Improving Inventory within a Business

Frances Brandofino, Madison Milton, Bailey Ramey, Van Vo

College of Business, University of Alabama in Huntsville

IS 491: IS Management & Strategy

Professor Karen Williams

November 18, 2024

Table of Contents

Project Overview

“The IS 491-01 Capstone Project capstone project will focus on investigating, researching, and applying Information System Management and Strategy concepts to a topic of the students’ choice. This is a group project with 3-4 maximum students per group.”

Improving Inventory within a Business

Background

Running and managing a business can arguably be described as one of the most challenging yet rewarding things to do. As a business owner you are responsible for the overall being of your store. This includes marketing, business development, business plans, finance and operation cost. It is imperative that you make smart decisions that will impact your business and most importantly your customers. One large aspect to help ensure this is keeping up with your stock/inventory.

Effective inventory management is essential for businesses, affecting both efficiency and customer satisfaction. Traditional methods, like manual tracking or basic spreadsheets, can lead to mistakes, delays, and extra costs. As businesses grow and add more products, these outdated methods become less effective, making it important to adopt better solutions. In today's time and with how fast the world is adapting to a more technical time, databases are a great solution for managing inventory challenges.

According to (Date, 2003, pg. 32) a database can be simply described as a computerized record-keeping system. The overall purpose of a database is to store information and allow users to retrieve and update said information. Databases have become a valuable solution for tackling these challenges. They offer a structured approach to storing, managing, and retrieving inventory data, allowing for real-time tracking of stock levels, sales, and supply chain activities. This helps businesses keep accurate records and enhance demand forecasting. Additionally, integrating databases with other systems, like point-of-sale and e-commerce platforms, improves visibility throughout the entire inventory process.

Scope

This paper looks at how database technology is used to manage inventory in different industries, such as retail, manufacturing, and logistics. The main areas covered are:

1. **Types of Databases and Technologies**: An introduction to different types of databases (like relational databases, SQL, and Python coding), explaining how each can help with inventory management.
2. **UML Diagrams:** An outline of how databases fully function on the backside. Exploring use case diagrams, domain model class diagrams, activity diagrams, data flow diagrams and system sequence diagrams.
3. **Inventory Management Processes**: A look at key inventory tasks, such as tracking stock, managing orders, forecasting demand and how databases can make these tasks more efficient.
4. **Real-time Data Management**: The importance of having access to real-time data to make quick, informed decisions about inventory, and how data analytics can help manage stock levels and reduce waste.
5. **Challenges and Trials**: Discusses potential issues with using databases for inventory management, such as security concerns and the need for user training.

Methods

Research & Findings

In developing an inventory management system for retailers, several methods of data management and tracking were researched by our group members. Below is a list of our research findings and out-dated and up to date methods:

1. **Manual Tracking**: Many businesses still depend on traditional, manual systems such as pen-and-paper to track their inventory and handle customer orders. While these methods are often affordable and easy to set up, they come with significant problems. As a business continues to grow and the demand for inventory increases, this method can become increasingly difficult to manage. Manual methods are more prone to errors, such as miscounting inventory or recording incorrect information, this can lead to major issues like running out of stock, over-ordering, or holding too much inventory.
2. **Spreadsheets:** The most popular spreadsheets used by businesses are excel and google sheets. Comparable to manual tracking, spreadsheets are low cost and easy to use. However, as businesses grow, spreadsheets become harder to manage. Since everything has to be typed in on a day-to-day basis they can be prone to errors and can result in data integrity issues when multiple users are involved. Additionally, they do not support real-time updates, making it difficult to maintain accurate records.
3. **Dedicated Inventory Management Software**: There are many pre-made inventory management solutions available on the internet, but they can create problems for businesses. Many of these systems are either too complex to easily learn and use or too expensive for many businesses depending on how much their budget is. For example,

some systems come with high licensing fees that businesses can't easily afford, while others include a wide range of advanced features that aren't relevant for their daily operations.

4. **Database Systems**: Relational database management systems like MySQL offer many benefits over traditional methods. These systems are designed to grow with the business, making it easy to handle large amounts of data as needs increase. They ensure that data remains accurate and organized, with built-in tools to reduce the risk of mistakes and automate repetitive tasks. Unlike manual methods, databases can be updated in real time, allowing businesses to track inventory accurately and manage orders more effectively. Additionally, when a database is accurately set up, it can be used to give business owners insight into how the business is performing, their sales, and where improvements could be made.

Benefits of Using Databases

Further research by our group showed us that using a database management system (including python & mySQL) offers several benefits over other methods. The benefits are listed below:

1. **Maintaining Data Accuracy:** Databases are designed with tools to help keep information consistent and accurate. Features like primary keys, foreign keys, and unique constraints work to prevent data conflicts and errors. This ensures that inventory records, orders, and sales data remain reliable, minimizing the risk of duplicate or inconsistent entries.
2. **Scalability**: A database system can efficiently handle large amounts of data, making it easy for small businesses to scale their operations as they grow. As inventory levels and

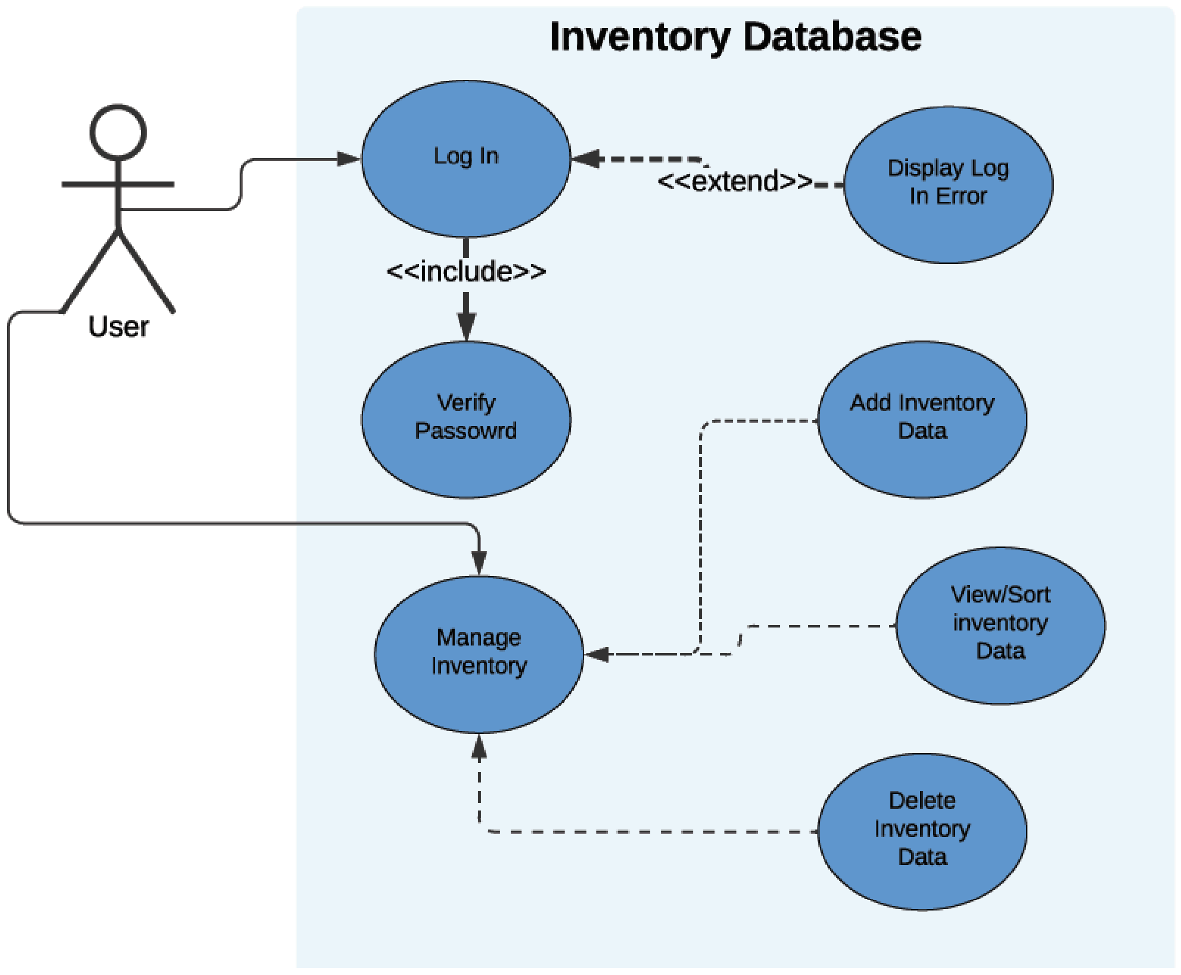
order quantities increase, the system can accommodate the added demand without a noticeable decline in performance.

3. **Automation**: With a database, it is possible to automate tasks such as updating inventory levels, generating reports, and sending alerts when stock reaches a threshold of a certain amount. This automation will help reduce the workload on staff and minimizes the potential for human errors.
4. **Real-Time Updates**: A database system ensures that inventory and order data are updated in real-time. This is important for businesses who need to track stock levels accurately to avoid overstocking or stockouts.
5. **Reporting and Insights:**: A database makes it easy to ask questions and generate reports. Sales data can be reviewed to find trends, customer preferences, and busy sales times, helping retailers make better decisions. Custom reports can also be generated to assess product performance and sales margins.
6. **Security**: Databases offer built-in security features, such as user authentication and access controls, to protect sensitive business data. Only authorized personnel can access or modify inventory and order records. This is essential because it gives the business a sense of security and allows them to know that their information is safe.
7. **Cost-Effective**: Creating a database is open-source and free to use, which makes it a cost-effective solution for businesses.. Combined with Python (which is also free and widely used), the system can be developed without incurring significant software licensing costs.

Approach

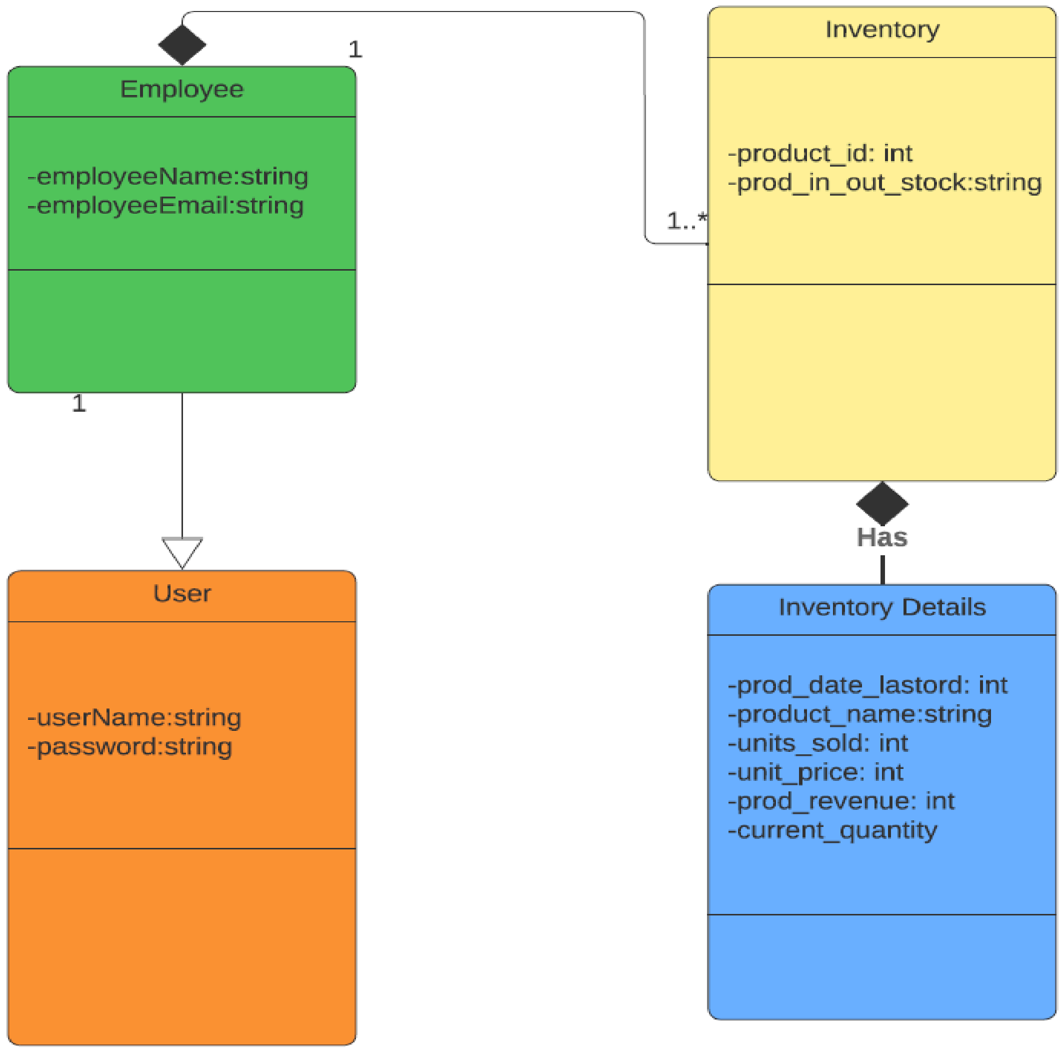
For the approach on how to structure our database system, we utilized lucidchart to map out our systems requirements and functionalities. We created a use case diagram, domain model class diagram, activity diagram, and sequence diagram. As each diagram is mapped out, it dives deeper into the structure of the database functionalities at different times of the database life cycle.

Use Case Diagram



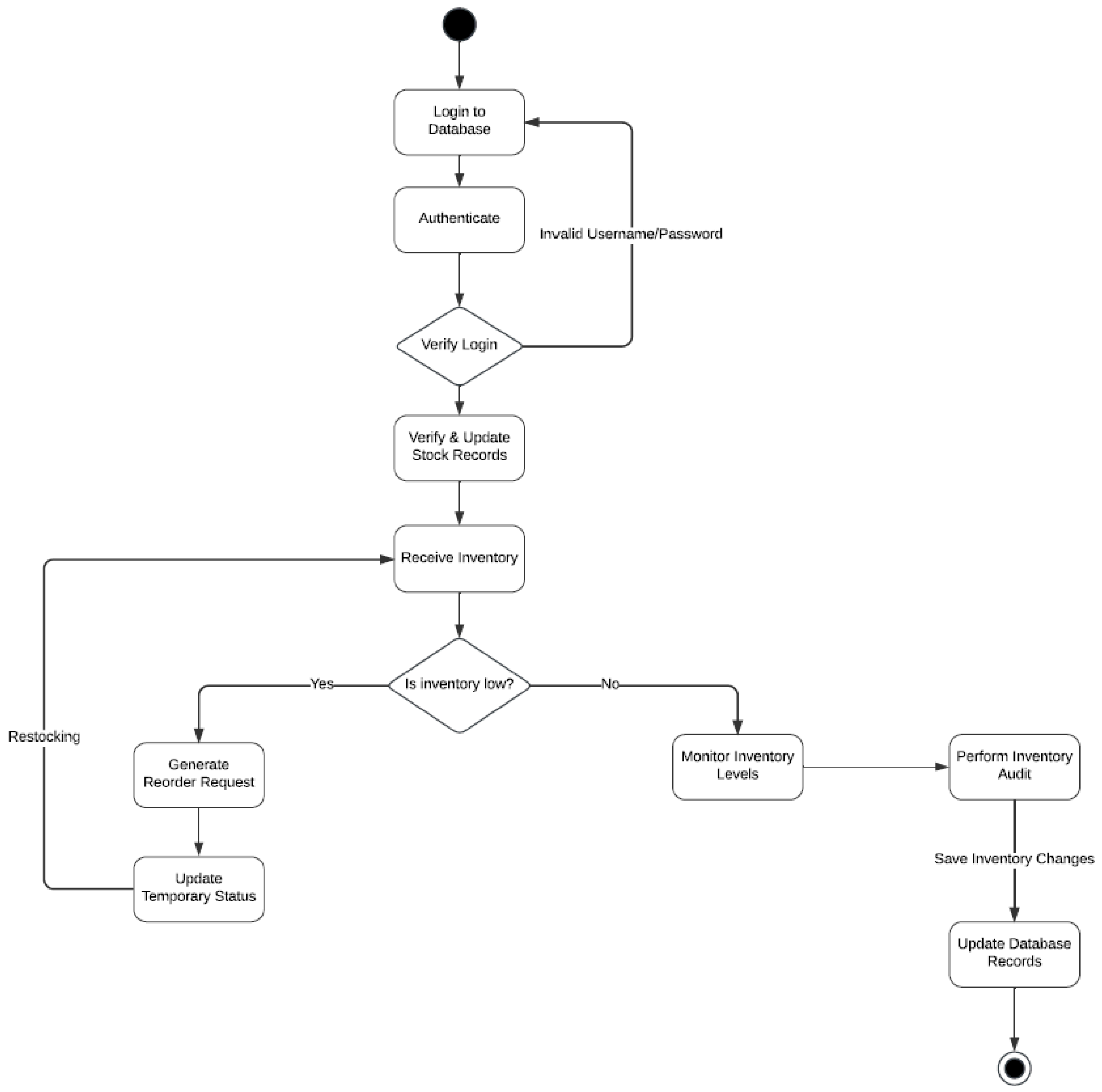
The primary users of this database will be employees of the company who need to utilize the inventory management system. All employees will be required to log in to the system. The users will have the ability to manage inventory, add inventory data, view/sort inventory data, and delta inventory data.

Domain Model Class Diagram



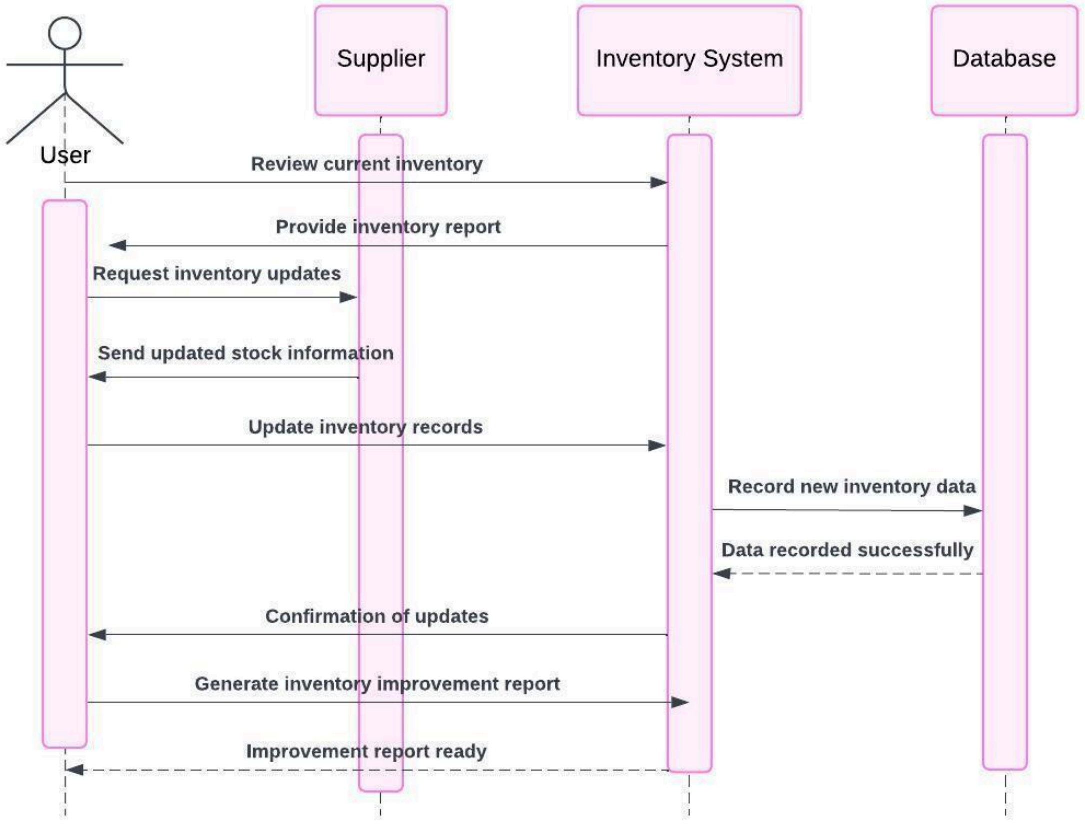
This database will be used to store, view, and maintain information that is relevant to the business utilizing inventory management. Employees will be able to maintain an accurate record of all the inventory items that they carry. Data will be grouped into several tables and further grouped based off the preceding information. Inventory levels are the most important aspect of the data because of the need to make sure that the business can stay stocked and not run out of supplies. Each inventory item can be modified by one or many users, and each inventory item will include product name, products last order date, units sold, unit pricing, products revenue, and current quantity in stock.

Activity Diagram



The activity diagram shows a much clearer general picture of how the system will operate. Employees will be able to easily access the data that they require in order to fulfill their duties. Employees will be able to login to the system. Upon successful login, they will be able receive/record inventory data as well as choose what to do with said data. If inventory is low, the employee will be able to decide to generate a reorder request. If inventory is not low, they can continue to monitor inventory levels for auditory purposes and add/delete/search functions as normal.

Sequence Diagram

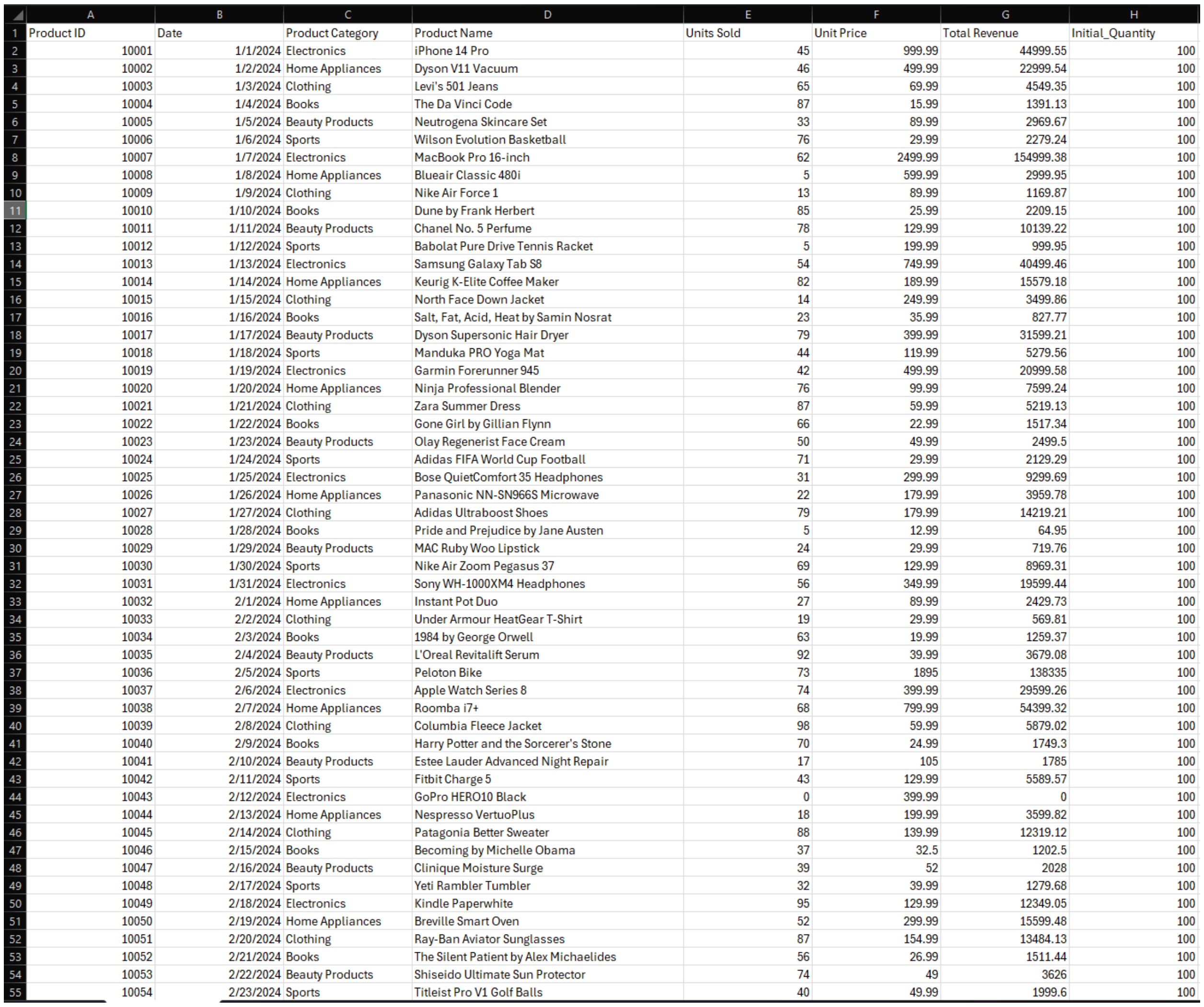


The system's sequence diagram shows the inputs and outputs of the system when an employee is using the database application to add, delete, view or change data. This diagram gives a clearer picture of the specifics of how the application will operate.

System Implementation

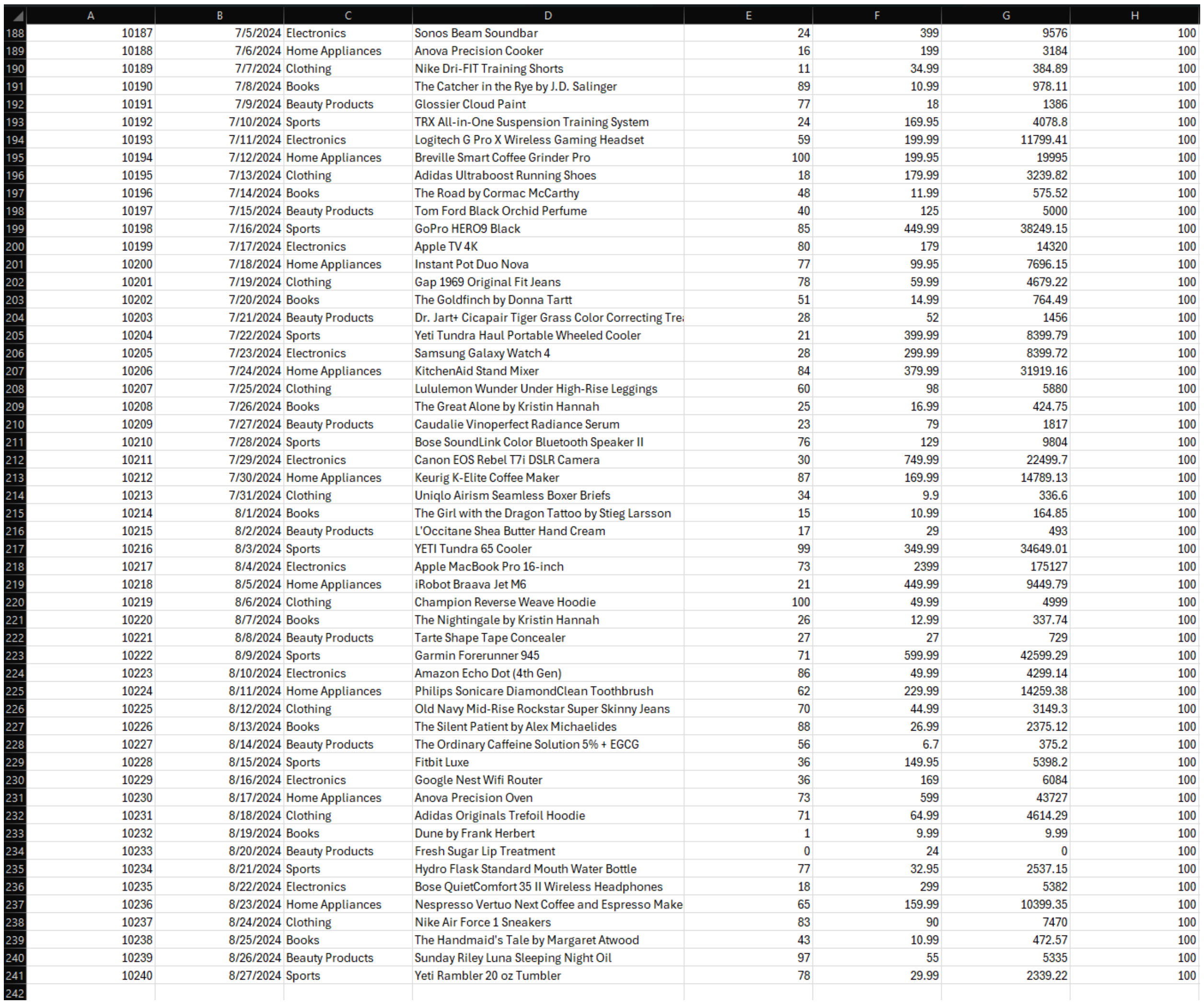
Python Code And Database

1. **“Online Sales Data.csv”** : We will be using an available CSV data file of a business taken from the Kaggle website for the project "Improving Inventory within a Business", and this file has a total of 240 products.



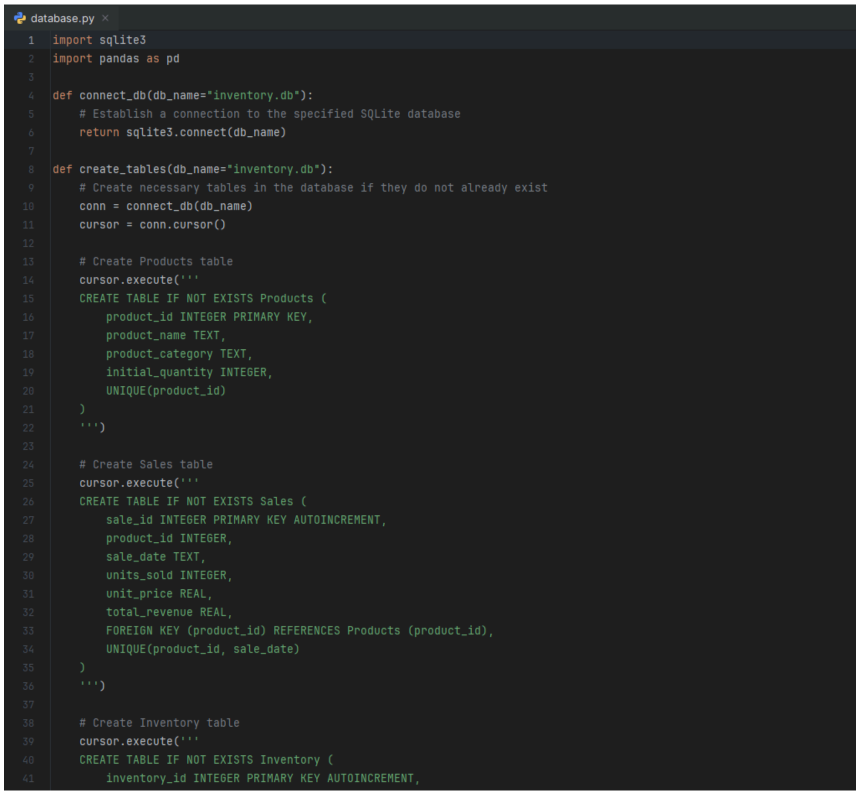
Product ID	Date	Product Category	Product Name	Units Sold	Unit Price	Total Revenue	Initial_Quantity
10001	1/1/2024	Electronics	iPhone 14 Pro	45	999.99	44999.55	100
10002	1/2/2024	Home Appliances	Dyson V11 Vacuum	46	499.99	22999.54	100
10003	1/3/2024	Clothing	Levi's 501 Jeans	65	69.99	4549.35	100
10004	1/4/2024	Books	The Da Vinci Code	87	15.99	1391.13	100
10005	1/5/2024	Beauty Products	Neutrogena Skincare Set	33	89.99	2969.67	100
10006	1/6/2024	Sports	Wilson Evolution Basketball	76	29.99	2279.24	100
10007	1/7/2024	Electronics	MacBook Pro 16-inch	62	2499.99	154999.38	100
10008	1/8/2024	Home Appliances	Blueair Classic 480i	5	599.99	2999.95	100
10009	1/9/2024	Clothing	Nike Air Force 1	13	89.99	1169.87	100
10010	1/10/2024	Books	Dune by Frank Herbert	85	25.99	2209.15	100
10011	1/11/2024	Beauty Products	Chanel No. 5 Perfume	78	129.99	10139.22	100
10012	1/12/2024	Sports	Babolat Pure Drive Tennis Racket	5	199.99	999.95	100
10013	1/13/2024	Electronics	Samsung Galaxy Tab S8	54	749.99	40499.46	100
10014	1/14/2024	Home Appliances	Keurig K-Elite Coffee Maker	82	189.99	15579.18	100
10015	1/15/2024	Clothing	North Face Down Jacket	14	249.99	3499.86	100
10016	1/16/2024	Books	Salt, Fat, Acid, Heat by Samin Nosrat	23	35.99	827.77	100
10017	1/17/2024	Beauty Products	Dyson Supersonic Hair Dryer	79	399.99	31599.21	100
10018	1/18/2024	Sports	Manduka PRO Yoga Mat	44	119.99	5279.56	100
10019	1/19/2024	Electronics	Garmin Forerunner 945	42	499.99	20999.58	100
10020	1/20/2024	Home Appliances	Ninja Professional Blender	76	99.99	7599.24	100
10021	1/21/2024	Clothing	Zara Summer Dress	87	59.99	5219.13	100
10022	1/22/2024	Books	Gone Girl by Gillian Flynn	66	22.99	1517.34	100
10023	1/23/2024	Beauty Products	Olay Regenerist Face Cream	50	49.99	2499.5	100
10024	1/24/2024	Sports	Adidas FIFA World Cup Football	71	29.99	2129.29	100
10025	1/25/2024	Electronics	Bose QuietComfort 35 Headphones	31	299.99	9299.69	100
10026	1/26/2024	Home Appliances	Panasonic NN-SN966S Microwave	22	179.99	3959.78	100
10027	1/27/2024	Clothing	Adidas Ultraboost Shoes	79	179.99	14219.21	100
10028	1/28/2024	Books	Pride and Prejudice by Jane Austen	5	12.99	64.95	100
10029	1/29/2024	Beauty Products	MAC Ruby Woo Lipstick	24	29.99	719.76	100
10030	1/30/2024	Sports	Nike Air Zoom Pegasus 37	69	129.99	8969.31	100
10031	1/31/2024	Electronics	Sony WH-1000XM4 Headphones	56	349.99	19599.44	100
10032	2/1/2024	Home Appliances	Instant Pot Duo	27	89.99	2429.73	100
10033	2/2/2024	Clothing	Under Armour HeatGear T-Shirt	19	29.99	569.81	100
10034	2/3/2024	Books	1984 by George Orwell	63	19.99	1259.37	100
10035	2/4/2024	Beauty Products	L'Oreal Revitalift Serum	92	39.99	3679.08	100
10036	2/5/2024	Sports	Peloton Bike	73	1895	138335	100
10037	2/6/2024	Electronics	Apple Watch Series 8	74	399.99	29599.26	100
10038	2/7/2024	Home Appliances	Roomba i7+	68	799.99	54399.32	100
10039	2/8/2024	Clothing	Columbia Fleece Jacket	98	59.99	5879.02	100
10040	2/9/2024	Books	Harry Potter and the Sorcerer's Stone	70	24.99	1749.3	100
10041	2/10/2024	Beauty Products	Estee Lauder Advanced Night Repair	17	105	1785	100
10042	2/11/2024	Sports	Fitbit Charge 5	43	129.99	5589.57	100
10043	2/12/2024	Electronics	GoPro HERO10 Black	0	399.99	0	100
10044	2/13/2024	Home Appliances	Nespresso VertuoPlus	18	199.99	3599.82	100
10045	2/14/2024	Clothing	Patagonia Better Sweater	88	139.99	12319.12	100
10046	2/15/2024	Books	Becoming by Michelle Obama	37	32.5	1202.5	100
10047	2/16/2024	Beauty Products	Clinique Moisture Surge	39	52	2028	100
10048	2/17/2024	Sports	Yeti Rambler Tumbler	32	39.99	1279.68	100
10049	2/18/2024	Electronics	Kindle Paperwhite	95	129.99	12349.05	100
10050	2/19/2024	Home Appliances	Breville Smart Oven	52	299.99	15599.48	100
10051	2/20/2024	Clothing	Ray-Ban Aviator Sunglasses	87	154.99	13484.13	100
10052	2/21/2024	Books	The Silent Patient by Alex Michaelides	56	26.99	1511.44	100
10053	2/22/2024	Beauty Products	Shiseido Ultimate Sun Protector	74	49	3626	100
10054	2/23/2024	Sports	Titleist Pro V1 Golf Balls	40	49.99	1999.6	100

…………

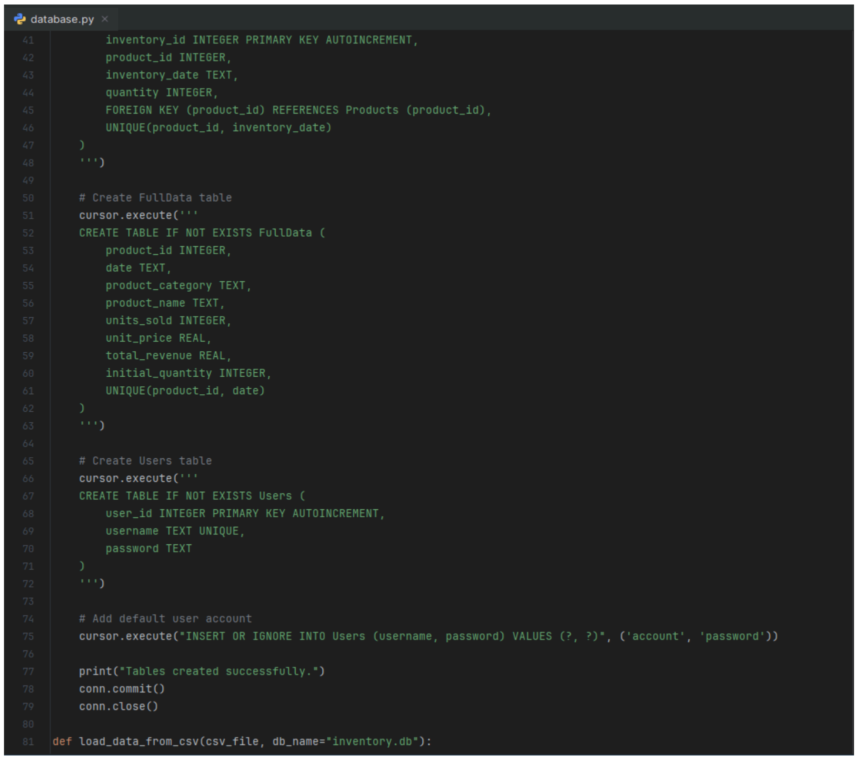


	A	B	C	D	E	F	G	H
188	10187	7/5/2024	Electronics	Sonos Beam Soundbar	24	399	9576	100
189	10188	7/6/2024	Home Appliances	Anova Precision Cooker	16	199	3184	100
190	10189	7/7/2024	Clothing	Nike Dri-FIT Training Shorts	11	34.99	384.89	100
191	10190	7/8/2024	Books	The Catcher in the Rye by J.D. Salinger	89	10.99	978.11	100
192	10191	7/9/2024	Beauty Products	Glossier Cloud Paint	77	18	1386	100
193	10192	7/10/2024	Sports	TRX All-in-One Suspension Training System	24	169.95	4078.8	100
194	10193	7/11/2024	Electronics	Logitech G Pro X Wireless Gaming Headset	59	199.99	11799.41	100
195	10194	7/12/2024	Home Appliances	Breville Smart Coffee Grinder Pro	100	199.95	19995	100
196	10195	7/13/2024	Clothing	Adidas Ultraboost Running Shoes	18	179.99	3239.82	100
197	10196	7/14/2024	Books	The Road by Cormac McCarthy	48	11.99	575.52	100
198	10197	7/15/2024	Beauty Products	Tom Ford Black Orchid Perfume	40	125	5000	100
199	10198	7/16/2024	Sports	GoPro HERO9 Black	85	449.99	38249.15	100
200	10199	7/17/2024	Electronics	Apple TV 4K	80	179	14320	100
201	10200	7/18/2024	Home Appliances	Instant Pot Duo Nova	77	99.95	7696.15	100
202	10201	7/19/2024	Clothing	Gap 1969 Original Fit Jeans	78	59.99	4679.22	100
203	10202	7/20/2024	Books	The Goldfinch by Donna Tartt	51	14.99	764.49	100
204	10203	7/21/2024	Beauty Products	Dr. Jart+ Cicapair Tiger Grass Color Correcting Tre	28	52	1456	100
205	10204	7/22/2024	Sports	Yeti Tundra Haul Portable Wheeled Cooler	21	399.99	8399.79	100
206	10205	7/23/2024	Electronics	Samsung Galaxy Watch 4	28	299.99	8399.72	100
207	10206	7/24/2024	Home Appliances	KitchenAid Stand Mixer	84	379.99	31919.16	100
208	10207	7/25/2024	Clothing	Lululemon Wunder Under High-Rise Leggings	60	98	5880	100
209	10208	7/26/2024	Books	The Great Alone by Kristin Hannah	25	16.99	424.75	100
210	10209	7/27/2024	Beauty Products	Caudalie Vinoperfect Radiance Serum	23	79	1817	100
211	10210	7/28/2024	Sports	Bose SoundLink Color Bluetooth Speaker II	76	129	9804	100
212	10211	7/29/2024	Electronics	Canon EOS Rebel T7i DSLR Camera	30	749.99	22499.7	100
213	10212	7/30/2024	Home Appliances	Keurig K-Elite Coffee Maker	87	169.99	14789.13	100
214	10213	7/31/2024	Clothing	Uniqlo Airism Seamless Boxer Briefs	34	9.9	336.6	100
215	10214	8/1/2024	Books	The Girl with the Dragon Tattoo by Stieg Larsson	15	10.99	164.85	100
216	10215	8/2/2024	Beauty Products	L'Occitane Shea Butter Hand Cream	17	29	493	100
217	10216	8/3/2024	Sports	YETI Tundra 65 Cooler	99	349.99	34649.01	100
218	10217	8/4/2024	Electronics	Apple MacBook Pro 16-inch	73	2399	175127	100
219	10218	8/5/2024	Home Appliances	iRobot Braava Jet M6	21	449.99	9449.79	100
220	10219	8/6/2024	Clothing	Champion Reverse Weave Hoodie	100	49.99	4999	100
221	10220	8/7/2024	Books	The Nightingale by Kristin Hannah	26	12.99	337.74	100
222	10221	8/8/2024	Beauty Products	Tarte Shape Tape Concealer	27	27	729	100
223	10222	8/9/2024	Sports	Garmin Forerunner 945	71	599.99	42599.29	100
224	10223	8/10/2024	Electronics	Amazon Echo Dot (4th Gen)	86	49.99	4299.14	100
225	10224	8/11/2024	Home Appliances	Philips Sonicare DiamondClean Toothbrush	62	229.99	14259.38	100
226	10225	8/12/2024	Clothing	Old Navy Mid-Rise Rockstar Super Skinny Jeans	70	44.99	3149.3	100
227	10226	8/13/2024	Books	The Silent Patient by Alex Michaelides	88	26.99	2375.12	100
228	10227	8/14/2024	Beauty Products	The Ordinary Caffeine Solution 5% + EGCG	56	6.7	375.2	100
229	10228	8/15/2024	Sports	Fitbit Luxe	36	149.95	5398.2	100
230	10229	8/16/2024	Electronics	Google Nest Wifi Router	36	169	6084	100
231	10230	8/17/2024	Home Appliances	Anova Precision Oven	73	599	43727	100
232	10231	8/18/2024	Clothing	Adidas Originals Trefoil Hoodie	71	64.99	4614.29	100
233	10232	8/19/2024	Books	Dune by Frank Herbert	1	9.99	9.99	100
234	10233	8/20/2024	Beauty Products	Fresh Sugar Lip Treatment	0	24	0	100
235	10234	8/21/2024	Sports	Hydro Flask Standard Mouth Water Bottle	77	32.95	2537.15	100
236	10235	8/22/2024	Electronics	Bose QuietComfort 35 II Wireless Headphones	18	299	5382	100
237	10236	8/23/2024	Home Appliances	Nespresso Vertuo Next Coffee and Espresso Make	65	159.99	10399.35	100
238	10237	8/24/2024	Clothing	Nike Air Force 1 Sneakers	83	90	7470	100
239	10238	8/25/2024	Books	The Handmaid's Tale by Margaret Atwood	43	10.99	472.57	100
240	10239	8/26/2024	Beauty Products	Sunday Riley Luna Sleeping Night Oil	97	55	5335	100
241	10240	8/27/2024	Sports	Yeti Rambler 20 oz Tumbler	78	29.99	2339.22	100
242								

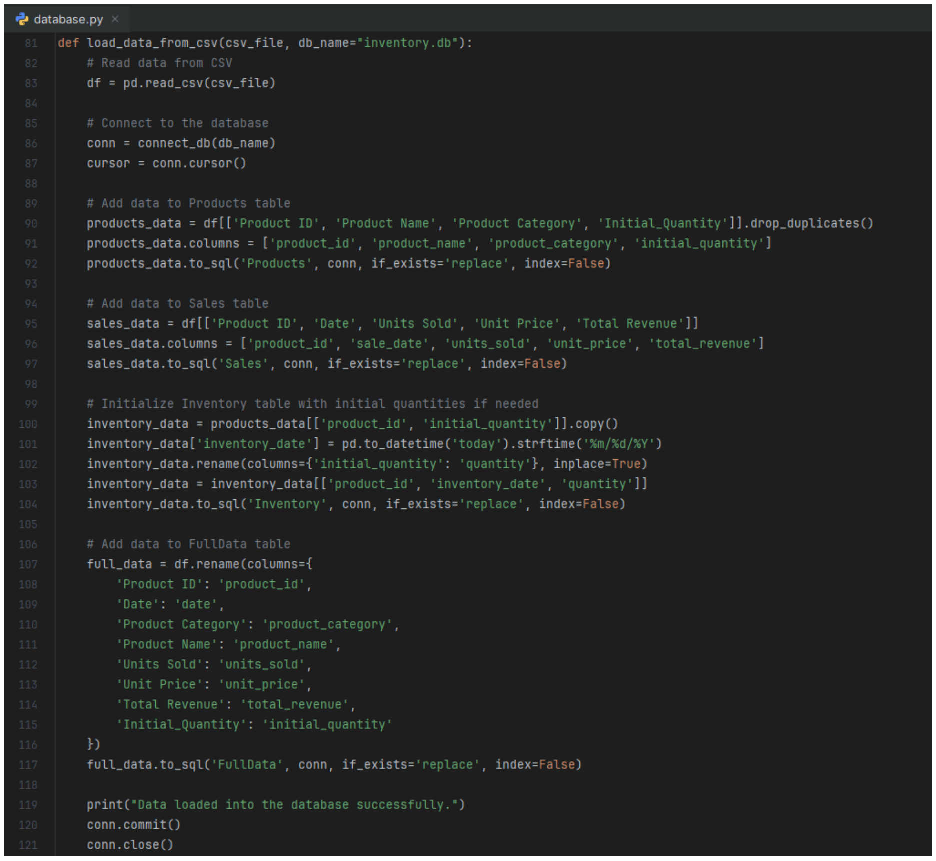
2. The **database.py** file: acts as an intermediary between the application and the database. It makes sure that a connection to the database is always available for such tasks as querying, adding, updating, and deleting data.


database.py

```python
import sqlite3
import pandas as pd

def connect_db(db_name="inventory.db"):
    # Establish a connection to the specified SQLite database
    return sqlite3.connect(db_name)

def create_tables(db_name="inventory.db"):
    # Create necessary tables in the database if they do not already exist
    conn = connect_db(db_name)
    cursor = conn.cursor()

    # Create Products table
    cursor.execute('''
    CREATE TABLE IF NOT EXISTS Products (
        product_id INTEGER PRIMARY KEY,
        product_name TEXT,
        product_category TEXT,
        initial_quantity INTEGER,
        UNIQUE(product_id)
    )
    ''')

    # Create Sales table
    cursor.execute('''
    CREATE TABLE IF NOT EXISTS Sales (
        sale_id INTEGER PRIMARY KEY AUTOINCREMENT,
        product_id INTEGER,
        sale_date TEXT,
        units_sold INTEGER,
        unit_price REAL,
        total_revenue REAL,
        FOREIGN KEY (product_id) REFERENCES Products (product_id),
        UNIQUE(product_id, sale_date)
    )
    ''')

    # Create Inventory table
    cursor.execute('''
    CREATE TABLE IF NOT EXISTS Inventory (
        inventory_id INTEGER PRIMARY KEY AUTOINCREMENT,
```

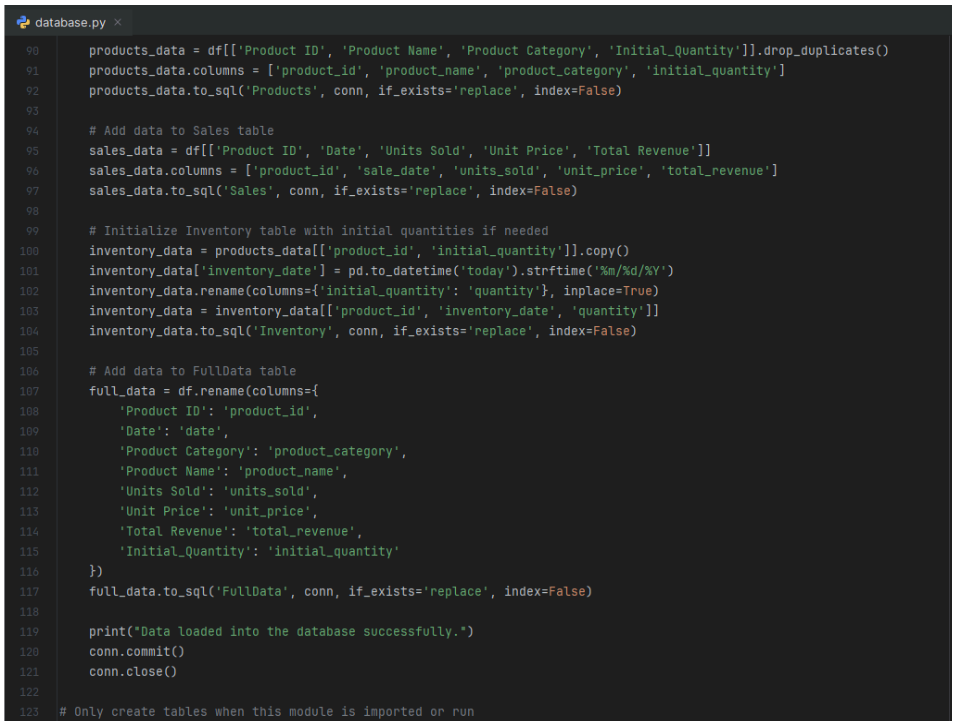

database.py

```python
        inventory_id INTEGER PRIMARY KEY AUTOINCREMENT,
        product_id INTEGER,
        inventory_date TEXT,
        quantity INTEGER,
        FOREIGN KEY (product_id) REFERENCES Products (product_id),
        UNIQUE(product_id, inventory_date)
    )
    ''')

    # Create FullData table
    cursor.execute('''
    CREATE TABLE IF NOT EXISTS FullData (
        product_id INTEGER,
        date TEXT,
        product_category TEXT,
        product_name TEXT,
        units_sold INTEGER,
        unit_price REAL,
        total_revenue REAL,
        initial_quantity INTEGER,
        UNIQUE(product_id, date)
    )
    ''')

    # Create Users table
    cursor.execute('''
    CREATE TABLE IF NOT EXISTS Users (
        user_id INTEGER PRIMARY KEY AUTOINCREMENT,
        username TEXT UNIQUE,
        password TEXT
    )
    ''')

    # Add default user account
    cursor.execute("INSERT OR IGNORE INTO Users (username, password) VALUES (?, ?)", ('account', 'password'))

    print("Tables created successfully.")
    conn.commit()
    conn.close()

def load_data_from_csv(csv_file, db_name="inventory.db"):
```



database.py

```python
def load_data_from_csv(csv_file, db_name="inventory.db"):
    # Read data from CSV
    df = pd.read_csv(csv_file)

    # Connect to the database
    conn = connect_db(db_name)
    cursor = conn.cursor()

    # Add data to Products table
    products_data = df[['Product ID', 'Product Name', 'Product Category', 'Initial_Quantity']].drop_duplicates()
    products_data.columns = ['product_id', 'product_name', 'product_category', 'initial_quantity']
    products_data.to_sql('Products', conn, if_exists='replace', index=False)

    # Add data to Sales table
    sales_data = df[['Product ID', 'Date', 'Units Sold', 'Unit Price', 'Total Revenue']]
    sales_data.columns = ['product_id', 'sale_date', 'units_sold', 'unit_price', 'total_revenue']
    sales_data.to_sql('Sales', conn, if_exists='replace', index=False)

    # Initialize Inventory table with initial quantities if needed
    inventory_data = products_data[['product_id', 'initial_quantity']].copy()
    inventory_data['inventory_date'] = pd.to_datetime('today').strftime('%m/%d/%Y')
    inventory_data.rename(columns={'initial_quantity': 'quantity'}, inplace=True)
    inventory_data = inventory_data[['product_id', 'inventory_date', 'quantity']]
    inventory_data.to_sql('Inventory', conn, if_exists='replace', index=False)

    # Add data to FullData table
    full_data = df.rename(columns={
        'Product ID': 'product_id',
        'Date': 'date',
        'Product Category': 'product_category',
        'Product Name': 'product_name',
        'Units Sold': 'units_sold',
        'Unit Price': 'unit_price',
        'Total Revenue': 'total_revenue',
        'Initial_Quantity': 'initial_quantity'
    })
    full_data.to_sql('FullData', conn, if_exists='replace', index=False)

    print("Data loaded into the database successfully.")
    conn.commit()
    conn.close()
```

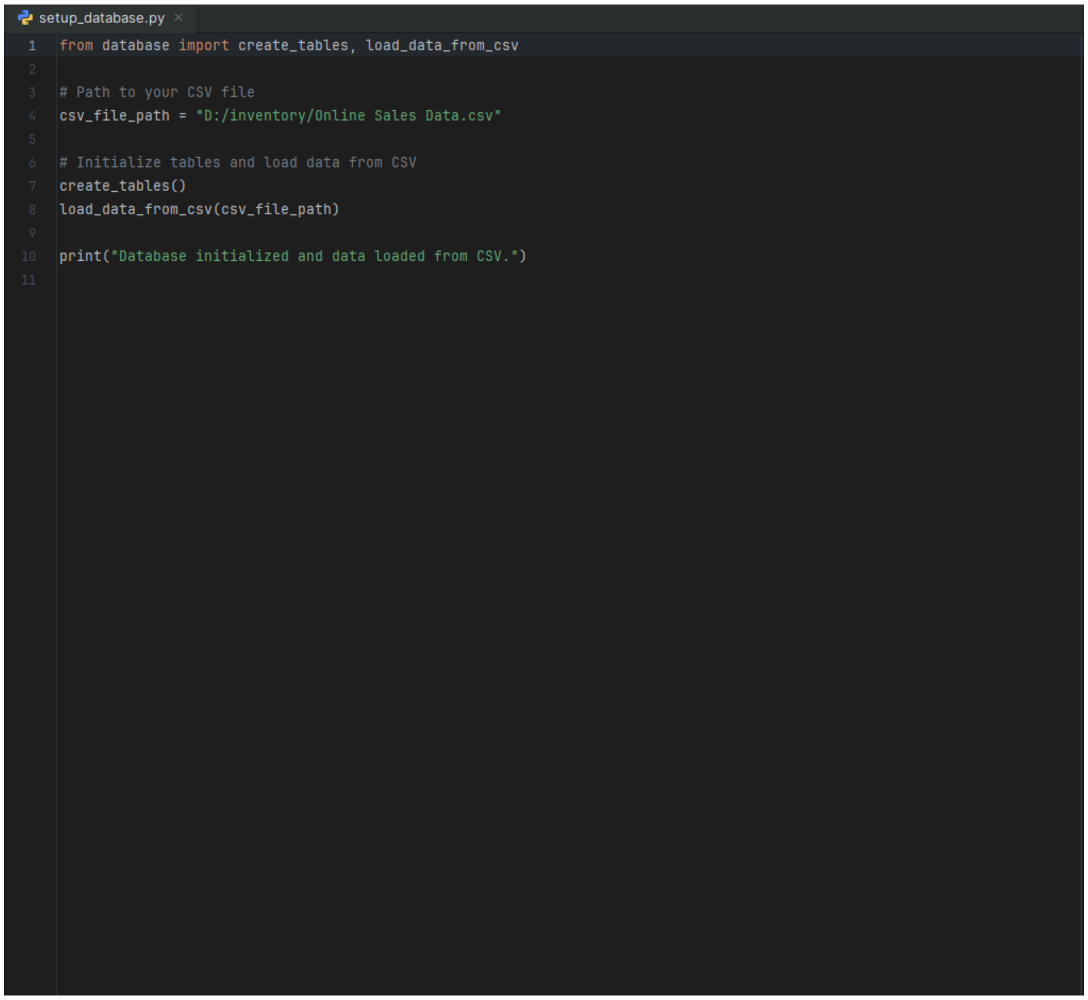


database.py

```python
    products_data = df[['Product ID', 'Product Name', 'Product Category', 'Initial_Quantity']].drop_duplicates()
    products_data.columns = ['product_id', 'product_name', 'product_category', 'initial_quantity']
    products_data.to_sql('Products', conn, if_exists='replace', index=False)

    # Add data to Sales table
    sales_data = df[['Product ID', 'Date', 'Units Sold', 'Unit Price', 'Total Revenue']]
    sales_data.columns = ['product_id', 'sale_date', 'units_sold', 'unit_price', 'total_revenue']
    sales_data.to_sql('Sales', conn, if_exists='replace', index=False)

    # Initialize Inventory table with initial quantities if needed
    inventory_data = products_data[['product_id', 'initial_quantity']].copy()
    inventory_data['inventory_date'] = pd.to_datetime('today').strftime('%m/%d/%Y')
    inventory_data.rename(columns={'initial_quantity': 'quantity'}, inplace=True)
    inventory_data = inventory_data[['product_id', 'inventory_date', 'quantity']]
    inventory_data.to_sql('Inventory', conn, if_exists='replace', index=False)

    # Add data to FullData table
    full_data = df.rename(columns={
        'Product ID': 'product_id',
        'Date': 'date',
        'Product Category': 'product_category',
        'Product Name': 'product_name',
        'Units Sold': 'units_sold',
        'Unit Price': 'unit_price',
        'Total Revenue': 'total_revenue',
        'Initial_Quantity': 'initial_quantity'
    })
    full_data.to_sql('FullData', conn, if_exists='replace', index=False)

    print("Data loaded into the database successfully.")
    conn.commit()
    conn.close()
```

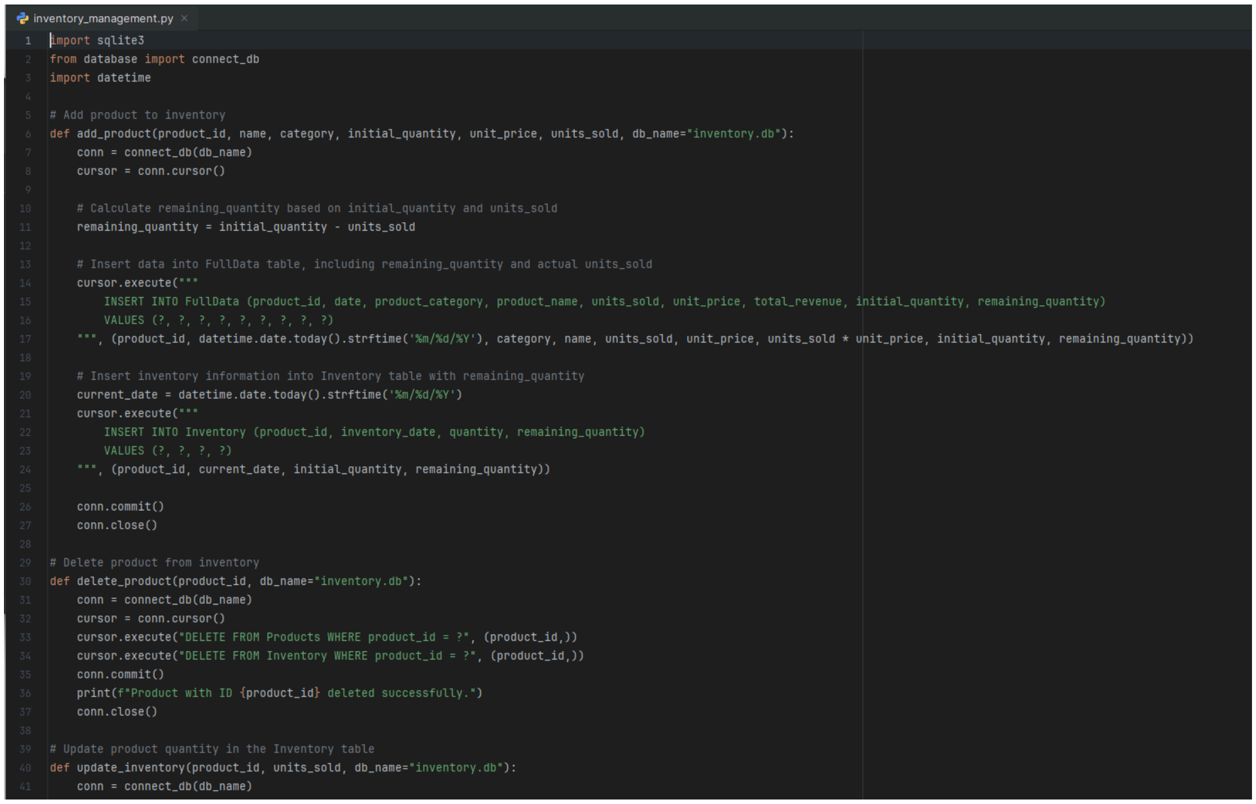
```python
# Only create tables when this module is imported or run
```

3. The **setup_database.py** file: creates tables like FullData, Products, Sales, Inventory and Users in the database if they don't already exist, and loads initial inventory data from the CSV file into the tables.



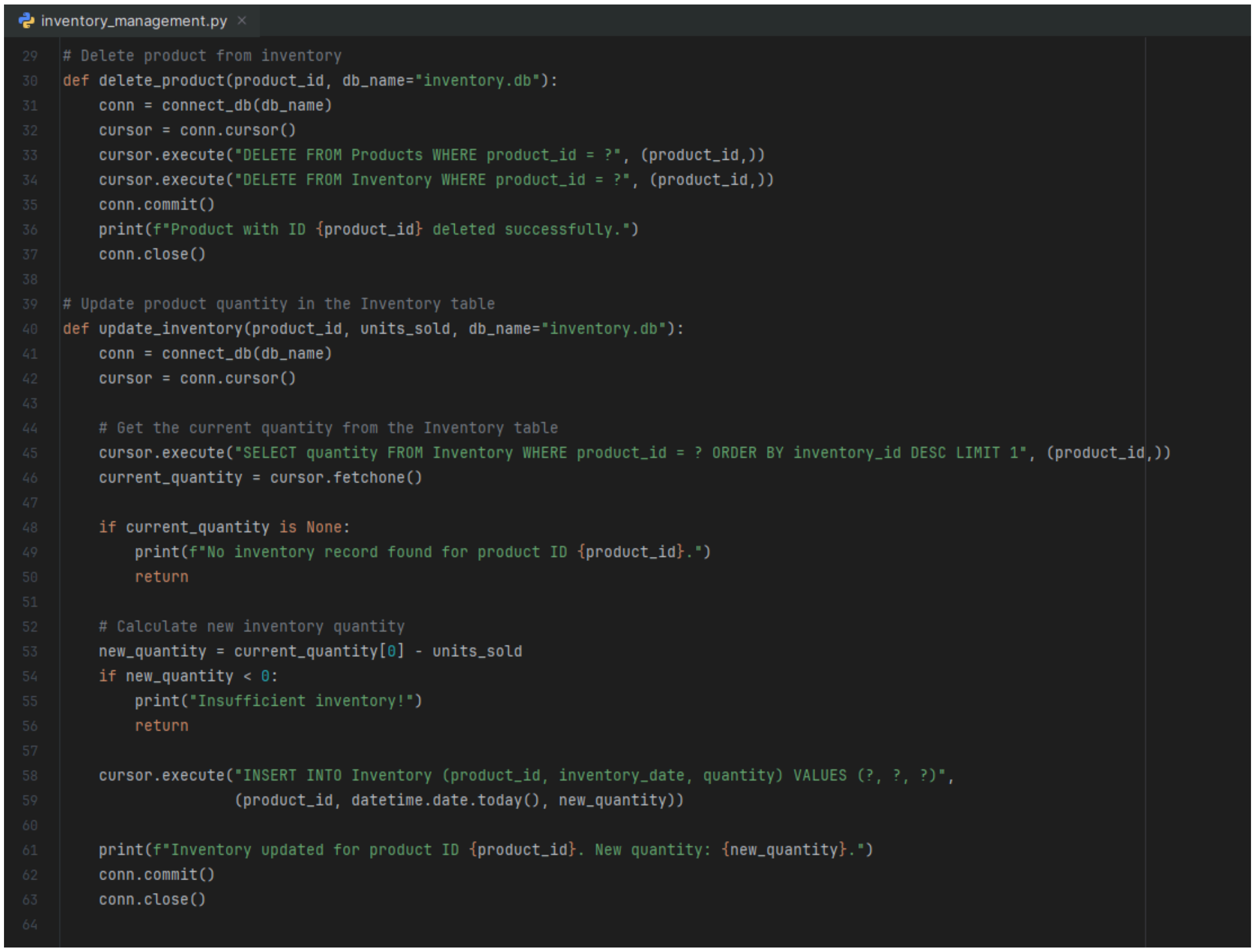
```python
from database import create_tables, load_data_from_csv

# Path to your CSV file
csv_file_path = "D:/inventory/Online Sales Data.csv"

# Initialize tables and load data from CSV
create_tables()
load_data_from_csv(csv_file_path)

print("Database initialized and data loaded from CSV.")
```

4. The **inventory_management.py** file: is responsible for managing the inventory related activities of the project. This file develops functions that allow us to rapidly perform adding, updating, and deletion of information about products in the database of the inventory. This file focuses on the processing of some product data for the purpose of having the correct accuracy of the product information with the current state of the warehouse.



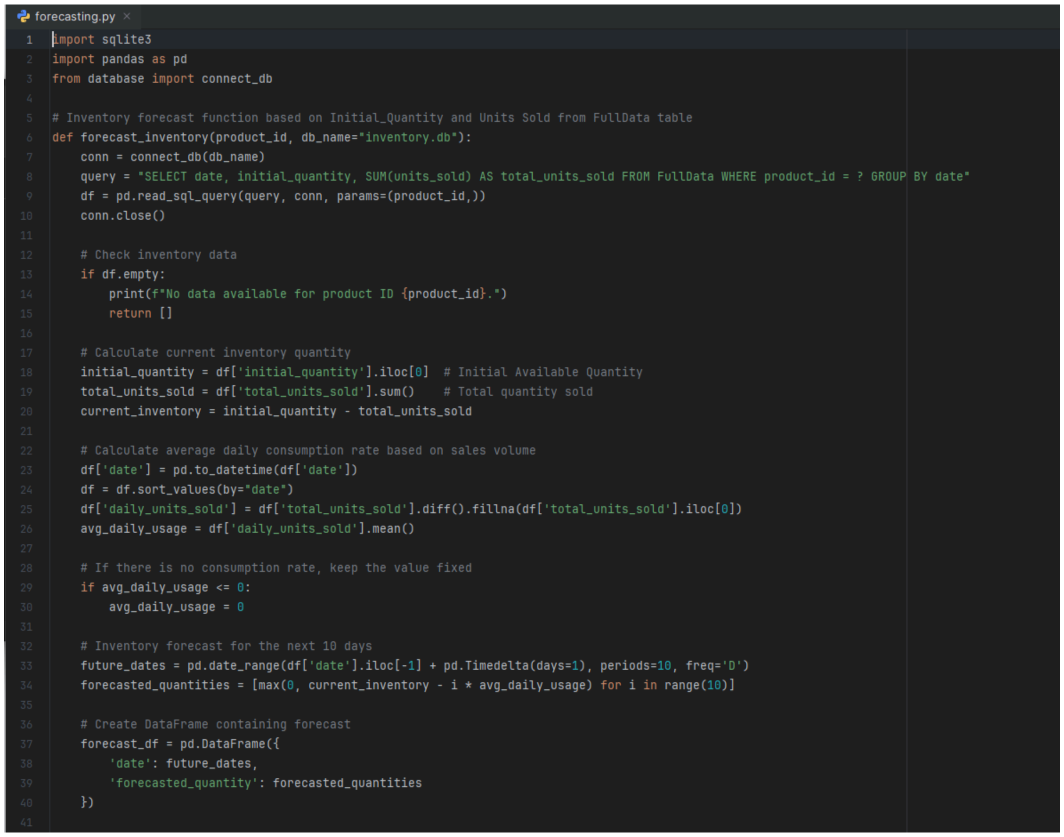
inventory_management.py

```python
import sqlite3
from database import connect_db
import datetime

# Add product to inventory
def add_product(product_id, name, category, initial_quantity, unit_price, units_sold, db_name="inventory.db"):
    conn = connect_db(db_name)
    cursor = conn.cursor()

    # Calculate remaining_quantity based on initial_quantity and units_sold
    remaining_quantity = initial_quantity - units_sold

    # Insert data into FullData table, including remaining_quantity and actual units_sold
    cursor.execute("""
        INSERT INTO FullData (product_id, date, product_category, product_name, units_sold, unit_price, total_revenue, initial_quantity, remaining_quantity)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
    """, (product_id, datetime.date.today().strftime('%m/%d/%Y'), category, name, units_sold, unit_price, units_sold * unit_price, initial_quantity, remaining_quantity))

    # Insert inventory information into Inventory table with remaining_quantity
    current_date = datetime.date.today().strftime('%m/%d/%Y')
    cursor.execute("""
        INSERT INTO Inventory (product_id, inventory_date, quantity, remaining_quantity)
        VALUES (?, ?, ?, ?)
    """, (product_id, current_date, initial_quantity, remaining_quantity))

    conn.commit()
    conn.close()

# Delete product from inventory
def delete_product(product_id, db_name="inventory.db"):
    conn = connect_db(db_name)
    cursor = conn.cursor()
    cursor.execute("DELETE FROM Products WHERE product_id = ?", (product_id,))
    cursor.execute("DELETE FROM Inventory WHERE product_id = ?", (product_id,))
    conn.commit()
    print(f"Product with ID {product_id} deleted successfully.")
    conn.close()

# Update product quantity in the Inventory table
def update_inventory(product_id, units_sold, db_name="inventory.db"):
    conn = connect_db(db_name)
```

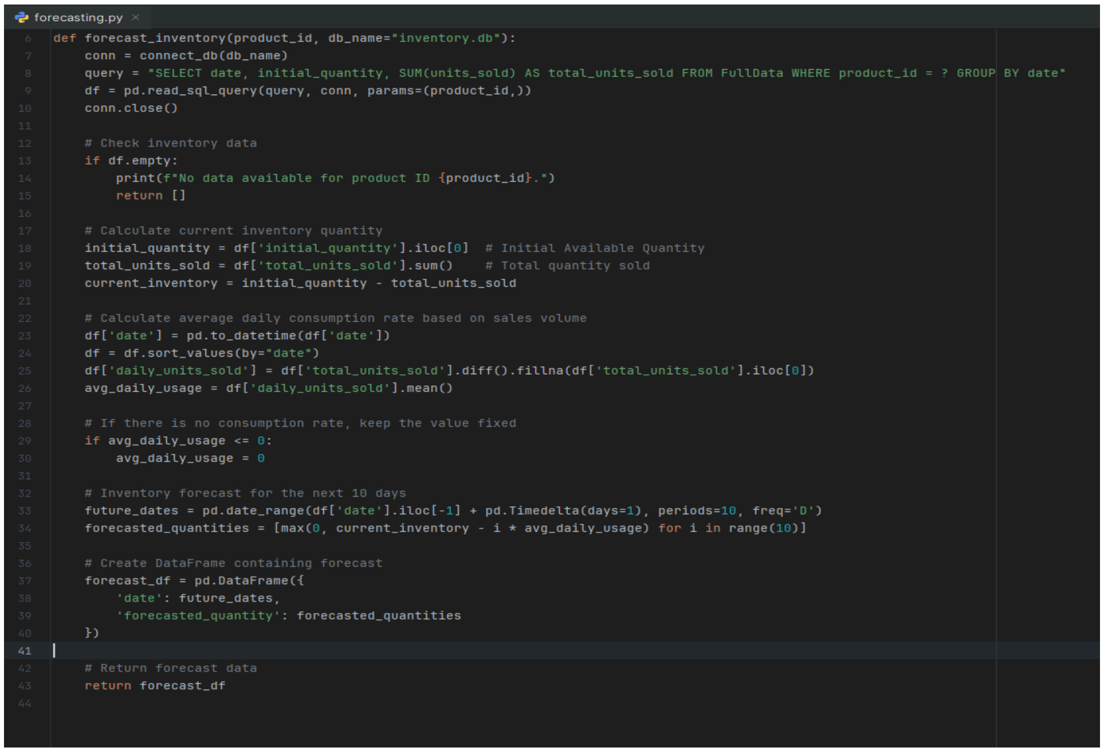

inventory_management.py

```python
# Delete product from inventory
def delete_product(product_id, db_name="inventory.db"):
    conn = connect_db(db_name)
    cursor = conn.cursor()
    cursor.execute("DELETE FROM Products WHERE product_id = ?", (product_id,))
    cursor.execute("DELETE FROM Inventory WHERE product_id = ?", (product_id,))
    conn.commit()
    print(f"Product with ID {product_id} deleted successfully.")
    conn.close()

# Update product quantity in the Inventory table
def update_inventory(product_id, units_sold, db_name="inventory.db"):
    conn = connect_db(db_name)
    cursor = conn.cursor()

    # Get the current quantity from the Inventory table
    cursor.execute("SELECT quantity FROM Inventory WHERE product_id = ? ORDER BY inventory_id DESC LIMIT 1", (product_id,))
    current_quantity = cursor.fetchone()

    if current_quantity is None:
        print(f"No inventory record found for product ID {product_id}.")
        return

    # Calculate new inventory quantity
    new_quantity = current_quantity[0] - units_sold
    if new_quantity < 0:
        print("Insufficient inventory!")
        return

    cursor.execute("INSERT INTO Inventory (product_id, inventory_date, quantity) VALUES (?, ?, ?)",
                   (product_id, datetime.date.today(), new_quantity))

    print(f"Inventory updated for product ID {product_id}. New quantity: {new_quantity}.")
    conn.commit()
    conn.close()
```

5. The **forecasting.py** file: This will contain functions related to the forecast of future inventory levels. It contains functions for the analysis of existing data for the estimation of demand and the quantity of goods needed in the forthcoming days to support businesses in creating plans for reasonable replenishment to avoid shortage or excess inventories within the warehouse.

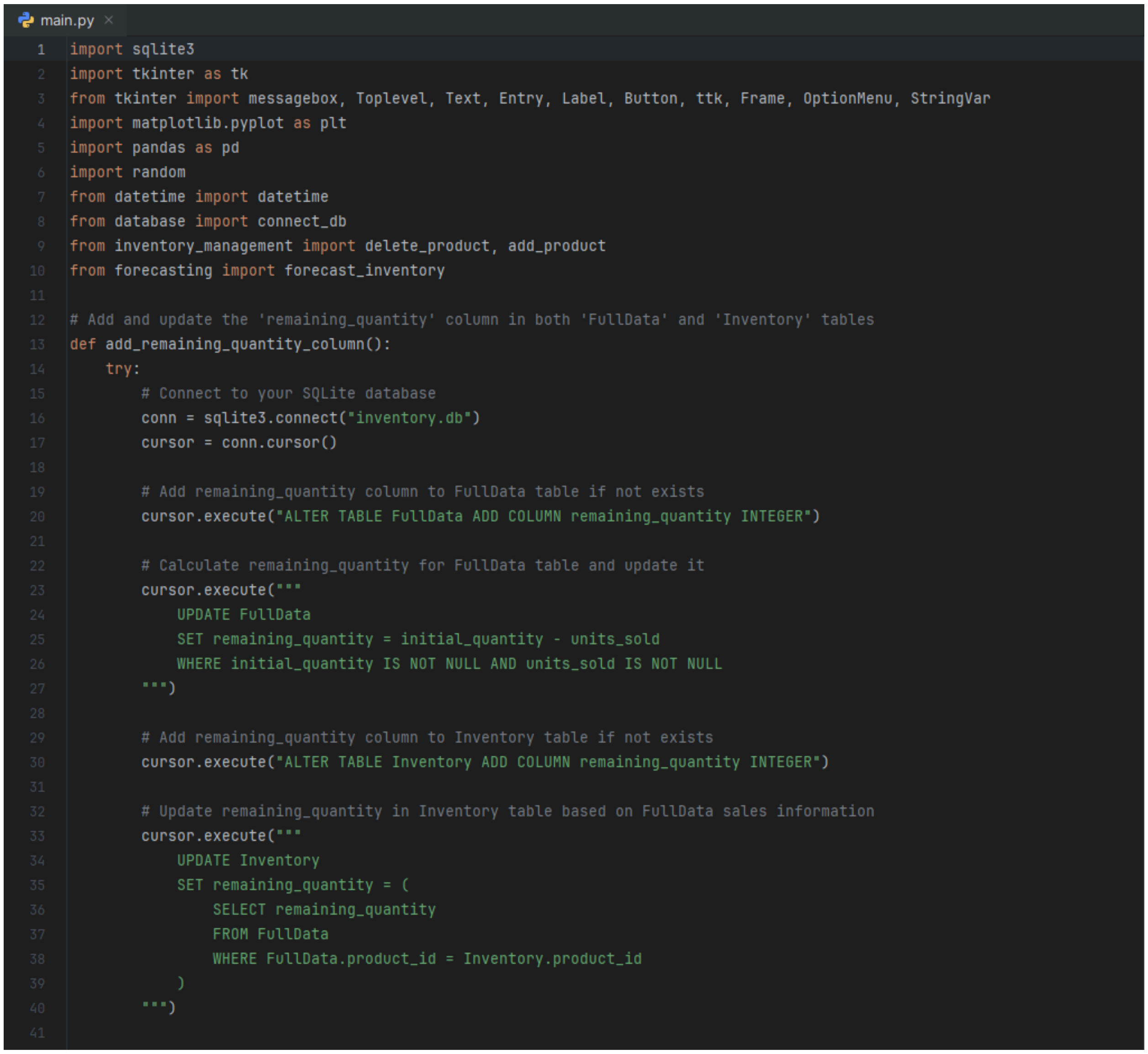

forecasting.py

```python
import sqlite3
import pandas as pd
from database import connect_db

# Inventory forecast function based on Initial_Quantity and Units Sold from FullData table
def forecast_inventory(product_id, db_name="inventory.db"):
    conn = connect_db(db_name)
    query = "SELECT date, initial_quantity, SUM(units_sold) AS total_units_sold FROM FullData WHERE product_id = ? GROUP BY date"
    df = pd.read_sql_query(query, conn, params=(product_id,))
    conn.close()

    # Check inventory data
    if df.empty:
        print(f"No data available for product ID {product_id}.")
        return []

    # Calculate current inventory quantity
    initial_quantity = df['initial_quantity'].iloc[0]  # Initial Available Quantity
    total_units_sold = df['total_units_sold'].sum()    # Total quantity sold
    current_inventory = initial_quantity - total_units_sold

    # Calculate average daily consumption rate based on sales volume
    df['date'] = pd.to_datetime(df['date'])
    df = df.sort_values(by="date")
    df['daily_units_sold'] = df['total_units_sold'].diff().fillna(df['total_units_sold'].iloc[0])
    avg_daily_usage = df['daily_units_sold'].mean()

    # If there is no consumption rate, keep the value fixed
    if avg_daily_usage <= 0:
        avg_daily_usage = 0

    # Inventory forecast for the next 10 days
    future_dates = pd.date_range(df['date'].iloc[-1] + pd.Timedelta(days=1), periods=10, freq='D')
    forecasted_quantities = [max(0, current_inventory - i * avg_daily_usage) for i in range(10)]

    # Create DataFrame containing forecast
    forecast_df = pd.DataFrame({
        'date': future_dates,
        'forecasted_quantity': forecasted_quantities
    })

```

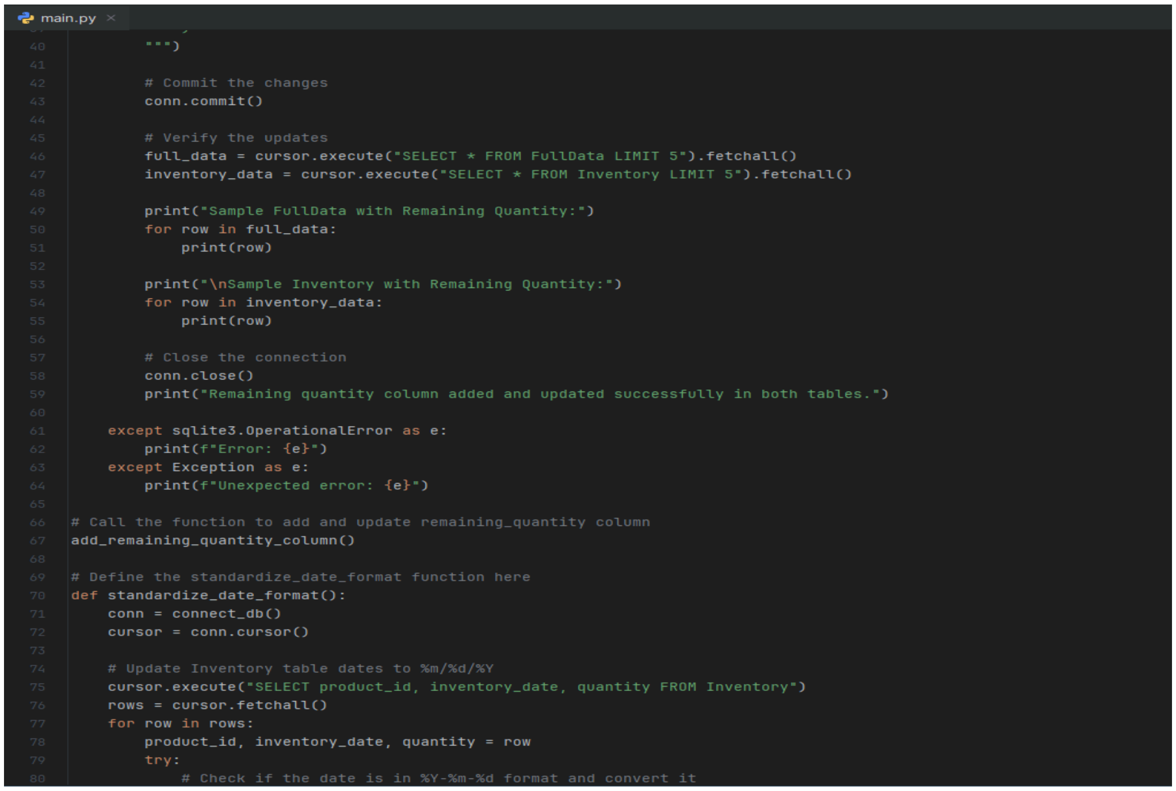

forecasting.py

```python
def forecast_inventory(product_id, db_name="inventory.db"):
    conn = connect_db(db_name)
    query = "SELECT date, initial_quantity, SUM(units_sold) AS total_units_sold FROM FullData WHERE product_id = ? GROUP BY date"
    df = pd.read_sql_query(query, conn, params=(product_id,))
    conn.close()

    # Check inventory data
    if df.empty:
        print(f"No data available for product ID {product_id}.")
        return []

    # Calculate current inventory quantity
    initial_quantity = df['initial_quantity'].iloc[0]  # Initial Available Quantity
    total_units_sold = df['total_units_sold'].sum()    # Total quantity sold
    current_inventory = initial_quantity - total_units_sold

    # Calculate average daily consumption rate based on sales volume
    df['date'] = pd.to_datetime(df['date'])
    df = df.sort_values(by="date")
    df['daily_units_sold'] = df['total_units_sold'].diff().fillna(df['total_units_sold'].iloc[0])
    avg_daily_usage = df['daily_units_sold'].mean()

    # If there is no consumption rate, keep the value fixed
    if avg_daily_usage <= 0:
        avg_daily_usage = 0

    # Inventory forecast for the next 10 days
    future_dates = pd.date_range(df['date'].iloc[-1] + pd.Timedelta(days=1), periods=10, freq='D')
    forecasted_quantities = [max(0, current_inventory - i * avg_daily_usage) for i in range(10)]

    # Create DataFrame containing forecast
    forecast_df = pd.DataFrame({
        'date': future_dates,
        'forecasted_quantity': forecasted_quantities
    })

    # Return forecast data
    return forecast_df

```

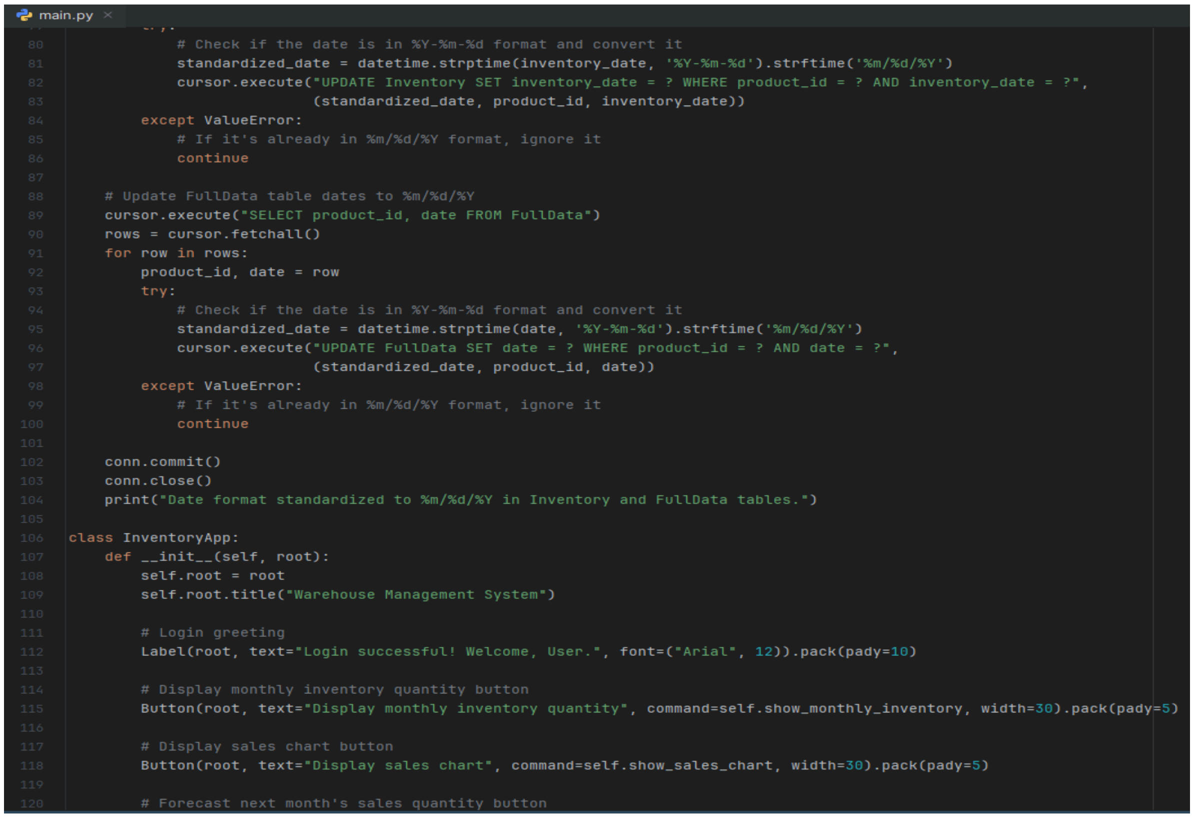
6. The **main.py** file: In the project is the main interface of the application. It serves to provide users with possibilities to use the functions such as adding, updating, deleting products,, it shows sales charts, forecasts the inventory level, and shows full data. It allows the user to directly manipulate and manage the inventory information and view necessary analytics.



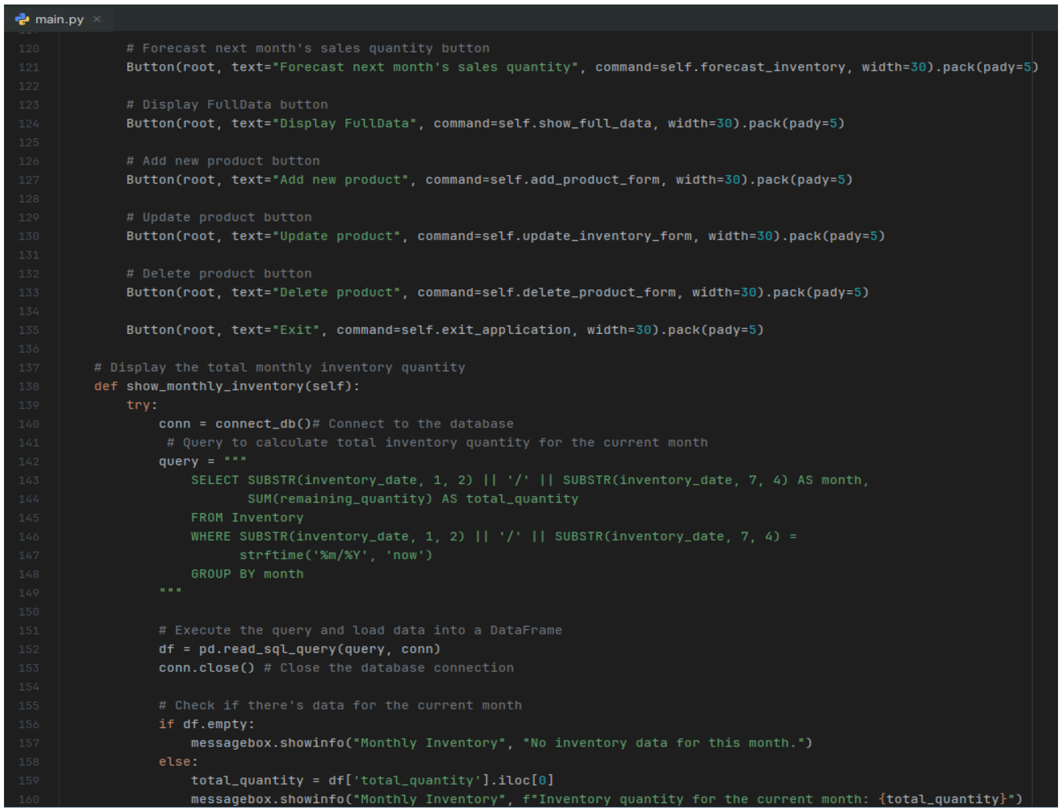
```python
import sqlite3
import tkinter as tk
from tkinter import messagebox, Toplevel, Text, Entry, Label, Button, ttk, Frame, OptionMenu, StringVar
import matplotlib.pyplot as plt
import pandas as pd
import random
from datetime import datetime
from database import connect_db
from inventory_management import delete_product, add_product
from forecasting import forecast_inventory

# Add and update the 'remaining_quantity' column in both 'FullData' and 'Inventory' tables
def add_remaining_quantity_column():
    try:
        # Connect to your SQLite database
        conn = sqlite3.connect("inventory.db")
        cursor = conn.cursor()

        # Add remaining_quantity column to FullData table if not exists
        cursor.execute("ALTER TABLE FullData ADD COLUMN remaining_quantity INTEGER")

        # Calculate remaining_quantity for FullData table and update it
        cursor.execute("""
            UPDATE FullData
            SET remaining_quantity = initial_quantity - units_sold
            WHERE initial_quantity IS NOT NULL AND units_sold IS NOT NULL
        """)

        # Add remaining_quantity column to Inventory table if not exists
        cursor.execute("ALTER TABLE Inventory ADD COLUMN remaining_quantity INTEGER")

        # Update remaining_quantity in Inventory table based on FullData sales information
        cursor.execute("""
            UPDATE Inventory
            SET remaining_quantity = (
                SELECT remaining_quantity
                FROM FullData
                WHERE FullData.product_id = Inventory.product_id
            )
        """)

```



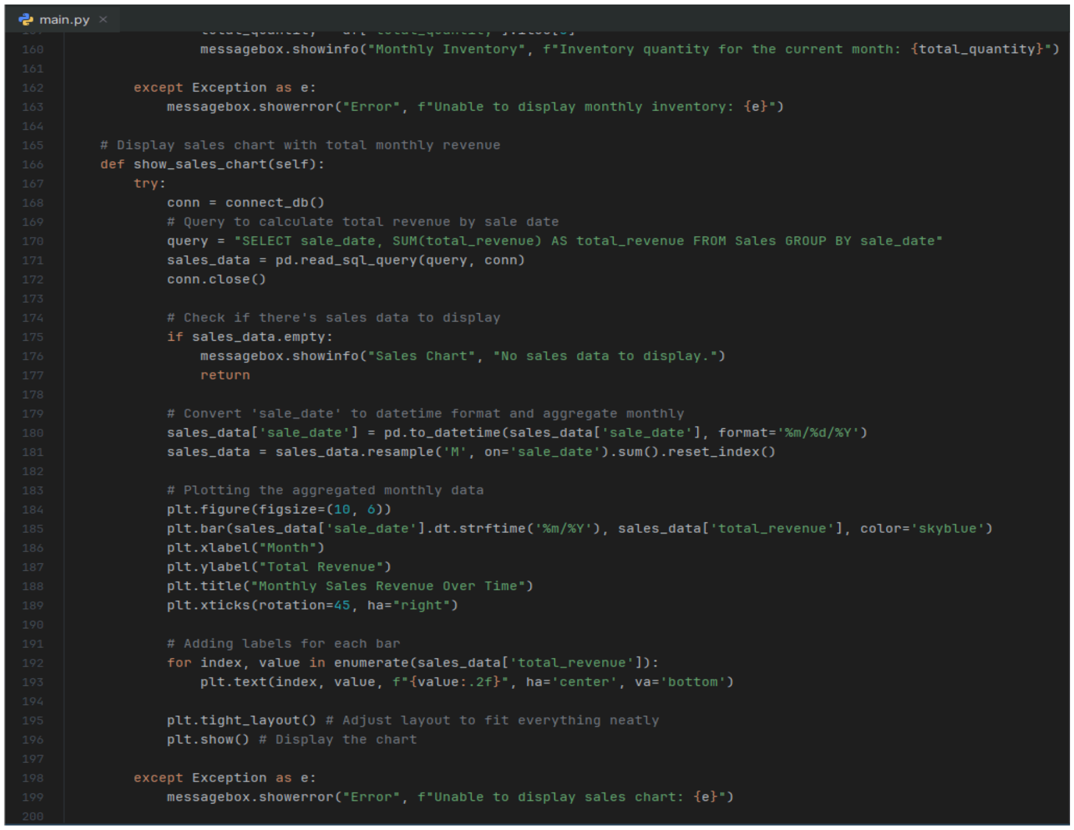
```python
        """)

        # Commit the changes
        conn.commit()

        # Verify the updates
        full_data = cursor.execute("SELECT * FROM FullData LIMIT 5").fetchall()
        inventory_data = cursor.execute("SELECT * FROM Inventory LIMIT 5").fetchall()

        print("Sample FullData with Remaining Quantity:")
        for row in full_data:
            print(row)

        print("\nSample Inventory with Remaining Quantity:")
        for row in inventory_data:
            print(row)

        # Close the connection
        conn.close()
        print("Remaining quantity column added and updated successfully in both tables.")

    except sqlite3.OperationalError as e:
        print(f"Error: {e}")
    except Exception as e:
        print(f"Unexpected error: {e}")

# Call the function to add and update remaining_quantity column
add_remaining_quantity_column()

# Define the standardize_date_format function here
def standardize_date_format():
    conn = connect_db()
    cursor = conn.cursor()

    # Update Inventory table dates to %m/%d/%Y
    cursor.execute("SELECT product_id, inventory_date, quantity FROM Inventory")
    rows = cursor.fetchall()
    for row in rows:
        product_id, inventory_date, quantity = row
        try:
            # Check if the date is in %Y-%m-%d format and convert it
```



```python
            # Check if the date is in %Y-%m-%d format and convert it
            standardized_date = datetime.strptime(inventory_date, '%Y-%m-%d').strftime('%m/%d/%Y')
            cursor.execute("UPDATE Inventory SET inventory_date = ? WHERE product_id = ? AND inventory_date = ?",
                           (standardized_date, product_id, inventory_date))
        except ValueError:
            # If it's already in %m/%d/%Y format, ignore it
            continue

    # Update FullData table dates to %m/%d/%Y
    cursor.execute("SELECT product_id, date FROM FullData")
    rows = cursor.fetchall()
    for row in rows:
        product_id, date = row
        try:
            # Check if the date is in %Y-%m-%d format and convert it
            standardized_date = datetime.strptime(date, '%Y-%m-%d').strftime('%m/%d/%Y')
            cursor.execute("UPDATE FullData SET date = ? WHERE product_id = ? AND date = ?",
                           (standardized_date, product_id, date))
        except ValueError:
            # If it's already in %m/%d/%Y format, ignore it
            continue

    conn.commit()
    conn.close()
    print("Date format standardized to %m/%d/%Y in Inventory and FullData tables.")

class InventoryApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Warehouse Management System")

        # Login greeting
        Label(root, text="Login successful! Welcome, User.", font=("Arial", 12)).pack(pady=10)

        # Display monthly inventory quantity button
        Button(root, text="Display monthly inventory quantity", command=self.show_monthly_inventory, width=30).pack(pady=5)

        # Display sales chart button
        Button(root, text="Display sales chart", command=self.show_sales_chart, width=30).pack(pady=5)

        # Forecast next month's sales quantity button
```

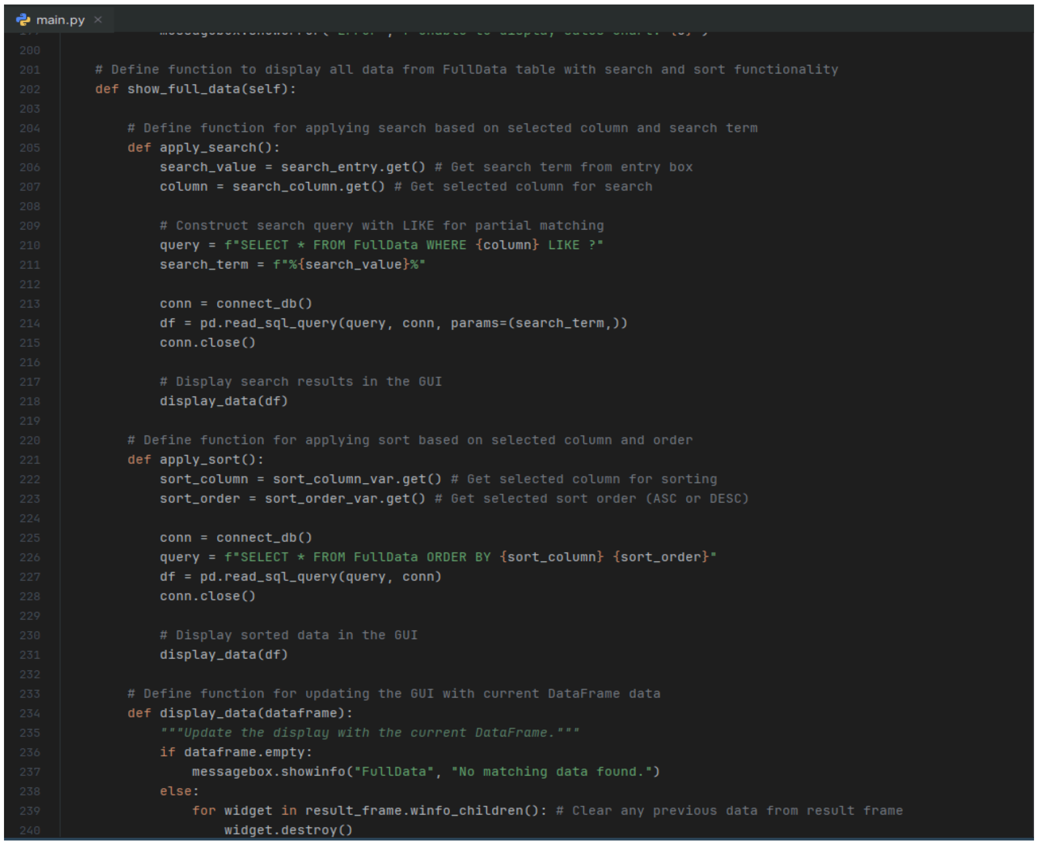


```python
        # Forecast next month's sales quantity button
        Button(root, text="Forecast next month's sales quantity", command=self.forecast_inventory, width=30).pack(pady=5)

        # Display FullData button
        Button(root, text="Display FullData", command=self.show_full_data, width=30).pack(pady=5)

        # Add new product button
        Button(root, text="Add new product", command=self.add_product_form, width=30).pack(pady=5)

        # Update product button
        Button(root, text="Update product", command=self.update_inventory_form, width=30).pack(pady=5)

        # Delete product button
        Button(root, text="Delete product", command=self.delete_product_form, width=30).pack(pady=5)

        Button(root, text="Exit", command=self.exit_application, width=30).pack(pady=5)

    # Display the total monthly inventory quantity
    def show_monthly_inventory(self):
        try:
            conn = connect_db()# Connect to the database
             # Query to calculate total inventory quantity for the current month
            query = """
                SELECT SUBSTR(inventory_date, 1, 2) || '/' || SUBSTR(inventory_date, 7, 4) AS month,
                       SUM(remaining_quantity) AS total_quantity
                FROM Inventory
                WHERE SUBSTR(inventory_date, 1, 2) || '/' || SUBSTR(inventory_date, 7, 4) =
                      strftime('%m/%Y', 'now')
                GROUP BY month
            """

            # Execute the query and load data into a DataFrame
            df = pd.read_sql_query(query, conn)
            conn.close() # Close the database connection

            # Check if there's data for the current month
            if df.empty:
                messagebox.showinfo("Monthly Inventory", "No inventory data for this month.")
            else:
                total_quantity = df['total_quantity'].iloc[0]
                messagebox.showinfo("Monthly Inventory", f"Inventory quantity for the current month: {total_quantity}")
```

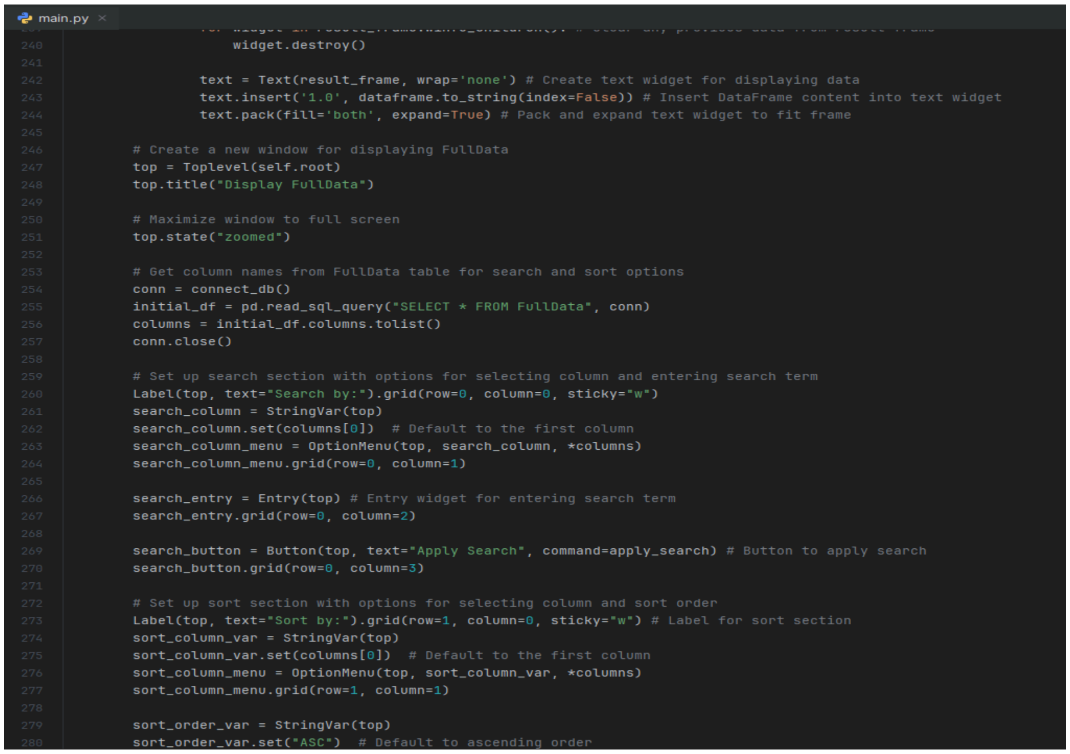


```python
                messagebox.showinfo("Monthly Inventory", f"Inventory quantity for the current month: {total_quantity}")
```

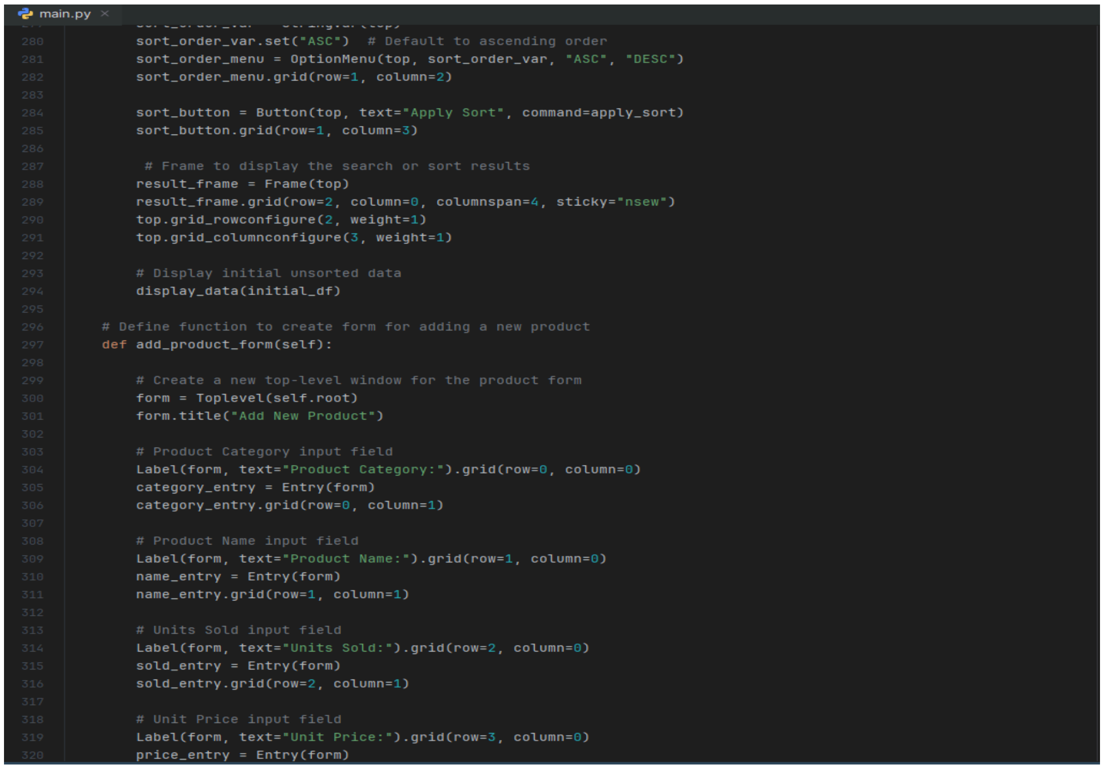
```python

        except Exception as e:
            messagebox.showerror("Error", f"Unable to display monthly inventory: {e}")

    # Display sales chart with total monthly revenue
    def show_sales_chart(self):
        try:
            conn = connect_db()
            # Query to calculate total revenue by sale date
            query = "SELECT sale_date, SUM(total_revenue) AS total_revenue FROM Sales GROUP BY sale_date"
            sales_data = pd.read_sql_query(query, conn)
            conn.close()

            # Check if there's sales data to display
            if sales_data.empty:
                messagebox.showinfo("Sales Chart", "No sales data to display.")
                return

            # Convert 'sale_date' to datetime format and aggregate monthly
            sales_data['sale_date'] = pd.to_datetime(sales_data['sale_date'], format='%m/%d/%Y')
            sales_data = sales_data.resample('M', on='sale_date').sum().reset_index()

            # Plotting the aggregated monthly data
            plt.figure(figsize=(10, 6))
            plt.bar(sales_data['sale_date'].dt.strftime('%m/%Y'), sales_data['total_revenue'], color='skyblue')
            plt.xlabel("Month")
            plt.ylabel("Total Revenue")
            plt.title("Monthly Sales Revenue Over Time")
            plt.xticks(rotation=45, ha="right")

            # Adding labels for each bar
            for index, value in enumerate(sales_data['total_revenue']):
                plt.text(index, value, f"{value:.2f}", ha='center', va='bottom')

            plt.tight_layout() # Adjust layout to fit everything neatly
            plt.show() # Display the chart

        except Exception as e:
            messagebox.showerror("Error", f"Unable to display sales chart: {e}")
```



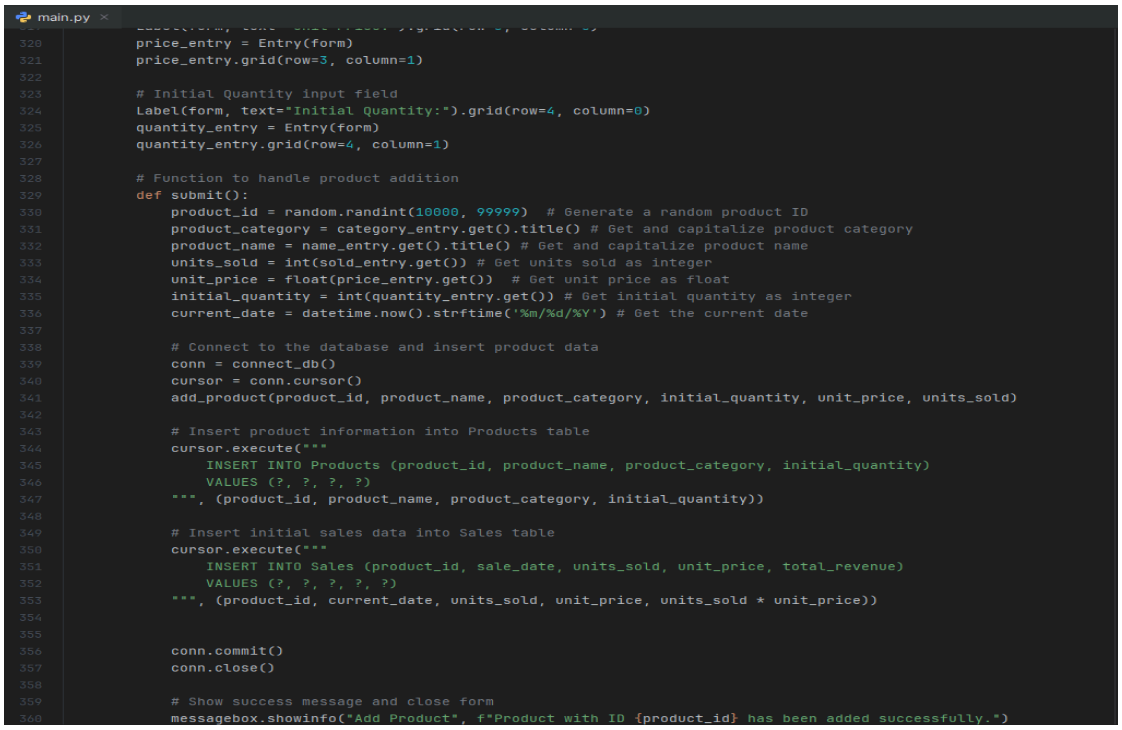
```python
    # Define function to display all data from FullData table with search and sort functionality
    def show_full_data(self):

        # Define function for applying search based on selected column and search term
        def apply_search():
            search_value = search_entry.get() # Get search term from entry box
            column = search_column.get() # Get selected column for search

            # Construct search query with LIKE for partial matching
            query = f"SELECT * FROM FullData WHERE {column} LIKE ?"
            search_term = f"%{search_value}%"

            conn = connect_db()
            df = pd.read_sql_query(query, conn, params=(search_term,))
            conn.close()

            # Display search results in the GUI
            display_data(df)

        # Define function for applying sort based on selected column and order
        def apply_sort():
            sort_column = sort_column_var.get() # Get selected column for sorting
            sort_order = sort_order_var.get() # Get selected sort order (ASC or DESC)

            conn = connect_db()
            query = f"SELECT * FROM FullData ORDER BY {sort_column} {sort_order}"
            df = pd.read_sql_query(query, conn)
            conn.close()

            # Display sorted data in the GUI
            display_data(df)

        # Define function for updating the GUI with current DataFrame data
        def display_data(dataframe):
            """Update the display with the current DataFrame."""
            if dataframe.empty:
                messagebox.showinfo("FullData", "No matching data found.")
            else:
                for widget in result_frame.winfo_children(): # Clear any previous data from result frame
                    widget.destroy()
```



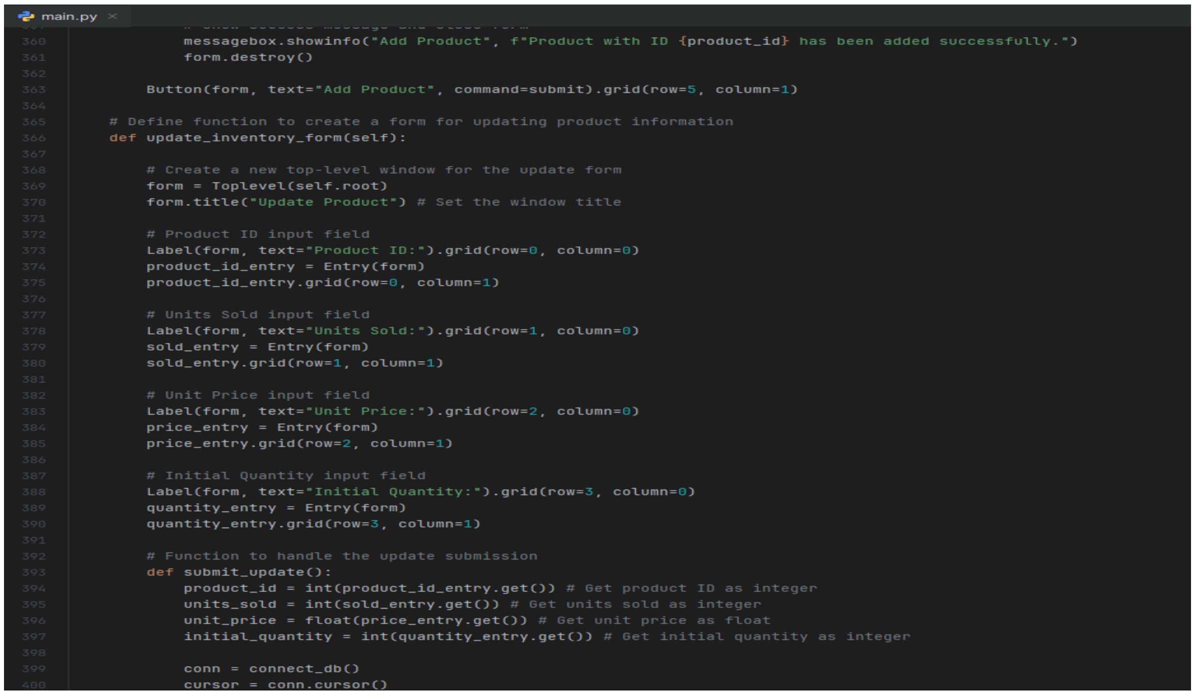
```python
                    widget.destroy()

                text = Text(result_frame, wrap='none') # Create text widget for displaying data
                text.insert('1.0', dataframe.to_string(index=False)) # Insert DataFrame content into text widget
                text.pack(fill='both', expand=True) # Pack and expand text widget to fit frame

        # Create a new window for displaying FullData
        top = Toplevel(self.root)
        top.title("Display FullData")

        # Maximize window to full screen
        top.state("zoomed")

        # Get column names from FullData table for search and sort options
        conn = connect_db()
        initial_df = pd.read_sql_query("SELECT * FROM FullData", conn)
        columns = initial_df.columns.tolist()
        conn.close()

        # Set up search section with options for selecting column and entering search term
        Label(top, text="Search by:").grid(row=0, column=0, sticky="w")
        search_column = StringVar(top)
        search_column.set(columns[0])  # Default to the first column
        search_column_menu = OptionMenu(top, search_column, *columns)
        search_column_menu.grid(row=0, column=1)

        search_entry = Entry(top) # Entry widget for entering search term
        search_entry.grid(row=0, column=2)

        search_button = Button(top, text="Apply Search", command=apply_search) # Button to apply search
        search_button.grid(row=0, column=3)

        # Set up sort section with options for selecting column and sort order
        Label(top, text="Sort by:").grid(row=1, column=0, sticky="w") # Label for sort section
        sort_column_var = StringVar(top)
        sort_column_var.set(columns[0])  # Default to the first column
        sort_column_menu = OptionMenu(top, sort_column_var, *columns)
        sort_column_menu.grid(row=1, column=1)

        sort_order_var = StringVar(top)
        sort_order_var.set("ASC")  # Default to ascending order
```



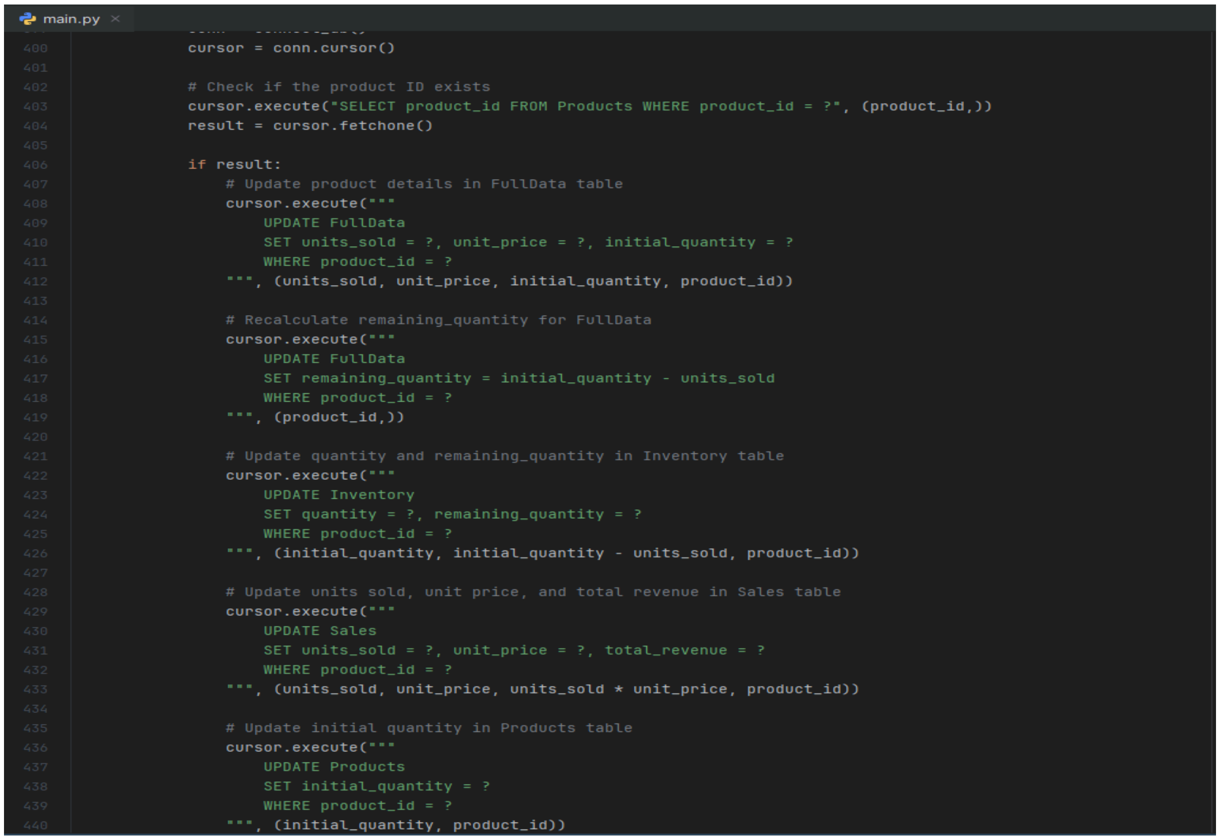
```python
        sort_order_var.set("ASC")  # Default to ascending order
        sort_order_menu = OptionMenu(top, sort_order_var, "ASC", "DESC")
        sort_order_menu.grid(row=1, column=2)

        sort_button = Button(top, text="Apply Sort", command=apply_sort)
        sort_button.grid(row=1, column=3)

         # Frame to display the search or sort results
        result_frame = Frame(top)
        result_frame.grid(row=2, column=0, columnspan=4, sticky="nsew")
        top.grid_rowconfigure(2, weight=1)
        top.grid_columnconfigure(3, weight=1)

        # Display initial unsorted data
        display_data(initial_df)

    # Define function to create form for adding a new product
    def add_product_form(self):

        # Create a new top-level window for the product form
        form = Toplevel(self.root)
        form.title("Add New Product")

        # Product Category input field
        Label(form, text="Product Category:").grid(row=0, column=0)
        category_entry = Entry(form)
        category_entry.grid(row=0, column=1)

        # Product Name input field
        Label(form, text="Product Name:").grid(row=1, column=0)
        name_entry = Entry(form)
        name_entry.grid(row=1, column=1)

        # Units Sold input field
        Label(form, text="Units Sold:").grid(row=2, column=0)
        sold_entry = Entry(form)
        sold_entry.grid(row=2, column=1)

        # Unit Price input field
        Label(form, text="Unit Price:").grid(row=3, column=0)
        price_entry = Entry(form)
```



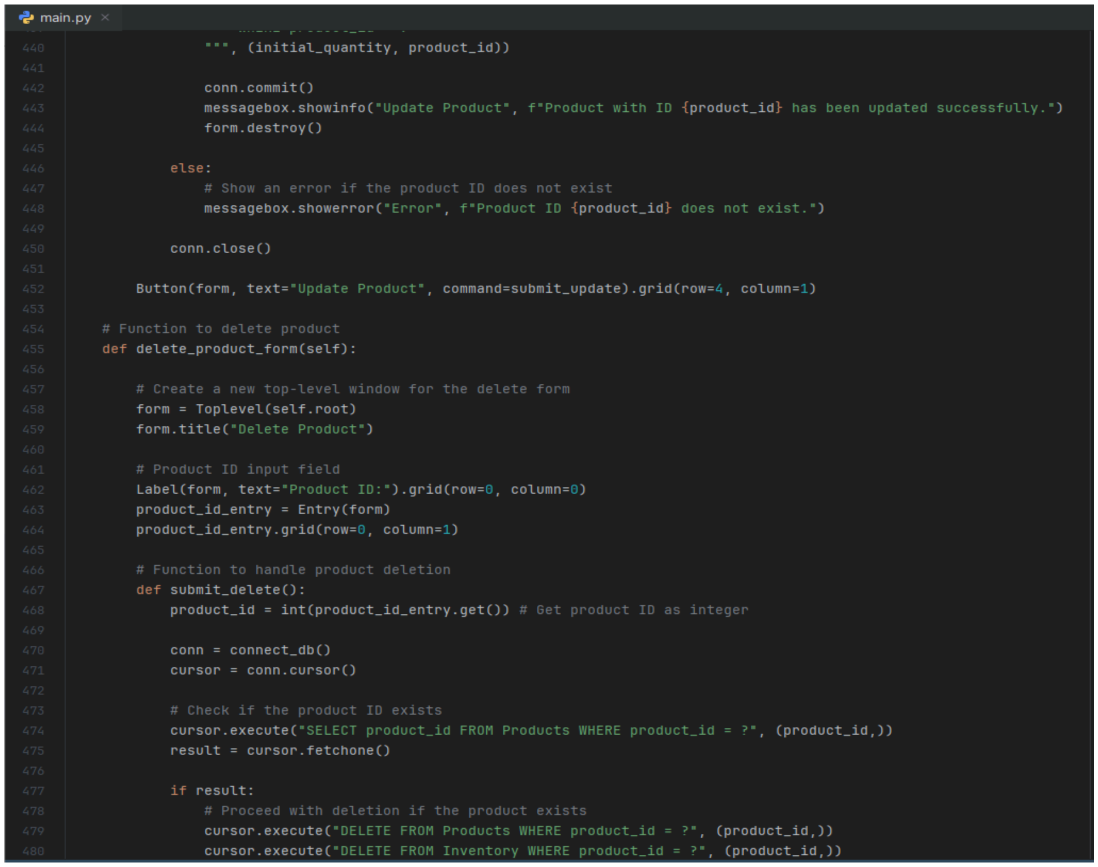
```python
        price_entry = Entry(form)
        price_entry.grid(row=3, column=1)

        # Initial Quantity input field
        Label(form, text="Initial Quantity:").grid(row=4, column=0)
        quantity_entry = Entry(form)
        quantity_entry.grid(row=4, column=1)

        # Function to handle product addition
        def submit():
            product_id = random.randint(10000, 99999)  # Generate a random product ID
            product_category = category_entry.get().title() # Get and capitalize product category
            product_name = name_entry.get().title() # Get and capitalize product name
            units_sold = int(sold_entry.get()) # Get units sold as integer
            unit_price = float(price_entry.get())  # Get unit price as float
            initial_quantity = int(quantity_entry.get()) # Get initial quantity as integer
            current_date = datetime.now().strftime('%m/%d/%Y') # Get the current date

            # Connect to the database and insert product data
            conn = connect_db()
            cursor = conn.cursor()
            add_product(product_id, product_name, product_category, initial_quantity, unit_price, units_sold)

            # Insert product information into Products table
            cursor.execute("""
                INSERT INTO Products (product_id, product_name, product_category, initial_quantity)
                VALUES (?, ?, ?, ?)
            """, (product_id, product_name, product_category, initial_quantity))

            # Insert initial sales data into Sales table
            cursor.execute("""
                INSERT INTO Sales (product_id, sale_date, units_sold, unit_price, total_revenue)
                VALUES (?, ?, ?, ?, ?)
            """, (product_id, current_date, units_sold, unit_price, units_sold * unit_price))


            conn.commit()
            conn.close()

            # Show success message and close form
            messagebox.showinfo("Add Product", f"Product with ID {product_id} has been added successfully.")
```



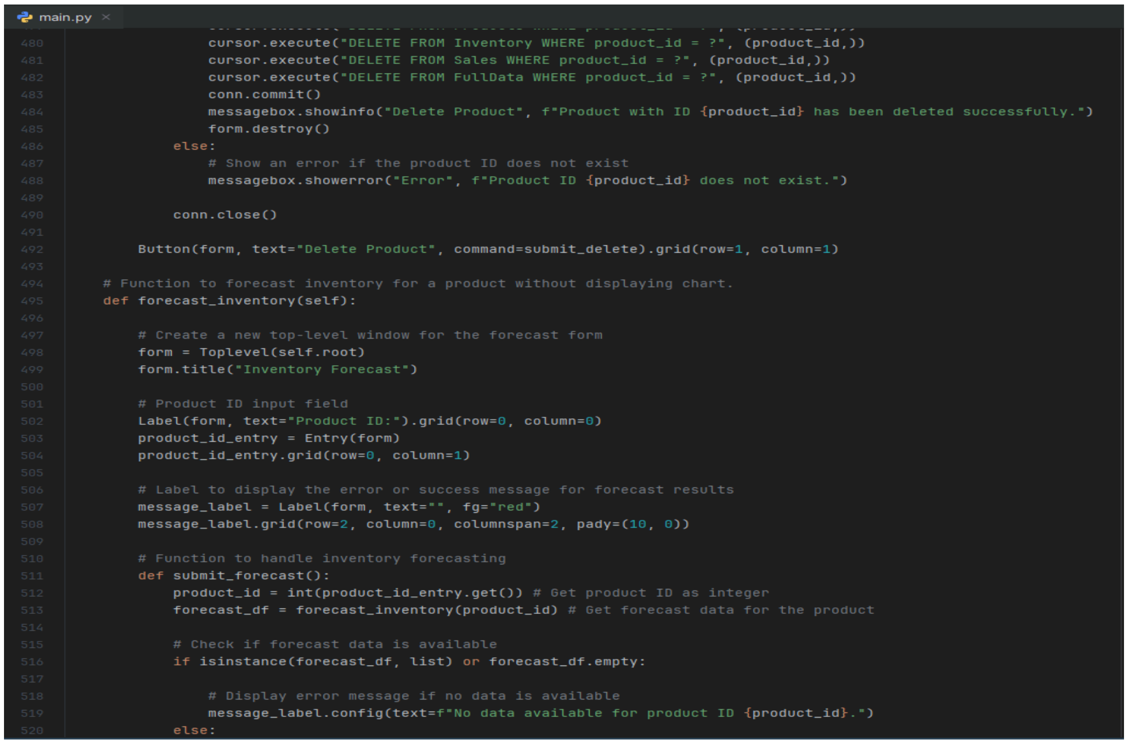
```
            messagebox.showinfo("Add Product", f"Product with ID {product_id} has been added successfully.")
            form.destroy()

        Button(form, text="Add Product", command=submit).grid(row=5, column=1)

    # Define function to create a form for updating product information
    def update_inventory_form(self):

        # Create a new top-level window for the update form
        form = Toplevel(self.root)
        form.title("Update Product") # Set the window title

        # Product ID input field
        Label(form, text="Product ID:").grid(row=0, column=0)
        product_id_entry = Entry(form)
        product_id_entry.grid(row=0, column=1)

        # Units Sold input field
        Label(form, text="Units Sold:").grid(row=1, column=0)
        sold_entry = Entry(form)
        sold_entry.grid(row=1, column=1)

        # Unit Price input field
        Label(form, text="Unit Price:").grid(row=2, column=0)
        price_entry = Entry(form)
        price_entry.grid(row=2, column=1)

        # Initial Quantity input field
        Label(form, text="Initial Quantity:").grid(row=3, column=0)
        quantity_entry = Entry(form)
        quantity_entry.grid(row=3, column=1)

        # Function to handle the update submission
        def submit_update():
            product_id = int(product_id_entry.get()) # Get product ID as integer
            units_sold = int(sold_entry.get()) # Get units sold as integer
            unit_price = float(price_entry.get()) # Get unit price as float
            initial_quantity = int(quantity_entry.get()) # Get initial quantity as integer

            conn = connect_db()
            cursor = conn.cursor()
```



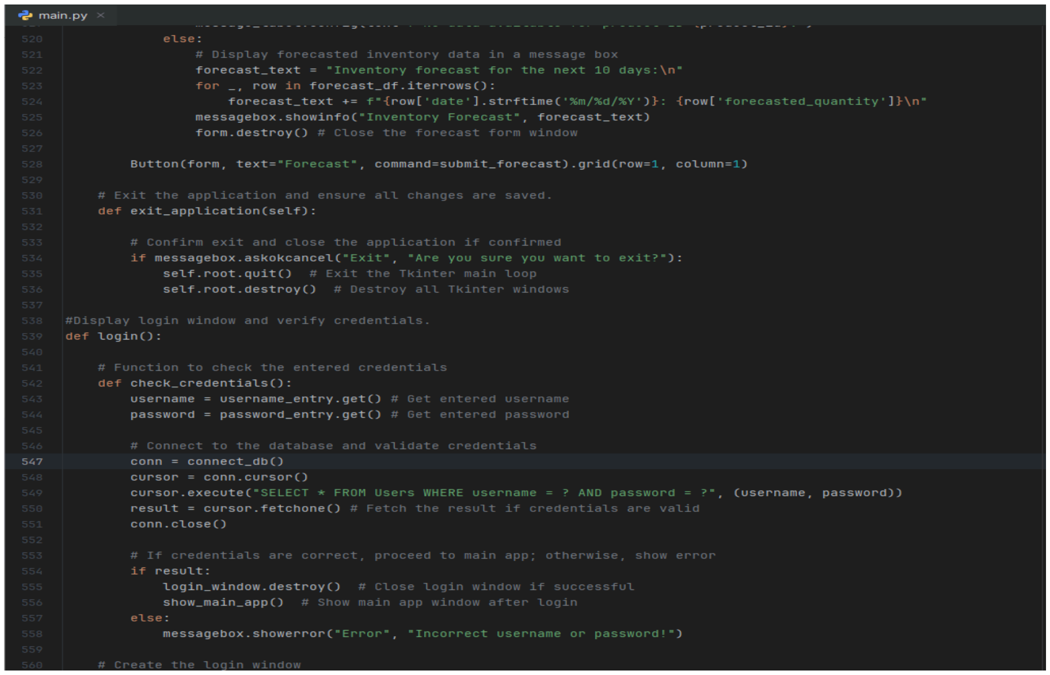
```
            cursor = conn.cursor()

            # Check if the product ID exists
            cursor.execute("SELECT product_id FROM Products WHERE product_id = ?", (product_id,))
            result = cursor.fetchone()

            if result:
                # Update product details in FullData table
                cursor.execute("""
                    UPDATE FullData
                    SET units_sold = ?, unit_price = ?, initial_quantity = ?
                    WHERE product_id = ?
                """, (units_sold, unit_price, initial_quantity, product_id))

                # Recalculate remaining_quantity for FullData
                cursor.execute("""
                    UPDATE FullData
                    SET remaining_quantity = initial_quantity - units_sold
                    WHERE product_id = ?
                """, (product_id,))

                # Update quantity and remaining_quantity in Inventory table
                cursor.execute("""
                    UPDATE Inventory
                    SET quantity = ?, remaining_quantity = ?
                    WHERE product_id = ?
                """, (initial_quantity, initial_quantity - units_sold, product_id))

                # Update units sold, unit price, and total revenue in Sales table
                cursor.execute("""
                    UPDATE Sales
                    SET units_sold = ?, unit_price = ?, total_revenue = ?
                    WHERE product_id = ?
                """, (units_sold, unit_price, units_sold * unit_price, product_id))

                # Update initial quantity in Products table
                cursor.execute("""
                    UPDATE Products
                    SET initial_quantity = ?
                    WHERE product_id = ?
                """, (initial_quantity, product_id))
```



```python
                """, (initial_quantity, product_id))

                conn.commit()
                messagebox.showinfo("Update Product", f"Product with ID {product_id} has been updated successfully.")
                form.destroy()

            else:
                # Show an error if the product ID does not exist
                messagebox.showerror("Error", f"Product ID {product_id} does not exist.")

            conn.close()

        Button(form, text="Update Product", command=submit_update).grid(row=4, column=1)

    # Function to delete product
    def delete_product_form(self):

        # Create a new top-level window for the delete form
        form = Toplevel(self.root)
        form.title("Delete Product")

        # Product ID input field
        Label(form, text="Product ID:").grid(row=0, column=0)
        product_id_entry = Entry(form)
        product_id_entry.grid(row=0, column=1)

        # Function to handle product deletion
        def submit_delete():
            product_id = int(product_id_entry.get()) # Get product ID as integer

            conn = connect_db()
            cursor = conn.cursor()

            # Check if the product ID exists
            cursor.execute("SELECT product_id FROM Products WHERE product_id = ?", (product_id,))
            result = cursor.fetchone()

            if result:
                # Proceed with deletion if the product exists
                cursor.execute("DELETE FROM Products WHERE product_id = ?", (product_id,))
                cursor.execute("DELETE FROM Inventory WHERE product_id = ?", (product_id,))
```

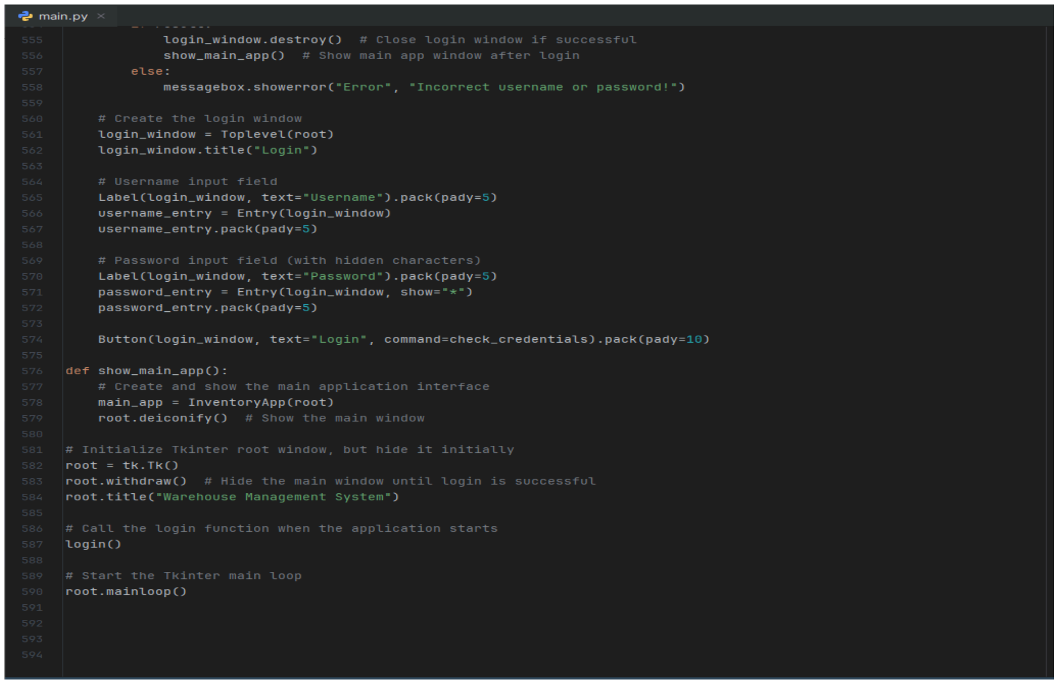


```python
                cursor.execute("DELETE FROM Inventory WHERE product_id = ?", (product_id,))
                cursor.execute("DELETE FROM Sales WHERE product_id = ?", (product_id,))
                cursor.execute("DELETE FROM FullData WHERE product_id = ?", (product_id,))
                conn.commit()
                messagebox.showinfo("Delete Product", f"Product with ID {product_id} has been deleted successfully.")
                form.destroy()
            else:
                # Show an error if the product ID does not exist
                messagebox.showerror("Error", f"Product ID {product_id} does not exist.")

            conn.close()

        Button(form, text="Delete Product", command=submit_delete).grid(row=1, column=1)

    # Function to forecast inventory for a product without displaying chart.
    def forecast_inventory(self):

        # Create a new top-level window for the forecast form
        form = Toplevel(self.root)
        form.title("Inventory Forecast")

        # Product ID input field
        Label(form, text="Product ID:").grid(row=0, column=0)
        product_id_entry = Entry(form)
        product_id_entry.grid(row=0, column=1)

        # Label to display the error or success message for forecast results
        message_label = Label(form, text="", fg="red")
        message_label.grid(row=2, column=0, columnspan=2, pady=(10, 0))

        # Function to handle inventory forecasting
        def submit_forecast():
            product_id = int(product_id_entry.get()) # Get product ID as integer
            forecast_df = forecast_inventory(product_id) # Get forecast data for the product

            # Check if forecast data is available
            if isinstance(forecast_df, list) or forecast_df.empty:

                # Display error message if no data is available
                message_label.config(text=f"No data available for product ID {product_id}.")
            else:
```



main.py

```python
            else:
                # Display forecasted inventory data in a message box
                forecast_text = "Inventory forecast for the next 10 days:\n"
                for _, row in forecast_df.iterrows():
                    forecast_text += f"{row['date'].strftime('%m/%d/%Y')}: {row['forecasted_quantity']}\n"
                messagebox.showinfo("Inventory Forecast", forecast_text)
                form.destroy() # Close the forecast form window

        Button(form, text="Forecast", command=submit_forecast).grid(row=1, column=1)

    # Exit the application and ensure all changes are saved.
    def exit_application(self):

        # Confirm exit and close the application if confirmed
        if messagebox.askokcancel("Exit", "Are you sure you want to exit?"):
            self.root.quit()  # Exit the Tkinter main loop
            self.root.destroy()  # Destroy all Tkinter windows

#Display login window and verify credentials.
def login():

    # Function to check the entered credentials
    def check_credentials():
        username = username_entry.get() # Get entered username
        password = password_entry.get() # Get entered password

        # Connect to the database and validate credentials
        conn = connect_db()
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM Users WHERE username = ? AND password = ?", (username, password))
        result = cursor.fetchone() # Fetch the result if credentials are valid
        conn.close()

        # If credentials are correct, proceed to main app; otherwise, show error
        if result:
            login_window.destroy()  # Close login window if successful
            show_main_app()  # Show main app window after login
        else:
            messagebox.showerror("Error", "Incorrect username or password!")

    # Create the login window
```



main.py

```python
            login_window.destroy()  # Close login window if successful
            show_main_app()  # Show main app window after login
        else:
            messagebox.showerror("Error", "Incorrect username or password!")

    # Create the login window
    login_window = Toplevel(root)
    login_window.title("Login")

    # Username input field
    Label(login_window, text="Username").pack(pady=5)
    username_entry = Entry(login_window)
    username_entry.pack(pady=5)

    # Password input field (with hidden characters)
    Label(login_window, text="Password").pack(pady=5)
    password_entry = Entry(login_window, show="*")
    password_entry.pack(pady=5)

    Button(login_window, text="Login", command=check_credentials).pack(pady=10)

def show_main_app():
    # Create and show the main application interface
    main_app = InventoryApp(root)
    root.deiconify()  # Show the main window

# Initialize Tkinter root window, but hide it initially
root = tk.Tk()
root.withdraw()  # Hide the main window until login is successful
root.title("Warehouse Management System")

# Call the login function when the application starts
login()

# Start the Tkinter main loop
root.mainloop()
```

Database In SQLite

The original database “inventory.db” file after loading data from the “Online Sales Data.csv file”. It will have tables such as FullData, Inventory, Products, Sales, Users.

1. **FullData tables**:

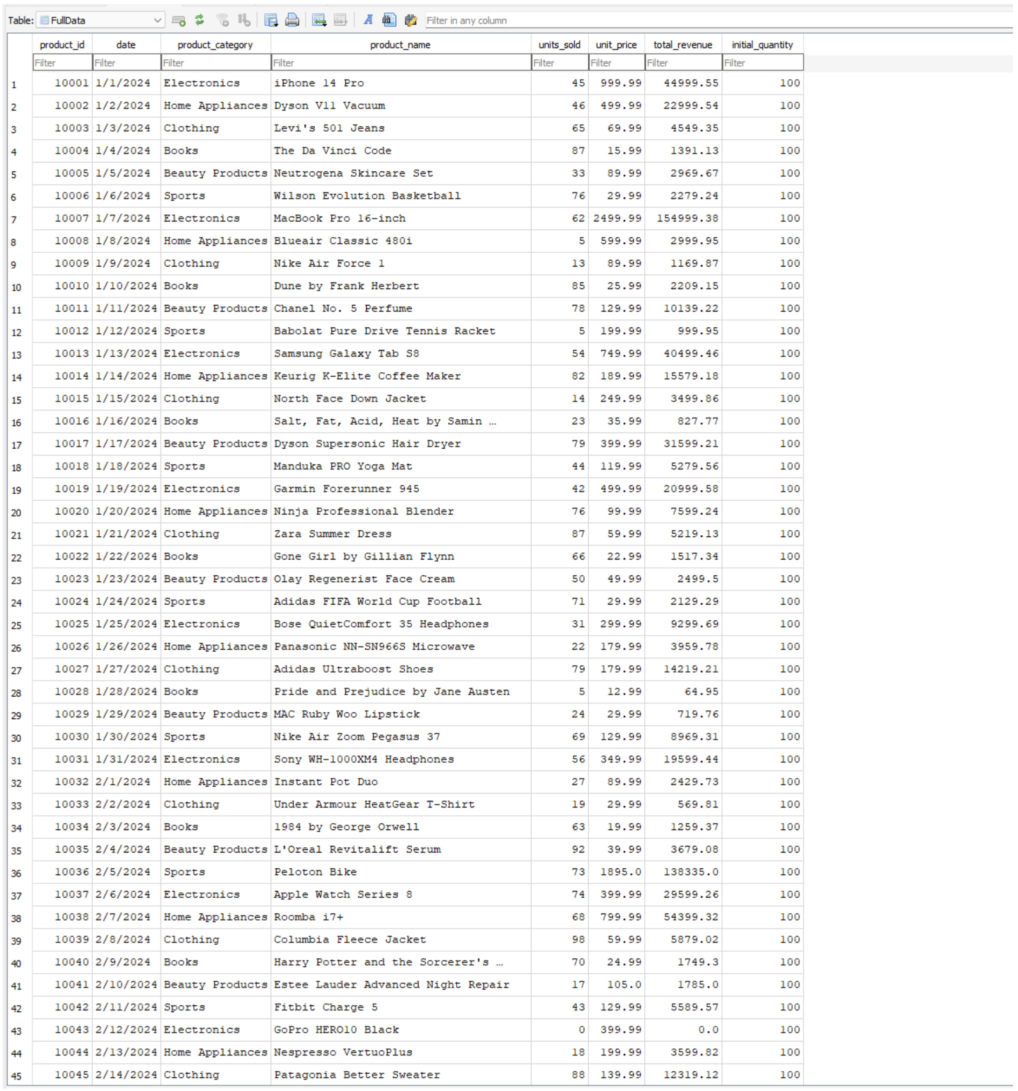
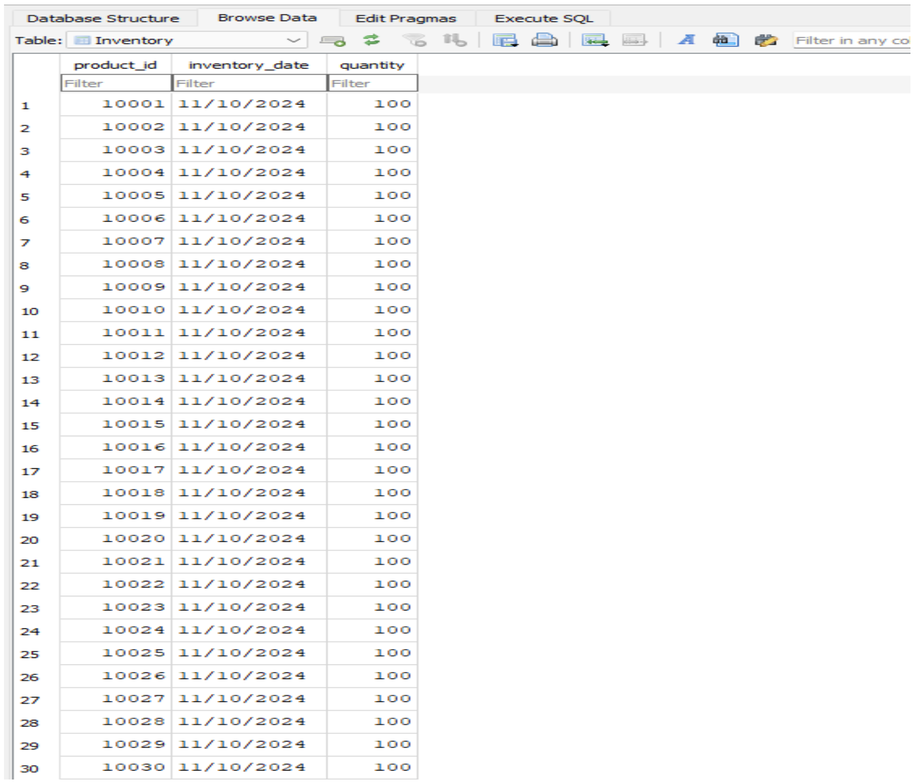


Table: FullData

Filter in any column

	product_id	date	product_category	product_name	units_sold	unit_price	total_revenue	initial_quantity
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	10001	1/1/2024	Electronics	iPhone 14 Pro	45	999.99	44999.55	100
2	10002	1/2/2024	Home Appliances	Dyson V11 Vacuum	46	499.99	22999.54	100
3	10003	1/3/2024	Clothing	Levi's 501 Jeans	65	69.99	4549.35	100
4	10004	1/4/2024	Books	The Da Vinci Code	87	15.99	1391.13	100
5	10005	1/5/2024	Beauty Products	Neutrogena Skincare Set	33	89.99	2969.67	100
6	10006	1/6/2024	Sports	Wilson Evolution Basketball	76	29.99	2279.24	100
7	10007	1/7/2024	Electronics	MacBook Pro 16-inch	62	2499.99	154999.38	100
8	10008	1/8/2024	Home Appliances	Blueair Classic 480i	5	599.99	2999.95	100
9	10009	1/9/2024	Clothing	Nike Air Force 1	13	89.99	1169.87	100
10	10010	1/10/2024	Books	Dune by Frank Herbert	85	25.99	2209.15	100
11	10011	1/11/2024	Beauty Products	Chanel No. 5 Perfume	78	129.99	10139.22	100
12	10012	1/12/2024	Sports	Babolat Pure Drive Tennis Racket	5	199.99	999.95	100
13	10013	1/13/2024	Electronics	Samsung Galaxy Tab S8	54	749.99	40499.46	100
14	10014	1/14/2024	Home Appliances	Keurig K-Elite Coffee Maker	82	189.99	15579.18	100
15	10015	1/15/2024	Clothing	North Face Down Jacket	14	249.99	3499.86	100
16	10016	1/16/2024	Books	Salt, Fat, Acid, Heat by Samin …	23	35.99	827.77	100
17	10017	1/17/2024	Beauty Products	Dyson Supersonic Hair Dryer	79	399.99	31599.21	100
18	10018	1/18/2024	Sports	Manduka PRO Yoga Mat	44	119.99	5279.56	100
19	10019	1/19/2024	Electronics	Garmin Forerunner 945	42	499.99	20999.58	100
20	10020	1/20/2024	Home Appliances	Ninja Professional Blender	76	99.99	7599.24	100
21	10021	1/21/2024	Clothing	Zara Summer Dress	87	59.99	5219.13	100
22	10022	1/22/2024	Books	Gone Girl by Gillian Flynn	66	22.99	1517.34	100
23	10023	1/23/2024	Beauty Products	Olay Regenerist Face Cream	50	49.99	2499.5	100
24	10024	1/24/2024	Sports	Adidas FIFA World Cup Football	71	29.99	2129.29	100
25	10025	1/25/2024	Electronics	Bose QuietComfort 35 Headphones	31	299.99	9299.69	100
26	10026	1/26/2024	Home Appliances	Panasonic NN-SN966S Microwave	22	179.99	3959.78	100
27	10027	1/27/2024	Clothing	Adidas Ultraboost Shoes	79	179.99	14219.21	100
28	10028	1/28/2024	Books	Pride and Prejudice by Jane Austen	5	12.99	64.95	100
29	10029	1/29/2024	Beauty Products	MAC Ruby Woo Lipstick	24	29.99	719.76	100
30	10030	1/30/2024	Sports	Nike Air Zoom Pegasus 37	69	129.99	8969.31	100
31	10031	1/31/2024	Electronics	Sony WH-1000XM4 Headphones	56	349.99	19599.44	100
32	10032	2/1/2024	Home Appliances	Instant Pot Duo	27	89.99	2429.73	100
33	10033	2/2/2024	Clothing	Under Armour HeatGear T-Shirt	19	29.99	569.81	100
34	10034	2/3/2024	Books	1984 by George Orwell	63	19.99	1259.37	100
35	10035	2/4/2024	Beauty Products	L'Oreal Revitalift Serum	92	39.99	3679.08	100
36	10036	2/5/2024	Sports	Peloton Bike	73	1895.0	138335.0	100
37	10037	2/6/2024	Electronics	Apple Watch Series 8	74	399.99	29599.26	100
38	10038	2/7/2024	Home Appliances	Roomba i7+	68	799.99	54399.32	100
39	10039	2/8/2024	Clothing	Columbia Fleece Jacket	98	59.99	5879.02	100
40	10040	2/9/2024	Books	Harry Potter and the Sorcerer's …	70	24.99	1749.3	100
41	10041	2/10/2024	Beauty Products	Estee Lauder Advanced Night Repair	17	105.0	1785.0	100
42	10042	2/11/2024	Sports	Fitbit Charge 5	43	129.99	5589.57	100
43	10043	2/12/2024	Electronics	GoPro HERO10 Black	0	399.99	0.0	100
44	10044	2/13/2024	Home Appliances	Nespresso VertuoPlus	18	199.99	3599.82	100
45	10045	2/14/2024	Clothing	Patagonia Better Sweater	88	139.99	12319.12	100

2. Inventory tables



Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: Inventory

	product_id	inventory_date	quantity
1	10001	11/10/2024	100
2	10002	11/10/2024	100
3	10003	11/10/2024	100
4	10004	11/10/2024	100
5	10005	11/10/2024	100
6	10006	11/10/2024	100
7	10007	11/10/2024	100
8	10008	11/10/2024	100
9	10009	11/10/2024	100
10	10010	11/10/2024	100
11	10011	11/10/2024	100
12	10012	11/10/2024	100
13	10013	11/10/2024	100
14	10014	11/10/2024	100
15	10015	11/10/2024	100
16	10016	11/10/2024	100
17	10017	11/10/2024	100
18	10018	11/10/2024	100
19	10019	11/10/2024	100
20	10020	11/10/2024	100
21	10021	11/10/2024	100
22	10022	11/10/2024	100
23	10023	11/10/2024	100
24	10024	11/10/2024	100
25	10025	11/10/2024	100
26	10026	11/10/2024	100
27	10027	11/10/2024	100
28	10028	11/10/2024	100
29	10029	11/10/2024	100
30	10030	11/10/2024	100

3. Products tables:

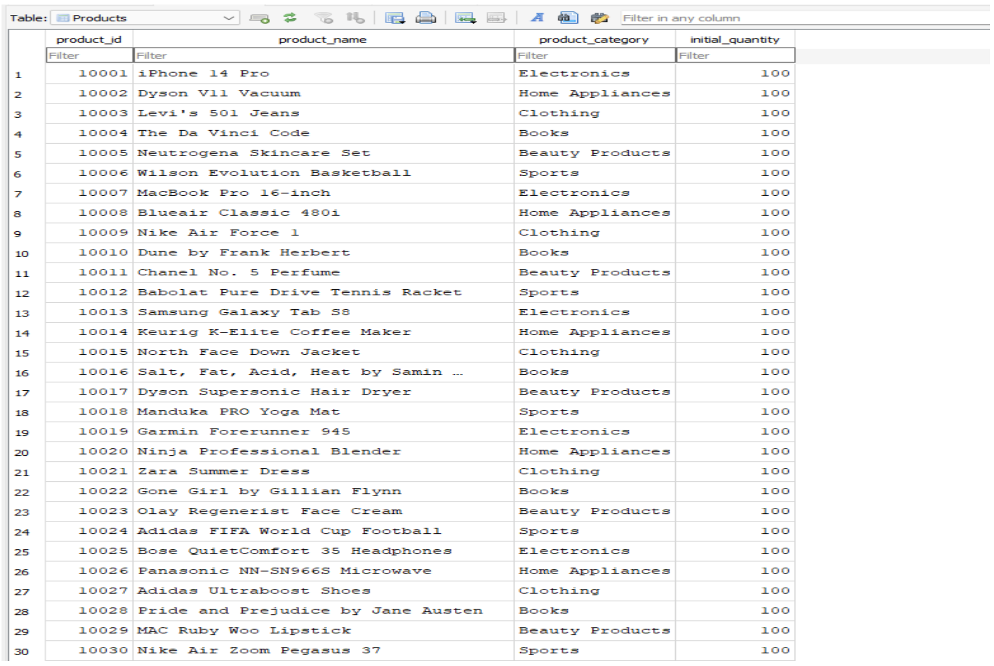
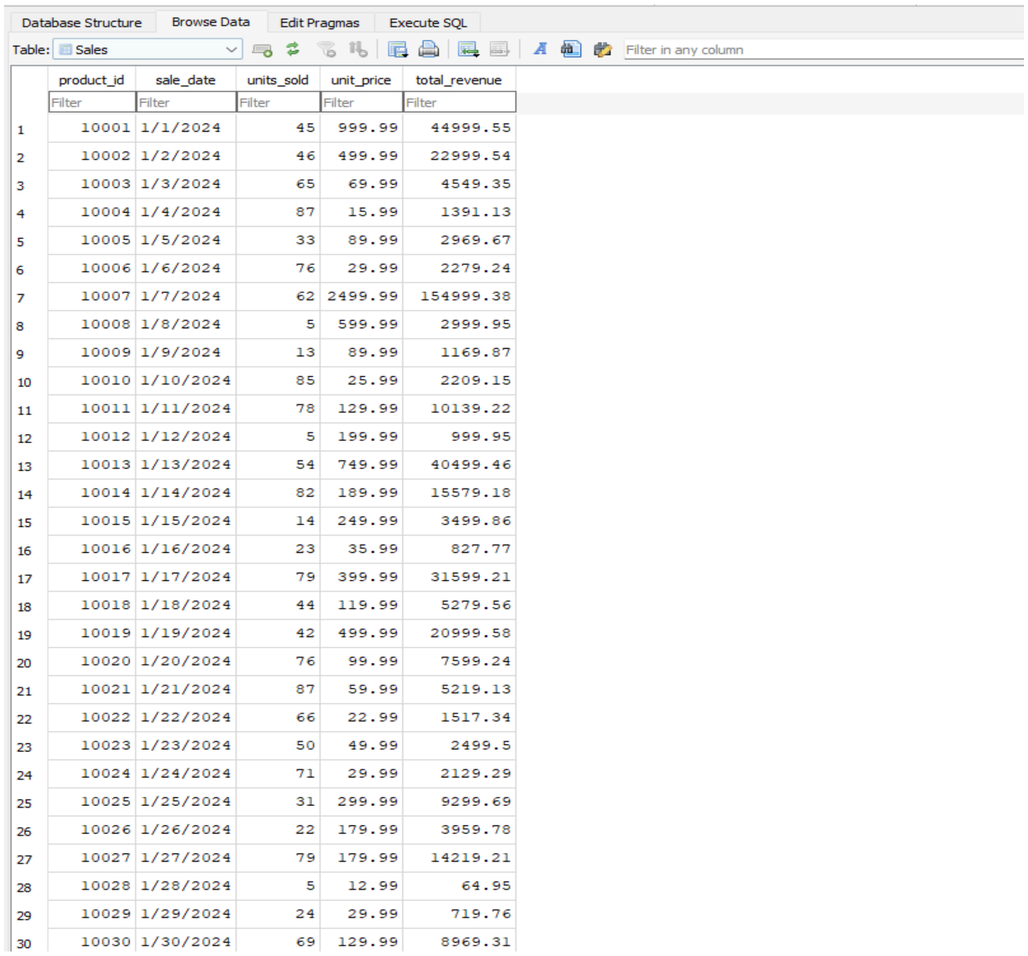


Table: Products

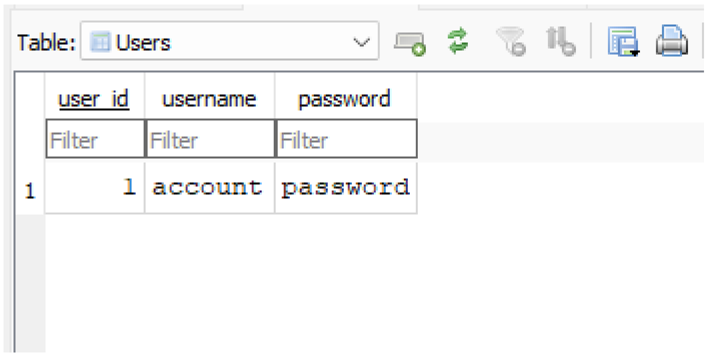
	product_id	product_name	product_category	initial_quantity
1	10001	iPhone 14 Pro	Electronics	100
2	10002	Dyson V11 Vacuum	Home Appliances	100
3	10003	Levi's 501 Jeans	Clothing	100
4	10004	The Da Vinci Code	Books	100
5	10005	Neutrogena Skincare Set	Beauty Products	100
6	10006	Wilson Evolution Basketball	Sports	100
7	10007	MacBook Pro 16-inch	Electronics	100
8	10008	Blueair Classic 480i	Home Appliances	100
9	10009	Nike Air Force 1	Clothing	100
10	10010	Dune by Frank Herbert	Books	100
11	10011	Chanel No. 5 Perfume	Beauty Products	100
12	10012	Babolat Pure Drive Tennis Racket	Sports	100
13	10013	Samsung Galaxy Tab S8	Electronics	100
14	10014	Keurig K-Elite Coffee Maker	Home Appliances	100
15	10015	North Face Down Jacket	Clothing	100
16	10016	Salt, Fat, Acid, Heat by Samin ...	Books	100
17	10017	Dyson Supersonic Hair Dryer	Beauty Products	100
18	10018	Manduka PRO Yoga Mat	Sports	100
19	10019	Garmin Forerunner 945	Electronics	100
20	10020	Ninja Professional Blender	Home Appliances	100
21	10021	Zara Summer Dress	Clothing	100
22	10022	Gone Girl by Gillian Flynn	Books	100
23	10023	Olay Regenerist Face Cream	Beauty Products	100
24	10024	Adidas FIFA World Cup Football	Sports	100
25	10025	Bose QuietComfort 35 Headphones	Electronics	100
26	10026	Panasonic NN-SN966S Microwave	Home Appliances	100
27	10027	Adidas Ultraboost Shoes	Clothing	100
28	10028	Pride and Prejudice by Jane Austen	Books	100
29	10029	MAC Ruby Woo Lipstick	Beauty Products	100
30	10030	Nike Air Zoom Pegasus 37	Sports	100

4. **Sales tables:**



	product_id	sale_date	units_sold	unit_price	total_revenue
1	10001	1/1/2024	45	999.99	44999.55
2	10002	1/2/2024	46	499.99	22999.54
3	10003	1/3/2024	65	69.99	4549.35
4	10004	1/4/2024	87	15.99	1391.13
5	10005	1/5/2024	33	89.99	2969.67
6	10006	1/6/2024	76	29.99	2279.24
7	10007	1/7/2024	62	2499.99	154999.38
8	10008	1/8/2024	5	599.99	2999.95
9	10009	1/9/2024	13	89.99	1169.87
10	10010	1/10/2024	85	25.99	2209.15
11	10011	1/11/2024	78	129.99	10139.22
12	10012	1/12/2024	5	199.99	999.95
13	10013	1/13/2024	54	749.99	40499.46
14	10014	1/14/2024	82	189.99	15579.18
15	10015	1/15/2024	14	249.99	3499.86
16	10016	1/16/2024	23	35.99	827.77
17	10017	1/17/2024	79	399.99	31599.21
18	10018	1/18/2024	44	119.99	5279.56
19	10019	1/19/2024	42	499.99	20999.58
20	10020	1/20/2024	76	99.99	7599.24
21	10021	1/21/2024	87	59.99	5219.13
22	10022	1/22/2024	66	22.99	1517.34
23	10023	1/23/2024	50	49.99	2499.5
24	10024	1/24/2024	71	29.99	2129.29
25	10025	1/25/2024	31	299.99	9299.69
26	10026	1/26/2024	22	179.99	3959.78
27	10027	1/27/2024	79	179.99	14219.21
28	10028	1/28/2024	5	12.99	64.95
29	10029	1/29/2024	24	29.99	719.76
30	10030	1/30/2024	69	129.99	8969.31

5. **Users tables:**

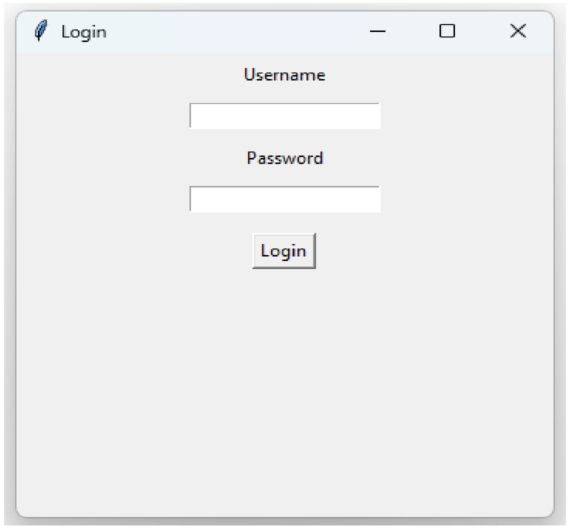


	user_id	username	password
1	1	account	password

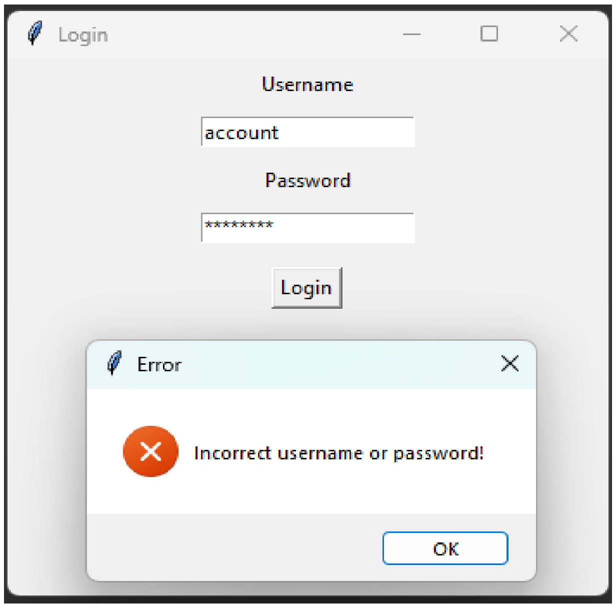
The Main Actions In The System

The following are some screenshots of how the system works. Each screenshot will be followed by a caption of what the screenshot entails, as well as a brief description where appropriate.

1. **Login:** The first screen when running the program is the login, and the user will enter the login information.



If the user enters an incorrect Username or Password, they will receive a message.



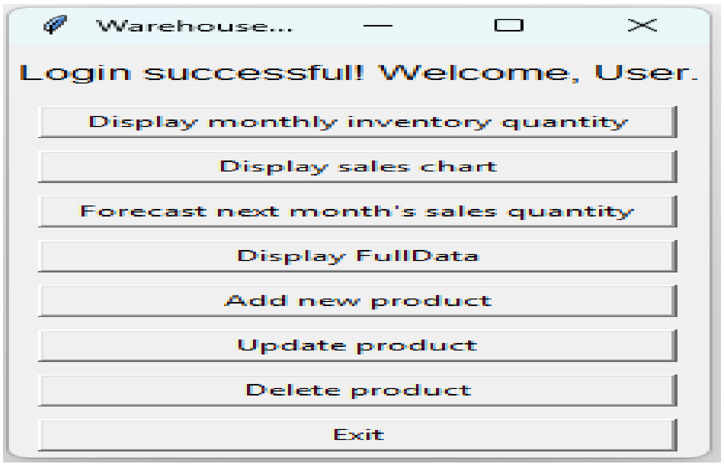
Username: account

Password: ********

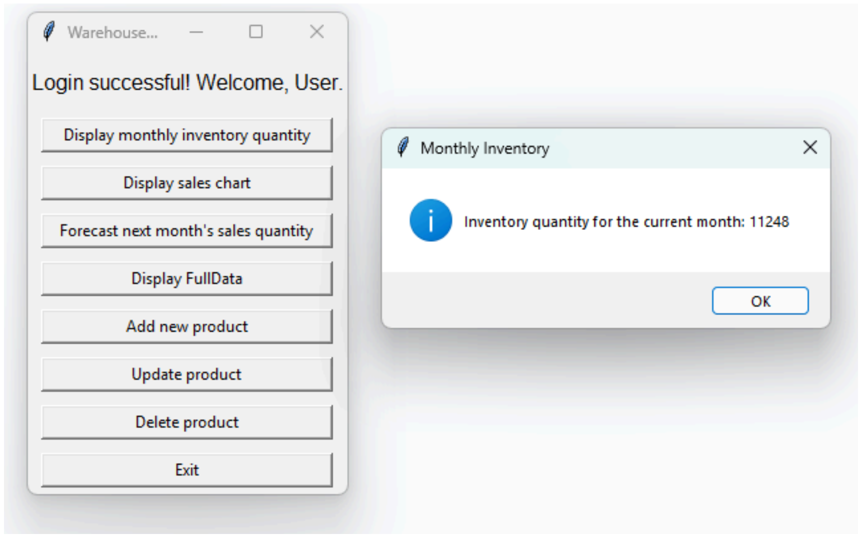
Incorrect username or password!

Note: If the user does not want to continue logging in, close the login window by clicking on the "x" and the program will exit.

2. **The main application:** Once the user has entered the login information correctly, they will be routed to the main screen with options given that the system would have provided. There will be 8 options for the user.

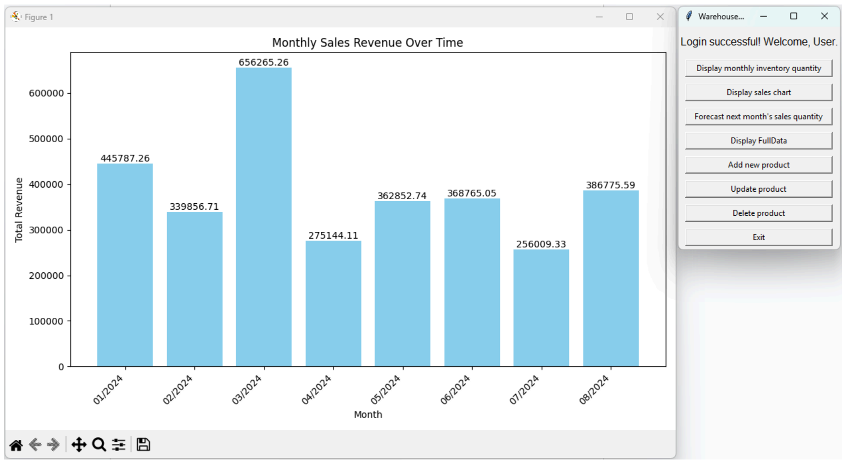

Warehouse...
Login successful! Welcome, User.
Display monthly inventory quantity
Display sales chart
Forecast next month's sales quantity
Display FullData
Add new product
Update product
Delete product
Exit

3. **Display monthly inventory quantity:** This option will display the current total inventory, giving users a quick look at the inventory on hand.



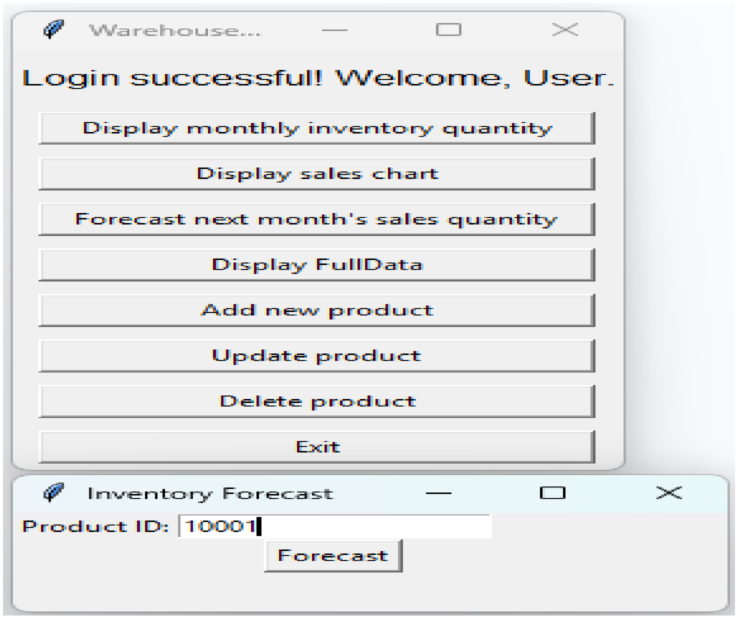
Inventory quantity for the current month: 11248

4. **Display sales chart:** Calculates and plots a bar chart of total sales for each month.

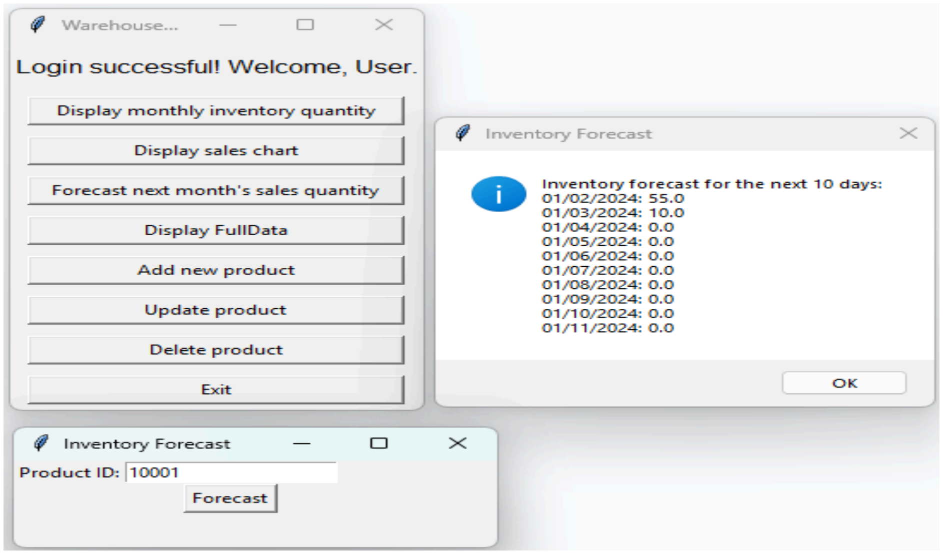


Note: Once users add new products or update old products or delete products, it will also recalculate and redraw the chart according to the new data.

5. **Forecast next month's sales quantity:** This option will forecast sales for the next 10 days based on historical data.



The user will enter the Product ID.



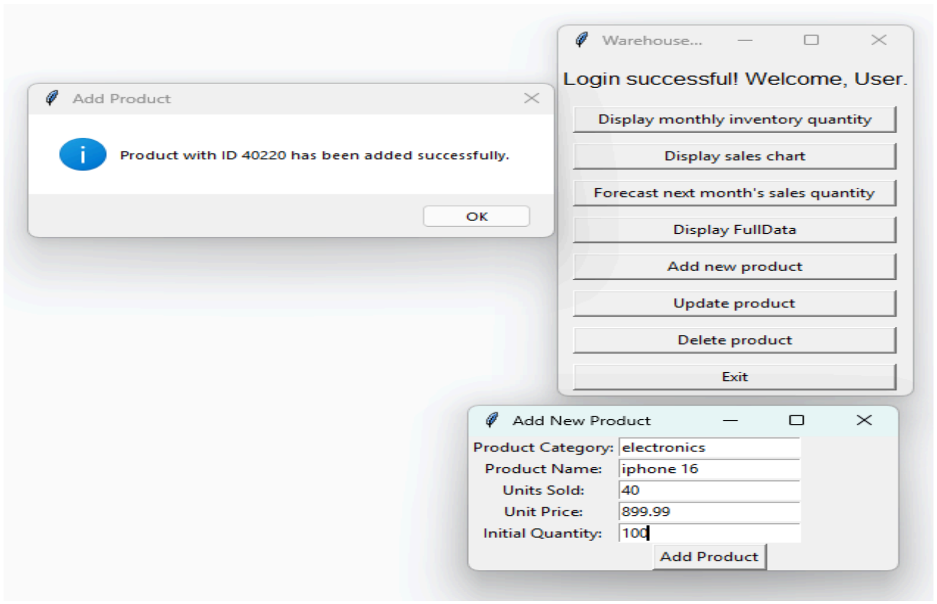
Login successful! Welcome, User.

Inventory forecast for the next 10 days:
01/02/2024: 55.0
01/03/2024: 10.0
01/04/2024: 0.0
01/05/2024: 0.0
01/06/2024: 0.0
01/07/2024: 0.0
01/08/2024: 0.0
01/09/2024: 0.0
01/10/2024: 0.0
01/11/2024: 0.0

Product ID: 10001

After the user enters the correct Product Id, the system will display sales for the next 10 days.

6. **Add new product:**The system will prompt the user to enter new product information and the new product will be added to the inventory.



Product with ID 40220 has been added successfully.

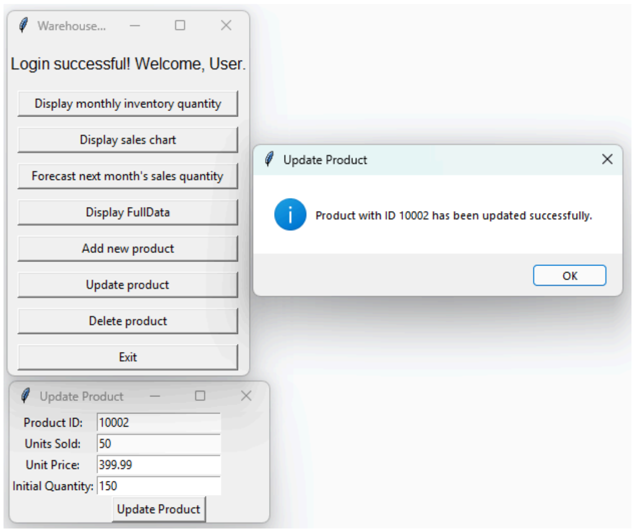
Login successful! Welcome, User.

Product Category: electronics
Product Name: iphone 16
Units Sold: 40
Unit Price: 899.99
Initial Quantity: 100

After entering new product information and clicking the "add product" button, the user will receive a notification.

Note: The new product will be added to the end of the original data with index number 241.

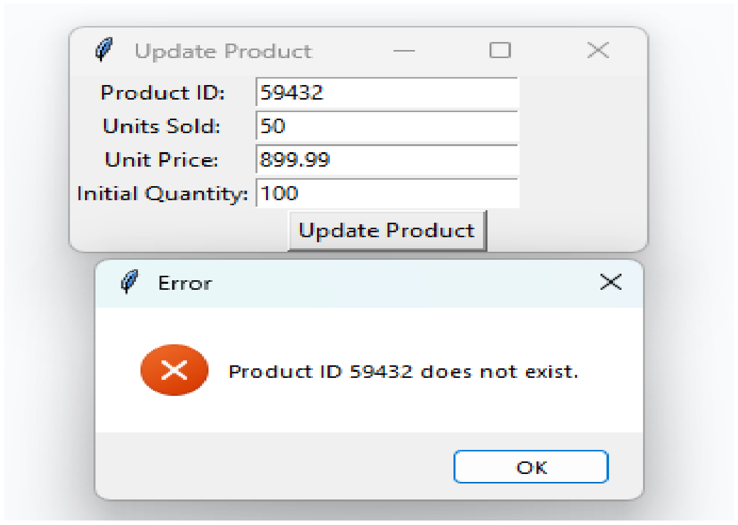
7. **Update product:** The system will prompt the user to enter product information that needs to be updated.



Product with ID 10002 has been updated successfully.

Product ID: 10002
Units Sold: 50
Unit Price: 399.99
Initial Quantity: 150

After the user has entered the product information that needs to be updated, the system will display a notification to the user.

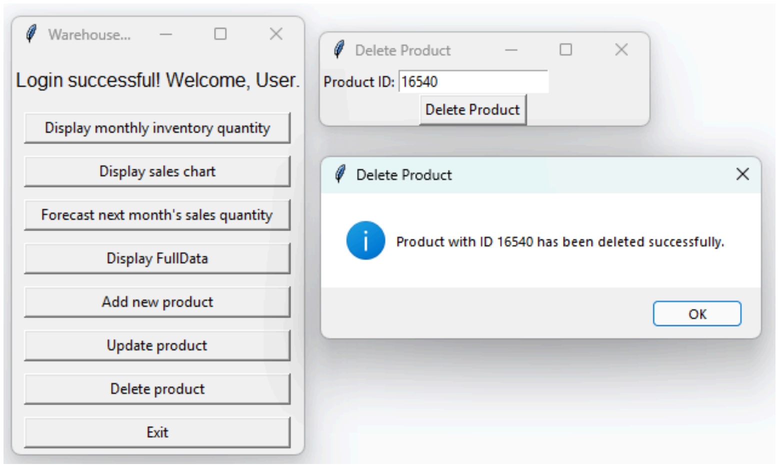


Product ID: 59432
Units Sold: 50
Unit Price: 899.99
Initial Quantity: 100

Product ID 59432 does not exist.

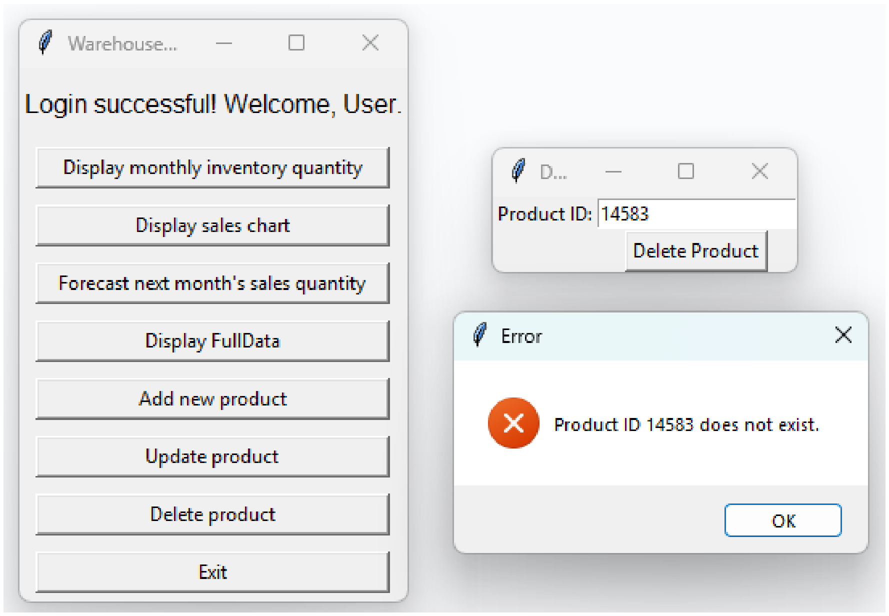
If the user enters a Product ID that does not exist, the system will notify the user.

8. **Delete product:** The system will prompt the user to enter the Product ID of the product they want to delete.



Product with ID 16540 has been deleted successfully.

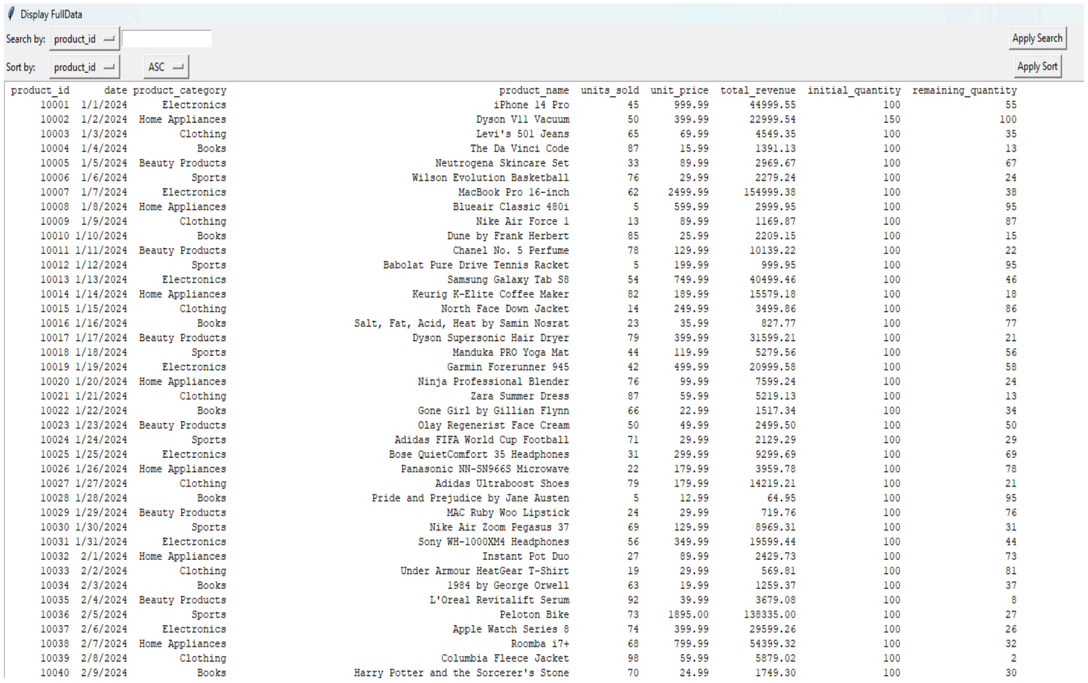
After the user enters the correct Product ID to be deleted, they will receive the message "deleted successfully".



Product ID 14583 does not exist.

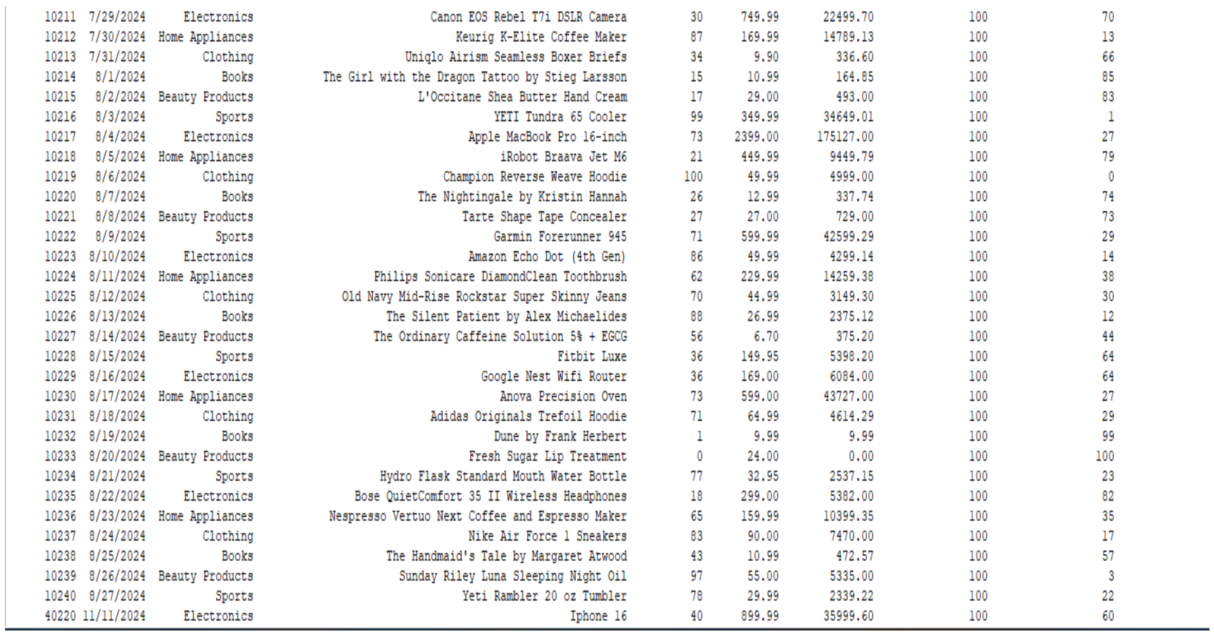
If the user enters a product that does not exist, the system will notify the user.

9. **Display FullData:** The system will display all data about product information, including products that have been added, products that have been updated, and products that have been deleted will no longer appear. In addition, there is a search and sort function to help users easily find products.



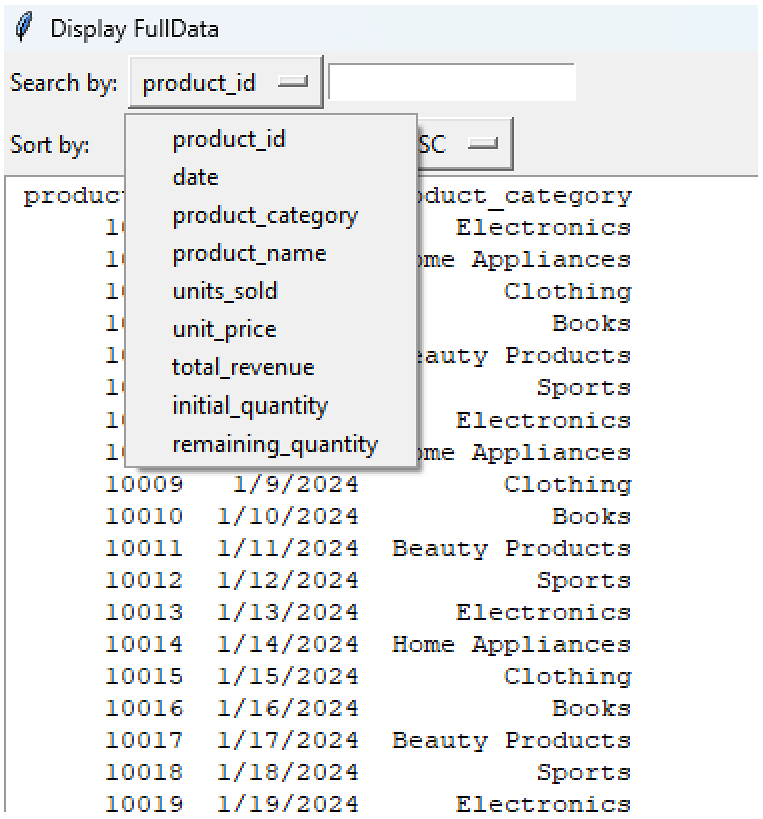
```
product_id       date product_category                          product_name units_sold unit_price total_revenue initial_quantity remaining_quantity
     10001   1/1/2024      Electronics                         iPhone 14 Pro         45     999.99      44999.55              100                 55
     10002   1/2/2024  Home Appliances                      Dyson V11 Vacuum         50     399.99      22999.54              150                100
     10003   1/3/2024         Clothing                      Levi's 501 Jeans         65      69.99       4549.35              100                 35
     10004   1/4/2024            Books                     The Da Vinci Code         87      15.99       1391.13              100                 13
     10005   1/5/2024  Beauty Products               Neutrogena Skincare Set         33      89.99       2969.67              100                 67
     10006   1/6/2024           Sports           Wilson Evolution Basketball         76      29.99       2279.24              100                 24
     10007   1/7/2024      Electronics                   MacBook Pro 16-inch         62    2499.99     154999.38              100                 38
     10008   1/8/2024  Home Appliances                  Blueair Classic 480i          5     599.99       2999.95              100                 95
     10009   1/9/2024         Clothing                      Nike Air Force 1         13      89.99       1169.87              100                 87
     10010  1/10/2024            Books                 Dune by Frank Herbert         85      25.99       2209.15              100                 15
     10011  1/11/2024  Beauty Products                  Chanel No. 5 Perfume         78     129.99      10139.22              100                 22
     10012  1/12/2024           Sports      Babolat Pure Drive Tennis Racket          5     199.99        999.95              100                 95
     10013  1/13/2024      Electronics                 Samsung Galaxy Tab S8         54     749.99      40499.46              100                 46
     10014  1/14/2024  Home Appliances           Keurig K-Elite Coffee Maker         82     189.99      15579.18              100                 18
     10015  1/15/2024         Clothing                North Face Down Jacket         14     249.99       3499.86              100                 86
     10016  1/16/2024            Books Salt, Fat, Acid, Heat by Samin Nosrat         23      35.99        827.77              100                 77
     10017  1/17/2024  Beauty Products           Dyson Supersonic Hair Dryer         79     399.99      31599.21              100                 21
     10018  1/18/2024           Sports                  Manduka PRO Yoga Mat         44     119.99       5279.56              100                 56
     10019  1/19/2024      Electronics                 Garmin Forerunner 945         42     499.99      20999.58              100                 58
     10020  1/20/2024  Home Appliances            Ninja Professional Blender         76      99.99       7599.24              100                 24
     10021  1/21/2024         Clothing                     Zara Summer Dress         87      59.99       5219.13              100                 13
     10022  1/22/2024            Books            Gone Girl by Gillian Flynn         66      22.99       1517.34              100                 34
     10023  1/23/2024  Beauty Products            Olay Regenerist Face Cream         50      49.99       2499.50              100                 50
     10024  1/24/2024           Sports        Adidas FIFA World Cup Football         71      29.99       2129.29              100                 29
     10025  1/25/2024      Electronics       Bose QuietComfort 35 Headphones         31     299.99       9299.69              100                 69
     10026  1/26/2024  Home Appliances         Panasonic NN-SN966S Microwave         22     179.99       3959.78              100                 78
     10027  1/27/2024         Clothing               Adidas Ultraboost Shoes         79     179.99      14219.21              100                 21
     10028  1/28/2024            Books    Pride and Prejudice by Jane Austen          5      12.99         64.95              100                 95
     10029  1/29/2024  Beauty Products                 MAC Ruby Woo Lipstick         24      29.99        719.76              100                 76
     10030  1/30/2024           Sports                 Nike Air Zoom Pegasus 37      69     129.99       8969.31              100                 31
     10031  1/31/2024      Electronics            Sony WH-1000XM4 Headphones         56     349.99      19599.44              100                 44
     10032   2/1/2024  Home Appliances                       Instant Pot Duo         27      89.99       2429.73              100                 73
     10033   2/2/2024         Clothing         Under Armour HeatGear T-Shirt         19      29.99        569.81              100                 81
     10034   2/3/2024            Books                 1984 by George Orwell         63      19.99       1259.37              100                 37
     10035   2/4/2024  Beauty Products              L'Oreal Revitalift Serum         92      39.99       3679.08              100                  8
     10036   2/5/2024           Sports                          Peloton Bike         73    1895.00     138335.00              100                 27
     10037   2/6/2024      Electronics                  Apple Watch Series 8         74     399.99      29599.26              100                 26
     10038   2/7/2024  Home Appliances                            Roomba i7+         68     799.99      54399.32              100                 32
     10039   2/8/2024         Clothing                Columbia Fleece Jacket         98      59.99       5879.02              100                  2
     10040   2/9/2024            Books Harry Potter and the Sorcerer's Stone         70      24.99       1749.30              100                 30
```

Note: Product ID 10002 has been updated.



10211	7/29/2024	Electronics	Canon EOS Rebel T7i DSLR Camera	30	749.99	22499.70	100	70
10212	7/30/2024	Home Appliances	Keurig K-Elite Coffee Maker	87	169.99	14789.13	100	13
10213	7/31/2024	Clothing	Uniqlo Airism Seamless Boxer Briefs	34	9.90	336.60	100	66
10214	8/1/2024	Books	The Girl with the Dragon Tattoo by Stieg Larsson	15	10.99	164.85	100	85
10215	8/2/2024	Beauty Products	L'Occitane Shea Butter Hand Cream	17	29.00	493.00	100	83
10216	8/3/2024	Sports	YETI Tundra 65 Cooler	99	349.99	34649.01	100	1
10217	8/4/2024	Electronics	Apple MacBook Pro 16-inch	73	2399.00	175127.00	100	27
10218	8/5/2024	Home Appliances	iRobot Braava Jet M6	21	449.99	9449.79	100	79
10219	8/6/2024	Clothing	Champion Reverse Weave Hoodie	100	49.99	4999.00	100	0
10220	8/7/2024	Books	The Nightingale by Kristin Hannah	26	12.99	337.74	100	74
10221	8/8/2024	Beauty Products	Tarte Shape Tape Concealer	27	27.00	729.00	100	73
10222	8/9/2024	Sports	Garmin Forerunner 945	71	599.99	42599.29	100	29
10223	8/10/2024	Electronics	Amazon Echo Dot (4th Gen)	86	49.99	4299.14	100	14
10224	8/11/2024	Home Appliances	Philips Sonicare DiamondClean Toothbrush	62	229.99	14259.38	100	38
10225	8/12/2024	Clothing	Old Navy Mid-Rise Rockstar Super Skinny Jeans	70	44.99	3149.30	100	30
10226	8/13/2024	Books	The Silent Patient by Alex Michaelides	88	26.99	2375.12	100	12
10227	8/14/2024	Beauty Products	The Ordinary Caffeine Solution 5% + EGCG	56	6.70	375.20	100	44
10228	8/15/2024	Sports	Fitbit Luxe	36	149.95	5398.20	100	64
10229	8/16/2024	Electronics	Google Nest Wifi Router	36	169.00	6084.00	100	64
10230	8/17/2024	Home Appliances	Anova Precision Oven	73	599.00	43727.00	100	27
10231	8/18/2024	Clothing	Adidas Originals Trefoil Hoodie	71	64.99	4614.29	100	29
10232	8/19/2024	Books	Dune by Frank Herbert	1	9.99	9.99	100	99
10233	8/20/2024	Beauty Products	Fresh Sugar Lip Treatment	0	24.00	0.00	100	100
10234	8/21/2024	Sports	Hydro Flask Standard Mouth Water Bottle	77	32.95	2537.15	100	23
10235	8/22/2024	Electronics	Bose QuietComfort 35 II Wireless Headphones	18	299.00	5382.00	100	82
10236	8/23/2024	Home Appliances	Nespresso Vertuo Next Coffee and Espresso Maker	65	159.99	10399.35	100	35
10237	8/24/2024	Clothing	Nike Air Force 1 Sneakers	83	90.00	7470.00	100	17
10238	8/25/2024	Books	The Handmaid's Tale by Margaret Atwood	43	10.99	472.57	100	57
10239	8/26/2024	Beauty Products	Sunday Riley Luna Sleeping Night Oil	97	55.00	5335.00	100	3
10240	8/27/2024	Sports	Yeti Rambler 20 oz Tumbler	78	29.99	2339.22	100	22
40220	11/11/2024	Electronics	Iphone 16	40	899.99	35999.60	100	60

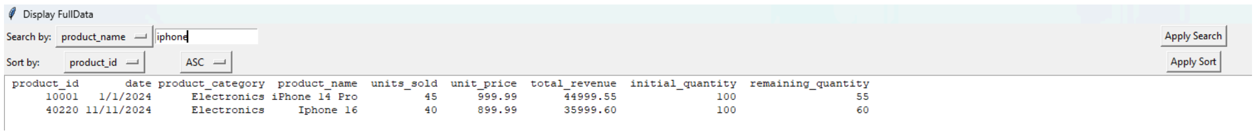
Note: Product ID 40220 is a newly added product.



Search by: product_id

Sort by: SC

```
produc                          duct_category
      1                           Electronics
      1                        me Appliances
      1                              Clothing
      1                                 Books
      1                        auty Products
      1                                Sports
      1                           Electronics
      1                        me Appliances
  10009     1/9/2024                 Clothing
  10010    1/10/2024                    Books
  10011    1/11/2024          Beauty Products
  10012    1/12/2024                   Sports
  10013    1/13/2024              Electronics
  10014    1/14/2024          Home Appliances
  10015    1/15/2024                 Clothing
  10016    1/16/2024                    Books
  10017    1/17/2024          Beauty Products
  10018    1/18/2024                   Sports
  10019    1/19/2024              Electronics
```



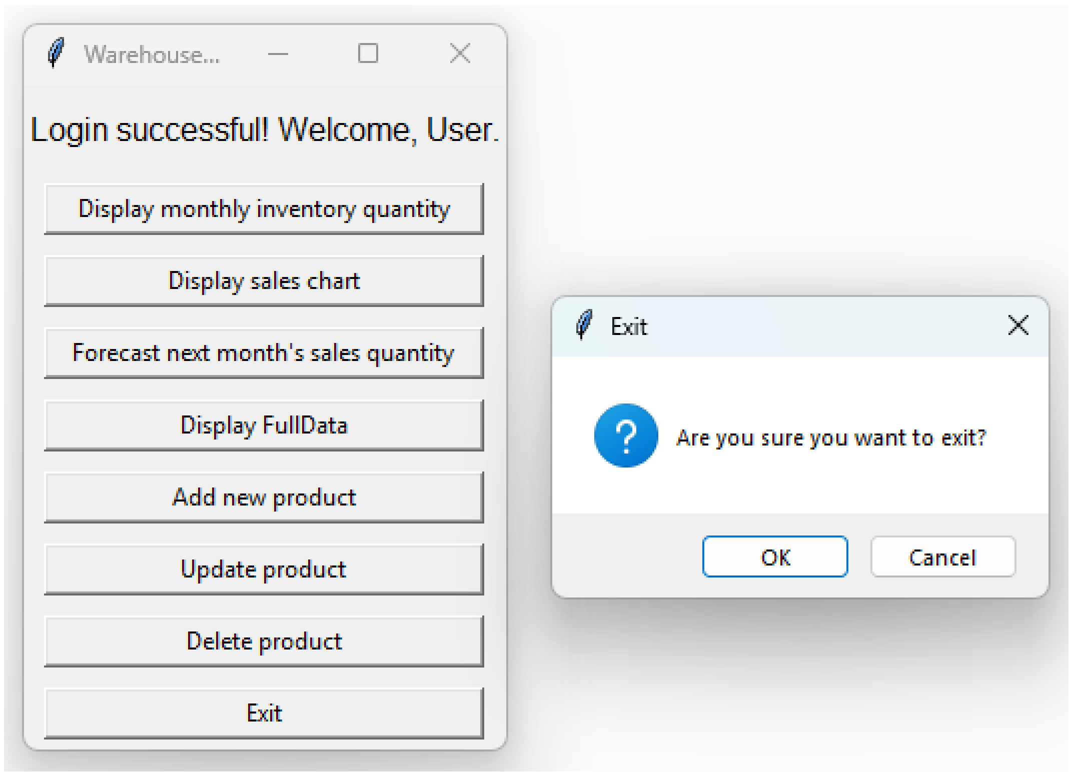
product_name · iphone · product_id · ASC

```
product_id       date product_category  product_name  units_sold  unit_price  total_revenue  initial_quantity  remaining_quantity
     10001   1/1/2024      Electronics iPhone 14 Pro          45      999.99       44999.55               100                  55
     40220 11/11/2024      Electronics     Iphone 16          40      899.99       35999.60               100                  60
```

Note: Select any column from "search by", then enter name or ID or price and click "Apply search" to search.

Note: Select a column from "Sort by" and select "ASC: ascending, A-Z" or "DESC: descending, Z-A" and click "Apply sort" to sort.

10. **Exit:** Exit program.



Login successful! Welcome, User.

Are you sure you want to exit?

Results

Data Summary

After Exiting the program, any new data, any changes, and any deleted items will all be persisted to the “inventory.db” database.

1. **Products table:** This contains the information of each product such as product_id, product_name, category, and initial_quantity.
2. **Sales Table:** This is the sales transactions table having details like product_id, sale_date, unit_sold, unit_price, and Total_revenue of the products sold.
3. **Inventory table:** It Manages inventory information into the same database file with information including product_id, inventory_date, remaining_quantity, and initial_quantity.
4. **FullData table:** includes all information regarding the product.
5. **Data Update and Maintenance:** The inclusion of the remaining_quantity column in FullData table and the Inventory table supports updating records after a sale. Date formats have been standardized for consistent reporting as well as calculations with data.

Performance Metrics

1. **Addition and Updating of Products:** The program ensures addition of fresh products into the warehouse and updating of the existing information related to a given set of products. Information changes like sales quantity, sales price, inventory should be synched across the tables based on which correct information is updated.

2. **Function to Delete a Product:** After information is entered for the product to delete, the system first checks if the specific "Product ID" exists; otherwise, it outputs a message that the product code doesn't exist. Then, it deletes after updating corresponding tables that unrelated data does not remain in the system.
3. **Statistics and Display Features:**

   _ Monthly sales revenue chart: This will allow easy observation of revenue trends over time by the users.

   _ Monthly inventory statistics: With this, users can maintain the record of how much quantity of the goods still remains in the warehouse.

   _ Search/sort by column: Users can easily find the information they are looking for about a product in the shortest time possible by sorting or searching data.

   _ Forecasting the expected number of products to be sold during the next 10 days, thus helping in a better way of controlling amounts of stocks to be kept, reducing risks and enhancing business operations.
4. **Performance and Stability:** The application processes data in memory and information is written back to the SQLite database, ensuring changes in data are saved. very efficient ware management software in a friendly, easy to use intuitive interface guarantees smooth operations with top speed.

Reduced Stock Shortages

Thanks to the system's efficiency, stock shortages have been mitigated. This is because of the system's forecasting, which takes into account previous data and predicts future sales. With such accurate forecasting, stock can be ordered and developed accordingly before needing to be sold. This is also helped by the fact that inventory is more accurately tracked, thus creating an easier time for other systems and other users to understand what the company may require at a glance.

Improved Cash Flow and Inventory Control

Due to the aforementioned inventory management, inventory and stock can more easily be surveyed and taken action for. Compared to previous systems, this system can more clearly display shortages or other problems that a company would need to address quickly and efficiently through the sales data chart display. Thus, less time is wasted trying to double or triple check inventory or future purchases, further saving companies time and money.

Conclusion

Throughout this project, the difficulty of databases and how to manage them was made apparent. As data gets bigger and companies grow with more products, there is an ever-growing need for effective and fast software that can handle this data and allow users to understand and make decisions quickly. This system achieves this goal, as well as other major concerns a business may have, such as forecasting future required purchases as well as easy input of new data.

Although this system may not be currently in use for any modern company, it is a very strong base for any future system that a company may require. It includes all required features that pertain to a business as well as documentation to further advance the system for custom needs. Unlike other systems, this one includes many visual aspects – a nearly required part to make it feel like an app that is user friendly. By ensuring everything is as visual as possible, users can make decisions quickly rather than having to parse through data like other systems may require.

It has also served an important purpose as being a powerful learning tool for the creators of this project. The project planning, collaboration, programming, and debugging have all been important skills required to complete this project and ensure the quality of the system. For example, the programming needed the use and learning of programs such as Pandas for data analysis, sqlite3 for database implementation, tkinter for user interface, and matplotlib for graph generation, as well as the implementation of Python libraries. Overall, the building of this system was an important stepping stone in the development of many business management and programming skills, as well as now serving as a proper foundation for future systems, apps, or programs.

Works Cited

- Date, C. J. (2003). *An Introduction to Database Systems, Fifth Edition*. Boston, MA: Addison Wesley.
- Standing Technology. (2018, November 25). Why your business needs a good database. *Standing Technology*.
- Subramoniam, S. and Krishnankutty, K.V. (2005), "EDSIM: expert database system for inventory management", *Kybernetes*, Vol. 34 No. 5, pp. 721-733. https://doi.org/10.1108/03684920510595409
- Online Sales Data.csv. (2024). “Online Sales Dataset - Popular Marketplace Data”. https://www.kaggle.com/datasets/shreyanshverma27/online-sales-dataset-popular-marketplace-data