E-commerce Inventory Analyzer

Scenario You work for an online store and have received a raw inventory list. Using only loops, conditionals, and direct object/array operations , you'll analyze and update this data to generate key business insights.

Data Setup

1. **Create** an array `products` containing **8–10** product objects. Each object should have:

```
{
  id: 'P001',                  // unique string
  name: 'Wireless Mouse',      // product name
  category: 'Electronics',     // e.g. 'Electronics', 'Books', 'Clothing', etc.
  price: 29.99,                // number
  stock: 12                    // integer
}
```

Tasks

1. **Compute Total Inventory Value**
   - Initialize a variable `totalValue = 0`.
   - **Loop** through `products` and for each item add `price * stock` to `totalValue`.
   - After the loop, log:

   ```
   Total inventory value: $XYZ
   ```

2. **Build Category Summary**
   - Initialize an empty object:

   ```
   const categorySummary = {};
   ```

   - **Loop** through each product:
     - Let `cat = product.category`.
     - If `categorySummary[cat]` doesn't exist, set it to `0`.
     - Add `price * stock` to `categorySummary[cat]`.
   - After the loop, log each category and its total value.

3. **Flag Low-Stock Items**

   - Decide on a threshold (e.g. `const LOW_STOCK = 5`).
   - **Loop** through `products` and **add** a new property on each object:

     ```
     product.lowStock = (product.stock < LOW_STOCK);
     ```

   - After, log the names of all products where `lowStock === true`.

4. **Apply Discount to Electronics**
   - **Loop** through `products`:
     - If `product.category === 'Electronics'`, reduce its price by 10%:

       ```
       product.price = product.price * 0.9;
       ```

   - Log the updated prices for all electronics.

5. **Filter Available Products**
   - Create an empty array:

     ```
     const available = [];
     ```

   - **Loop** through `products`:
     - If `product.stock > 0`, **push** the entire product object into `available`.
   - Log how many items are available and list their names.

6. **Remove Discontinued Items**
   - Decide: any product with `price <= 0` or `stock === 0` is “discontinued.”
   - **Loop** backwards (from end to start) through `products`:
     - If discontinued, **use** `products.splice(index, 1)` to remove it.
   - After cleanup, log the new length of `products`.

7. **Bonus: Clone & Extend a Product**
   - Pick the first product in `products`.
   - Create a shallow clone using the spread operator:

     ```
     const clone = { ...products[0], warranty: '1 year' };
     ```

  - Log both the original and the clone to verify only the clone has `warranty`.