Project 1: Inventory Management System

**"I run a small retail store and struggle to keep track of my inventory. Can you create a software solution that
helps me manage my products, sales, and inventory more efficiently?"**

Features Requested:
**Product Management: "I need a system where I can easily add new products, update their details, and
remove outdated ones."**
**Stock Tracking: "Can the software track how many of each item I have in stock and notify me when I need to
restock?"**
Sales Recording: "I want to record each sale and have it automatically update my inventory levels."
**Report Generation: "It would be great to generate reports showing what's selling well and what's low in
stock."**

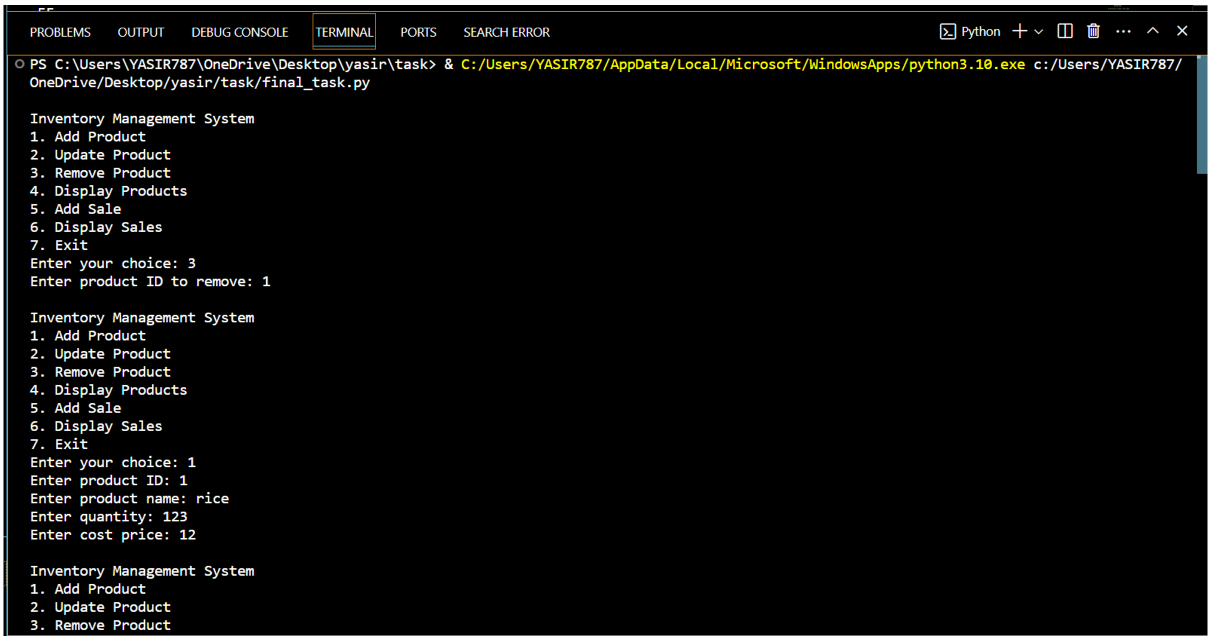
Solution

```
import sqlite3
import pandas as pd

# Connect to the database
conn = sqlite3.connect("inventory.db")
cursor = conn.cursor()

# Create the products table if it doesn't exist
cursor.execute("""CREATE TABLE IF NOT EXISTS products (
   product_id TEXT PRIMARY KEY,
   product_name TEXT,
   quantity INTEGER,
   price REAL
)""")

# Create the sales table if it doesn't exist
cursor.execute("""CREATE TABLE IF NOT EXISTS sales (
   sale_id INTEGER PRIMARY KEY AUTOINCREMENT,
   product_id TEXT,
   quantity INTEGER,
   price REAL,
   sale_date DATETIME
)""")

# Function to add a new product
def add_product():
   product_id = input("Enter product ID: ")
```

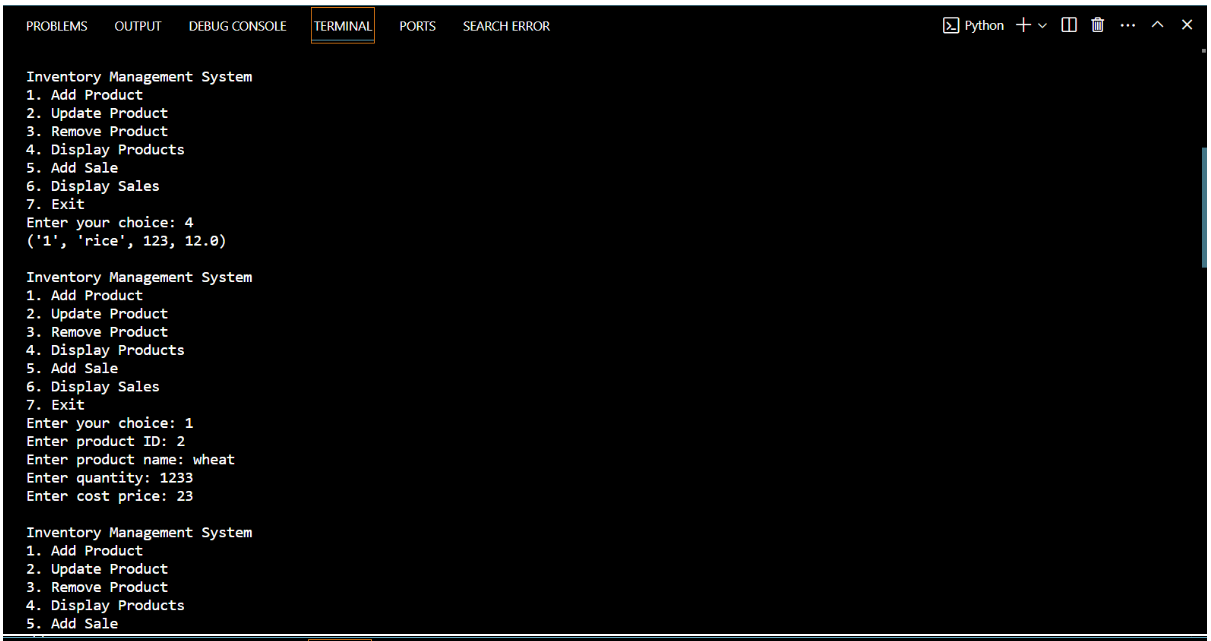
```
    product_name = input("Enter product name: ")
    quantity = int(input("Enter quantity: "))
    price = float(input("Enter cost price: "))
    cursor.execute("""INSERT INTO products VALUES (?, ?, ?, ?)""", (product_id, product_name,
quantity, price))
    conn.commit()

# Function to update product details
def update_product():
    product_id = input("Enter product ID to update: ")
    product_name = input("Enter new product name: ")
    quantity = int(input("Enter new quantity: "))
    price = float(input("Enter new cost price: "))
    cursor.execute("""UPDATE products SET product_name = ?, quantity = ?, price = ? WHERE
product_id = ?""", (product_name, quantity, price, product_id))
    conn.commit()

# Function to remove a product
def remove_product():
    product_id = input("Enter product ID to remove: ")
    cursor.execute("""DELETE FROM products WHERE product_id = ?""", (product_id,))
    conn.commit()

# Function to display all products
def display_products():
    cursor.execute("""SELECT * FROM products""")
    rows = cursor.fetchall()
    for row in rows:
        print(row)

# Function to add a new sale
def add_sale():
    product_id = input("Enter product ID: ")
    quantity = int(input("Enter quantity sold: "))
    price = float(input("Enter sales price: "))
    sale_date = input("Enter sale date (YYYY-MM-DD): ")

    # Update the product quantity in the products table
    cursor.execute("""UPDATE products SET quantity = quantity - ? WHERE product_id = ?""",
(quantity, product_id))

    # Insert the sale record into the sales table
    cursor.execute("""INSERT INTO sales (product_id, quantity, price, sale_date) VALUES (?, ?, ?,
?)""", (product_id, quantity, price, sale_date))

    conn.commit()

# Function to display all sales
```

```
def display_sales():
    cursor.execute("""SELECT * FROM sales""")
    rows = cursor.fetchall()
    for row in rows:
        print(row)

# Main program loop
while True:
    print("\nInventory Management System")
    print("1. Add Product")
    print("2. Update Product")
    print("3. Remove Product")
    print("4. Display Products")
    print("5. Add Sale")
    print("6. Display Sales")
    print("7. Exit")

    choice = int(input("Enter your choice: "))
    if choice == 1:
        add_product()
    elif choice == 2:
        update_product()
    elif choice == 3:
        remove_product()
    elif choice == 4:
        display_products()
    elif choice == 5:
        add_sale()
    elif choice == 6:
        display_sales()
    elif choice == 7:
        break

# Close the database connection
conn.close()
```

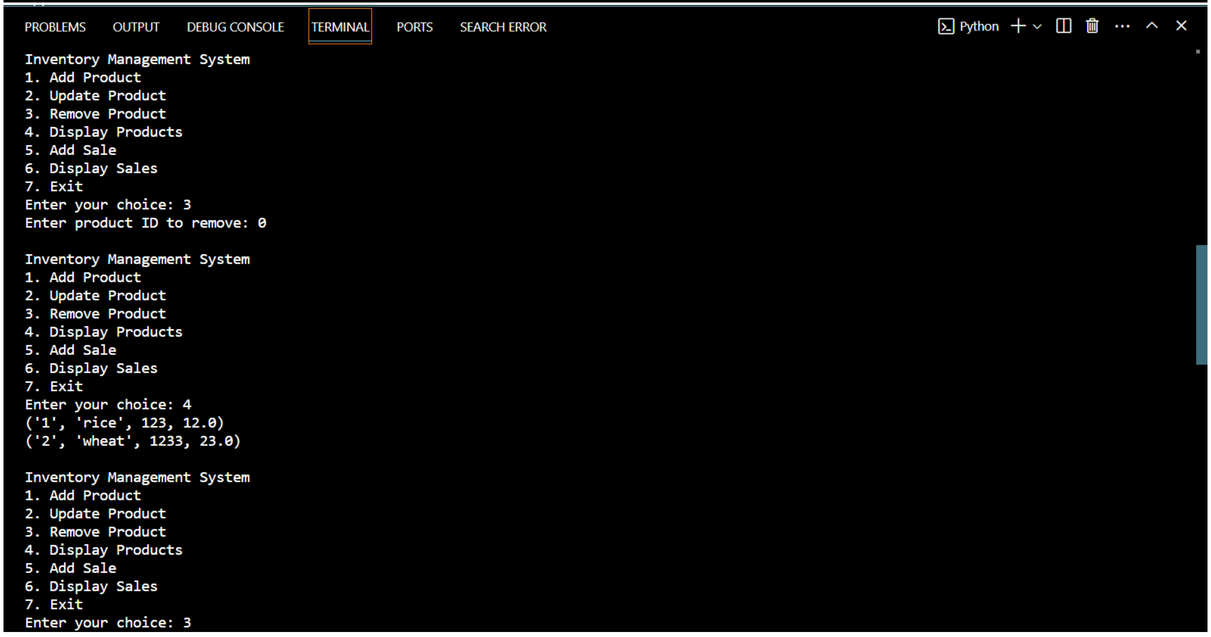
Outputs



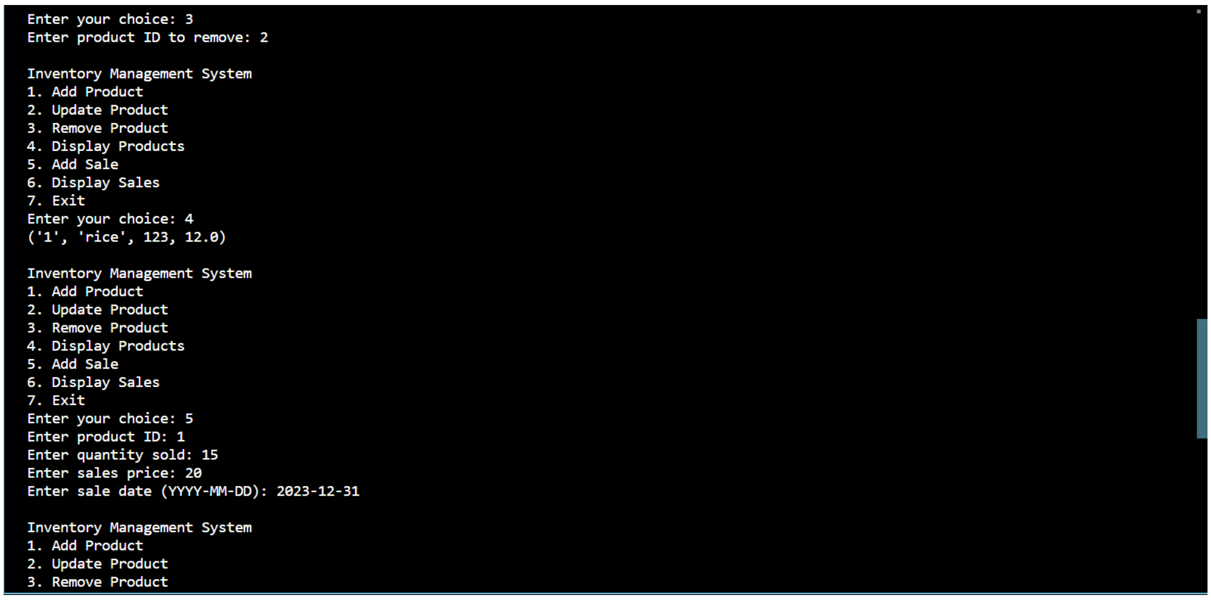
```
PS C:\Users\YASIR787\OneDrive\Desktop\yasir\task> & C:/Users/YASIR787/AppData/Local/Microsoft/WindowsApps/python3.10.exe c:/Users/YASIR787/
OneDrive/Desktop/yasir/task/final_task.py

Inventory Management System
1. Add Product
2. Update Product
3. Remove Product
4. Display Products
5. Add Sale
6. Display Sales
7. Exit
Enter your choice: 3
Enter product ID to remove: 1

Inventory Management System
1. Add Product
2. Update Product
3. Remove Product
4. Display Products
5. Add Sale
6. Display Sales
7. Exit
Enter your choice: 1
Enter product ID: 1
Enter product name: rice
Enter quantity: 123
Enter cost price: 12

Inventory Management System
1. Add Product
2. Update Product
3. Remove Product
```



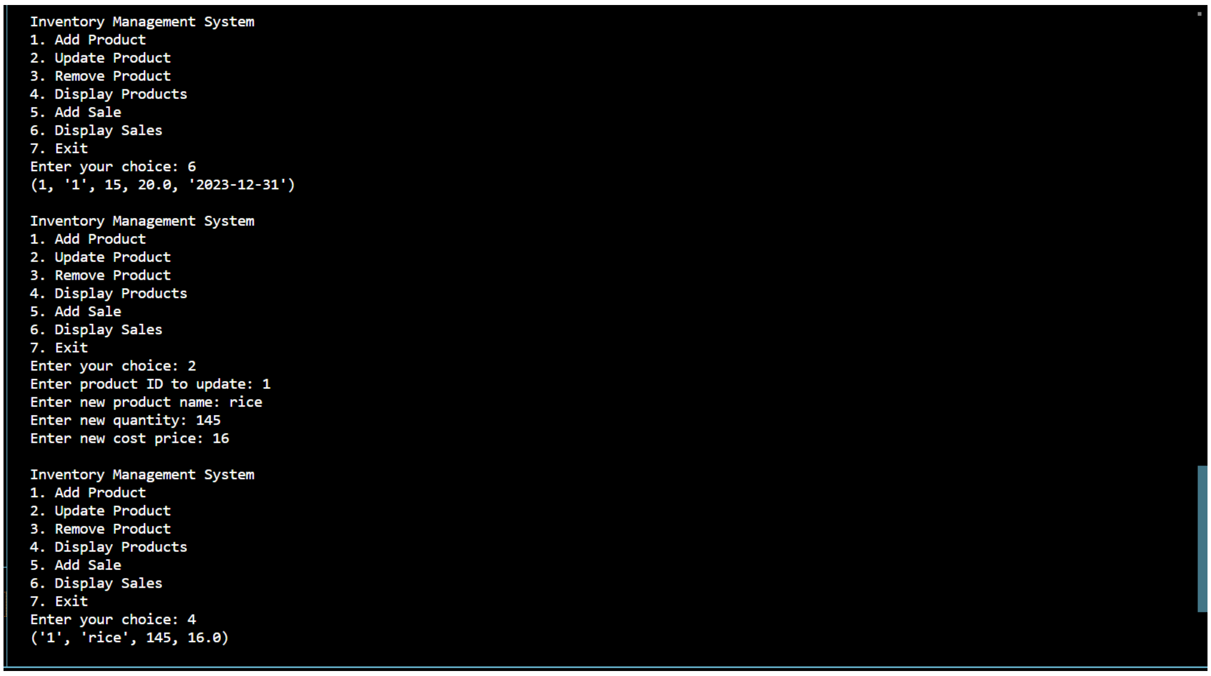
```
Inventory Management System
1. Add Product
2. Update Product
3. Remove Product
4. Display Products
5. Add Sale
6. Display Sales
7. Exit
Enter your choice: 4
('1', 'rice', 123, 12.0)

Inventory Management System
1. Add Product
2. Update Product
3. Remove Product
4. Display Products
5. Add Sale
6. Display Sales
7. Exit
Enter your choice: 1
Enter product ID: 2
Enter product name: wheat
Enter quantity: 1233
Enter cost price: 23

Inventory Management System
1. Add Product
2. Update Product
3. Remove Product
4. Display Products
5. Add Sale
```



```
Inventory Management System
1. Add Product
2. Update Product
3. Remove Product
4. Display Products
5. Add Sale
6. Display Sales
7. Exit
Enter your choice: 3
Enter product ID to remove: 0

Inventory Management System
1. Add Product
2. Update Product
3. Remove Product
4. Display Products
5. Add Sale
6. Display Sales
7. Exit
Enter your choice: 4
('1', 'rice', 123, 12.0)
('2', 'wheat', 1233, 23.0)

Inventory Management System
1. Add Product
2. Update Product
3. Remove Product
4. Display Products
5. Add Sale
6. Display Sales
7. Exit
Enter your choice: 3
```

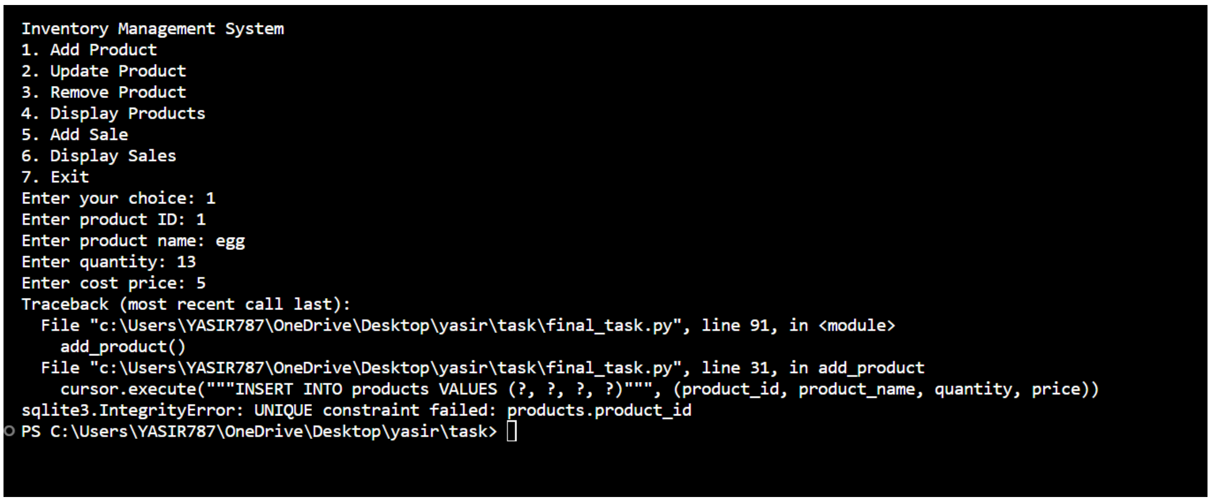


```
Enter your choice: 3
Enter product ID to remove: 2

Inventory Management System
1. Add Product
2. Update Product
3. Remove Product
4. Display Products
5. Add Sale
6. Display Sales
7. Exit
Enter your choice: 4
('1', 'rice', 123, 12.0)

Inventory Management System
1. Add Product
2. Update Product
3. Remove Product
4. Display Products
5. Add Sale
6. Display Sales
7. Exit
Enter your choice: 5
Enter product ID: 1
Enter quantity sold: 15
Enter sales price: 20
Enter sale date (YYYY-MM-DD): 2023-12-31

Inventory Management System
1. Add Product
2. Update Product
3. Remove Product
```



```
Inventory Management System
1. Add Product
2. Update Product
3. Remove Product
4. Display Products
5. Add Sale
6. Display Sales
7. Exit
Enter your choice: 6
(1, '1', 15, 20.0, '2023-12-31')

Inventory Management System
1. Add Product
2. Update Product
3. Remove Product
4. Display Products
5. Add Sale
6. Display Sales
7. Exit
Enter your choice: 2
Enter product ID to update: 1
Enter new product name: rice
Enter new quantity: 145
Enter new cost price: 16

Inventory Management System
1. Add Product
2. Update Product
3. Remove Product
4. Display Products
5. Add Sale
6. Display Sales
7. Exit
Enter your choice: 4
('1', 'rice', 145, 16.0)
```

If we use the same id for two products it will show the error



```
Inventory Management System
1. Add Product
2. Update Product
3. Remove Product
4. Display Products
5. Add Sale
6. Display Sales
7. Exit
Enter your choice: 1
Enter product ID: 1
Enter product name: egg
Enter quantity: 13
Enter cost price: 5
Traceback (most recent call last):
  File "c:\Users\YASIR787\OneDrive\Desktop\yasir\task\final_task.py", line 91, in <module>
    add_product()
  File "c:\Users\YASIR787\OneDrive\Desktop\yasir\task\final_task.py", line 31, in add_product
    cursor.execute("""INSERT INTO products VALUES (?, ?, ?, ?)""", (product_id, product_name, quantity, price))
sqlite3.IntegrityError: UNIQUE constraint failed: products.product_id
PS C:\Users\YASIR787\OneDrive\Desktop\yasir\task> 
```

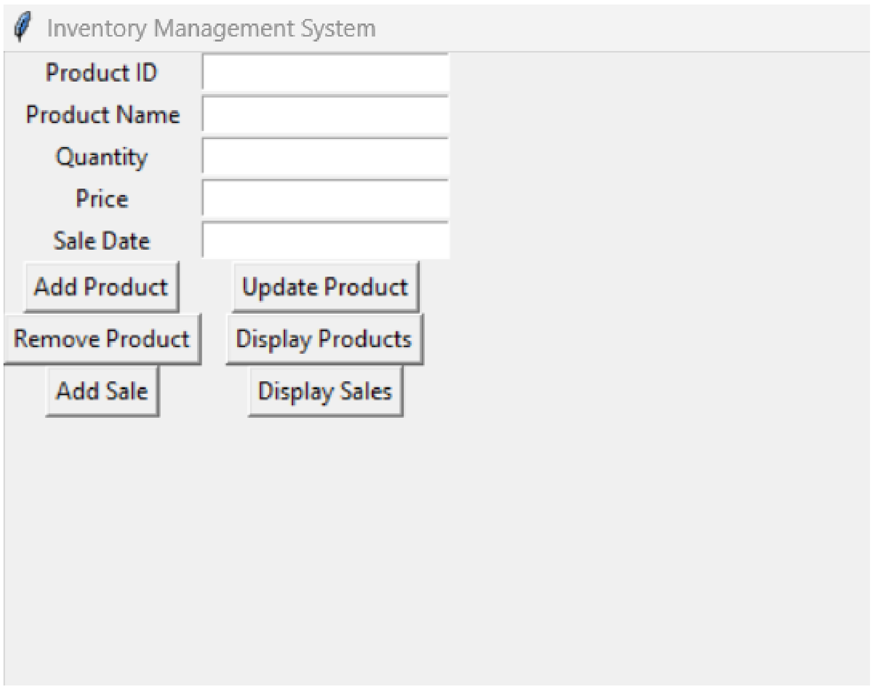
Solution with simple gui interface

```
import sqlite3
import tkinter as tk
from tkinter import messagebox

# Connect to the database
conn = sqlite3.connect("inventory.db")
cursor = conn.cursor()

# Create the products table if it doesn't exist
cursor.execute("""CREATE TABLE IF NOT EXISTS products (
   product_id TEXT PRIMARY KEY,
   product_name TEXT,
   quantity INTEGER,
   price REAL
)""")

# Create the sales table if it doesn't exist
cursor.execute("""CREATE TABLE IF NOT EXISTS sales (
   sale_id INTEGER PRIMARY KEY AUTOINCREMENT,
   product_id TEXT,
   quantity INTEGER,
   price REAL,
   sale_date DATETIME
)""")

# Function to add a new product
def add_product():
   product_id = entry_product_id.get()
   product_name = entry_product_name.get()
   quantity = entry_quantity.get()
   price = entry_price.get()
   cursor.execute("""INSERT INTO products VALUES (?, ?, ?, ?)""", (product_id, product_name,
quantity, price))
   conn.commit()
   messagebox.showinfo("Success", "Product added successfully")

# Function to update product details
def update_product():
   product_id = entry_product_id.get()
   product_name = entry_product_name.get()
   quantity = entry_quantity.get()
   price = entry_price.get()
   cursor.execute("""UPDATE products SET product_name = ?, quantity = ?, price = ? WHERE
product_id = ?""", (product_name, quantity, price, product_id))
   conn.commit()
   messagebox.showinfo("Success", "Product updated successfully")
```

```
# Function to remove a product
def remove_product():
  product_id = entry_product_id.get()
  cursor.execute("""DELETE FROM products WHERE product_id = ?""", (product_id,))
  conn.commit()
  messagebox.showinfo("Success", "Product removed successfully")

# Function to display all products
def display_products():
  cursor.execute("""SELECT * FROM products""")
  rows = cursor.fetchall()
  for row in rows:
    print(row)

# Function to add a new sale
def add_sale():
  product_id = entry_product_id.get()
  quantity = entry_quantity.get()
  price = entry_price.get()
  sale_date = entry_sale_date.get()

  cursor.execute("""UPDATE products SET quantity = quantity - ? WHERE product_id = ?""",
(quantity, product_id))

  cursor.execute("""INSERT INTO sales (product_id, quantity, price, sale_date) VALUES (?, ?, ?,
?)""", (product_id, quantity, price, sale_date))

  conn.commit()
  messagebox.showinfo("Success", "Sale added successfully")

# Function to display all sales
def display_sales():
  cursor.execute("""SELECT * FROM sales""")
  rows = cursor.fetchall()
  for row in rows:
    print(row)

# Create the main application window
root = tk.Tk()
root.title("Inventory Management System")

# Create and place labels and entry widgets for user input
tk.Label(root, text="Product ID").grid(row=0, column=0)
entry_product_id = tk.Entry(root)
entry_product_id.grid(row=0, column=1)

tk.Label(root, text="Product Name").grid(row=1, column=0)
```

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

tk.Label(root, text="Quantity").grid(row=2, column=0)
entry_quantity = tk.Entry(root)
entry_quantity.grid(row=2, column=1)

tk.Label(root, text="Price").grid(row=3, column=0)
entry_price = tk.Entry(root)
entry_price.grid(row=3, column=1)

tk.Label(root, text="Sale Date").grid(row=4, column=0)
entry_sale_date = tk.Entry(root)
entry_sale_date.grid(row=4, column=1)

# Create buttons for different actions
tk.Button(root, text="Add Product", command=add_product).grid(row=5, column=0)
tk.Button(root, text="Update Product", command=update_product).grid(row=5, column=1)
tk.Button(root, text="Remove Product", command=remove_product).grid(row=6, column=0)
tk.Button(root, text="Display Products", command=display_products).grid(row=6, column=1)
tk.Button(root, text="Add Sale", command=add_sale).grid(row=7, column=0)
tk.Button(root, text="Display Sales", command=display_sales).grid(row=7, column=1)

# Run the Tkinter event loop
root.mainloop()

# Close the database connection
conn.close()
```

Output



Inventory Management System
Product ID
Product Name
Quantity
Price
Sale Date
Add Product
Update Product
Remove Product
Display Products
Add Sale
Display Sales