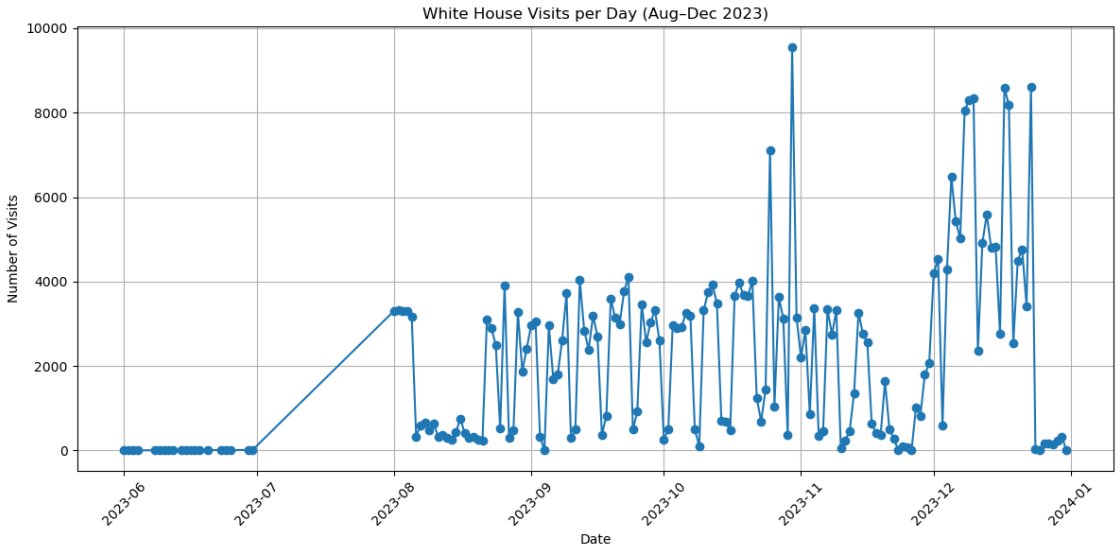
DSC640Weeks3and4

July 1, 2025

[3]:
```
import pandas as pd

# File paths
file_paths = [
    r"C:\Users\Armin\Documents\Homework\DSC640\2023.08_WAVES-ACCESS-RECORDS.
 ↪csv",
    r"C:\Users\Armin\Documents\Homework\DSC640\2023.09_WAVES-ACCESS-RECORDS.
 ↪csv",
    r"C:\Users\Armin\Documents\Homework\DSC640\2023.10_WAVES-ACCESS-RECORDS.
 ↪csv",
    r"C:\Users\Armin\Documents\Homework\DSC640\2023.11_WAVES-ACCESS-RECORDS.
 ↪csv",
    r"C:\Users\Armin\Documents\Homework\DSC640\2023.12_WAVES-ACCESS-RECORDS.csv"
]

# Load and concatenate all CSVs with low_memory=False
dfs = [pd.read_csv(fp, low_memory=False) for fp in file_paths]
combined_df = pd.concat(dfs, ignore_index=True)

# Convert datetime columns
combined_df['Appointment Start Date'] = pd.to_datetime(combined_df['Appointment␣
 ↪Start Date'], errors='coerce')
combined_df['Last Entry Date'] = pd.to_datetime(combined_df['Last Entry Date'],␣
 ↪errors='coerce')

# Create helper columns
combined_df['Visit Date'] = combined_df['Appointment Start Date'].dt.date
combined_df['Weekday'] = combined_df['Appointment Start Date'].dt.day_name()
combined_df['Month'] = combined_df['Appointment Start Date'].dt.to_period('M').
 ↪astype(str)
```

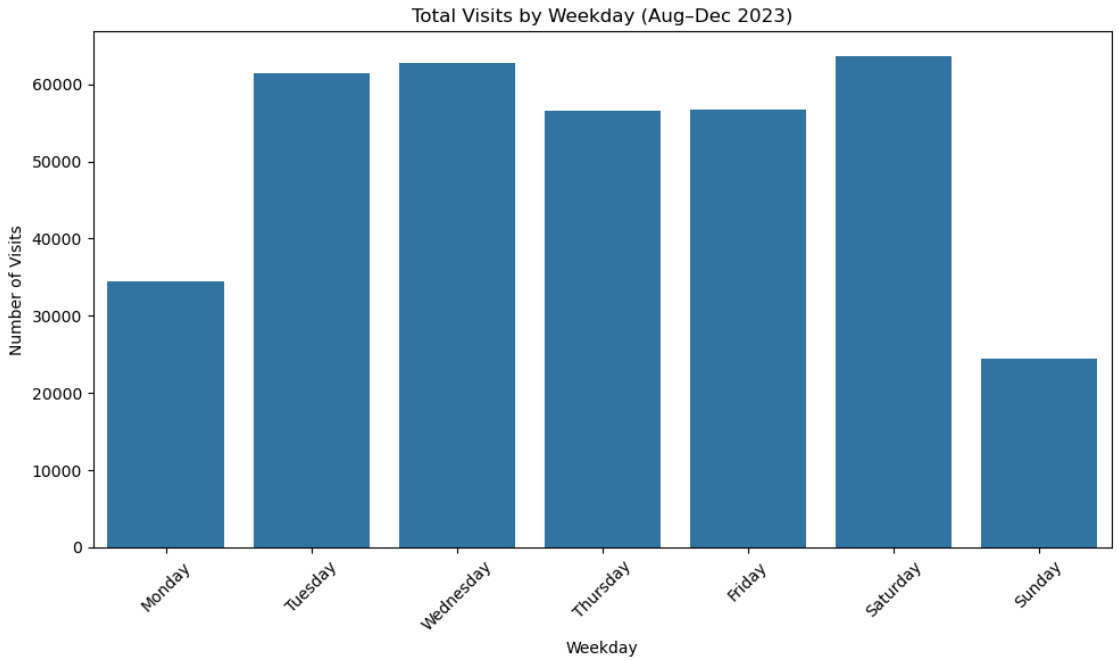
[17]:
```
# Line chart that includes the visits per day.
import matplotlib.pyplot as plt

# Group and plot
visits_per_day = combined_df.groupby('Visit Date').size().
 ↪reset_index(name='Visit Count')
```

```
visits_per_day.dropna(inplace=True)
visits_per_day = visits_per_day.sort_values('Visit Date')

plt.figure(figsize=(12, 6))
plt.plot(visits_per_day['Visit Date'], visits_per_day['Visit Count'],␣
 ↪marker='o')
plt.title('White House Visits per Day (Aug-Dec 2023)')
plt.xlabel('Date')
plt.ylabel('Number of Visits')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```

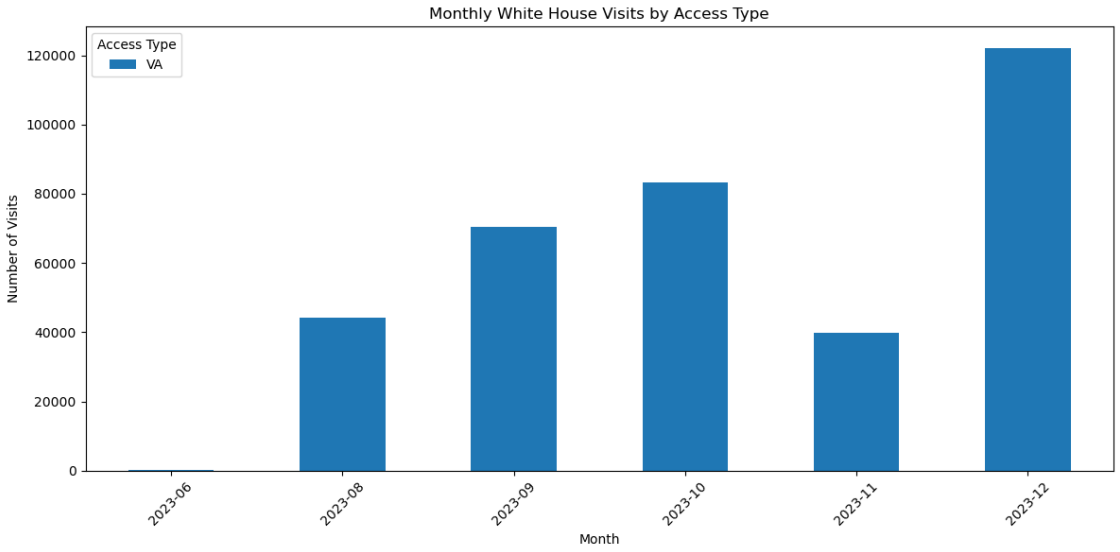


[19]:
```
# Bar chart that includes the visits by weekday
import seaborn as sns

weekday_visits = combined_df.groupby('Weekday').size().reindex([
    'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'
]).reset_index(name='Visit Count')

plt.figure(figsize=(10, 6))
sns.barplot(data=weekday_visits, x='Weekday', y='Visit Count')
plt.title('Total Visits by Weekday (Aug-Dec 2023)')
plt.xlabel('Weekday')
plt.ylabel('Number of Visits')
plt.xticks(rotation=45)
plt.tight_layout()
```

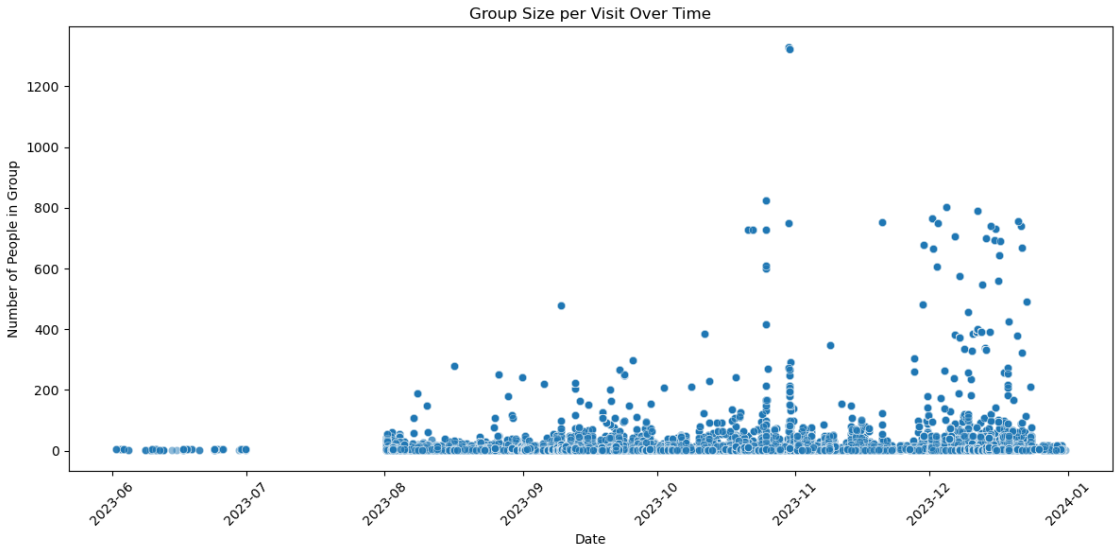
```
plt.show()
```



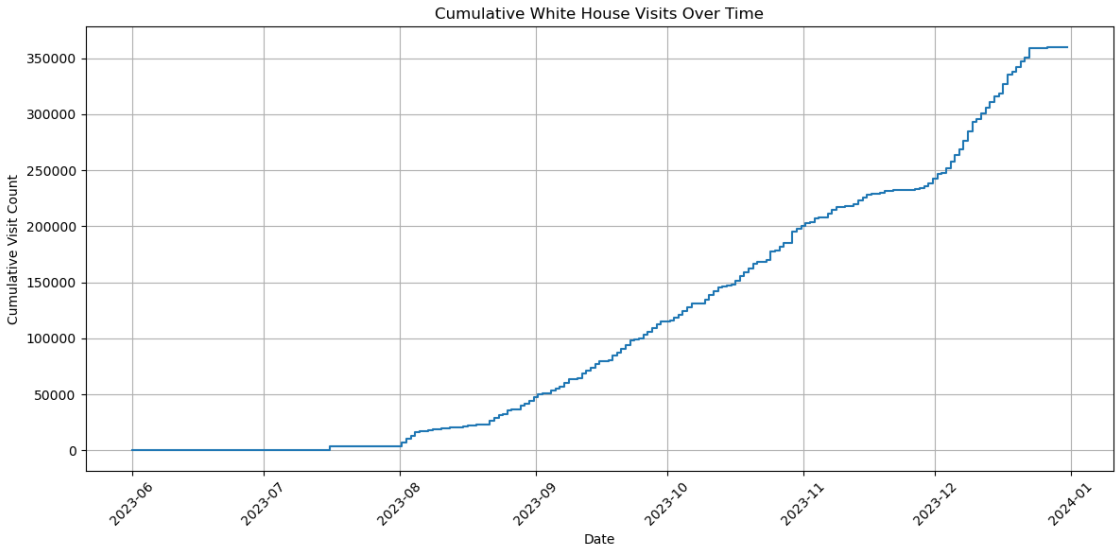
[21]:
```
# Stacked bar chart that includes the monthly visits by the type of access
monthly_access = combined_df.groupby(['Month', 'Access Type']).size().
 ↪unstack(fill_value=0)

monthly_access.plot(kind='bar', stacked=True, figsize=(12, 6))
plt.title('Monthly White House Visits by Access Type')
plt.xlabel('Month')
plt.ylabel('Number of Visits')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[23]:
```
# Scatteplot of the group size over time per each visit
plt.figure(figsize=(12, 6))
sns.scatterplot(data=combined_df, x='Appointment Start Date', y='Total People',␣
↪alpha=0.5)
plt.title('Group Size per Visit Over Time')
plt.xlabel('Date')
plt.ylabel('Number of People in Group')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

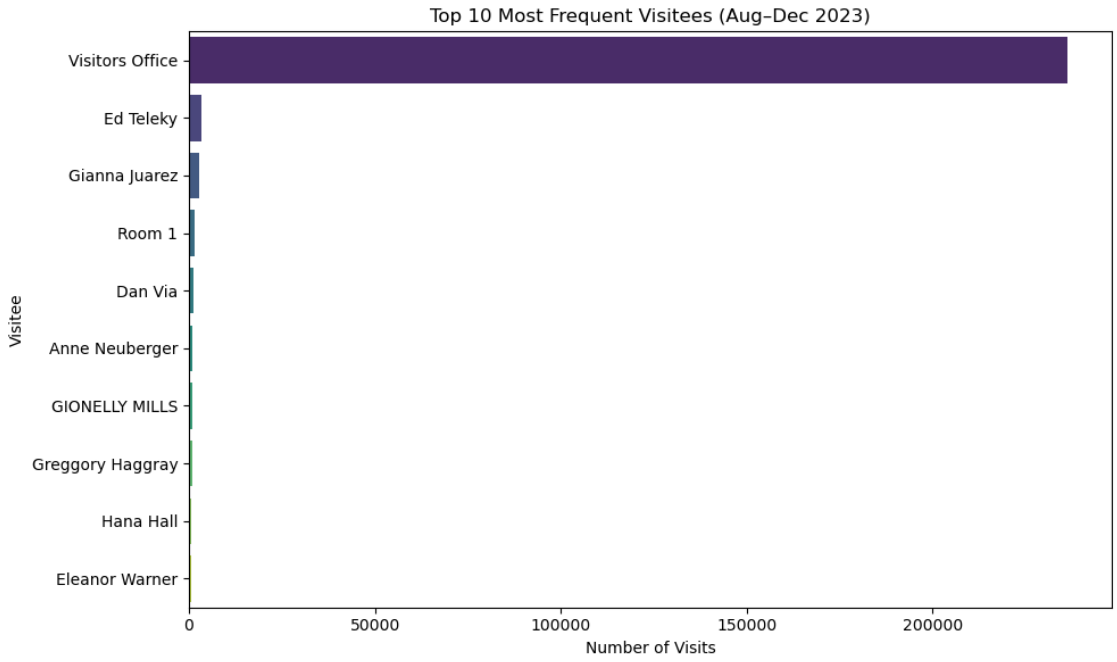


[25]:
```
# Step chart of the cumulative visits
visits_per_day['Cumulative Visits'] = visits_per_day['Visit Count'].cumsum()

plt.figure(figsize=(12, 6))
plt.step(visits_per_day['Visit Date'], visits_per_day['Cumulative Visits'],␣
 ↪where='mid')
plt.title('Cumulative White House Visits Over Time')
plt.xlabel('Date')
plt.ylabel('Cumulative Visit Count')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



[27]:
```
# Bar chart of the top 10 visitees
top_visitees = (
    combined_df['Visitee First Name'] + " " + combined_df['Visitee Last Name']
).value_counts().head(10).reset_index()
top_visitees.columns = ['Visitee Name', 'Visit Count']

plt.figure(figsize=(10, 6))
sns.barplot(data=top_visitees, x='Visit Count', y='Visitee Name',␣
 ↪palette='viridis')
plt.title('Top 10 Most Frequent Visitees (Aug-Dec 2023)')
plt.xlabel('Number of Visits')
plt.ylabel('Visitee')
plt.tight_layout()
plt.show()
```

```
C:\Users\Armin\AppData\Local\Temp\ipykernel_21252\3935897819.py:8:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(data=top_visitees, x='Visit Count', y='Visitee Name',
palette='viridis')
```



[ ]: