INVENTORY MANAGEMENT SYSTEM

```
import tkinter as tk
from tkinter import messagebox
from datetime import datetime

# Sample data (can be replaced with SQLite/SQLAlchemy for persistent storage)
inventory = [
    {"id": 1, "name": "Product A", "price": 10.0, "quantity": 100},
    {"id": 2, "name": "Product B", "price": 15.0, "quantity": 50},
    {"id": 3, "name": "Product C", "price": 20.0, "quantity": 75},
]

sales = []

# Function to display products in a listbox
def display_products():
    product_listbox.delete(0, tk.END)
    for product in inventory:
        product_listbox.insert(tk.END, f"{product['name']} - ${product['price']} - Quantity: {product['quantity']}")

# Function to add a new product
def add_product():
    name = name_entry.get()
    price = float(price_entry.get())
    quantity = int(quantity_entry.get())

    # Add validation here

    new_product = {
        "id": len(inventory) + 1,
        "name": name,
        "price": price,
        "quantity": quantity
    }
    inventory.append(new_product)
    messagebox.showinfo("Success", "Product added successfully!")
    clear_entries()
    refresh_product_list()

# Function to handle selling a product
def sell_product():
    selected_index = product_listbox.curselection()
    if selected_index:
        selected_product = inventory[selected_index[0]]
        quantity_to_sell = int(sell_quantity_entry.get())
        if selected_product['quantity'] >= quantity_to_sell:
            selected_product['quantity'] -= quantity_to_sell
            sales.append({
                "product": selected_product['name'],
                "customer": customer_entry.get(),  # Customer name from entry
                "quantity": quantity_to_sell,
                "total_price": selected_product['price'] * quantity_to_sell,
                "date": datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            })
            messagebox.showinfo("Success", f"{quantity_to_sell}
{selected_product['name']} sold to {customer_entry.get()} successfully!")
            display_products()
            clear_entries()
        else:
            messagebox.showerror("Error", "Not enough stock to complete the
sale.")
    else:
        messagebox.showerror("Error", "Please select a product to sell.")

# Function to display sales history
def display_sales():
    sales_listbox.delete(0, tk.END)
    for sale in sales:
        sales_listbox.insert(tk.END, f"{sale['date']} - {sale['customer']} bought
{sale['quantity']} of {sale['product']} for ${sale['total_price']}")

# Function to clear entry fields
def clear_entries():
    name_entry.delete(0, tk.END)
    price_entry.delete(0, tk.END)
    quantity_entry.delete(0, tk.END)
    sell_quantity_entry.delete(0, tk.END)
    customer_entry.delete(0, tk.END)

# Create GUI
root = tk.Tk()
root.title("Inventory Management System")

# Labels and entries for adding products
tk.Label(root, text="Add Product").grid(row=0, column=0, columnspan=2)
tk.Label(root, text="Name").grid(row=1, column=0)
tk.Label(root, text="Price").grid(row=2, column=0)
tk.Label(root, text="Quantity").grid(row=3, column=0)

name_entry = tk.Entry(root)
name_entry.grid(row=1, column=1)
```

```
price_entry = tk.Entry(root)
price_entry.grid(row=2, column=1)
quantity_entry = tk.Entry(root)
quantity_entry.grid(row=3, column=1)

add_button = tk.Button(root, text="Add Product", command=add_product)
add_button.grid(row=4, column=0, columnspan=2)

# Listbox to display products
tk.Label(root, text="Products").grid(row=5, column=0, columnspan=2)
product_listbox = tk.Listbox(root, width=50)
product_listbox.grid(row=6, column=0, columnspan=2)
display_products()

# Labels and entries for selling products
tk.Label(root, text="Sell Product").grid(row=7, column=0, columnspan=2)
tk.Label(root, text="Customer Name").grid(row=8, column=0)
customer_entry = tk.Entry(root)
customer_entry.grid(row=8, column=1)
tk.Label(root, text
```