**Build an Interactive Sales Dashboard with Streamlit and Python**

[[Mohamed Yosef](https://medium.com/@mohamedyosef101?source=post_page-----4f02fe49b470--------------------------------)](https://medium.com/@mohamedyosef101?source=post_page-----4f02fe49b470--------------------------------)

[Mohamed Yosef](https://medium.com/@mohamedyosef101?source=post_page-----4f02fe49b470--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F122eca15ad5c&operation=register&redirect=https%3A%2F%2Fmedium.com%2F%40mohamedyosef101%2Fbuild-an-interactive-sales-dashboard-with-streamlit-and-python-4f02fe49b470&user=Mohamed+Yosef&userId=122eca15ad5c&source=post_page-122eca15ad5c----4f02fe49b470---------------------post_header-----------)

4 min read

·

Aug 23

22

As data professionals, we often create reports and visualizations that provide insights into trends and metrics. However, these are usually static representations like PDFs, slides, and notebooks.

**What if you could turn your analysis into an interactive web app that others could easily use?**

In this post, I’ll walk through how I built a sales dashboard using Streamlit — a powerful framework in Python for creating web apps for data science and machine learning.

**Table of content:**

[Why Streamlit?](https://medium.com/@mohamedyosef101/build-an-interactive-sales-dashboard-with-streamlit-and-python-4f02fe49b470#9e8d)  
[Walkthrough](https://medium.com/@mohamedyosef101/build-an-interactive-sales-dashboard-with-streamlit-and-python-4f02fe49b470#b925)  
[1. Import libraries](https://medium.com/@mohamedyosef101/build-an-interactive-sales-dashboard-with-streamlit-and-python-4f02fe49b470#68b0)  
[2. Load data and page config](https://medium.com/@mohamedyosef101/build-an-interactive-sales-dashboard-with-streamlit-and-python-4f02fe49b470#c26e)  
[3. Layout sidebar filters](https://medium.com/@mohamedyosef101/build-an-interactive-sales-dashboard-with-streamlit-and-python-4f02fe49b470#06ff)  
[4. Create dashboard sections](https://medium.com/@mohamedyosef101/build-an-interactive-sales-dashboard-with-streamlit-and-python-4f02fe49b470#ab56)  
[5. Deploy online](https://medium.com/@mohamedyosef101/build-an-interactive-sales-dashboard-with-streamlit-and-python-4f02fe49b470#d01f)



Image by the author

**Why Streamlit?**

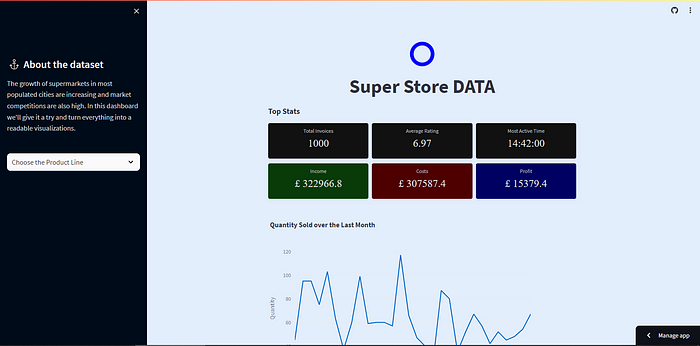
Some key advantages of Streamlit:

* Quickly turn scripts and notebooks into web apps
* Interactive widgets like sliders, selectors, date pickers
* Beautiful charts and graphs using Plotly and other libraries
* Responsive layouts that work on desktop and mobile
* Easy styling with themes, CSS, and components
* Seamless deployment and sharing

This makes Streamlit a perfect choice for building data-driven web dashboards with live filtering, visualization, and exploration.

**Walkthrough**

As supermarkets grow in populated cities, market competition increases. In this dashboard, we visualize supermarket sales data to uncover insights.



[Live Demo](https://super-dashboard.streamlit.app/)

**About the Data**

The dataset contains historical sales records from 3 supermarket branches over 3 months in 2023.

**Changes in the database**

The data has been slightly altered — years were incremented by 4 years and cities were changed to another names.

UPDATE superSales  
SET Order\_date = DATEADD(YEAR, 4, Order\_date)  
WHERE YEAR(Order\_date) = 2019  
  
-- change cites names  
Update superSales  
set City = case Branch  
 WHEN 'A' THEN 'Plymouth'  
 WHEN 'B' THEN 'Bristol'  
 WHEN 'C' THEN 'Glasgow'  
 END;  
select City from superSales;

Let’s Start with Streamlit

**1.Import the Libraries**

We’ll need core data analysis libraries like Pandas, NumPy as well as plotting libraries like Plotly Express. Streamlit is imported as st.

import streamlit as st  
import pandas as pd  
import plotly.express as px

**2. Load data and page config**

I loaded a supermarket sales CSV file using Pandas, did some preprocessing like setting page config as well as creating the layout variables.

# Load the data  
superSales = pd.read\_csv('data/superSales.csv')  
  
# Setting page config  
st.set\_page\_config(page\_title="Super Store Dashboard",   
 page\_icon="https://upload.wikimedia.org/wikipedia/commons/thumb/2/2f/Map-circle-blue.svg/1024px-Map-circle-blue.svg.png",  
 initial\_sidebar\_state="expanded",  
 )  
  
# the layout Variables  
hero = st.container()  
topRow = st.container()  
midRow = st.container()  
chartRow = st.container()  
footer = st.container()

**3. Layout sidebar filters**

Sidebars in Streamlit provide an easy way to add filters that users can tweak to update the dashboard.

# Sidebar  
with st.sidebar:  
 st.markdown(f'''  
 <style>  
 section[data-testid="stSidebar"] {{  
 width: 500px;  
 background-color: #000b1a;  
 }}  
 section[data-testid="stSidebar"] h1 {{  
 color: #e3eefc;  
 }}  
 section[data-testid="stSidebar"] p {{  
 color: #ddd;  
 text-align: left;  
 }}  
 section[data-testid="stSidebar"] svg {{  
 fill: #ddd;  
 }}  
 </style>  
 ''',unsafe\_allow\_html=True)  
 st.title(":anchor: About the dataset")  
 st.markdown("The growth of supermarkets in most populated cities are increasing and market competitions are also high. In this dashboard we'll give it a try and turn everything into a readable visualizations.")

I added selectbox filters to choose Product Line….

# The Selectbox  
 Product\_lines = superSales['Product\_line'].unique()  
 line = st.selectbox('',['Choose the Product Line'] + list(Product\_lines))  
 if line == 'Choose the Product Line':  
 chosen\_line = superSales  
 else:  
 chosen\_line = superSales[superSales['Product\_line'] == line]  
  
 # Customizing the select box  
 st.markdown(f'''  
 <style>  
 .stSelectbox div div {{  
 background-color: #fafafa;  
 color: #333;  
 }}  
 .stSelectbox div div:hover {{  
 cursor: pointer  
 }}  
 .stSelectbox div div .option {{  
 background-color: red;  
 color: #111;  
 }}  
 .stSelectbox div div svg {{  
 fill: black;  
 }}  
 </style>  
 ''', unsafe\_allow\_html=True)

**4. Create dashboard sections**

The main content is structured into sections:

with chartRow:  
 # Filter for the month  
 superSales['Order\_date'] = pd.to\_datetime(superSales['Order\_date'])  
 mar\_data = (superSales['Order\_date'].dt.month == 3)  
 lineQuantity = chosen\_line[(mar\_data)]  
  
 # Quantity for each day  
 quantity\_per\_day = lineQuantity.groupby('Order\_date')['Quantity'].sum().reset\_index()  
  
 # some space  
 st.markdown('<div></div>', unsafe\_allow\_html=True)  
   
 # Create a line chart for Quantity over the last month using Plotly  
 fig\_quantity = px.line(  
 quantity\_per\_day,   
 x='Order\_date',   
 y='Quantity',   
 title='Quantity Sold over the Last Month'  
 )  
 fig\_quantity.update\_layout(  
 margin\_r=100,  
 )  
 st.plotly\_chart(fig\_quantity)

**5. Deploy online**

Once ready, I deployed the app using Streamlit sharing which provides free hosting for Streamlit apps!

This allows me to share the interactive dashboard with anyone.

**For more about the deployment:**[**https://youtu.be/B0MUXtmSpiA**](https://youtu.be/B0MUXtmSpiA)

**Key Takeaways**

Building this dashboard was a great learning experience:

* Streamlit is an amazing tool for turning analysis into web apps
* Creating interactive dashboards amplifies insights from data
* Python and Pandas are very powerful for preparing and analyzing data
* Styling and theming help improve the presentation and UX
* Sharing these apps enables broader reach of analysis and findings

**I hope this inspires you to try Streamlit as well! Let me know if you have any other questions.**

Thank you for reading…

🔔 Stay up to date by [**following me**](https://medium.com/@mohamedyosef101) now or connecting on [**LinkedIn**](https://linkedin.com/in/mohamedyosef101).

I’ll be sharing more case studies and best practices to help you succeed with data-driven product decisions.

Have a good day,

[Mohamed Yosef](https://medium.com/u/122eca15ad5c?source=post_page-----4f02fe49b470--------------------------------)