"""

import streamlit as st

st.title('Streamlit Project')

st.header('Ebay cleaned data set')

import pandas as pd

import numpy as np

import matplotlib as mp

import plotly.express as px

ebay = pd.read\_csv("/Users/anani/Data Management/Streamlit individual/ebay\_streamlit\_data.csv")

st.header("Ebay Data")

#Creating a filter to filter through the data set

# First filter: Multiselect for Storage Type

storage\_filter = st.multiselect(

    "Filter by Storage Type",

    options=ebay['Storage Type'].unique(),

    default=[]

)

# Second filter: Multiselect for Brand

brand\_filter = st.multiselect(

    "Filter by Brand",

    options=ebay['Brand'].unique(),

    default=[]

)

# Apply filters

filtered\_ebay = ebay.copy()

if storage\_filter:

    filtered\_ebay = filtered\_ebay[filtered\_ebay['Storage Type'].isin(storage\_filter)]

if brand\_filter:

    filtered\_ebay = filtered\_ebay[filtered\_ebay['Brand'].isin(brand\_filter)]

st.write("Filtered Data:")

st.write(filtered\_ebay)

# streamlit graph

# Count the frequency of each 'Type'

type\_counts = ebay['Type'].value\_counts().reset\_index()

type\_counts.columns = ['Type', 'Frequency']

# Create the bar chart using Streamlit

st.header("Laptop Inventory: Categorized by Type")

st.bar\_chart(type\_counts.set\_index('Type'))

# plotly graph

st.header("Ebay Data: Storage Size vs. Price")

# Define the Plotly scatter chart

fig = px.scatter(

    ebay,

    x="Price",

    y="SSD Capacity (GB)",

    color="Price",

    color\_continuous\_scale="Blues",

    title="Relationship Between SSD Capacity (GB) and Price",

    labels={"Price": "Price (USD)","SSD Capacity (GB)": "Storage Type"},

)

# Create tabs for Streamlit theme and Plotly native theme

tab1, tab2 = st.tabs(["Streamlit theme (default)", "Plotly native theme"])

with tab1:

    st.plotly\_chart(fig, theme="streamlit", use\_container\_width=True)

with tab2:

    st.plotly\_chart(fig, theme=None, use\_container\_width=True)

# Prepare the data for plotting

chart\_data = pd.DataFrame({

    "RAM Size (GB)": ebay["RAM Size (GB)"],

    "SSD Capacity (GB)": ebay["SSD Capacity (GB)"],

    "Hard Drive Capacity (GB)": ebay["Hard Drive Capacity (GB)"],

    "Price": ebay["Price"]

})

#Dynamic plotly graph

# First filter: Type selection

type\_options = ebay['Type'].unique()

selected\_type = st.selectbox("Select a Type to filter:", options=type\_options)

# Filter data based on the selected Type

filtered\_data\_by\_type = ebay[ebay['Type'] == selected\_type]

# Second filter: Brand selection

brand\_options = filtered\_data\_by\_type['Brand'].unique()

selected\_brand = st.multiselect(

    "Select one or more Brands to filter:",

    options=brand\_options,

    default=brand\_options  # Default to show all brands

)

# Filter data based on selected Brand(s)

filtered\_data = filtered\_data\_by\_type[filtered\_data\_by\_type['Brand'].isin(selected\_brand)]

# Create a Plotly scatter plot

fig = px.scatter(

    filtered\_data,

    x="Type",

    y="Price",

    color="Brand",

    size="Price",

    title=f"Price vs. Type for {selected\_type}",

    labels={"Type": "Type", "Price": "Price (USD)"},

)

# Customize the layout

fig.update\_layout(

    xaxis\_title="Type",

    yaxis\_title="Price (USD)",

    legend\_title="Brand",

    template="plotly\_white",

)

# Display the chart

st.plotly\_chart(fig, use\_container\_width=True)

# Regression graphs

import numpy as np

import statsmodels.api as sm

import matplotlib.pyplot as plt

# Reloading data set

ebay = pd.read\_csv("/Users/anani/Data Management/Streamlit individual/ebay\_streamlit\_data.csv")

# Keep only the specified columns

columns\_to\_keep = ['Brand', 'Type', 'Storage Type', 'RAM Size (GB)', 'SSD Capacity (GB)', 'Hard Drive Capacity (GB)', 'Screen Size (in)', 'Processor Speed (GHz)', 'Price']

filtered\_ebay = ebay[columns\_to\_keep]

# Prepare the regression model (ensure the data is available)

X\_new = filtered\_ebay.drop(columns=['Price', 'Brand', 'Type', 'Storage Type', 'RAM Size (GB)', 'Hard Drive Capacity (GB)'])

X\_new = sm.add\_constant(X\_new)

y\_new = filtered\_ebay['Price']

# Apply log transformation to the dependent variable

logy\_new = np.log(y\_new)

# Fit the model

logreg2\_new = sm.OLS(logy\_new, X\_new).fit()

# Extract key regression metrics

st.subheader("Regression Summary (Cleaned)")

# R-squared and Adjusted R-squared

r\_squared = logreg2\_new.rsquared

adj\_r\_squared = logreg2\_new.rsquared\_adj

# Coefficients and p-values

coef\_table = logreg2\_new.summary2().tables[1][["Coef.", "P>|t|"]]

# Model significance

f\_stat = logreg2\_new.fvalue

f\_pvalue = logreg2\_new.f\_pvalue

# Display key metrics

st.write("\*\*Model Performance Metrics\*\*")

st.write(f"R-squared: {r\_squared:.4f}")

st.write(f"Adjusted R-squared: {adj\_r\_squared:.4f}")

st.write(f"F-statistic: {f\_stat:.4f} (p-value: {f\_pvalue:.4g})")

# Display coefficients in a table

st.write("\*\*Coefficients and P-values\*\*")

st.table(coef\_table)

# Calculate residuals

residuals = logreg2\_new.resid

# Generate Q-Q Plot

st.subheader("Q-Q Plot")

fig1 = sm.qqplot(residuals, fit=True, line='45')

st.pyplot(fig1)

# Residuals vs. Fitted Plot

st.subheader("Residuals vs. Fitted Values")

fig2, ax = plt.subplots(figsize=(10, 6))

ax.scatter(logreg2\_new.fittedvalues, residuals)

ax.set\_xlabel('Fitted Values')

ax.set\_ylabel('Residuals')

ax.set\_title('Residuals vs. Fitted Values')

ax.axhline(y=0, color='r', linestyle='--')  # Add a horizontal line at y=0

ax.grid(True)

st.pyplot(fig2)

"""