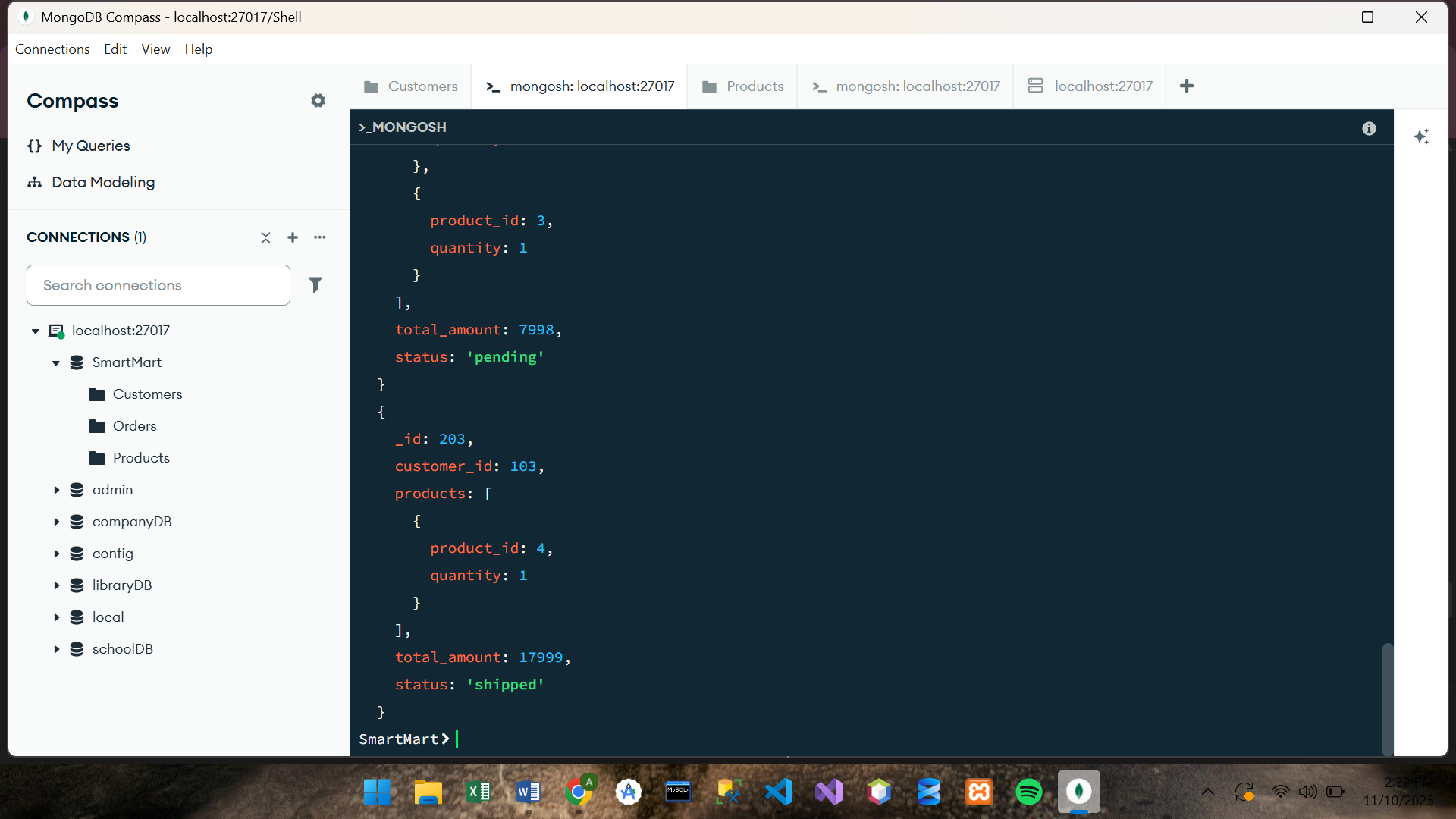


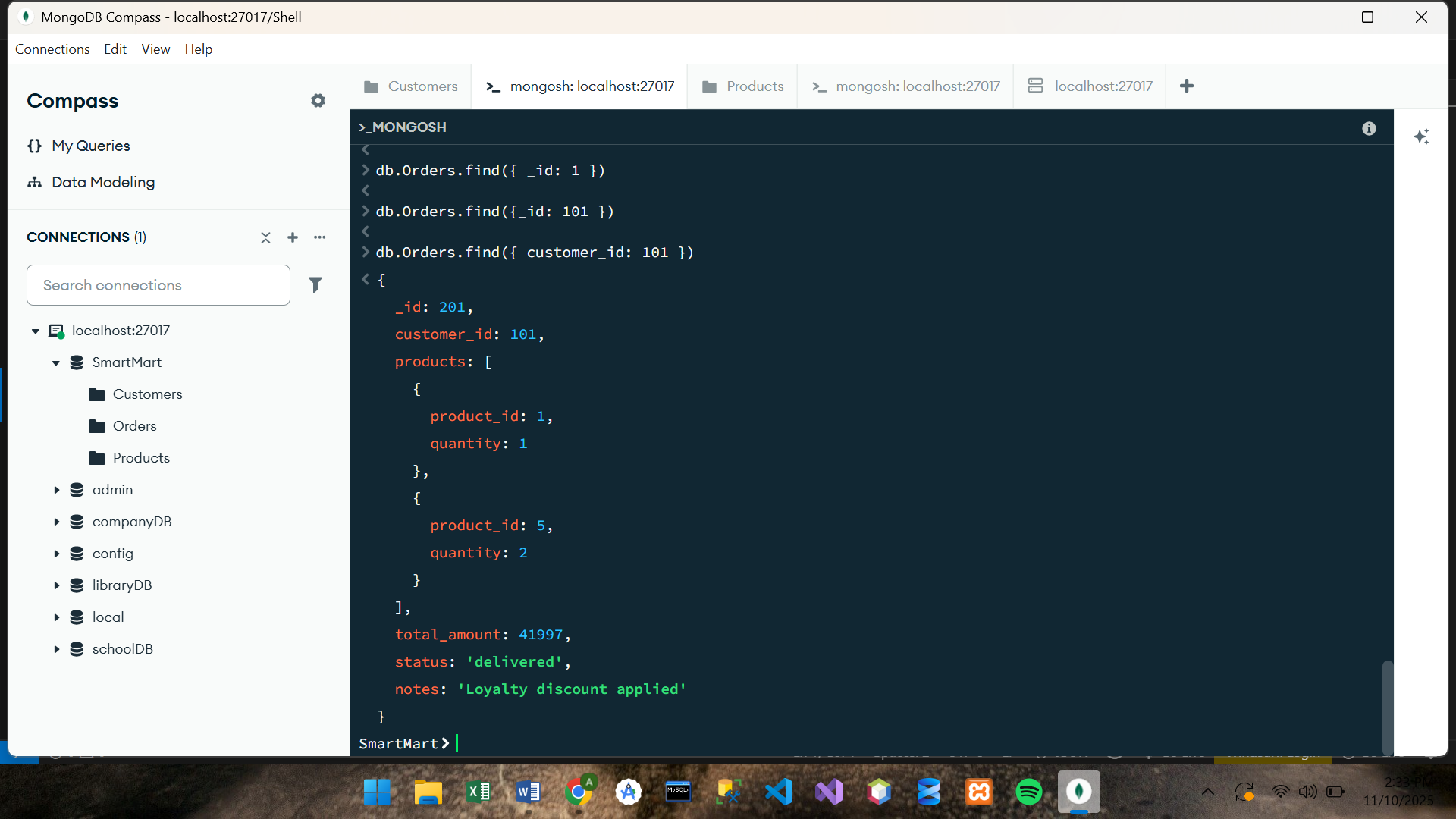
1. Orders

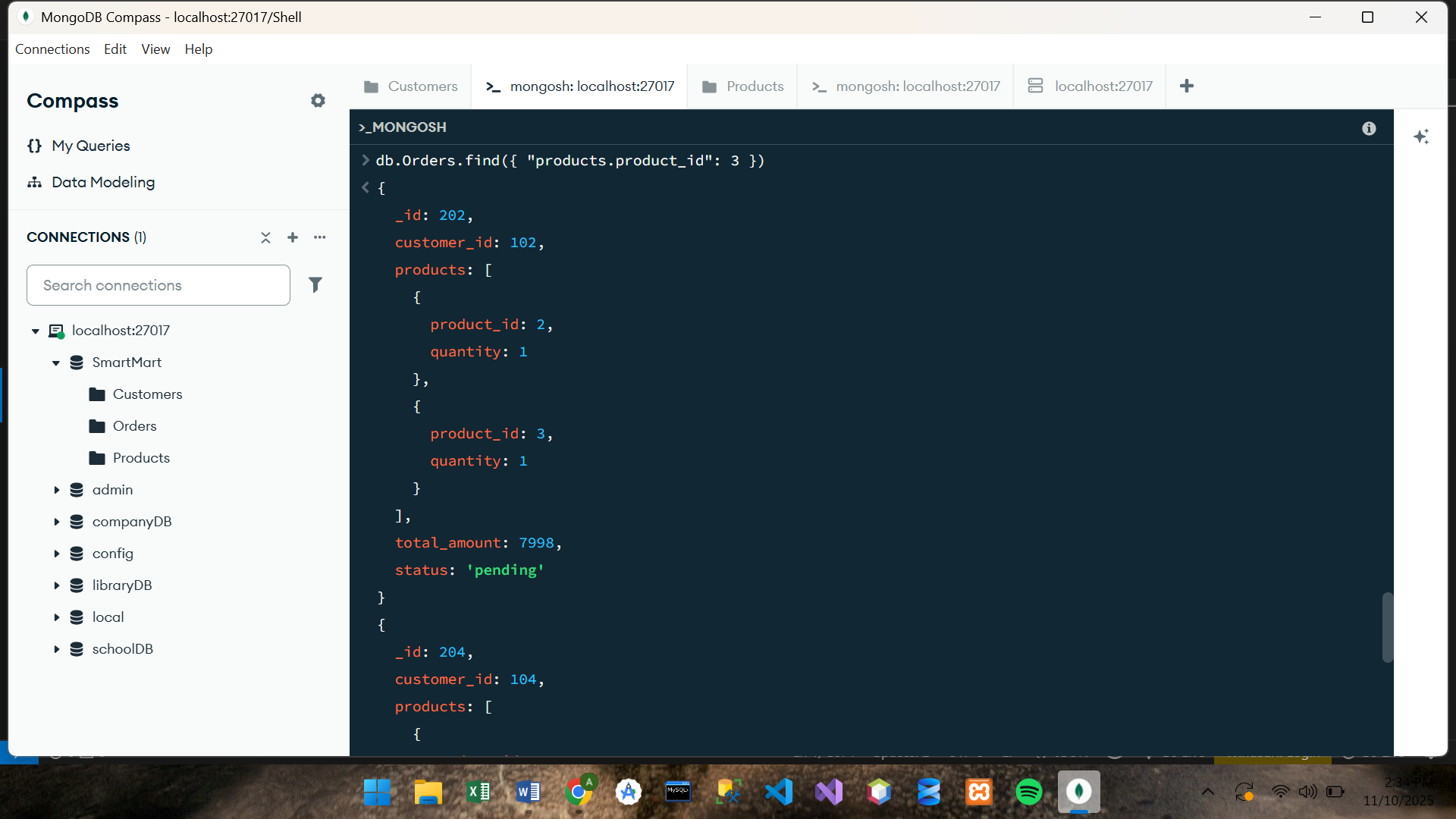
****

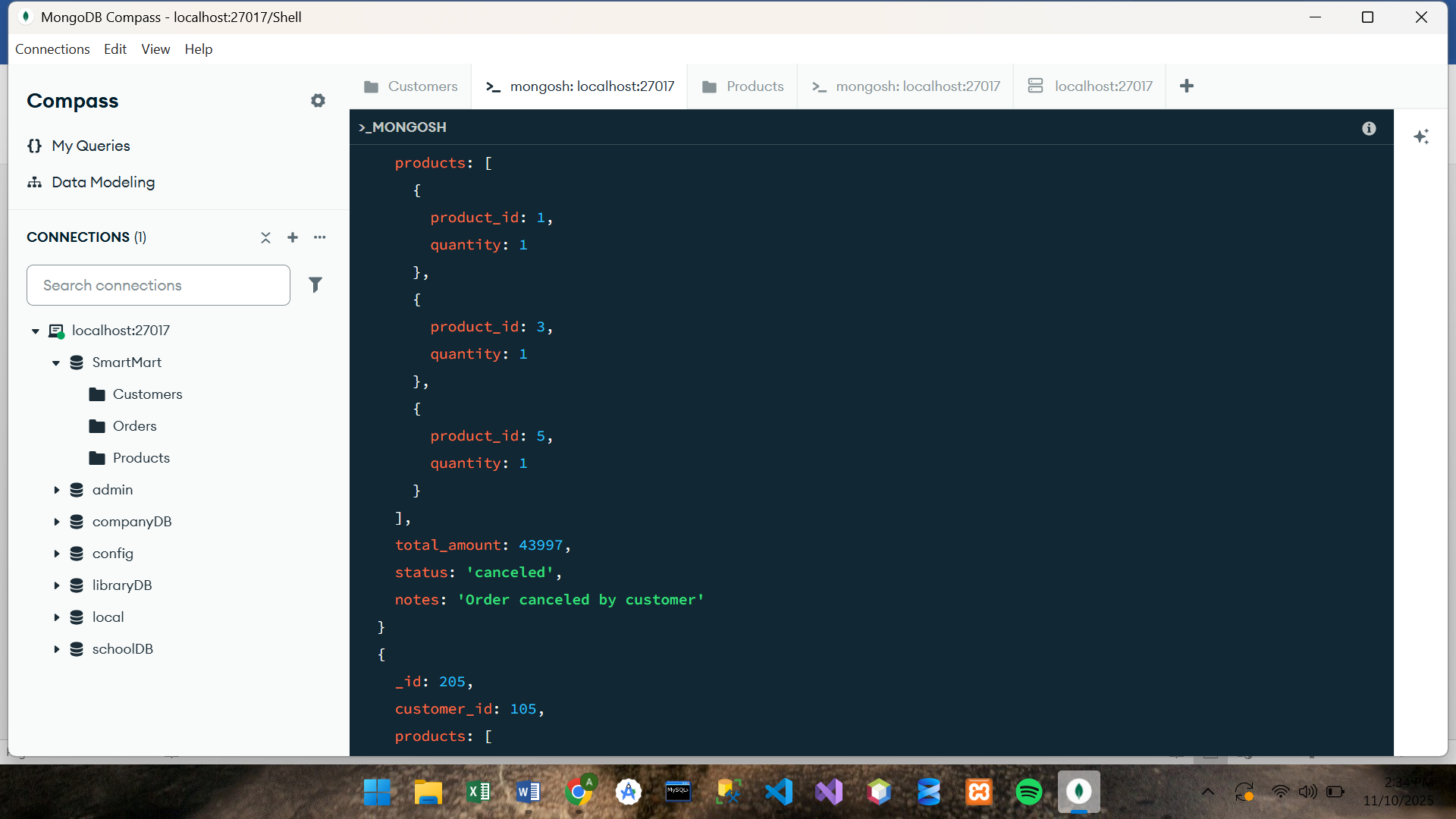
****

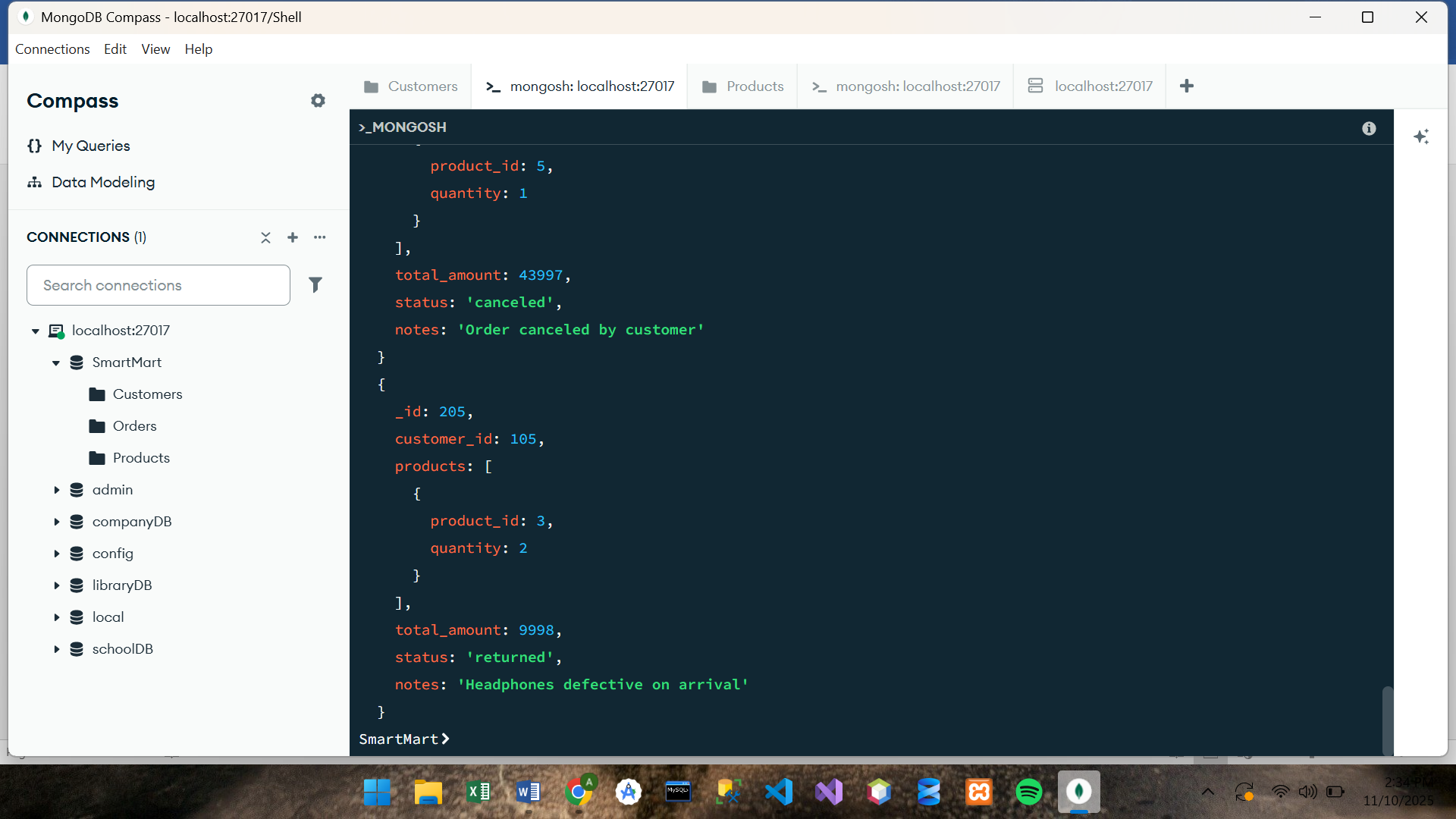
****

****

****

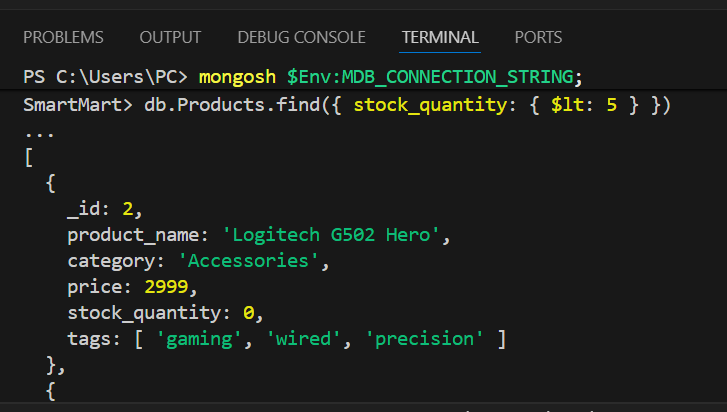
****

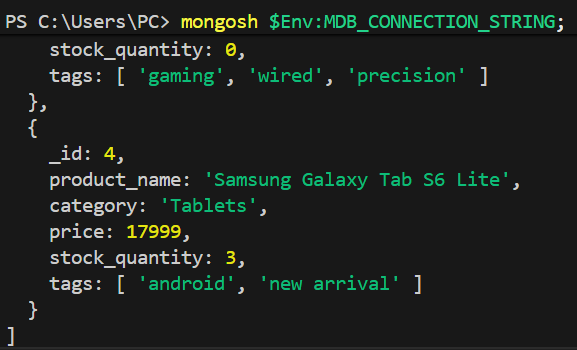
****

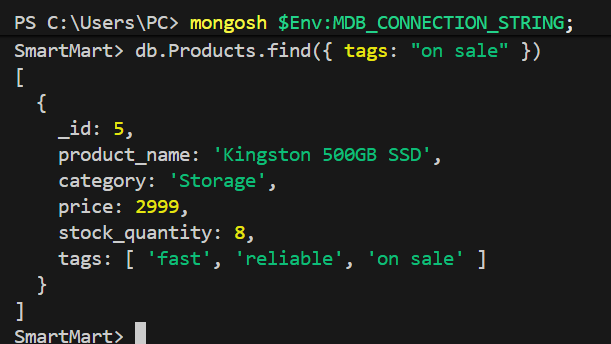
****

**C. Using Operators**

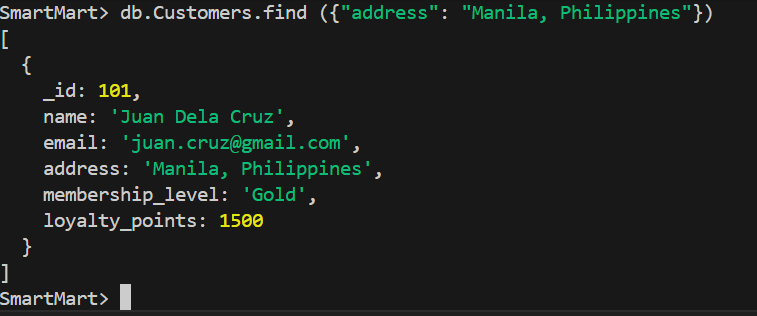
1. Finding products that are low in stock and on sale.

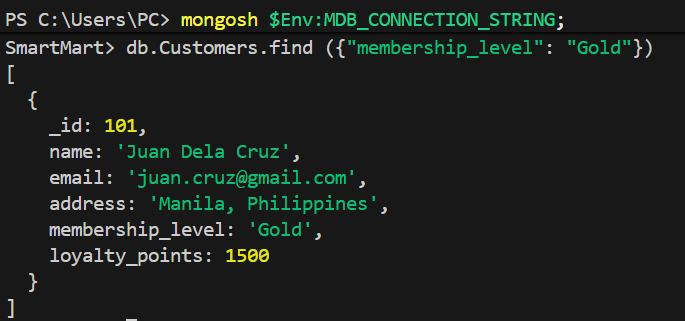






1. Finding customers from a certain city or with a premium membership.





1. Finding orders that are either pending or canceled.

