Beda.White House

December 21, 2024

[83]:
```
# Import necessary libraries
import pandas as pd
import requests
from io import StringIO
```

[85]:
```
# Base GitHub repository URL
base_url = "https://raw.githubusercontent.com/cheribeda/Data-Presentation/main/"

# List of all file names in your GitHub repository
file_names = [
    "2021_WAVES-ACCESS-RECORDS White House.csv",
    "2022.01_WAVES-ACCESS-RECORDS.csv",
    "2022.02_WAVES-ACCESS-RECORDS.csv",
    "2022.03_WAVES-ACCESS-RECORDS-.csv",
    "2022.04_WAVES-ACCESS-RECORDS.csv",
    "2022.05-WAVES-ACCESS-RECORDS.csv",
    "2022.06_WAVES-ACCESS-RECORDS.csv",
    "2022.07_WAVES-ACCESS-RECORDS.csv",
    "2022.08_WAVES-ACCESS-RECORDS.csv",
    "2022.09_WAVES-ACCESS-RECORDS.csv",
    "2022.10_WAVES-ACCESS-RECORDS.csv",
    "2022.11_WAVES-ACCESS-RECORDS.csv",
    "2022.12_WAVES-ACCESS-RECORDS.csv",
    "2023.01_WAVES-ACCESS-RECORDS.csv",
    "2023.02_WAVES-ACCESS-RECORDS.csv",
    "2023.03_WAVES-ACCESS-RECORDS.csv",
    "2023.04_WAVES-ACCESS-RECORDS.csv",
    "2023.05_WAVES-ACCESS-RECORDS.csv",
    "2023.06_WAVES-ACCESS-RECORDS.csv",
    "2023.07_WAVES-ACCESS-RECORDS.csv",
    "2023.08_WAVES-ACCESS-RECORDS.csv",
    "2023.09_WAVES-ACCESS-RECORDS.csv",
    "2023.10_WAVES-ACCESS-RECORDS.csv",
    "2023.11_WAVES-ACCESS-RECORDS.csv",
    "2023.12_WAVES-ACCESS-RECORDS.csv",
]

# Initialize an empty list to store dataframes
```

```
dataframes = []

# Load each file from GitHub
for file_name in file_names:
    file_url = f"{base_url}{file_name}"
    try:
        # Download the file
        response = requests.get(file_url)
        response.raise_for_status()  # Ensure the request was successful
        # Load into a DataFrame using StringIO
        df = pd.read_csv(StringIO(response.text))
        # Standardize column names
        df.columns = [col.strip().lower().replace(' ', '_') for col in df.
 ↪columns]
        dataframes.append(df)
        print(f"Successfully loaded: {file_name}")
    except Exception as e:
        print(f"Error loading {file_name}: {e}")

# Combine all datasets into a single DataFrame
combined_df = pd.concat(dataframes, ignore_index=True)

# Preview the combined DataFrame
print("Combined DataFrame Shape:", combined_df.shape)
print("Combined DataFrame Columns:", combined_df.columns)
print(combined_df.head())

# Save the combined dataset locally
combined_df.to_csv("combined_white_house_logs.csv", index=False)
print("Combined dataset saved as 'combined_white_house_logs.csv'")
```

```
Successfully loaded: 2021_WAVES-ACCESS-RECORDS White House.csv
Successfully loaded: 2022.01_WAVES-ACCESS-RECORDS.csv
Successfully loaded: 2022.02_WAVES-ACCESS-RECORDS.csv
Successfully loaded: 2022.03_WAVES-ACCESS-RECORDS-.csv
Successfully loaded: 2022.04_WAVES-ACCESS-RECORDS.csv
Successfully loaded: 2022.05-WAVES-ACCESS-RECORDS.csv
Successfully loaded: 2022.06_WAVES-ACCESS-RECORDS.csv
Successfully loaded: 2022.07_WAVES-ACCESS-RECORDS.csv
Successfully loaded: 2022.08_WAVES-ACCESS-RECORDS.csv
```

```
/var/folders/qh/fxk6l4hj7qn8crw0sbt5nypc0000gn/T/ipykernel_56496/3589418560.py:4
4: DtypeWarning: Columns (7,8) have mixed types. Specify dtype option on import
or set low_memory=False.
  df = pd.read_csv(StringIO(response.text))
```

```
Successfully loaded: 2022.09_WAVES-ACCESS-RECORDS.csv
```

```
/var/folders/qh/fxk6l4hj7qn8crw0sbt5nypc0000gn/T/ipykernel_56496/3589418560.py:4
```

```
4: DtypeWarning: Columns (13) have mixed types. Specify dtype option on import
or set low_memory=False.
  df = pd.read_csv(StringIO(response.text))

Successfully loaded: 2022.10_WAVES-ACCESS-RECORDS.csv
Successfully loaded: 2022.11_WAVES-ACCESS-RECORDS.csv

/var/folders/qh/fxk6l4hj7qn8crw0sbt5nypc0000gn/T/ipykernel_56496/3589418560.py:4
4: DtypeWarning: Columns (7,8) have mixed types. Specify dtype option on import
or set low_memory=False.
  df = pd.read_csv(StringIO(response.text))

Successfully loaded: 2022.12_WAVES-ACCESS-RECORDS.csv

/var/folders/qh/fxk6l4hj7qn8crw0sbt5nypc0000gn/T/ipykernel_56496/3589418560.py:4
4: DtypeWarning: Columns (7,8) have mixed types. Specify dtype option on import
or set low_memory=False.
  df = pd.read_csv(StringIO(response.text))

Successfully loaded: 2023.01_WAVES-ACCESS-RECORDS.csv

/var/folders/qh/fxk6l4hj7qn8crw0sbt5nypc0000gn/T/ipykernel_56496/3589418560.py:4
4: DtypeWarning: Columns (7,8) have mixed types. Specify dtype option on import
or set low_memory=False.
  df = pd.read_csv(StringIO(response.text))

Successfully loaded: 2023.02_WAVES-ACCESS-RECORDS.csv

/var/folders/qh/fxk6l4hj7qn8crw0sbt5nypc0000gn/T/ipykernel_56496/3589418560.py:4
4: DtypeWarning: Columns (7,8,13) have mixed types. Specify dtype option on
import or set low_memory=False.
  df = pd.read_csv(StringIO(response.text))

Successfully loaded: 2023.03_WAVES-ACCESS-RECORDS.csv

/var/folders/qh/fxk6l4hj7qn8crw0sbt5nypc0000gn/T/ipykernel_56496/3589418560.py:4
4: DtypeWarning: Columns (7,8,13) have mixed types. Specify dtype option on
import or set low_memory=False.
  df = pd.read_csv(StringIO(response.text))

Successfully loaded: 2023.04_WAVES-ACCESS-RECORDS.csv

/var/folders/qh/fxk6l4hj7qn8crw0sbt5nypc0000gn/T/ipykernel_56496/3589418560.py:4
4: DtypeWarning: Columns (7,8,13) have mixed types. Specify dtype option on
import or set low_memory=False.
  df = pd.read_csv(StringIO(response.text))

Successfully loaded: 2023.05_WAVES-ACCESS-RECORDS.csv

/var/folders/qh/fxk6l4hj7qn8crw0sbt5nypc0000gn/T/ipykernel_56496/3589418560.py:4
4: DtypeWarning: Columns (7,8) have mixed types. Specify dtype option on import
or set low_memory=False.
  df = pd.read_csv(StringIO(response.text))

Successfully loaded: 2023.06_WAVES-ACCESS-RECORDS.csv
```

```
/var/folders/qh/fxk6l4hj7qn8crw0sbt5nypc0000gn/T/ipykernel_56496/3589418560.py:4
4: DtypeWarning: Columns (7,8) have mixed types. Specify dtype option on import
or set low_memory=False.
  df = pd.read_csv(StringIO(response.text))

Successfully loaded: 2023.07_WAVES-ACCESS-RECORDS.csv

/var/folders/qh/fxk6l4hj7qn8crw0sbt5nypc0000gn/T/ipykernel_56496/3589418560.py:4
4: DtypeWarning: Columns (7,8) have mixed types. Specify dtype option on import
or set low_memory=False.
  df = pd.read_csv(StringIO(response.text))

Successfully loaded: 2023.08_WAVES-ACCESS-RECORDS.csv

/var/folders/qh/fxk6l4hj7qn8crw0sbt5nypc0000gn/T/ipykernel_56496/3589418560.py:4
4: DtypeWarning: Columns (7,8,13) have mixed types. Specify dtype option on
import or set low_memory=False.
  df = pd.read_csv(StringIO(response.text))

Successfully loaded: 2023.09_WAVES-ACCESS-RECORDS.csv

/var/folders/qh/fxk6l4hj7qn8crw0sbt5nypc0000gn/T/ipykernel_56496/3589418560.py:4
4: DtypeWarning: Columns (7,8) have mixed types. Specify dtype option on import
or set low_memory=False.
  df = pd.read_csv(StringIO(response.text))

Successfully loaded: 2023.10_WAVES-ACCESS-RECORDS.csv

/var/folders/qh/fxk6l4hj7qn8crw0sbt5nypc0000gn/T/ipykernel_56496/3589418560.py:4
4: DtypeWarning: Columns (7,8,13) have mixed types. Specify dtype option on
import or set low_memory=False.
  df = pd.read_csv(StringIO(response.text))

Successfully loaded: 2023.11_WAVES-ACCESS-RECORDS.csv

/var/folders/qh/fxk6l4hj7qn8crw0sbt5nypc0000gn/T/ipykernel_56496/3589418560.py:4
4: DtypeWarning: Columns (7,8) have mixed types. Specify dtype option on import
or set low_memory=False.
  df = pd.read_csv(StringIO(response.text))

Successfully loaded: 2023.12_WAVES-ACCESS-RECORDS.csv
Combined DataFrame Shape: (1207458, 44)
Combined DataFrame Columns: Index(['namelast', 'namefirst', 'namemid', 'uin',
'bdgnbr', 'access_type',
       'toa', 'poa', 'tod', 'pod', 'appt_made_date', 'appt_start_date',
       'appt_end_date', 'appt_cancel_date', 'total_people', 'last_updatedby',
       'post', 'lastentrydate', 'terminal_suffix', 'visitee_namelast',
       'visitee_namefirst', 'meeting_loc', 'meeting_room', 'caller_name_last',
       'caller_name_first', 'caller_room', 'releasedate', 'unnamed:_27',
       'unnamed:_28', 'last_name', 'first_name', 'middle_initial',
       'appointment_made_date', 'appointment_start_date',
       'appointment_end_date', 'appointment_cancel_date', 'last_updated_by',
       'last_entry_date', 'visitee_last_name', 'visitee_first_name',
```

```
       'meeting_location', 'caller_last_name', 'caller_first_name',
       'release_date'],
      dtype='object')
  namelast namefirst namemid     uin  bdgnbr access_type               toa  \
0   AAKHUU  BOLORMAA       N  U38116     NaN          VA               NaN
1   AASSAR       MIA       L  U37794     NaN          VA  12/17/2021 12:23
2   ABALOS    JANILA       L  U38186     NaN          VA               NaN
3  ABARCAR      KARA       N  U37879     NaN          VA  12/19/2021 17:25
4   ABBOTT   NICOLAS       P  U36630     NaN          VA               NaN

   poa  tod  pod  … appointment_end_date appointment_cancel_date  \
0  NaN  NaN  NaN  …                  NaN                     NaN
1  NaN  NaN  NaN  …                  NaN                     NaN
2  NaN  NaN  NaN  …                  NaN                     NaN
3  NaN  NaN  NaN  …                  NaN                     NaN
4  NaN  NaN  NaN  …                  NaN                     NaN

  last_updated_by last_entry_date  visitee_last_name visitee_first_name  \
0             NaN             NaN                NaN                NaN
1             NaN             NaN                NaN                NaN
2             NaN             NaN                NaN                NaN
3             NaN             NaN                NaN                NaN
4             NaN             NaN                NaN                NaN

  meeting_location caller_last_name caller_first_name release_date
0              NaN              NaN               NaN          NaN
1              NaN              NaN               NaN          NaN
2              NaN              NaN               NaN          NaN
3              NaN              NaN               NaN          NaN
4              NaN              NaN               NaN          NaN

[5 rows x 44 columns]
Combined dataset saved as 'combined_white_house_logs.csv'
```

[87]:
```python
# Load the combined dataset with low_memory=False
combined_df = pd.read_csv("combined_white_house_logs.csv", low_memory=False)
```

[88]:
```python
# Convert datetime columns using .loc to avoid SettingWithCopyWarning
for col in datetime_columns:
    cleaned_df.loc[:, col] = pd.to_datetime(cleaned_df[col], errors='coerce')
```

[89]:
```python
# Load the combined dataset with low_memory=False
combined_df = pd.read_csv("combined_white_house_logs.csv", low_memory=False)

# Keep relevant columns
columns_to_keep = [
    'namelast', 'namefirst', 'uin', 'access_type', 'toa', 'tod',
```

```
    'appt_start_date', 'appt_end_date', 'meeting_loc', 'meeting_room',
    'caller_name_last', 'caller_name_first', 'total_people'
]
cleaned_df = combined_df[columns_to_keep]

# Convert datetime columns using .loc
datetime_columns = ['toa', 'tod', 'appt_start_date', 'appt_end_date']
for col in datetime_columns:
    cleaned_df.loc[:, col] = pd.to_datetime(cleaned_df[col], errors='coerce')

# Drop rows with critical missing data
cleaned_df = cleaned_df.dropna(subset=['appt_start_date', 'appt_end_date',␣
 ↪'access_type'])

# Preview cleaned data
print("Cleaned Data Preview:")
print(cleaned_df.head())
```

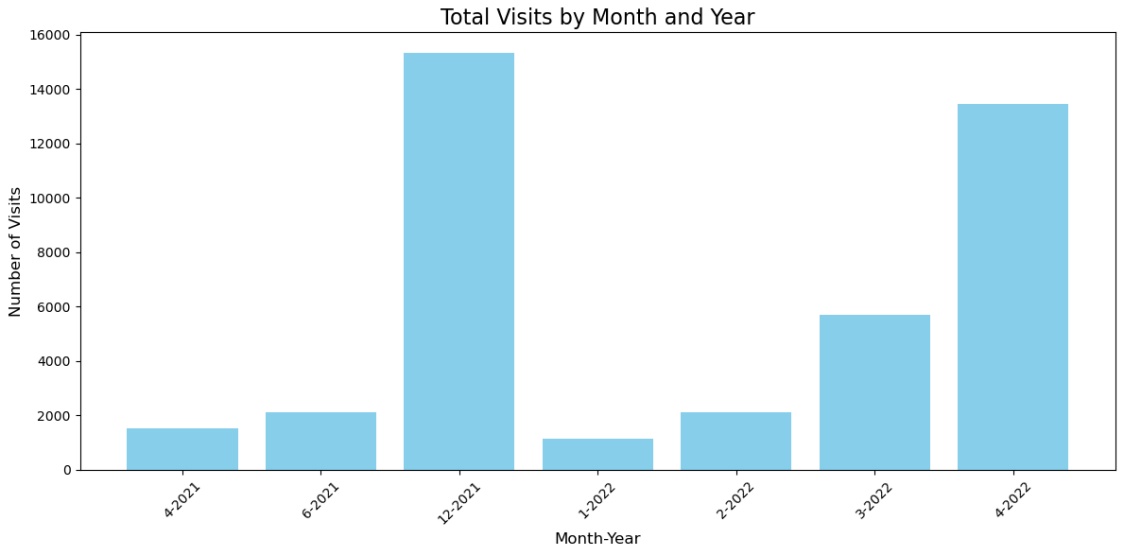
```
Cleaned Data Preview:
  namelast namefirst     uin access_type                 toa  tod  \
0   AAKHUU  BOLORMAA  U38116          VA                 NaT  NaT
1   AASSAR       MIA  U37794          VA 2021-12-17 12:23:00  NaT
2   ABALOS    JANILA  U38186          VA                 NaT  NaT
3  ABARCAR      KARA  U37879          VA 2021-12-19 17:25:00  NaT
4   ABBOTT   NICOLAS  U36630          VA                 NaT  NaT

       appt_start_date        appt_end_date meeting_loc meeting_room  \
0  2021-12-19 00:00:00  2021-12-19 11:59:00          WH      EW TOUR
1  2021-12-17 00:00:00  2021-12-17 11:59:00          WH     EW - RES
2  2021-12-19 00:00:00  2021-12-19 11:59:00          WH      EW TOUR
3  2021-12-19 00:00:00  2021-12-19 11:59:00          WH      EW TOUR
4  2021-12-06 00:00:00  2021-12-06 11:59:00        OEOB          SCA

  caller_name_last caller_name_first  total_people
0         SCHWARTZ            PEYTON          55.0
1         SCHWARTZ            PEYTON          27.0
2             COLE             KATIE           1.0
3             COLE             KATIE          83.0
4      GIAMMARELLA            ALYSSA          15.0
```

[93]:
```
# Ensure 'appt_start_date' is in datetime format
cleaned_df['appt_start_date'] = pd.to_datetime(cleaned_df['appt_start_date'],␣
 ↪errors='coerce')

# Check for any invalid datetime entries
invalid_dates = cleaned_df[cleaned_df['appt_start_date'].isna()]
print("Rows with Invalid Dates in 'appt_start_date':")
```

```
print(invalid_dates)

# Proceed with adding 'month' and 'year' columns
cleaned_df['month'] = cleaned_df['appt_start_date'].dt.month
cleaned_df['year'] = cleaned_df['appt_start_date'].dt.year

# Group by year and month to count visits
monthly_visits = cleaned_df.groupby(['year', 'month']).size().
 ↪reset_index(name='visit_count')

# Display the monthly visits
print("Monthly Visits Summary:")
print(monthly_visits.head())
```

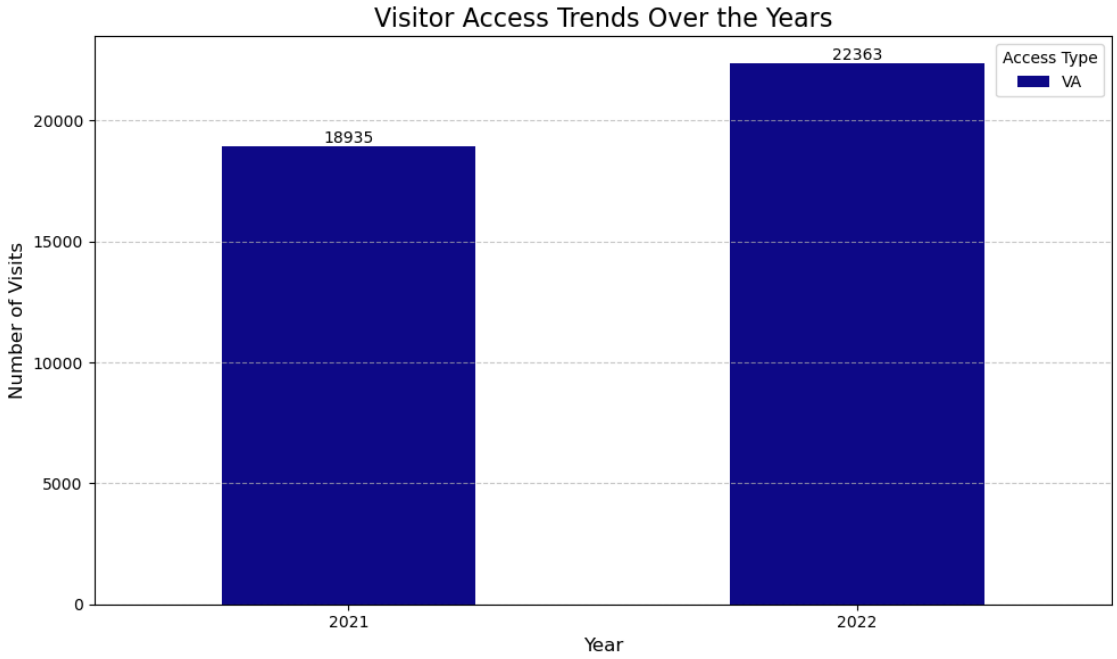
```
Rows with Invalid Dates in 'appt_start_date':
Empty DataFrame
Columns: [namelast, namefirst, uin, access_type, toa, tod, appt_start_date,
appt_end_date, meeting_loc, meeting_room, caller_name_last, caller_name_first,
total_people]
Index: []
Monthly Visits Summary:
   year  month  visit_count
0  2021      4         1509
1  2021      6         2110
2  2021     12        15316
3  2022      1         1139
4  2022      2         2098
```

[95]:
```
# Combine month and year for labels
monthly_visits['month_year'] = monthly_visits['month'].astype(str) + '-' +␣
 ↪monthly_visits['year'].astype(str)

# Bar Chart
plt.figure(figsize=(12, 6))
plt.bar(monthly_visits['month_year'], monthly_visits['visit_count'],␣
 ↪color='skyblue')
plt.title("Total Visits by Month and Year", fontsize=16)
plt.xlabel("Month-Year", fontsize=12)
plt.ylabel("Number of Visits", fontsize=12)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[99]:
```
import matplotlib.pyplot as plt

# Group visits by year
access_types_by_year = cleaned_df.groupby(['year', 'access_type']).size().
 ↪unstack(fill_value=0)

# Plot Stacked Bar Chart
fig, ax = plt.subplots(figsize=(10, 6))
bars = access_types_by_year.plot(kind='bar', stacked=True, ax=ax,␣
 ↪colormap='plasma')

# Add data labels
for bar in bars.patches:
    if bar.get_height() > 0:
        ax.annotate(
            f'{int(bar.get_height())}',
            (bar.get_x() + bar.get_width() / 2, bar.get_height()),
            ha='center',
            va='bottom',
            fontsize=10
        )

# Customize Title and Labels
plt.title("Visitor Access Trends Over the Years", fontsize=16)
plt.xlabel("Year", fontsize=12)
plt.ylabel("Number of Visits", fontsize=12)
plt.xticks(rotation=0)
plt.legend(title="Access Type", loc="upper right")
```

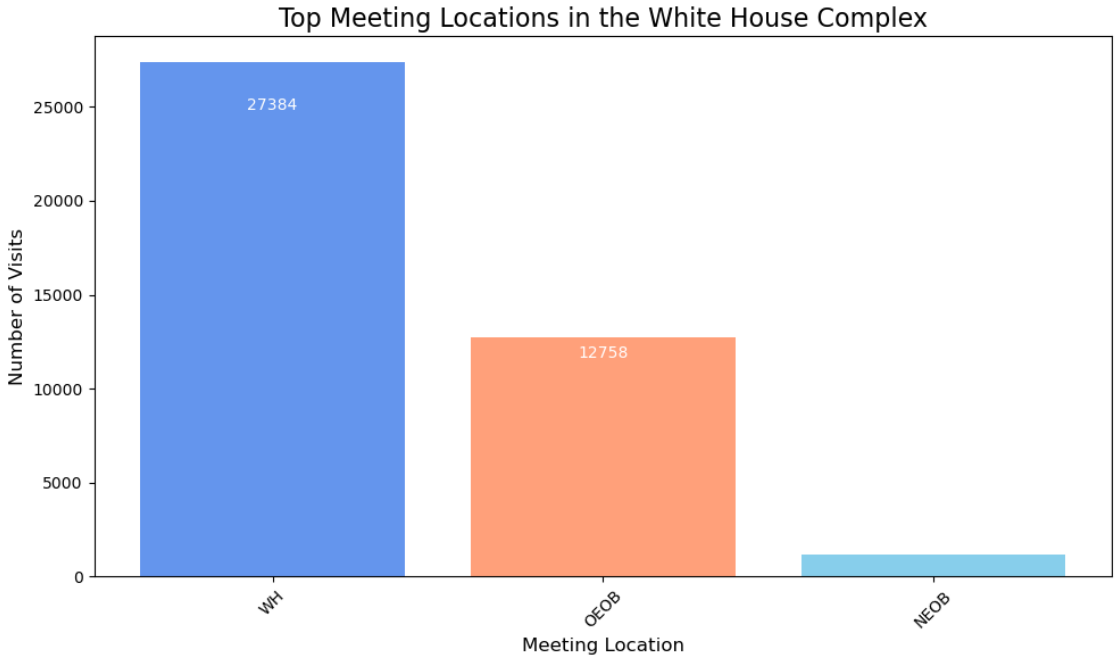
```
plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()
plt.show()
```



[101]:
```
# Check the number of valid (non-NaT) values in 'toa'
print("Number of valid TOA entries:", cleaned_df['toa'].notna().sum())

# Check a sample of the 'toa' column to confirm its contents
print("Sample TOA values:")
print(cleaned_df['toa'].head(10))
```

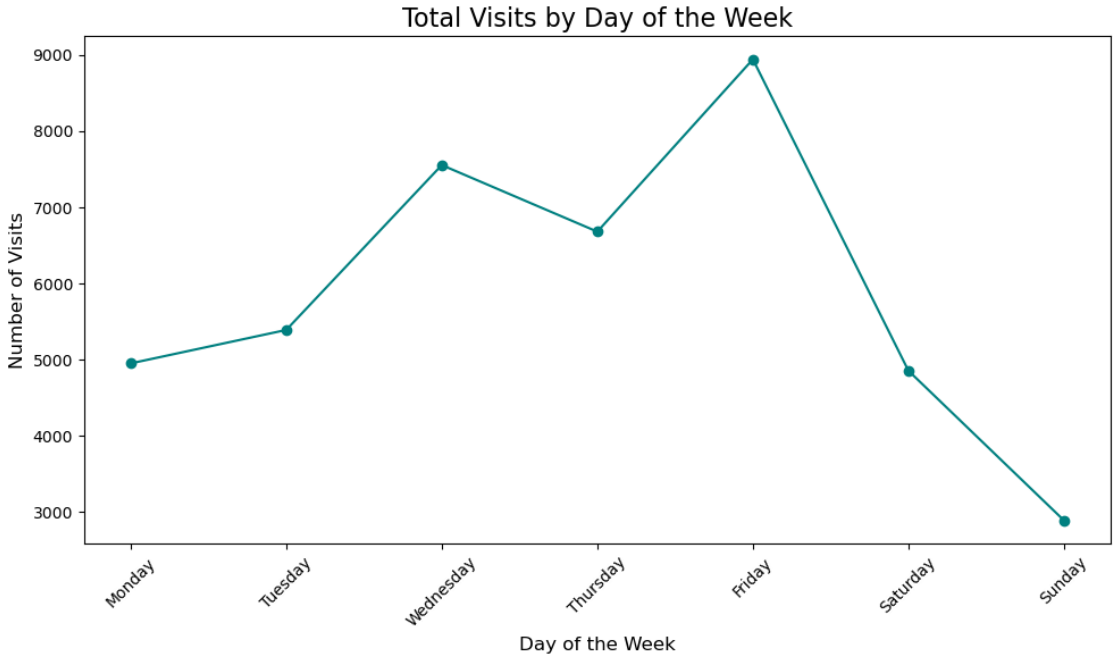
```
Number of valid TOA entries: 15224
Sample TOA values:
0                    NaT
1    2021-12-17 12:23:00
2                    NaT
3    2021-12-19 17:25:00
4                    NaT
5                    NaT
6    2021-12-17 20:36:00
7    2021-12-17 20:36:00
8                    NaT
9    2021-12-17 15:56:00
Name: toa, dtype: object
```

[136]:
```
# Bar chart: Top Meeting Locations
plt.figure(figsize=(10, 6))
bars = plt.bar(top_meeting_locations.index, top_meeting_locations.values,␣
 ↪color=['#6495ED', '#FFA07A', '#87CEEB'])

# Add data labels
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, height - (height * 0.1),␣
 ↪f'{int(height)}',
             ha='center', va='bottom', fontsize=10, color='white')

# Customize the chart
plt.title("Top Meeting Locations in the White House Complex", fontsize=16)
plt.xlabel("Meeting Location", fontsize=12)
plt.ylabel("Number of Visits", fontsize=12)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

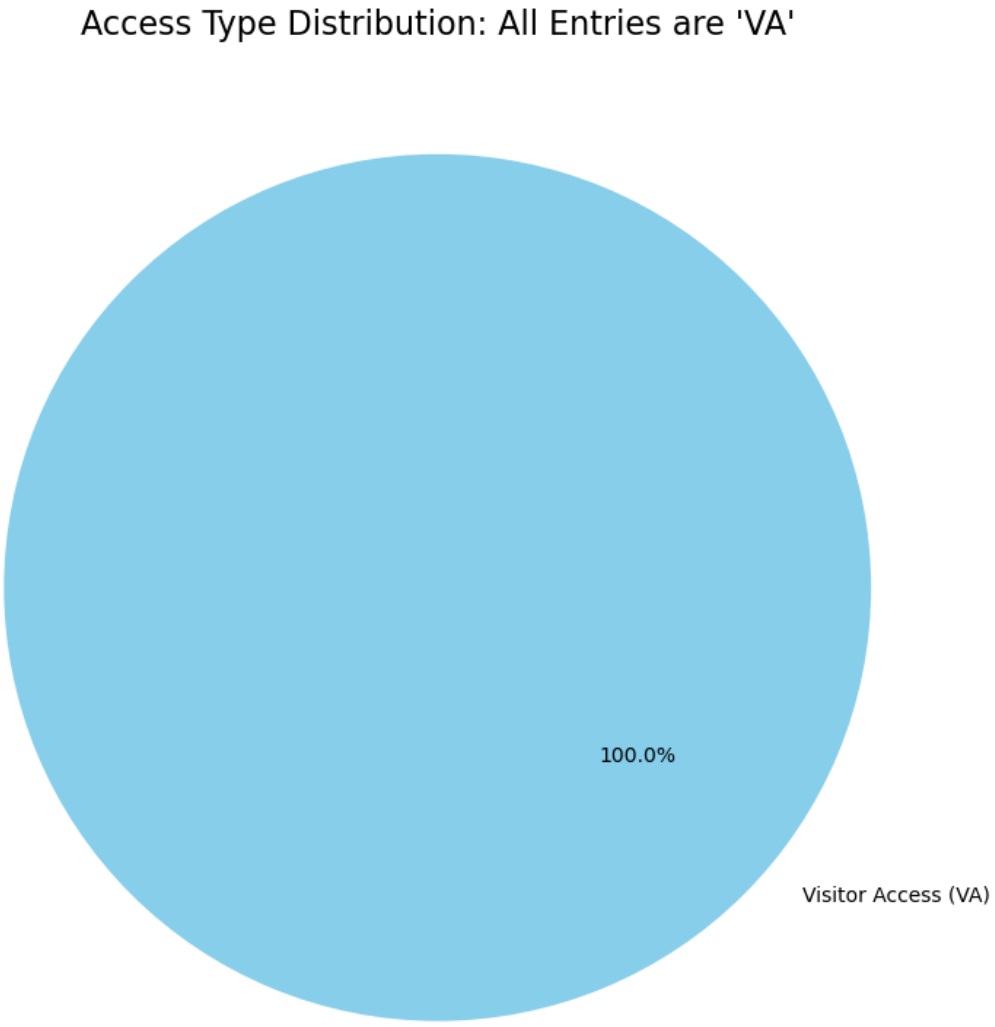


[61]:
```
# Visits by day of the week
visits_by_day = cleaned_df['day_of_week'].value_counts().reindex(
    ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday',␣
 ↪'Sunday']
)
```

```
# Line Chart
plt.figure(figsize=(10, 6))
visits_by_day.plot(kind='line', marker='o', color='teal')
plt.title("Total Visits by Day of the Week", fontsize=16)
plt.xlabel("Day of the Week", fontsize=12)
plt.ylabel("Number of Visits", fontsize=12)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

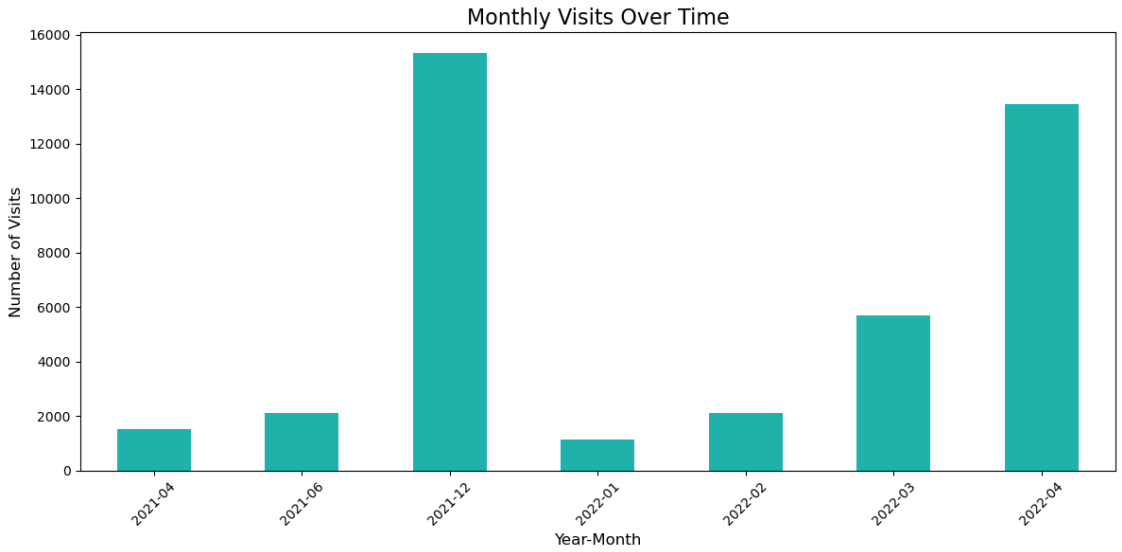


[107]:
```
# Single-category pie chart
plt.figure(figsize=(8, 8))
plt.pie([1], labels=['Visitor Access (VA)'], autopct='%1.1f%%', startangle=140,␣
↪colors=['skyblue'])
plt.title("Access Type Distribution: All Entries are 'VA'", fontsize=16)
plt.tight_layout()
plt.show()
```

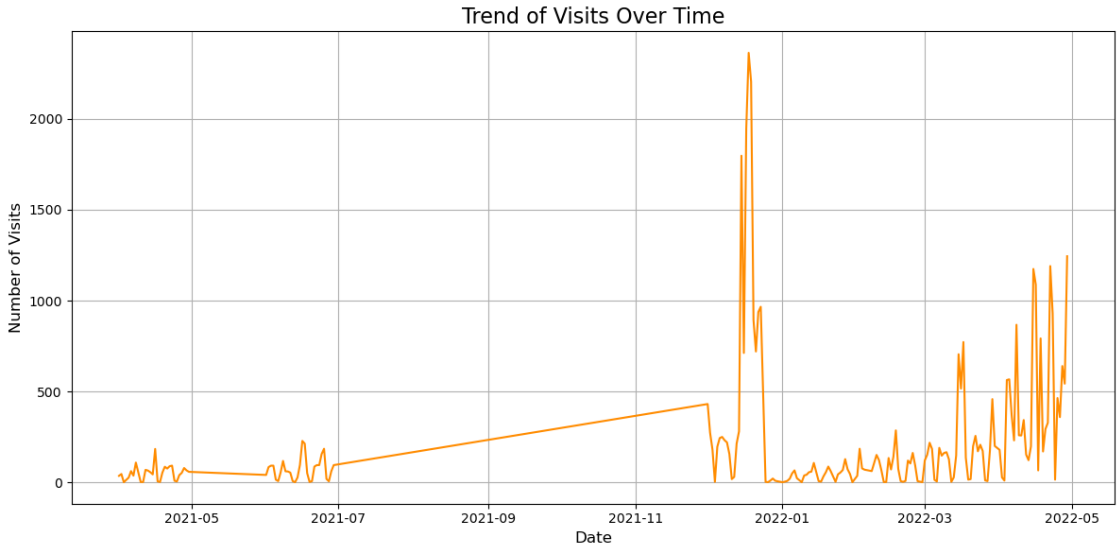


[109]:
```
# Extract year and month from 'appt_start_date'
cleaned_df['year_month'] = cleaned_df['appt_start_date'].dt.to_period('M')

# Group by year-month and count visits
visits_by_month = cleaned_df['year_month'].value_counts().sort_index()

# Convert PeriodIndex to string for plotting
visits_by_month.index = visits_by_month.index.astype(str)

# Create Stacked Bar Chart
plt.figure(figsize=(12, 6))
visits_by_month.plot(kind='bar', color='lightseagreen')
```

```
plt.title("Monthly Visits Over Time", fontsize=16)
plt.xlabel("Year-Month", fontsize=12)
plt.ylabel("Number of Visits", fontsize=12)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[111]:
```
# Group visits by 'appt_start_date' (daily basis)
daily_visits = cleaned_df['appt_start_date'].dt.date.value_counts().sort_index()

# Plot Line Chart: Daily Visits Over Time
plt.figure(figsize=(12, 6))
daily_visits.plot(kind='line', color='darkorange')
plt.title("Trend of Visits Over Time", fontsize=16)
plt.xlabel("Date", fontsize=12)
plt.ylabel("Number of Visits", fontsize=12)
plt.grid(True)
plt.tight_layout()
plt.show()
```



[122]:
```
print(cleaned_df.columns)
```

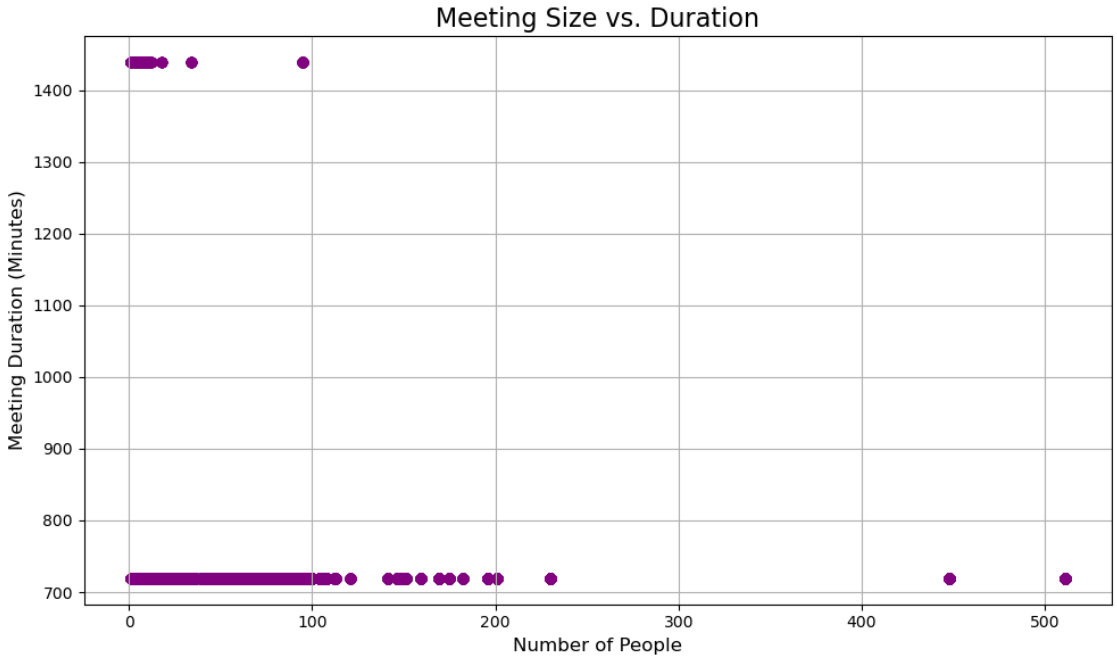
```
Index(['namelast', 'namefirst', 'uin', 'access_type', 'toa', 'tod',
       'appt_start_date', 'appt_end_date', 'meeting_loc', 'meeting_room',
       'caller_name_last', 'caller_name_first', 'total_people', 'month',
       'year', 'year_month'],
      dtype='object')
```

[128]:
```
# Ensure start and end dates are in datetime format
cleaned_df['appt_start_date'] = pd.to_datetime(cleaned_df['appt_start_date'],␣
 ↪errors='coerce')
cleaned_df['appt_end_date'] = pd.to_datetime(cleaned_df['appt_end_date'],␣
 ↪errors='coerce')

# Calculate meeting duration in minutes
cleaned_df['meeting_duration'] = (cleaned_df['appt_end_date'] -␣
 ↪cleaned_df['appt_start_date']).dt.total_seconds() / 60
```

[130]:
```
# Filter for valid meeting durations and total people
scatter_df = cleaned_df.dropna(subset=['meeting_duration', 'total_people'])
scatter_df = scatter_df[scatter_df['meeting_duration'] > 0]
```

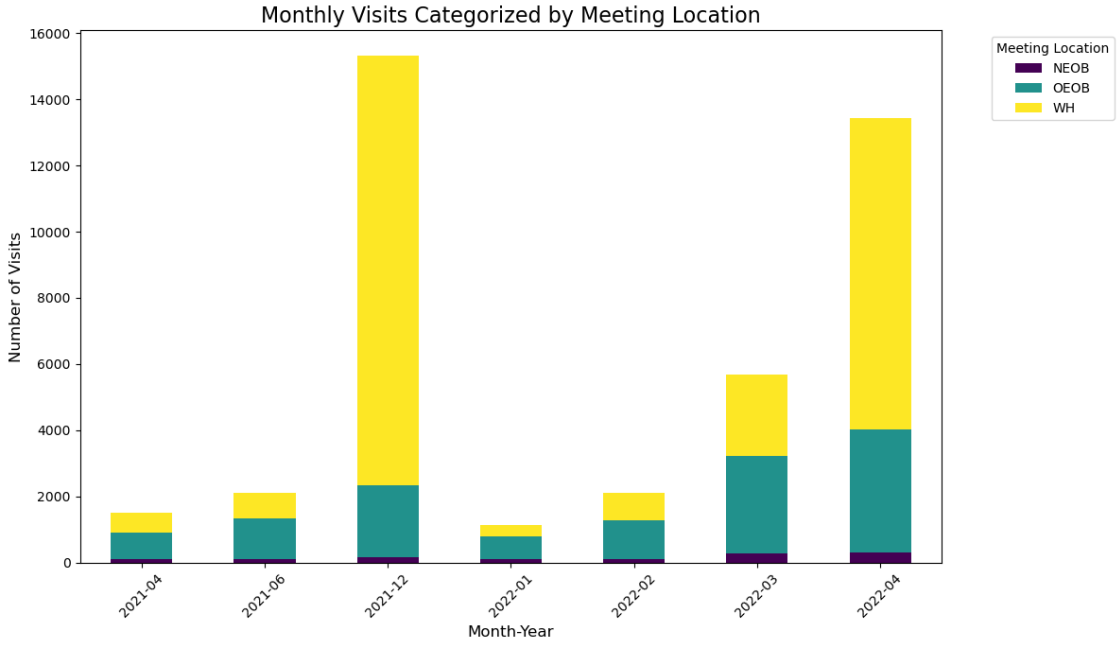
[132]:
```
# Create scatterplot
plt.figure(figsize=(10, 6))
plt.scatter(scatter_df['total_people'], scatter_df['meeting_duration'], alpha=0.
 ↪5, color='purple')
plt.title("Meeting Size vs. Duration", fontsize=16)
plt.xlabel("Number of People", fontsize=12)
plt.ylabel("Meeting Duration (Minutes)", fontsize=12)
```

```
plt.grid(True)
plt.tight_layout()
plt.show()
```



[134]:
```
import seaborn as sns

# Group by year-month and meeting location
stacked_bar_data = (
    cleaned_df
    .dropna(subset=['meeting_loc', 'year', 'month'])
    .groupby(['year', 'month', 'meeting_loc'])
    .size()
    .reset_index(name='visits')
)

# Pivot for stacked bar chart
stacked_bar_pivot = stacked_bar_data.pivot_table(
    index=['year', 'month'],
    columns='meeting_loc',
    values='visits',
    fill_value=0
)

# Plot stacked bar chart
```

```
stacked_bar_pivot.plot(kind='bar', stacked=True, figsize=(12, 7),␣
 ↪colormap='viridis')
plt.title("Monthly Visits Categorized by Meeting Location", fontsize=16)
plt.xlabel("Month-Year", fontsize=12)
plt.ylabel("Number of Visits", fontsize=12)
plt.xticks(ticks=range(len(stacked_bar_pivot.index)), labels=[f"{y}-{m:02}" for␣
 ↪y, m in stacked_bar_pivot.index], rotation=45)
plt.legend(title="Meeting Location", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



[ ]: