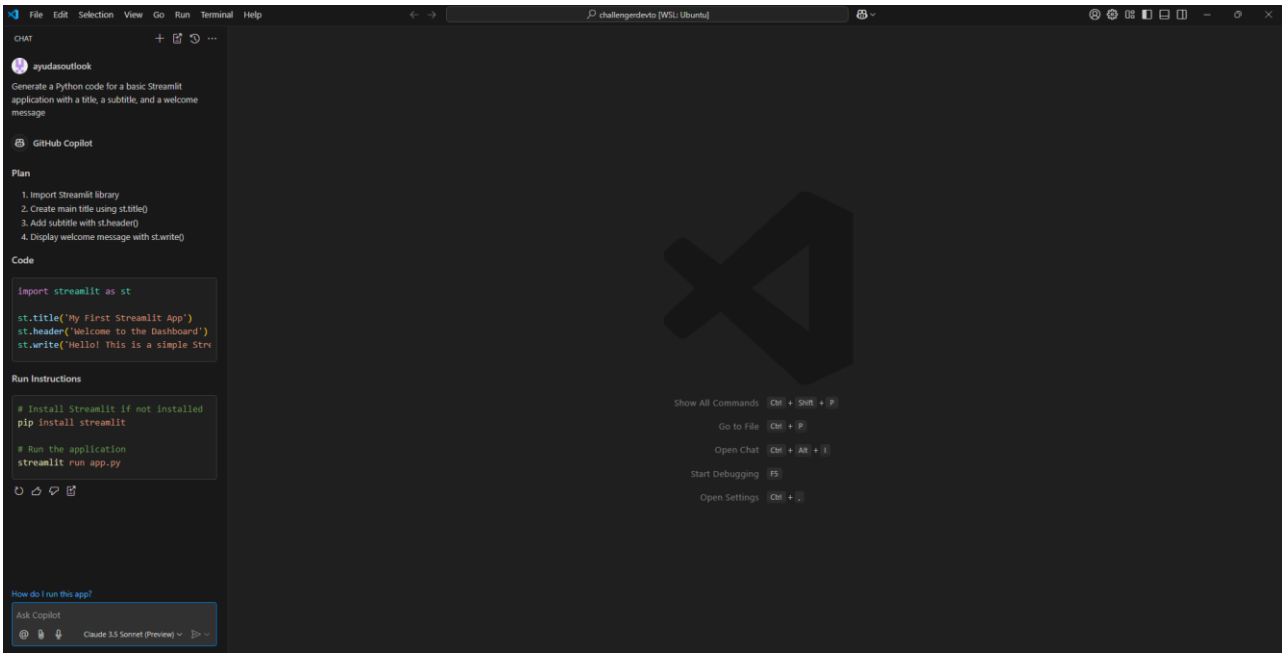
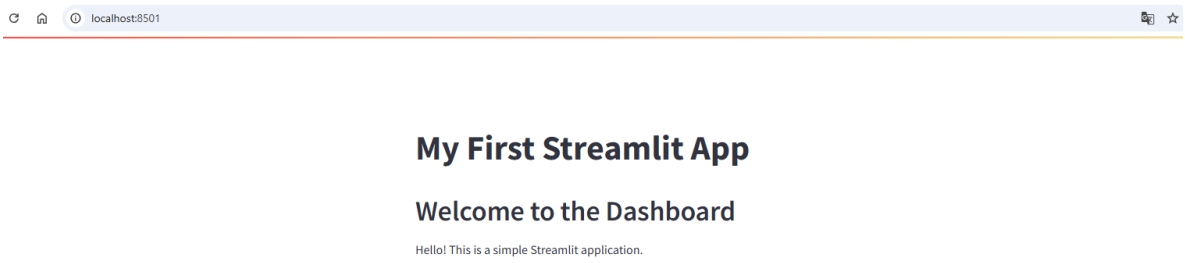

File Edit Selection View Go Run Terminal Help
challengerdevto [WSL: Ubuntu]
CHAT
ayudasoutlook
Generate a Python code for a basic Streamlit application with a title, a subtitle, and a welcome message
GitHub Copilot
Plan
1. Import Streamlit library
2. Create main title using st.title()
3. Add subtitle with st.header()
4. Display welcome message with st.write()
Code
import streamlit as st
st.title('My First Streamlit App')
st.header('Welcome to the Dashboard')
st.write('Hello! This is a simple Str
Run Instructions
Install Streamlit if not installed
pip install streamlit
Run the application
streamlit run app.py
How do I run this app?
Ask Copilot
Claude 3.5 Sonnet (Preview)
Show All Commands Ctrl + Shift + P
Go to File Ctrl + P
Open Chat Ctrl + Alt + I
Start Debugging F5
Open Settings Ctrl + ,


localhost:8501
My First Streamlit App
Welcome to the Dashboard
Hello! This is a simple Streamlit application.

ayudasoutlook

Create a function in Streamlit to upload a CSV or Excel file using a file uploader widget. Display a preview of the first rows of the uploaded file.

GitHub Copilot

Plan

1. Create a function to handle file upload
2. Support both CSV and Excel formats
3. Add file uploader widget
4. Add file type validation
5. Load data using pandas
6. Display preview of first rows
7. Add error handling

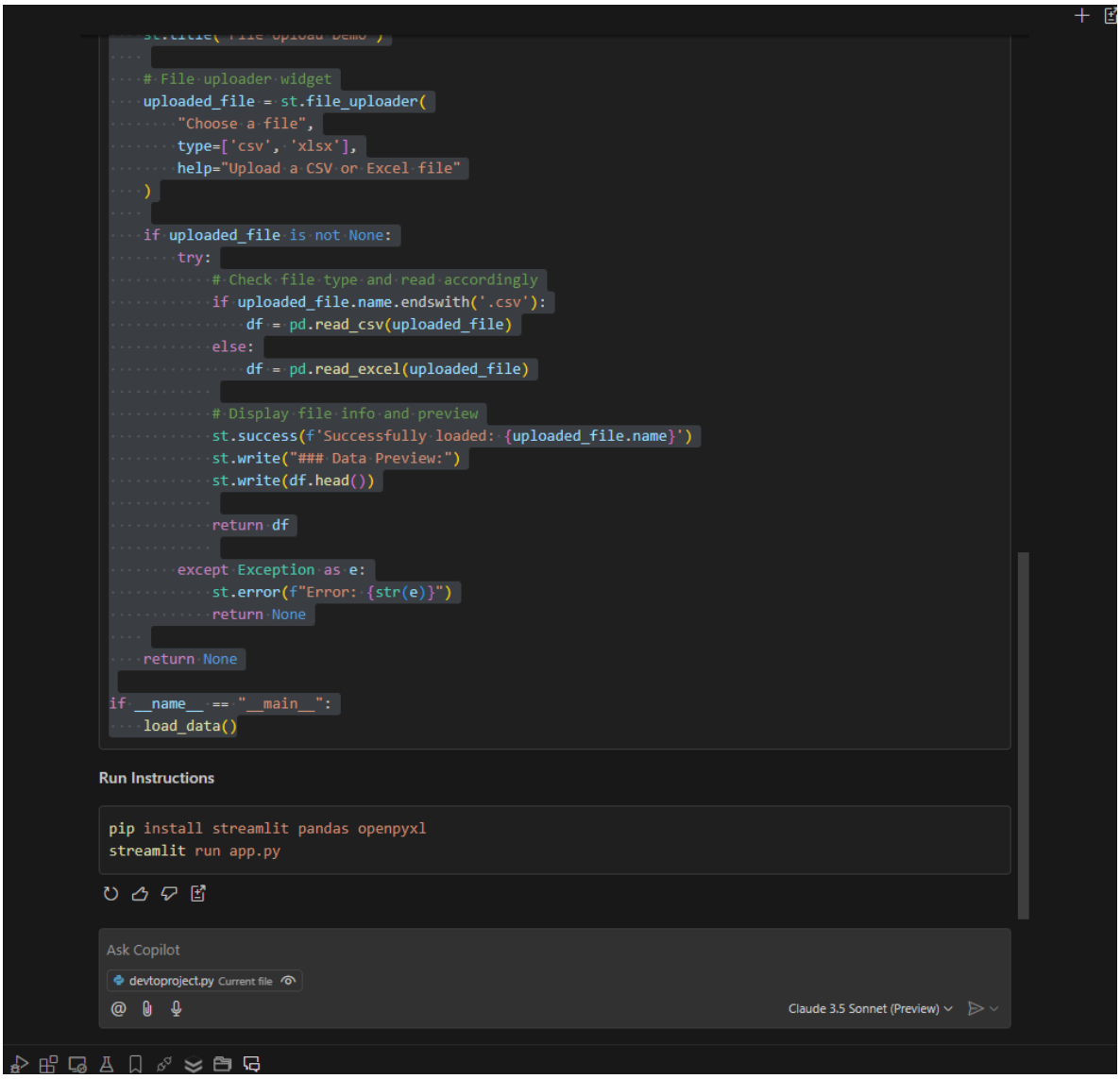
Code

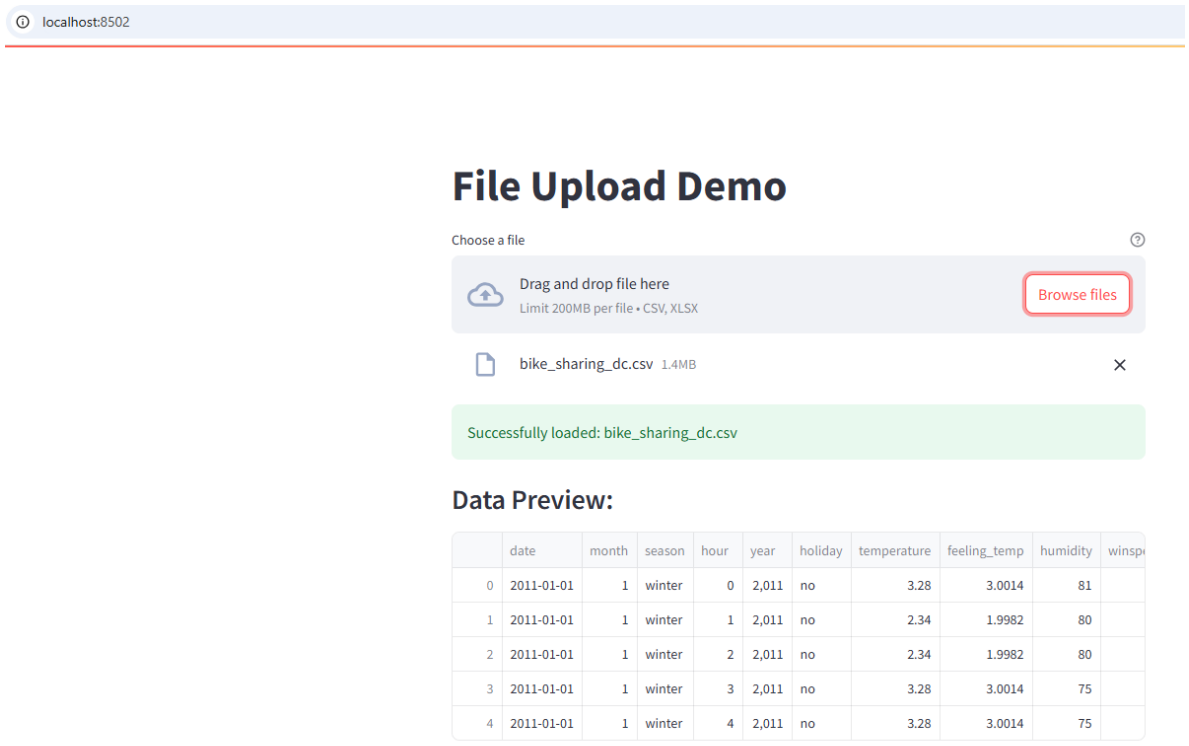
```python
import streamlit as st
import pandas as pd

def load_data():
    st.title('File Upload Demo')

    # File uploader widget
    uploaded_file = st.file_uploader(
        "Choose a file",
        type=['csv', 'xlsx'],
        help="Upload a CSV or Excel file"
    )

    if uploaded_file is not None:
        try:
            # Check file type and read accordingly
            if uploaded_file.name.endswith('.csv'):
                df = pd.read_csv(uploaded_file)
            else:
                df = pd.read_excel(uploaded_file)
```

Ask Copilot

devtoproject.py Current file

Claude 3.5 Sonnet (Preview)


st.title("File Upload Demo")
# File uploader widget
uploaded_file = st.file_uploader(
"Choose a file",
type=['csv', 'xlsx'],
help="Upload a CSV or Excel file"
)
if uploaded_file is not None:
try:
# Check file type and read accordingly
if uploaded_file.name.endswith('.csv'):
df = pd.read_csv(uploaded_file)
else:
df = pd.read_excel(uploaded_file)
# Display file info and preview
st.success(f'Successfully loaded: {uploaded_file.name}')
st.write("### Data Preview:")
st.write(df.head())
return df
except Exception as e:
st.error(f"Error: {str(e)}")
return None
return None
if __name__ == "__main__":
load_data()
Run Instructions
pip install streamlit pandas openpyxl
streamlit run app.py
Ask Copilot
devtoproject.py Current file
Claude 3.5 Sonnet (Preview)



	date	month	season	hour	year	holiday	temperature	feeling_temp	humidity	winsp
0	2011-01-01	1	winter	0	2,011	no	3.28	3.0014	81	
1	2011-01-01	1	winter	1	2,011	no	2.34	1.9982	80	
2	2011-01-01	1	winter	2	2,011	no	2.34	1.9982	80	
3	2011-01-01	1	winter	3	2,011	no	3.28	3.0014	75	
4	2011-01-01	1	winter	4	2,011	no	3.28	3.0014	75	

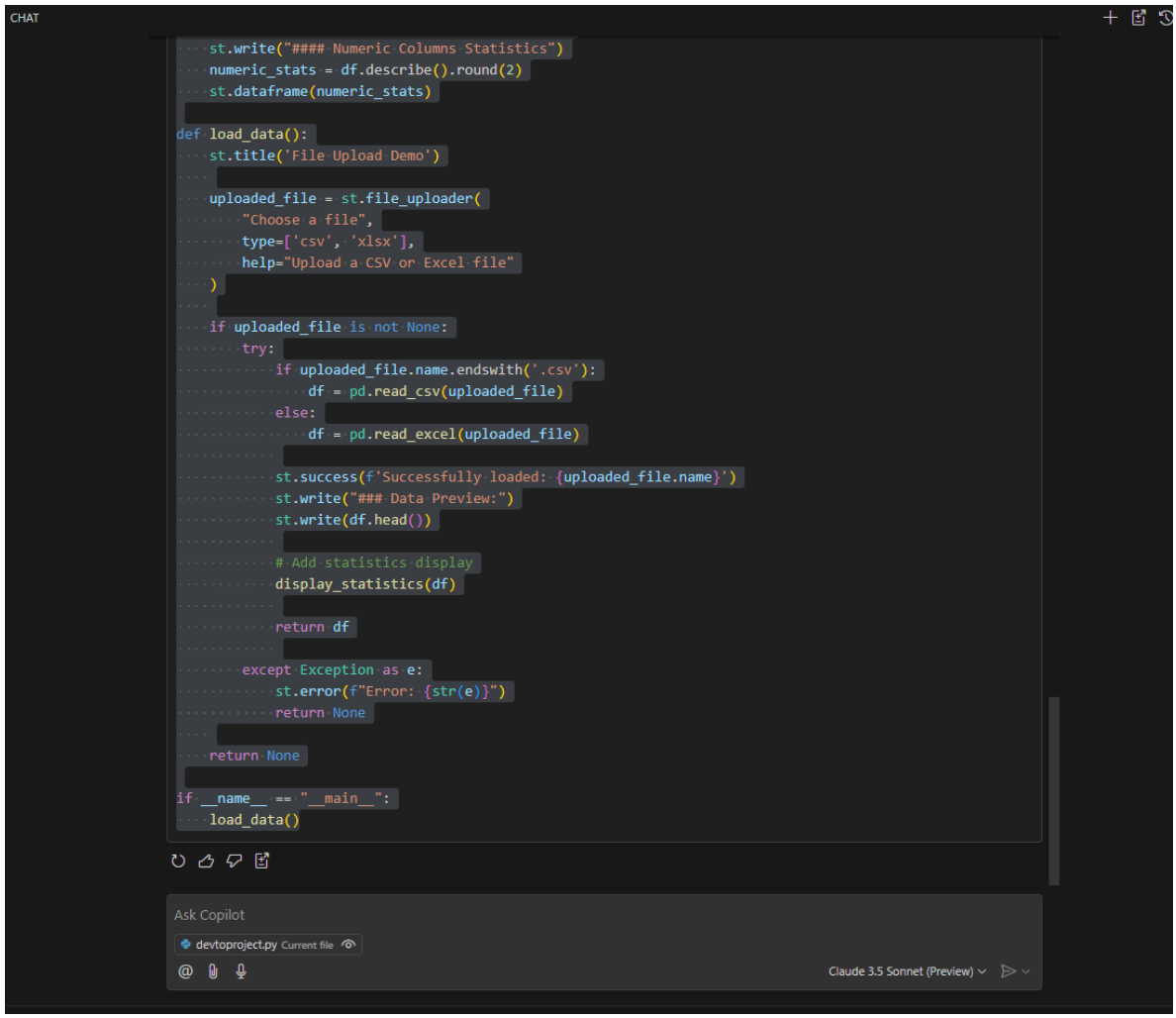
Plan

1. Create a function to display data statistics
2. Calculate basic dataset info (rows, columns, null values)
3. Generate descriptive statistics for numeric columns
4. Display metrics in organized sections using Streamlit components
5. Add this functionality to the existing load_data function

```python
import streamlit as st
import pandas as pd

def display_statistics(df):
    st.write("### Dataset Statistics")

    # Basic dataset info
    col1, col2, col3 = st.columns(3)
    with col1:
        st.metric("Rows", df.shape[0])
    with col2:
        st.metric("Columns", df.shape[1])
    with col3:
        st.metric("Total Null Values", df.isnull().sum().sum())

    # Null values by column
    st.write("#### Null Values by Column")
    null_df = pd.DataFrame({
        'Column': df.columns,
        'Null Count': df.isnull().sum(),
        'Null Percentage': (df.isnull().sum() / len(df) * 100).round(2)
    })
    st.dataframe(null_df)

    # Descriptive statistics for numeric columns
    st.write("#### Numeric Columns Statistics")
    numeric_stats = df.describe().round(2)
    st.dataframe(numeric_stats)

def load_data():
```

Ask Copilot

devtoproject.py Current file


CHAT
st.write("#### Numeric Columns Statistics")
numeric_stats = df.describe().round(2)
st.dataframe(numeric_stats)
def load_data():
st.title('File Upload Demo')
uploaded_file = st.file_uploader(
"Choose a file",
type=['csv', 'xlsx'],
help="Upload a CSV or Excel file"
)
if uploaded_file is not None:
try:
if uploaded_file.name.endswith('.csv'):
df = pd.read_csv(uploaded_file)
else:
df = pd.read_excel(uploaded_file)
st.success(f'Successfully loaded: {uploaded_file.name}')
st.write("### Data Preview:")
st.write(df.head())
# Add statistics display
display_statistics(df)
return df
except Exception as e:
st.error(f"Error: {str(e)}")
return None
return None
if __name__ == "__main__":
load_data()
Ask Copilot
devtoproject.py Current file
Claude 3.5 Sonnet (Preview)

Deploy

File Upload Demo

Choose a file

Drag and drop file here
Limit 200MB per file • CSV, XLSX

Browse files

bike_sharing_dc.csv 1.4MB

Successfully loaded: bike_sharing_dc.csv

Data Preview:

	date	month	season	hour	year	holiday	temperature	feeling_temp	humidity	winsp
0	2011-01-01	1	winter	0	2,011	no	3.28	3.0014	81	
1	2011-01-01	1	winter	1	2,011	no	2.34	1.9982	80	
2	2011-01-01	1	winter	2	2,011	no	2.34	1.9982	80	
3	2011-01-01	1	winter	3	2,011	no	3.28	3.0014	75	
4	2011-01-01	1	winter	4	2,011	no	3.28	3.0014	75	

Dataset Statistics

Rows
17379

Columns
16

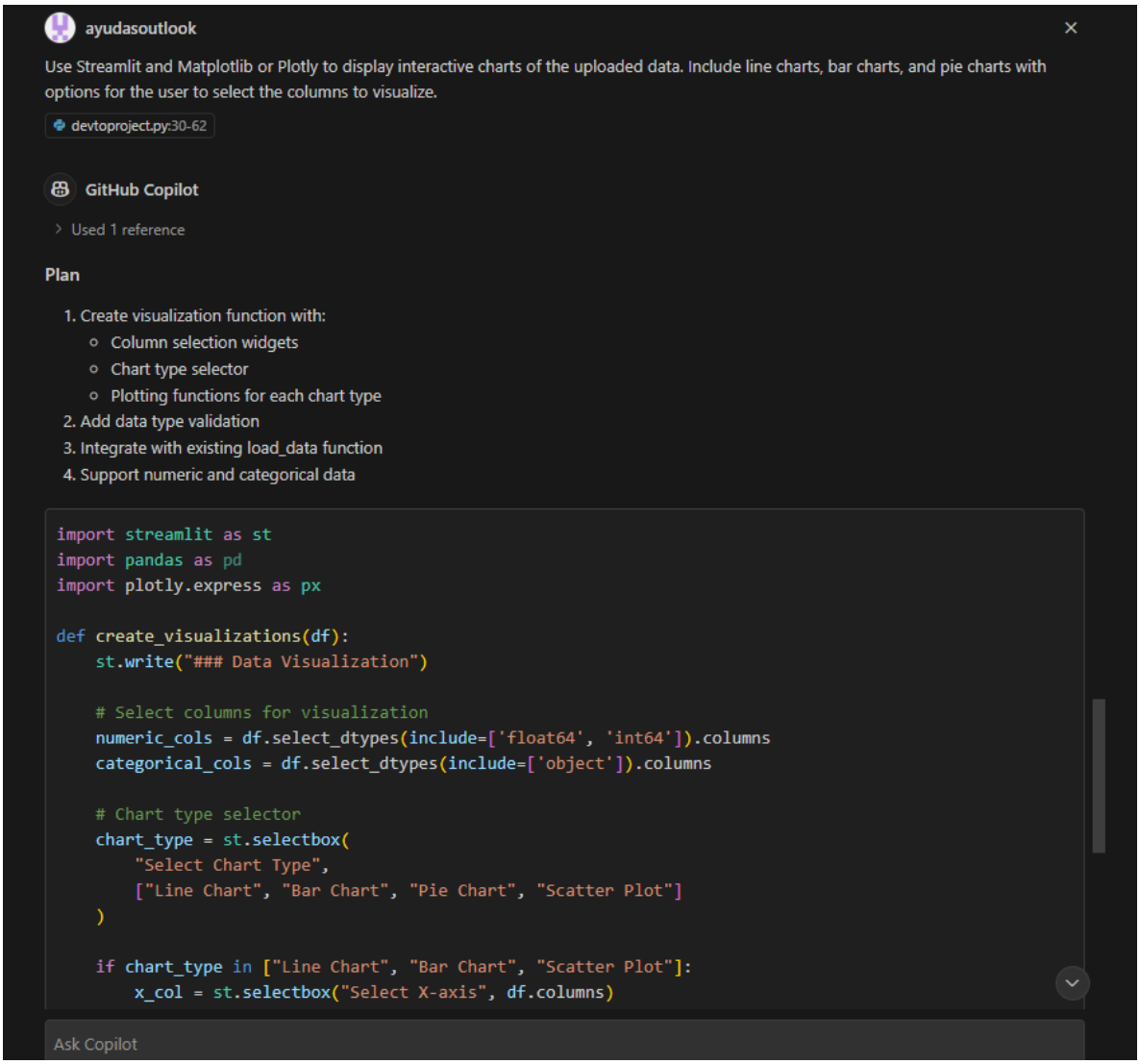
Total Null Values
0

Null Values by Column

	Column	Null Count	Null Percentage
date	date	0	0
month	month	0	0
season	season	0	0
hour	hour	0	0
year	year	0	0
holiday	holiday	0	0
temperature	temperature	0	0
feeling_temp	feeling_temp	0	0
humidity	humidity	0	0
winspeed	winspeed	0	0

Numeric Columns Statistics

	month	hour	year	temperature	feeling_temp	humidity	winspeed	casual	registered	cou
count	17,379	17,379	17,379	17,379	17,379	17,379	17,379	17,379	17,379	17,
mean	6.54	11.55	2,011.5	15.36	15.4	62.72	12.74	35.68	153.79	189
std	3.44	6.91	0.5	9.05	11.34	19.29	8.2	49.31	151.36	181
min	1	0	2,011	-7.06	-16	0	0	0	0	
25%	4	6	2,011	7.98	6	48	7	4	34	
50%	7	12	2,012	15.5	16	63	13	17	115	
75%	10	18	2,012	23.02	25	78	17	48	220	
max	12	23	2,012	39	50	100	57	367	886	


ayudasoutlook
Use Streamlit and Matplotlib or Plotly to display interactive charts of the uploaded data. Include line charts, bar charts, and pie charts with options for the user to select the columns to visualize.
devtoproject.py:30-62
GitHub Copilot
Used 1 reference
Plan
1. Create visualization function with:
Column selection widgets
Chart type selector
Plotting functions for each chart type
2. Add data type validation
3. Integrate with existing load_data function
4. Support numeric and categorical data
import streamlit as st
import pandas as pd
import plotly.express as px
def create_visualizations(df):
st.write("### Data Visualization")
# Select columns for visualization
numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns
categorical_cols = df.select_dtypes(include=['object']).columns
# Chart type selector
chart_type = st.selectbox(
"Select Chart Type",
["Line Chart", "Bar Chart", "Pie Chart", "Scatter Plot"]
)
if chart_type in ["Line Chart", "Bar Chart", "Scatter Plot"]:
x_col = st.selectbox("Select X-axis", df.columns)
Ask Copilot

```python
    chart_type = st.selectbox(
        "Select Chart Type",
        ["Line Chart", "Bar Chart", "Pie Chart", "Scatter Plot"]
    )

    if chart_type in ["Line Chart", "Bar Chart", "Scatter Plot"]:
        x_col = st.selectbox("Select X-axis", df.columns)
        y_col = st.selectbox("Select Y-axis", numeric_cols)

        if chart_type == "Line Chart":
            fig = px.line(df, x=x_col, y=y_col, title=f'{y_col} vs {x_col}')
        elif chart_type == "Bar Chart":
            fig = px.bar(df, x=x_col, y=y_col, title=f'{y_col} by {x_col}')
        else:  # Scatter Plot
            fig = px.scatter(df, x=x_col, y=y_col, title=f'{y_col} vs {x_col}')

    elif chart_type == "Pie Chart":
        value_col = st.selectbox("Select Value Column", numeric_cols)
        name_col = st.selectbox("Select Name Column", categorical_cols)
        fig = px.pie(df, values=value_col, names=name_col, title=f'{value_col} Distribution')

    st.plotly_chart(fig)

def load_data():
    ...existing code...
            st.success(f'Successfully loaded: {uploaded_file.name}')
            st.write("### Data Preview:")
            st.write(df.head())

            # Add statistics display
            display_statistics(df)

            # Add visualizations
            create_visualizations(df)

            return df
    ...existing code...

if __name__ == "__main__":
    load_data()
```

Ask Copilot

Deploy

File Upload Demo

Choose a file

Drag and drop file here
Limit 200MB per file • CSV, XLSX

Browse files

bike_sharing_dc.csv 1.4MB

Successfully loaded: bike_sharing_dc.csv

Data Preview:

	date	month	season	hour	year	holiday	temperature	feeling_temp	humidity	winsp
0	2011-01-01	1	winter	0	2,011	no	3.28	3.0014	81	
1	2011-01-01	1	winter	1	2,011	no	2.34	1.9982	80	
2	2011-01-01	1	winter	2	2,011	no	2.34	1.9982	80	
3	2011-01-01	1	winter	3	2,011	no	3.28	3.0014	75	
4	2011-01-01	1	winter	4	2,011	no	3.28	3.0014	75	

Dataset Statistics

Rows
17379

Columns
16

Total Null Values
0

Null Values by Column

	Column	Null Count	Null Percentage
date	date	0	0
month	month	0	0
season	season	0	0
hour	hour	0	0
year	year	0	0
holiday	holiday	0	0
temperature	temperature	0	0
feeling_temp	feeling_temp	0	0
humidity	humidity	0	0
winspeed	winspeed	0	0

Numeric Columns Statistics

	month	hour	year	temperature	feeling_temp	humidity	winspeed	casual	registered	cou
count	17,379	17,379	17,379	17,379	17,379	17,379	17,379	17,379	17,379	17,
mean	6.54	11.55	2,011.5	15.36	15.4	62.72	12.74	35.68	153.79	189
std	3.44	6.91	0.5	9.05	11.34	19.29	8.2	49.31	151.36	181
min	1	0	2,011	-7.06	-16	0	0	0	0	
25%	4	6	2,011	7.98	6	48	7	4	34	
50%	7	12	2,012	15.5	16	63	13	17	115	
75%	10	18	2,012	23.02	25	78	17	48	220	
max	12	23	2,012	39	50	100	57	367	886	

Data Visualization

Select Chart Type

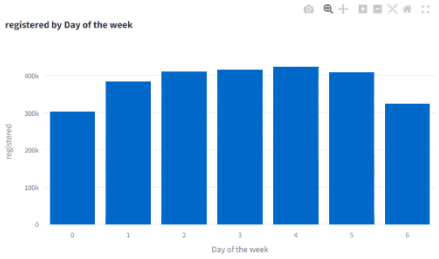
Bar Chart

Select X-axis

Day of the week

Select Y-axis

registered



ayudasoutlook

Create Streamlit widgets like selectbox, sliders, or multiselect to allow users to filter the uploaded data by categories, date ranges, or numeric values. Display the filtered table and update the charts dynamically.

devtoproject.py:73-98

GitHub Copilot

Used 1 reference

Plan

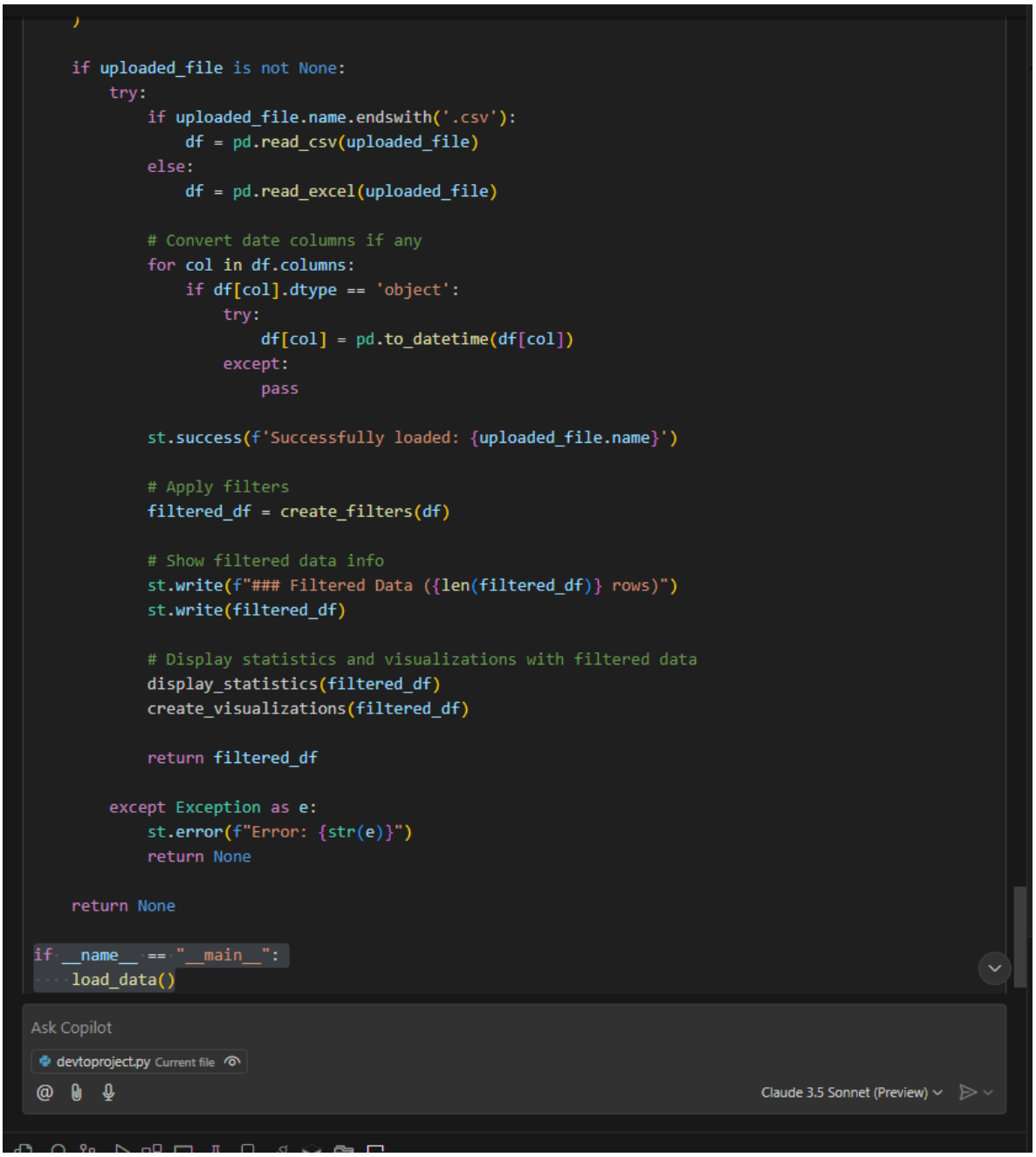
1. Create function for data filtering with:
   - Filter types based on column data types
   - Numeric range sliders
   - Category multiselect
   - Date range selector
2. Add dynamic table updates
3. Connect filters to visualizations
4. Handle edge cases and data validation
5. Update main flow to include filtering

```python
import streamlit as st
import pandas as pd
from datetime import datetime

def create_filters(df):
    st.sidebar.write("### Data Filters")
    filtered_df = df.copy()

    # Filter by categorical columns
    for col in df.select_dtypes(include=['object']).columns:
        unique_values = df[col].unique()
        if len(unique_values) < 50:  # Only show if manageable number of categories
            selected_values = st.sidebar.multiselect(
                f'Select {col}',
                unique_values,
                default=unique_values
```

Ask Copilot

CHAT

```python
    # Filter by numeric columns
    for col in df.select_dtypes(include=['float64', 'int64']).columns:
        min_val = float(df[col].min())
        max_val = float(df[col].max())
        range_vals = st.sidebar.slider(
            f'Range for {col}',
            min_val, max_val,
            (min_val, max_val)
        )
        filtered_df = filtered_df[
            (filtered_df[col] >= range_vals[0]) &
            (filtered_df[col] <= range_vals[1])
        ]

    # Filter by date columns if any
    for col in df.select_dtypes(include=['datetime64']).columns:
        min_date = df[col].min()
        max_date = df[col].max()
        date_range = st.sidebar.date_input(
            f'Range for {col}',
            [min_date, max_date]
        )
        if len(date_range) == 2:
            filtered_df = filtered_df[
                (filtered_df[col].dt.date >= date_range[0]) &
                (filtered_df[col].dt.date <= date_range[1])
            ]

    return filtered_df

def load_data():
    st.title('Data Analysis Dashboard')

    uploaded_file = st.file_uploader(
        "Choose a file",
        type=['csv', 'xlsx'],
        help="Upload a CSV or Excel file"
    )

    if uploaded_file is not None:
```



```
    )

    if uploaded_file is not None:
        try:
            if uploaded_file.name.endswith('.csv'):
                df = pd.read_csv(uploaded_file)
            else:
                df = pd.read_excel(uploaded_file)

            # Convert date columns if any
            for col in df.columns:
                if df[col].dtype == 'object':
                    try:
                        df[col] = pd.to_datetime(df[col])
                    except:
                        pass

            st.success(f'Successfully loaded: {uploaded_file.name}')

            # Apply filters
            filtered_df = create_filters(df)

            # Show filtered data info
            st.write(f"### Filtered Data ({len(filtered_df)} rows)")
            st.write(filtered_df)

            # Display statistics and visualizations with filtered data
            display_statistics(filtered_df)
            create_visualizations(filtered_df)

            return filtered_df

        except Exception as e:
            st.error(f"Error: {str(e)}")
            return None

    return None

if __name__ == "__main__":
    load_data()
```

Ask Copilot

devtoproject.py Current file

Claude 3.5 Sonnet (Preview)

Deploy

Data Filters

Select season

winter × spring × summer × fall ×

Select holiday

no × yes ×

Select am or pm

am × pm ×

Range for month

1.00 — 12.00

Range for hour

0.00 — 23.00

Range for year

2011.00 — 2012.00

Range for temperature

-7.06 — 39.00

Range for feeling_temp

-16.00 — 50.00

Range for humidity

0.00 — 100.00

Data Analysis Dashboard

Choose a file

Drag and drop file here
Limit 200MB per file • CSV, XLSX

Browse files

bike_sharing_dc.csv 1.4MB

Successfully loaded: bike_sharing_dc.csv

Filtered Data (17379 rows)

	date	month	season	hour	year	holiday	temperature	feeling_temp	humidity
29	2011-01-02 00:00:00	1	winter	6	2,011	no	11.74	11.9972	77
30	2011-01-02 00:00:00	1	winter	7	2,011	no	10.8	11.0006	76
31	2011-01-02 00:00:00	1	winter	8	2,011	no	10.8	11.0006	71
32	2011-01-02 00:00:00	1	winter	9	2,011	no	9.86	9.9974	76
33	2011-01-02 00:00:00	1	winter	10	2,011	no	8.92	7.001	81
34	2011-01-02 00:00:00	1	winter	11	2,011	no	8.92	5.9978	71
35	2011-01-02 00:00:00	1	winter	12	2,011	no	8.92	5.9978	66
36	2011-01-02 00:00:00	1	winter	13	2,011	no	8.92	7.001	66
37	2011-01-02 00:00:00	1	winter	14	2,011	no	8.92	7.001	76
38	2011-01-02 00:00:00	1	winter	15	2,011	no	7.98	5.9978	81

Dataset Statistics

Rows: 17379

Columns: 16

Total Null Values: 0

Null Values by Column

	Column	Null Count	Null Percentage
date	date	0	0
month	month	0	0
season	season	0	0
hour	hour	0	0
year	year	0	0
holiday	holiday	0	0
temperature	temperature	0	0
feeling_temp	feeling_temp	0	0
humidity	humidity	0	0
winspeed	winspeed	0	0

Numeric Columns Statistics

	date	month	hour	year	temperature	feeling_temp	humidity	wins
count	17379	17,379	17,379	17,379	17,379	17,379	17,379	1
mean	2012-01-02 04:08:34.552045568	6.54	11.55	2,011.5	15.36	15.4	62.72	
min	2011-01-01 00:00:00	1	0	2,011	-7.06	-16	0	
25%	2011-07-04 00:00:00	4	6	2,011	7.98	6	48	
50%	2012-01-02 00:00:00	7	12	2,012	15.5	16	63	
75%	2012-07-02 00:00:00	10	18	2,012	23.02	25	78	
max	2012-12-31 00:00:00	12	23	2,012	39	50	100	
std	None	3.44	6.91	0.5	9.05	11.34	19.29	

Data Visualization

Select Chart Type: Line Chart

Select X-axis: date

Select Y-axis: month

month vs date

ayudasoutlook

Add a section to perform advanced analyses, such as correlations between columns, basic linear regression, or clustering using k-means. Use libraries like Scikit-learn or Statsmodels and integrate the results into the application

devtoproject.py:141-154

GitHub Copilot

Used 1 reference

Plan

1. Create advanced analysis function with:
   - Correlation matrix using seaborn
   - Simple linear regression with scikit-learn
   - K-means clustering for numeric data
2. Add visualization of results
3. Allow user to select analysis type
4. Handle data preprocessing
5. Add error handling for non-numeric data

```python
import streamlit as st
import pandas as pd
import seaborn as sns
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import plotly.express as px
import plotly.figure_factory as ff

def perform_advanced_analysis(df):
    st.write("### Advanced Analysis")

    analysis_type = st.selectbox(
        "Select Analysis Type",
        ["Correlation Analysis", "Linear Regression", "K-means Clustering"]
    )
```

CHAT

```
    elif analysis_type == "Linear Regression":
        numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns
        x_col = st.selectbox("Select Independent Variable (X)", numeric_cols)
        y_col = st.selectbox("Select Dependent Variable (Y)", numeric_cols)

        X = df[x_col].values.reshape(-1, 1)
        y = df[y_col].values

        model = LinearRegression()
        model.fit(X, y)

        fig = px.scatter(df, x=x_col, y=y_col, title=f'Linear Regression: {y_col} vs {x_col}')
        fig.add_scatter(x=df[x_col], y=model.predict(X),
                        name='Regression Line',
                        line=dict(color='red'))

        st.plotly_chart(fig)
        st.write(f"R² Score: {model.score(X, y):.4f}")
        st.write(f"Coefficient: {model.coef_[0]:.4f}")
        st.write(f"Intercept: {model.intercept_:.4f}")

    elif analysis_type == "K-means Clustering":
        numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns
        col1 = st.selectbox("Select First Feature", numeric_cols)
        col2 = st.selectbox("Select Second Feature", numeric_cols)
        n_clusters = st.slider("Number of Clusters", 2, 10, 3)

        X = df[[col1, col2]]
        scaler = StandardScaler()
        X_scaled = scaler.fit_transform(X)

        kmeans = KMeans(n_clusters=n_clusters, random_state=42)
        df['Cluster'] = kmeans.fit_predict(X_scaled)

        fig = px.scatter(df, x=col1, y=col2, color='Cluster',
                         title=f'K-means Clustering ({n_clusters} clusters)')
        st.plotly_chart(fig)

def load_data():
```

```
        st.plotly_chart(fig)
        st.write(f"R² Score: {model.score(X, y):.4f}")
        st.write(f"Coefficient: {model.coef_[0]:.4f}")
        st.write(f"Intercept: {model.intercept_:.4f}")

    elif analysis_type == "K-means Clustering":
        numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns
        col1 = st.selectbox("Select First Feature", numeric_cols)
        col2 = st.selectbox("Select Second Feature", numeric_cols)
        n_clusters = st.slider("Number of Clusters", 2, 10, 3)

        X = df[[col1, col2]]
        scaler = StandardScaler()
        X_scaled = scaler.fit_transform(X)

        kmeans = KMeans(n_clusters=n_clusters, random_state=42)
        df['Cluster'] = kmeans.fit_predict(X_scaled)

        fig = px.scatter(df, x=col1, y=col2, color='Cluster',
                         title=f'K-means Clustering ({n_clusters} clusters)')
        st.plotly_chart(fig)

def load_data():
    ...existing code...
            # Display statistics, visualizations, and advanced analysis
            display_statistics(filtered_df)
            create_visualizations(filtered_df)
            perform_advanced_analysis(filtered_df)

            return filtered_df
    ...existing code...
```

Required dependencies:

```
pip install scikit-learn seaborn
```

localhost:8502

Data Filters

Select season

winter × spring × summer × fall ×

Select holiday

no × yes ×

Select am or pm

am × pm ×

Range for month

1.00 12.00

1.00 12.00

Range for hour

0.00 23.00

0.00 23.00

Range for year

2011.00 2012.00

2011.00 2012.00

Range for temperature

-7.06 39.00

-7.06 39.00

Range for feeling_temp

-16.00 50.00

-16.00 50.00

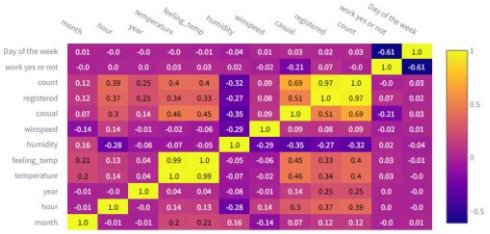
Range for humidity

0.00 100.00

Advanced Analysis

Select Analysis Type

Correlation Analysis



CHAT

Add a button in Streamlit to let users download the processed or filtered data in CSV format.

devtoproject.py:191-222

GitHub Copilot

> Used 1 reference

Plan

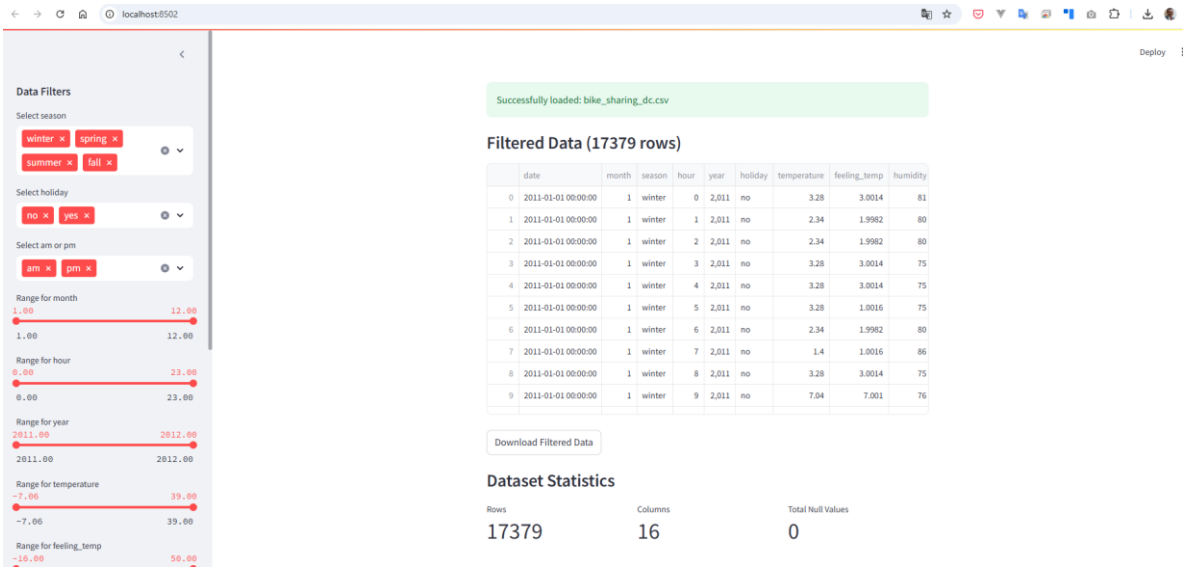
1. Add download functionality:
   - Create download button below filtered data display
   - Convert filtered DataFrame to CSV
   - Use st.download_button for the export
   - Add file naming with timestamp
2. Integrate into existing data flow

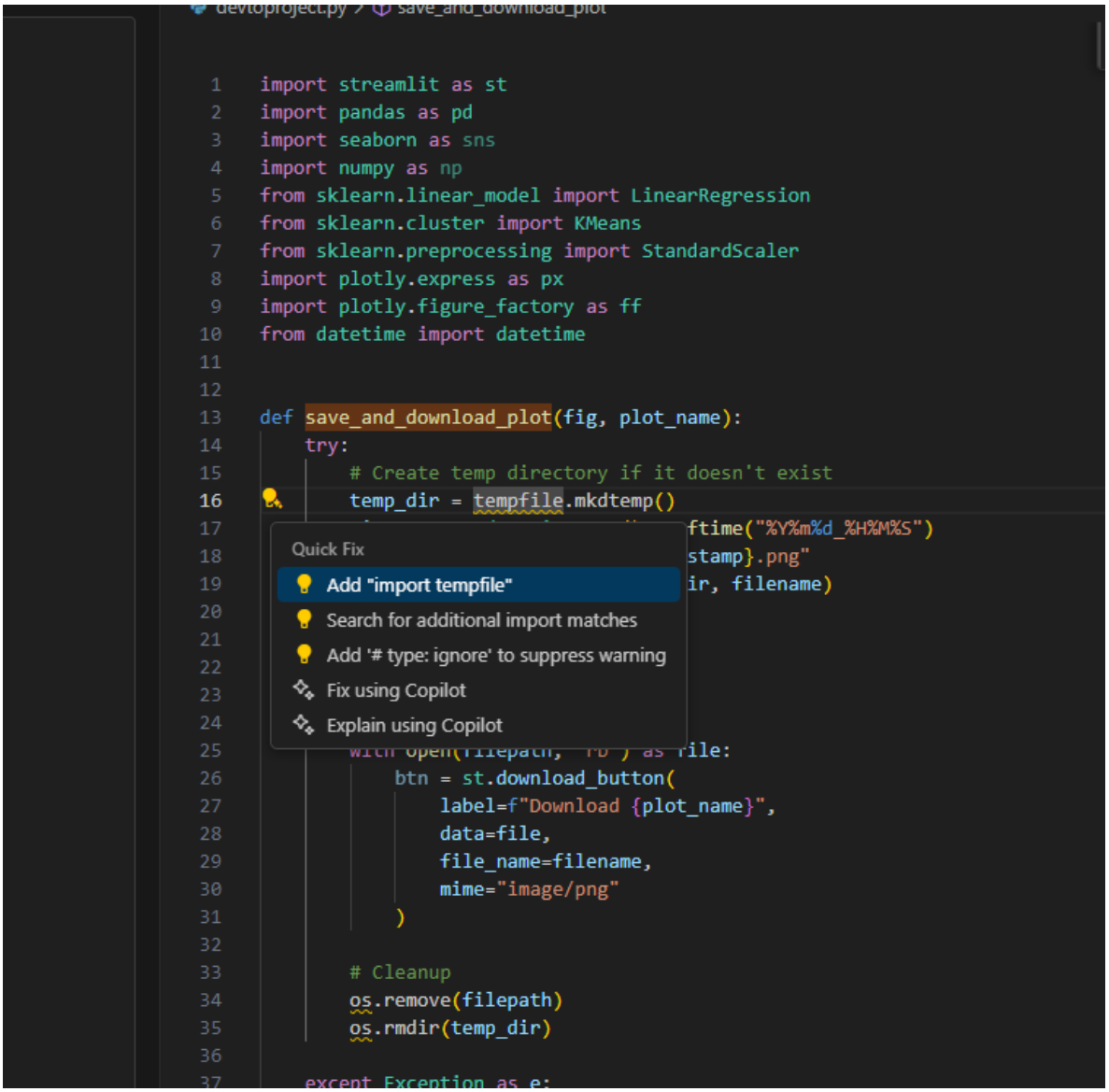
```python
import streamlit as st
import pandas as pd
from datetime import datetime

def load_data():
    ...existing code...
            # Show filtered data info
            st.write(f"### Filtered Data ({len(filtered_df)} rows)")
            st.write(filtered_df)

            # Add download button
            csv = filtered_df.to_csv(index=False)
            timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
            filename = f"filtered_data_{timestamp}.csv"

            st.download_button(
                label="Download Filtered Data",
                data=csv,
                file_name=filename,
                mime="text/csv"
            )

            # Display statistics and visualizations with filtered data
```

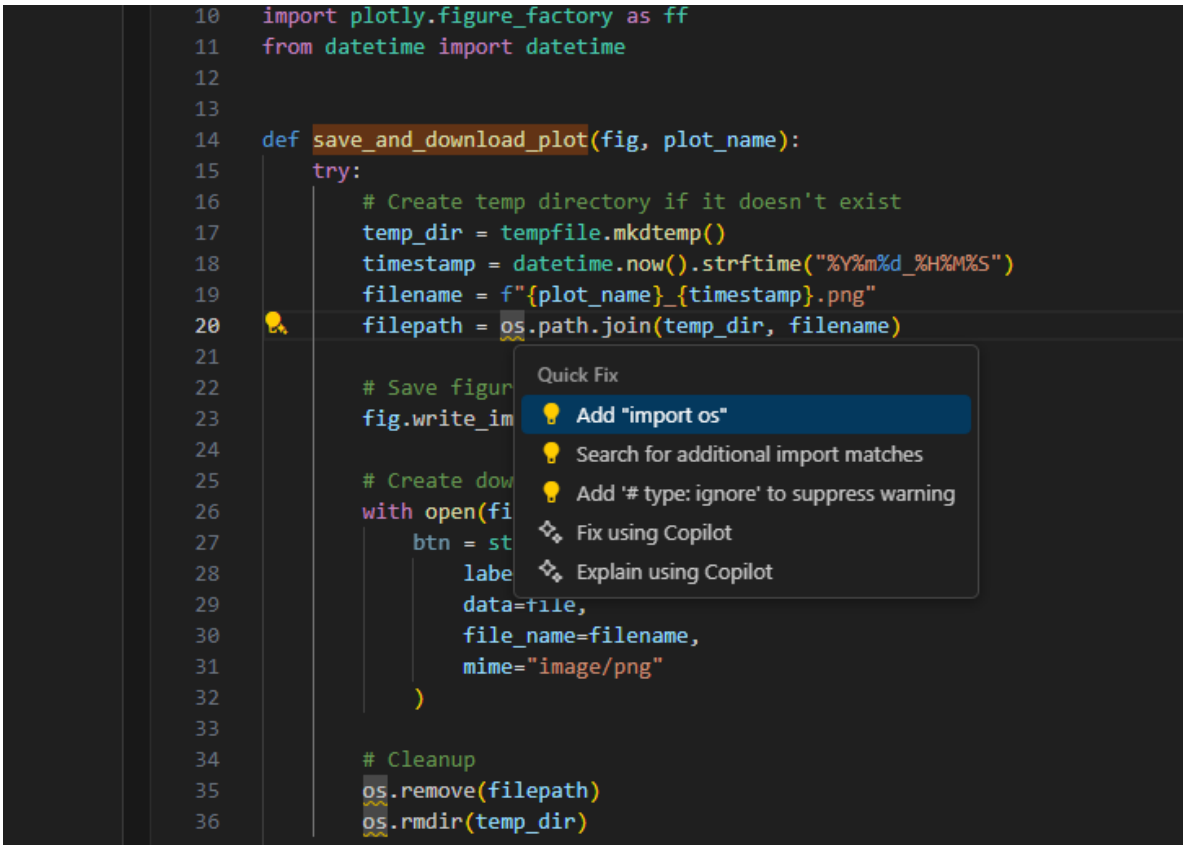
Ask Copilot

devtoproject.py Current file

Claude 3.5 Sonnet (Preview)


localhost:8502
Deploy
Data Filters
Select season
winter
spring
summer
fall
Select holiday
no
yes
Select am or pm
am
pm
Range for month
1.00
12.00
1.00
12.00
Range for hour
0.00
23.00
0.00
23.00
Range for year
2011.00
2012.00
2011.00
2012.00
Range for temperature
-7.06
39.00
-7.06
39.00
Range for feeling_temp
-16.00
50.00
Successfully loaded: bike_sharing_dc.csv
Filtered Data (17379 rows)
date month season hour year holiday temperature feeling_temp humidity
0 2011-01-01 00:00:00 1 winter 0 2,011 no 3.28 3.0014 81
1 2011-01-01 00:00:00 1 winter 1 2,011 no 2.34 1.9982 80
2 2011-01-01 00:00:00 1 winter 2 2,011 no 2.34 1.9982 80
3 2011-01-01 00:00:00 1 winter 3 2,011 no 3.28 3.0014 75
4 2011-01-01 00:00:00 1 winter 4 2,011 no 3.28 3.0014 75
5 2011-01-01 00:00:00 1 winter 5 2,011 no 3.28 1.0016 75
6 2011-01-01 00:00:00 1 winter 6 2,011 no 2.34 1.9982 80
7 2011-01-01 00:00:00 1 winter 7 2,011 no 1.4 1.0016 86
8 2011-01-01 00:00:00 1 winter 8 2,011 no 3.28 3.0014 75
9 2011-01-01 00:00:00 1 winter 9 2,011 no 7.04 7.001 76
Download Filtered Data
Dataset Statistics
Rows
17379
Columns
16
Total Null Values
0


devtoproject.py > save_and_download_plot
import streamlit as st
import pandas as pd
import seaborn as sns
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import plotly.express as px
import plotly.figure_factory as ff
from datetime import datetime
def save_and_download_plot(fig, plot_name):
try:
# Create temp directory if it doesn't exist
temp_dir = tempfile.mkdtemp()
ftime("%Y%m%d_%H%M%S")
stamp}.png"
ir, filename)
Quick Fix
Add "import tempfile"
Search for additional import matches
Add '# type: ignore' to suppress warning
Fix using Copilot
Explain using Copilot
with open(filepath, "rb") as file:
btn = st.download_button(
label=f"Download {plot_name}",
data=file,
file_name=filename,
mime="image/png"
)
# Cleanup
os.remove(filepath)
os.rmdir(temp_dir)
except Exception as e:


import plotly.figure_factory as ff
from datetime import datetime
def save_and_download_plot(fig, plot_name):
try:
# Create temp directory if it doesn't exist
temp_dir = tempfile.mkdtemp()
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
filename = f"{plot_name}_{timestamp}.png"
filepath = os.path.join(temp_dir, filename)
Quick Fix
Add "import os"
Search for additional import matches
Add '# type: ignore' to suppress warning
Fix using Copilot
Explain using Copilot
Save figur
fig.write_im
Create dow
with open(fi
btn = st
labe
data=file,
file_name=filename,
mime="image/png"
)
# Cleanup
os.remove(filepath)
os.rmdir(temp_dir)

ayudasoutlook

Include a function to save the generated charts as image files (PNG or JPG) and add a button in Streamlit to download them.

devtoproject.py:1-27

GitHub Copilot

Used 1 reference

Plan

1. Create a function to save charts:
   - Accept plotly figure object as input
   - Save in PNG format
   - Create temp directory for storage
   - Add download button for each chart
2. Modify visualization functions to include save options
3. Handle file naming and cleanup
4. Add error handling

```python
import streamlit as st
import pandas as pd
import plotly.express as px
import plotly.figure_factory as ff
import os
from datetime import datetime
import tempfile

def save_and_download_plot(fig, plot_name):
    try:
        # Create temp directory if it doesn't exist
        temp_dir = tempfile.mkdtemp()
        timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
        filename = f"{plot_name}_{timestamp}.png"
        filepath = os.path.join(temp_dir, filename)
```

```
            btn = st.download_button(
                label=f"Download {plot_name}",
                data=file,
                file_name=filename,
                mime="image/png"
            )

        # Cleanup
        os.remove(filepath)
        os.rmdir(temp_dir)

    except Exception as e:
        st.error(f"Error saving plot: {str(e)}")

def create_visualizations(df):
    st.write("### Data Visualization")

    # ... existing visualization code ...

    # Add download option for the plot
    st.plotly_chart(fig)
    save_and_download_plot(fig, chart_type.replace(" ", "_").lower())

def perform_advanced_analysis(df):
    # ... existing analysis code ...

    if analysis_type == "Correlation Analysis":
        # ... existing correlation code ...
        st.plotly_chart(fig)
        save_and_download_plot(fig, "correlation_heatmap")

    elif analysis_type == "Linear Regression":
        # ... existing regression code ...
        st.plotly_chart(fig)
        save_and_download_plot(fig, "regression_plot")

    elif analysis_type == "K-means Clustering":
        # ... existing clustering code ...
        st.plotly_chart(fig)
        save_and_download_plot(fig, "clustering_plot")
```

Ask Copilot

CHAT

```
        os.remove(filepath)
        os.rmdir(temp_dir)

    except Exception as e:
        st.error(f"Error saving plot: {str(e)}")

def create_visualizations(df):
    st.write("### Data Visualization")

    # ... existing visualization code ...

    # Add download option for the plot
    st.plotly_chart(fig)
    save_and_download_plot(fig, chart_type.replace(" ", "_").lower())

def perform_advanced_analysis(df):
    # ... existing analysis code ...

    if analysis_type == "Correlation Analysis":
        # ... existing correlation code ...
        st.plotly_chart(fig)
        save_and_download_plot(fig, "correlation_heatmap")

    elif analysis_type == "Linear Regression":
        # ... existing regression code ...
        st.plotly_chart(fig)
        save_and_download_plot(fig, "regression_plot")

    elif analysis_type == "K-means Clustering":
        # ... existing clustering code ...
        st.plotly_chart(fig)
        save_and_download_plot(fig, "clustering_plot")
```

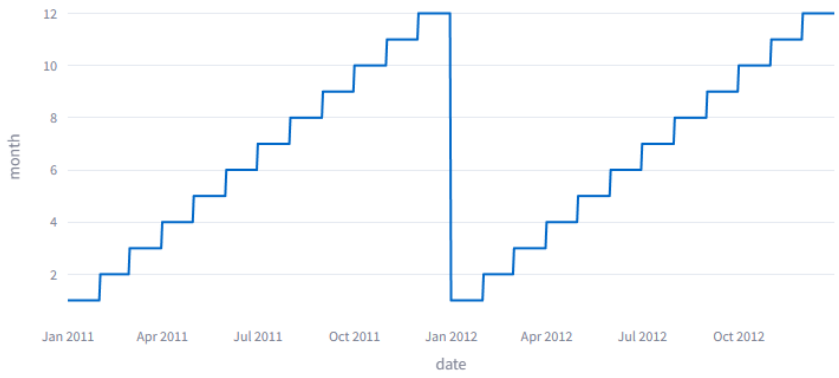
Required dependencies:

```
pip install kaleido
```

Select Y-axis

month

month vs date



Download line_chart

Advanced Analysis

Select Analysis Type

Correlation Analysis

ayudasoutlook

Set up the Streamlit app with multiple sections, such as 'Upload Data,' 'Visualization,' and 'Analysis.' Use st.sidebar to add an interactive navigation menu

devtoproject.py:8-45

GitHub Copilot

Used 1 reference

Plan

1. Structure app with sections:
   - Data Upload & Preview
   - Data Statistics
   - Visualization
   - Advanced Analysis
2. Create sidebar navigation
3. Use session state to store data
4. Add section management functions
5. Implement main app flow

```python
import streamlit as st
import pandas as pd

def main():
    st.set_page_config(page_title="Data Analysis Dashboard", layout="wide")

    # Initialize session state
    if 'data' not in st.session_state:
        st.session_state.data = None

    # Sidebar navigation
```

Ask Copilot

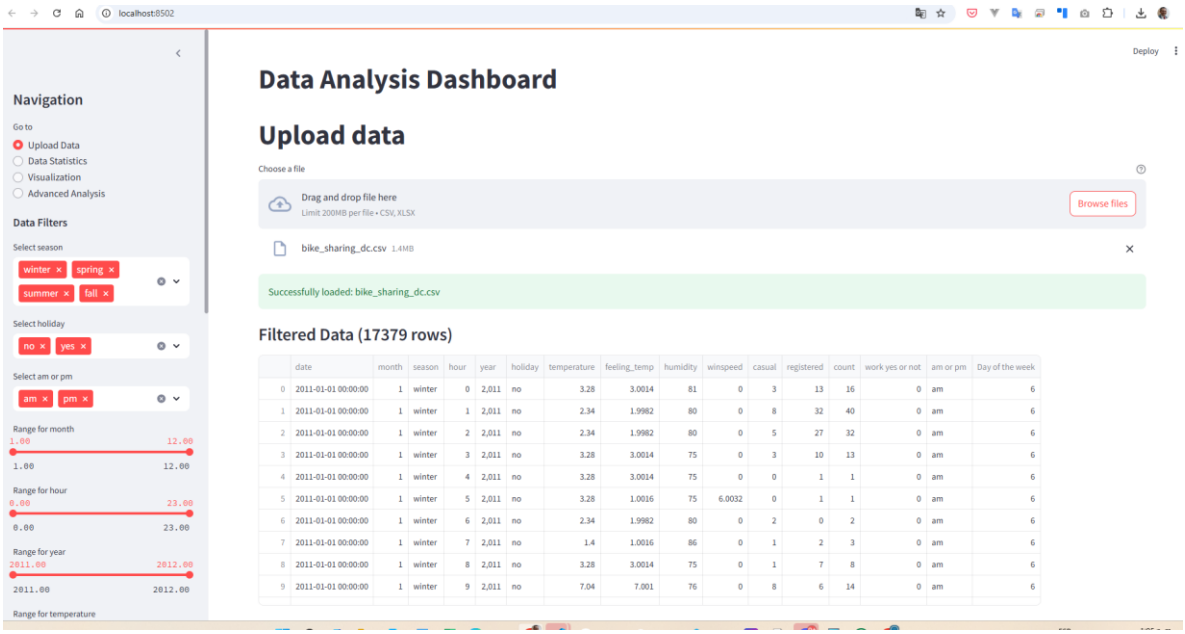
CHAT

```
    st.sidebar.title("Navigation")
    page = st.sidebar.radio(
        "Go to",
        ["Upload Data", "Data Statistics", "Visualization", "Advanced Analysis"]
    )

    # Main title
    st.title("Data Analysis Dashboard")

    # Handle navigation
    if page == "Upload Data":
        df = load_data()
        if df is not None:
            st.session_state.data = df

    elif page == "Data Statistics":
        if st.session_state.data is not None:
            display_statistics(st.session_state.data)
        else:
            st.warning("Please upload data first")

    elif page == "Visualization":
        if st.session_state.data is not None:
            create_visualizations(st.session_state.data)
        else:
            st.warning("Please upload data first")

    elif page == "Advanced Analysis":
        if st.session_state.data is not None:
            perform_advanced_analysis(st.session_state.data)
        else:
            st.warning("Please upload data first")

if __name__ == "__main__":
    main()

# ...existing code...
```

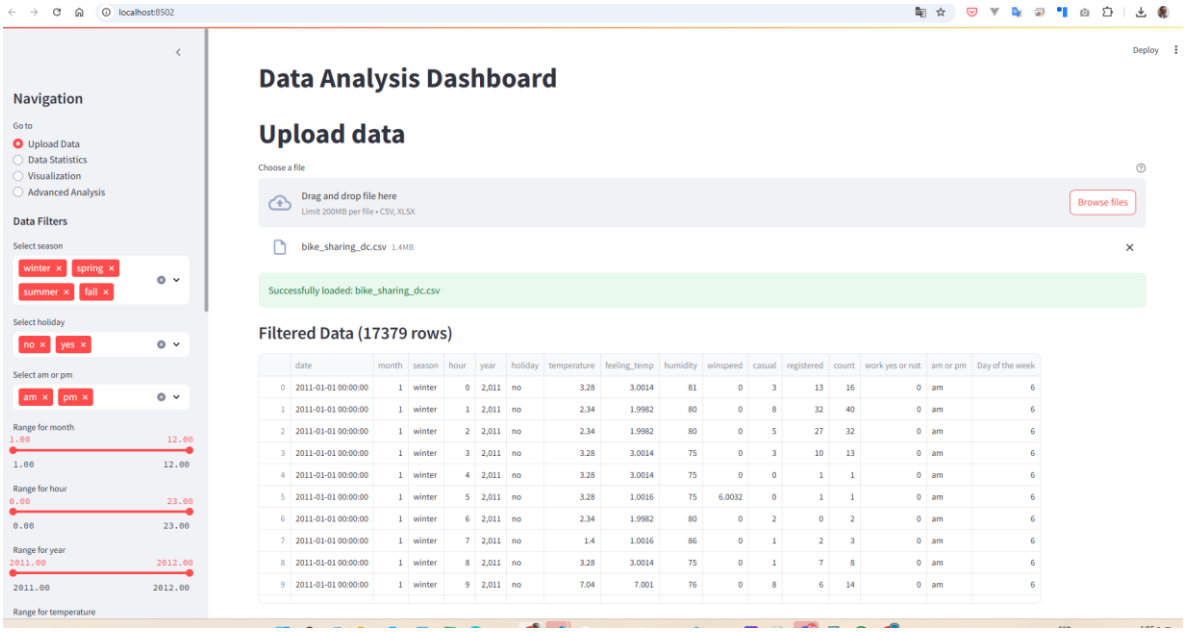
Ask Copilot

devtoproject.py Current file

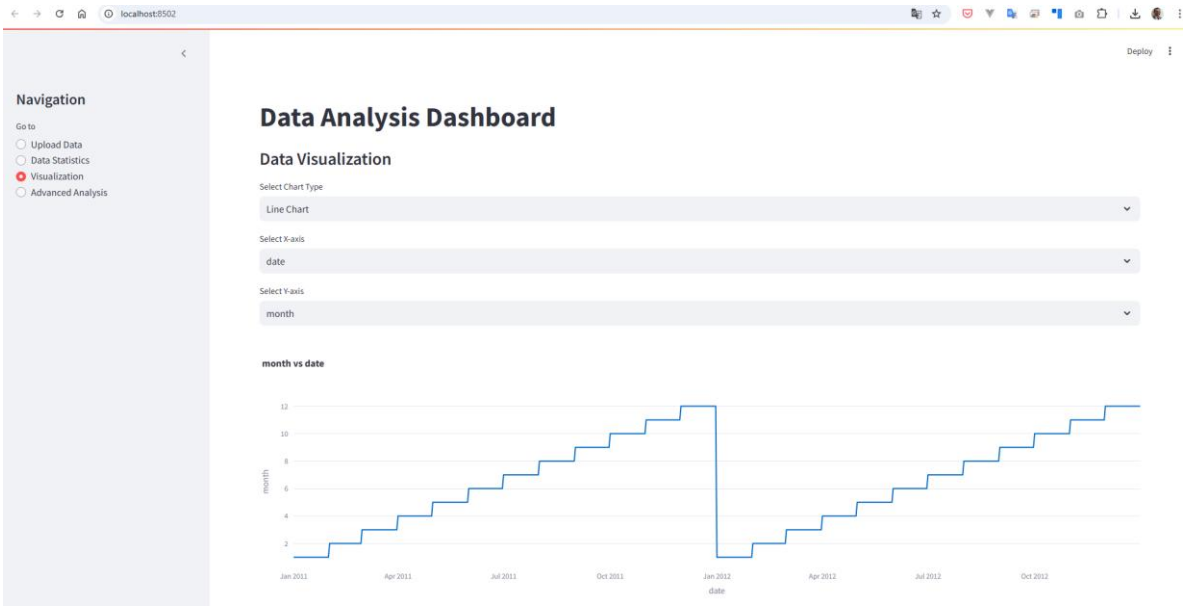
Claude 3.5 Sonnet (Preview)

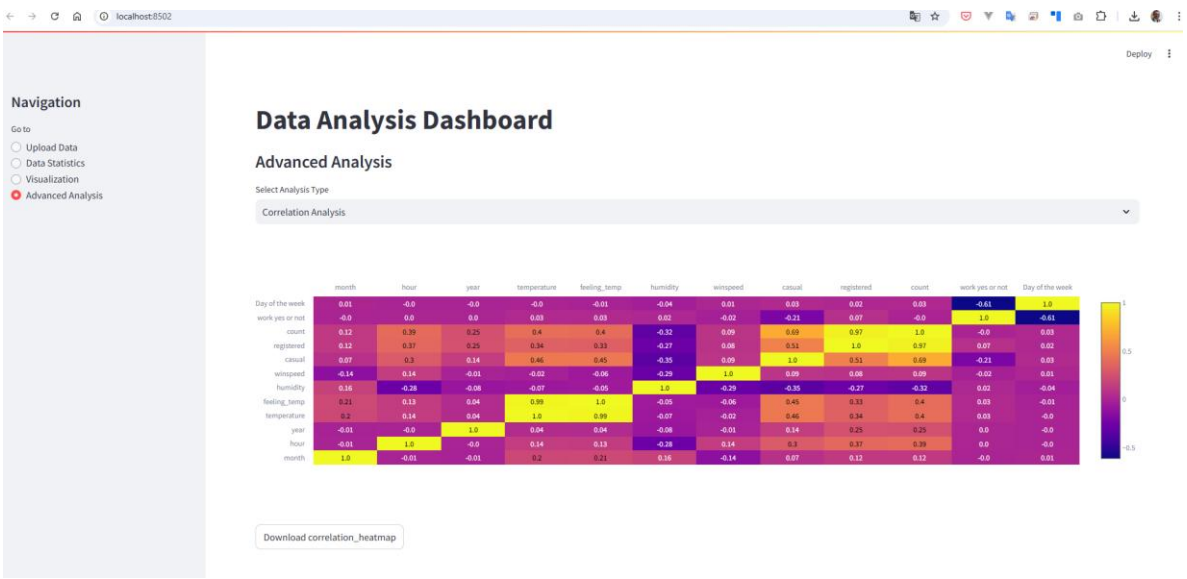


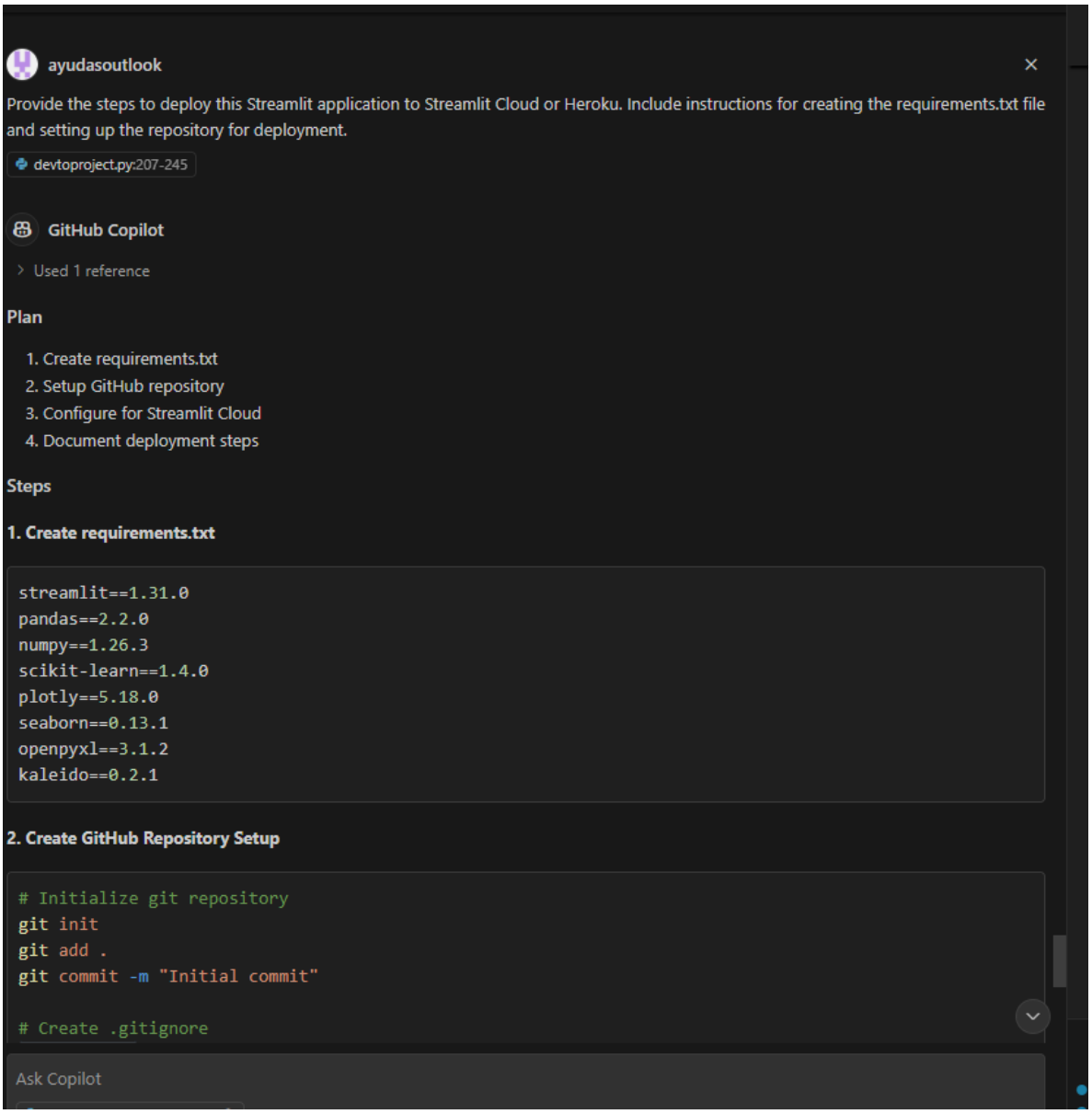
	date	month	season	hour	year	holiday	temperature	feeling_temp	humidity	winspeed	casual	registered	count	work yes or not	am or pm	Day of the week
0	2011-01-01 00:00:00	1	winter	0	2,011	no	3.28	3.0014	81	0	3	13	16	0	am	6
1	2011-01-01 00:00:00	1	winter	1	2,011	no	2.34	1.9982	80	0	8	32	40	0	am	6
2	2011-01-01 00:00:00	1	winter	2	2,011	no	2.34	1.9982	80	0	5	27	32	0	am	6
3	2011-01-01 00:00:00	1	winter	3	2,011	no	3.28	3.0014	75	0	3	10	13	0	am	6
4	2011-01-01 00:00:00	1	winter	4	2,011	no	3.28	3.0014	75	0	0	1	1	0	am	6
5	2011-01-01 00:00:00	1	winter	5	2,011	no	3.28	1.0016	75	6.0032	0	1	1	0	am	6
6	2011-01-01 00:00:00	1	winter	6	2,011	no	2.34	1.9982	80	0	2	0	2	0	am	6
7	2011-01-01 00:00:00	1	winter	7	2,011	no	1.4	1.0016	86	0	1	2	3	0	am	6
8	2011-01-01 00:00:00	1	winter	8	2,011	no	3.28	3.0014	75	0	1	7	8	0	am	6
9	2011-01-01 00:00:00	1	winter	9	2,011	no	7.04	7.001	76	0	8	6	14	0	am	6

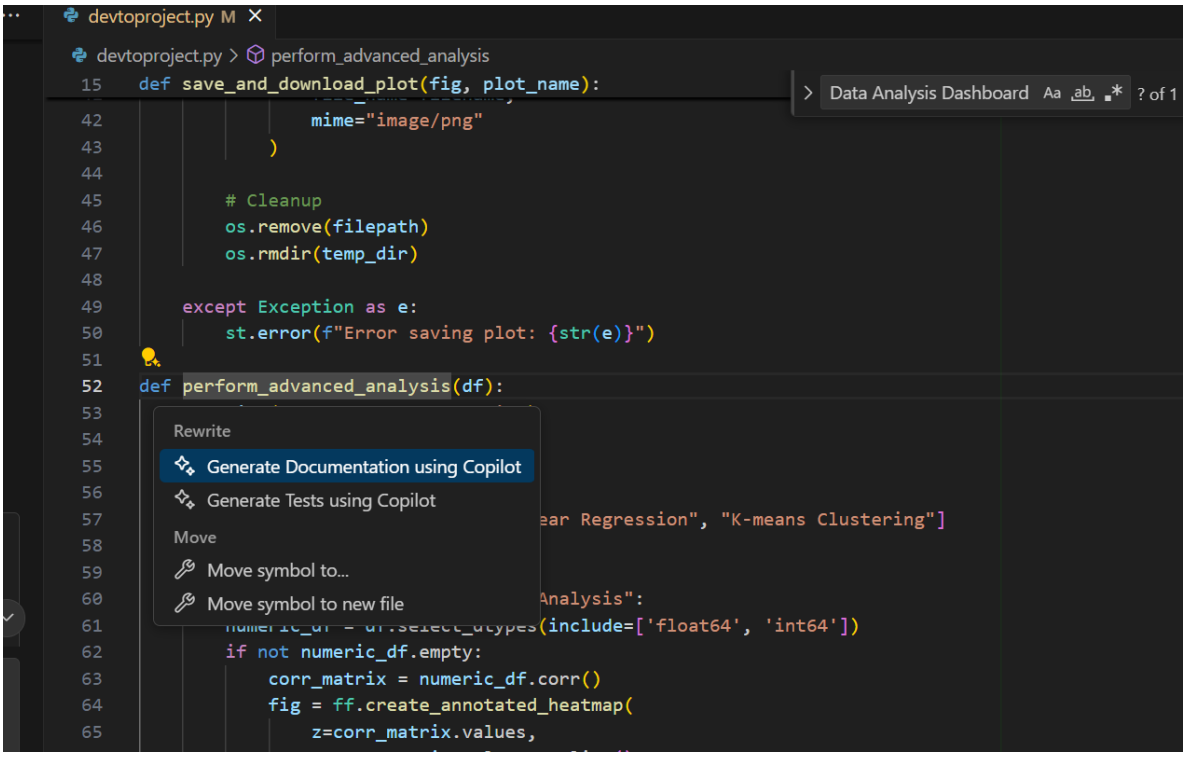


	date	month	season	hour	year	holiday	temperature	feeling_temp	humidity	winspeed	casual	registered	count	work yes or not	am or pm	Day of the week
0	2011-01-01 00:00:00	1	winter	0	2,011	no	3.28	3.0014	81	0	3	13	16	0	am	6
1	2011-01-01 00:00:00	1	winter	1	2,011	no	2.34	1.9982	80	0	8	32	40	0	am	6
2	2011-01-01 00:00:00	1	winter	2	2,011	no	2.34	1.9982	80	0	5	27	32	0	am	6
3	2011-01-01 00:00:00	1	winter	3	2,011	no	3.28	3.0014	75	0	3	10	13	0	am	6
4	2011-01-01 00:00:00	1	winter	4	2,011	no	3.28	3.0014	75	0	0	1	1	0	am	6
5	2011-01-01 00:00:00	1	winter	5	2,011	no	3.28	1.0016	75	6.0032	0	1	1	0	am	6
6	2011-01-01 00:00:00	1	winter	6	2,011	no	2.34	1.9982	80	0	2	0	2	0	am	6
7	2011-01-01 00:00:00	1	winter	7	2,011	no	1.4	1.0016	86	0	1	2	3	0	am	6
8	2011-01-01 00:00:00	1	winter	8	2,011	no	3.28	3.0014	75	0	1	7	8	0	am	6
9	2011-01-01 00:00:00	1	winter	9	2,011	no	7.04	7.001	76	0	8	6	14	0	am	6


localhost:8502
Deploy
Navigation
Go to
Upload Data
Data Statistics
Visualization
Advanced Analysis
Data Analysis Dashboard
Data Visualization
Select Chart Type
Line Chart
Select X-axis
date
Select Y-axis
month
month vs date
month
date
Jan 2011
Apr 2011
Jul 2011
Oct 2011
Jan 2012
Apr 2012
Jul 2012
Oct 2012


localhost:8502
Deploy
Navigation
Go to
Upload Data
Data Statistics
Visualization
Advanced Analysis
Data Analysis Dashboard
Advanced Analysis
Select Analysis Type
Correlation Analysis
Download correlation_heatmap


ayudasoutlook
Provide the steps to deploy this Streamlit application to Streamlit Cloud or Heroku. Include instructions for creating the requirements.txt file and setting up the repository for deployment.
devtoproject.py:207-245
GitHub Copilot
Used 1 reference
Plan
1. Create requirements.txt
2. Setup GitHub repository
3. Configure for Streamlit Cloud
4. Document deployment steps
Steps
1. Create requirements.txt
streamlit==1.31.0
pandas==2.2.0
numpy==1.26.3
scikit-learn==1.4.0
plotly==5.18.0
seaborn==0.13.1
openpyxl==3.1.2
kaleido==0.2.1
2. Create GitHub Repository Setup
Initialize git repository
git init
git add .
git commit -m "Initial commit"
Create .gitignore
Ask Copilot


devtoproject.py M
devtoproject.py > perform_advanced_analysis
Data Analysis Dashboard
? of 1
def save_and_download_plot(fig, plot_name):
mime="image/png"
)
# Cleanup
os.remove(filepath)
os.rmdir(temp_dir)
except Exception as e:
st.error(f"Error saving plot: {str(e)}")
def perform_advanced_analysis(df):
Rewrite
Generate Documentation using Copilot
Generate Tests using Copilot
Move
Move symbol to...
Move symbol to new file
ear Regression", "K-means Clustering"]
Analysis":
(include=['float64', 'int64'])
if not numeric_df.empty:
corr_matrix = numeric_df.corr()
fig = ff.create_annotated_heatmap(
z=corr_matrix.values,