You are developing a simple inventory management system for a small store. The system needs to handle different types of data and ensure that certain data remains immutable or has restricted modifications.

Part 1: Data Types

Product Information: Create variables to store the following information for a product

productName: The name of the product (e.g., "T-Shirt")

productId: A unique ID for the product (e.g., Symbol("uniqueId"))

price: The price of the product (e.g., 25.99)

isOnSale: A boolean indicating whether the product is on sale (e.g., true)

quantity: The number of items in stock (e.g., 50)

Data Type Identification: Use console.log(typeof variableName) to check the data type of each variable you created

Inventory Array: Create an array called inventory to store multiple product objects. Each product object should have the properties defined above. Add at least three different products to the array.

Part 2: Immutability with

Freezing a Product: Choose one of the product objects in the inventory array and freeze it using Object.freeze()

Attempted Modifications: Try to modify the following properties of the frozen product

- ***Change the price.***
- ***Add a new property, such as discount.***
- ***Delete the quantity property.***

Use c**onsole.log()** to display whether these modifications were successful.

Nested Objects: Add a nested object called details to one of the product objects before freezing it. The details object should contain properties like color and size. **Freeze** the product object and then try to modify the color property inside the details object.

Explain why this modification is still possible and how you could prevent it

Deep Freeze: Implement a deep freeze function to freeze the product object, including its nested objects. Test that modifications to nested properties are no longer possible.

Part 3: Restricted Modifications with

Sealing a Product: Choose another product object in the inventory array and seal it using Object.seal()

- ***Attempted Modifications****: Try to perform the following actions on the sealed product*
- ***Modify the price.***
- *Add a new property, such as description.*
- *Delete the quantity property.*

Use console.log() to display whether each action was successful.

Comparison: Explain the differences in behavior between Object.freeze() and Object.seal()

Part 4: Variable Assignment and Mutability

Primitive vs. Non-Primitive: Create a **primitive variable** (e.g., a number or string) and a **non-primitive variable** (e.g., an object or array)

Assignment: Assign the primitive variable to another variable and change the value of the new variable. Show that the original variable remains **unchanged**

Reference: Assign the non-primitive variable to another variable and modify a property of the new variable. ***Explain how this affects the original variable***

Part 5: Best Practices

Choosing the Right Method: Describe scenarios where you would use **const, Object.freeze(), or Object.seal()** to manage data immutability and restricted modifications

Variable Naming*: Follow the rules for naming variables in JavaScript. Provide examples of valid and invalid variable names*