Dashboarding

Kunal Khurana

2024-06-03

Table of contents

```
import streamlit as st
import plotly.express as px
import pandas as pd
import os
import warnings
```

```
warnings.filterwarnings('ignore')
```

```
st.set_page_config(page_title="Superstore!!!", page_icon=":bar_chart", layout="wide")
```

```
st.title(" :bar_chart: Superstore sales analysis!!!")
st.markdown('<style>div.block-container{padding-top:2rem;}</style>',unsafe_allow_html=True
```

```
fl = st.file_uploader(":file_folder: Upload a file",type=(["csv","txt","xlsx","xls"]))
```

```
if fl is not None:
    filename = fl.name
    st.write(filename)
    df = pd.read_csv(filename)
else:
    os.chdir(r"E:\dashboarding")
    df = pd.read_csv("Sample - Superstore.csv", encoding = "ISO-8859-1")
```

```
col1, col2 = st.columns((2))
df["Order Date"] = pd.to_datetime(df["Order Date"])
```

Getting the min and max date

```
startDate = pd.to_datetime(df["Order Date"]).min()
endDate = pd.to_datetime(df["Order Date"]).max()
```

```
with col1:
    date1 = pd.to_datetime(st.date_input("Start Date", startDate))
with col2:
    date2 = pd.to_datetime(st.date_input("End Date", endDate))

df = df[(df["Order Date"] >= date1) & (df["Order Date"] <= date2)].copy()
```

Creating a side filter based on group and series_title

```
st.sidebar.header("Choose your filter: ")
# Create for Region
region = st.sidebar.multiselect("Pick your Region", df["Region"].unique())
if not region:
    df2 = df.copy()
else:
    df2 = df[df["Region"].isin(region)]


# Create for State
state = st.sidebar.multiselect("Pick the State", df2["State"].unique())
if not state:
    df3 = df2.copy()
else:
    df3 = df2[df2["State"].isin(state)]

# Create for City

city = st.sidebar.multiselect("Pick the City",df3["City"].unique())
```

Filter the data based on Region, State and City

```
if not region and not state and not city:
    filtered_df = df
elif not state and not city:
    filtered_df = df[df["Region"].isin(region)]
elif not region and not city:
    filtered_df = df[df["State"].isin(state)]
elif state and city:
    filtered_df = df3[df["State"].isin(state) & df3["City"].isin(city)]
elif region and city:
    filtered_df = df3[df["Region"].isin(region) & df3["City"].isin(city)]
elif region and state:
    filtered_df = df3[df["Region"].isin(region) & df3["State"].isin(state)]
elif city:
    filtered_df = df3[df3["City"].isin(city)]
else:
    filtered_df = df3[df3["Region"].isin(region) & df3["State"].isin(state) & df3["City"].

category_df = filtered_df.groupby(by = ["Category"], as_index = False)["Sales"].sum()
```

```
with col1:
    st.subheader("Category wise Sales")
    fig = px.bar(category_df, x = "Category", y = "Sales", text = ['${:,.2f}'.format(x) fo
                 template = "seaborn")
    st.plotly_chart(fig,use_container_width=True, height = 200)
```

```
with col2:
    st.subheader("Region wise Sales")
    fig = px.pie(filtered_df, values = "Sales", names = "Region", hole = 0.5)
    fig.update_traces(text = filtered_df["Region"], textposition = "outside")
    st.plotly_chart(fig,use_container_width=True)
```

```
cl1, cl2 = st.columns((2))
with cl1:
    with st.expander("Category_ViewData"):
        st.write(category_df.style.background_gradient(cmap="Blues"))
        csv = category_df.to_csv(index = False).encode('utf-8')
        st.download_button("Download Data", data = csv, file_name = "Category.csv", mime =
                            help = 'Click here to download the data as a CSV file')
```

```
with cl2:
    with st.expander("Region_ViewData"):
        region = filtered_df.groupby(by = "Region", as_index = False)["Sales"].sum()
        st.write(region.style.background_gradient(cmap="Oranges"))
        csv = region.to_csv(index = False).encode('utf-8')
        st.download_button("Download Data", data = csv, file_name = "Region.csv", mime = "
                        help = 'Click here to download the data as a CSV file')


filtered_df["month_year"] = filtered_df["Order Date"].dt.to_period("M")
st.subheader("Time Series Analysis")
```

```
linechart = pd.DataFrame(filtered_df.groupby(filtered_df["month_year"].dt.strftime("%Y : %
fig2 = px.line(linechart, x = "month_year", y = "Sales", labels = {"Sales" : "Amount"}, he
st.plotly_chart(fig2, use_container_width=True)
```

```
with st.expander("View Data of TimeSeries:"):
    st.write(linechart.T.style.background_gradient(cmap="Blues"))
    csv = linechart.to_csv(index=False).encode("utf-8")
    st.download_button('Download Data', data= csv, file_name= 'TimeSeries.csv', mime='text
```

Create a tree map based on Tegion, category, sub-category for navigation

```
st.subheader("Hierarchical view of Sales using TreeMap")
fig3 = px.treemap(filtered_df, path=["Region", "Category", "Sub-Category"], values= "Sales
                  color="Sub-Category")
fig3.update_layout(width = 1200, height = 650)
st.plotly_chart(fig3, use_contatiner_widht= True)
```

```
chart1, chart2 = st.columns((2))
with chart1:
    st.subheader("Segment wise sales")
    fig = px.pie(filtered_df, values = "Sales", names = "Segment", template = "plotly_dark
    fig.update_traces(text = filtered_df["Segment"], textposition = "inside")
    st.plotly_chart(fig, use_container_width=True)
```

```
with chart2:
    st.subheader("Category wise sales")
    fig = px.pie(filtered_df, values = "Sales", names = "Category", template = "gridon")
    fig.update_traces(text = filtered_df["Category"], textposition = "inside")
    st.plotly_chart(fig, use_container_width=True)
```

```
import plotly.figure_factory as ff
st.subheader(":point_right: Month wise Sub-Category Sales Summary")
with st.expander ("Summary_Table"):
    df_sample = df[0:5][["Region", "State", "City", "Category", "Sales", "Profit", "Quanti
    fig = ff.create_table(df_sample, colorscale = "cividis")
    st.plotly_chart(fig, use_container_width=True)

    st.markdown("Month wise sub-Category Table")
    filtered_df["month"] = filtered_df["Order Date"].dt.month_name()
    sub_category_Year = pd.pivot_table(data = filtered_df, values = "Sales", index=["Sub-C
                                   columns="month")
    st.write(sub_category_Year.style.background_gradient(cmap="Blues"))


# create a scatter plot
data1 = px.scatter(filtered_df, x= "Sales", y= "Profit", size= "Quantity")
data1['layout'].update(title="Realationship between Sales and Profit using Scatter Plot!",
                       titlefont=dict(size=20), xaxis=dict(title="Sales", titlefont=dict(s
                       yaxis= dict(title="Profit", titlefont= dict(size=19)))
st.plotly_chart(data1, use_container_width= True)
```

Download the whole dataset

```
with st.expander("View Data"):
    st.write(filtered_df.iloc[:500, 1:20:2].style.background_gradient(cmap="Oranges"))
```

Reference

1. Python Interactive Dashboard Development using Streamlit and Plotly