911 Calls Data Analysis Project

Introduction:

In this project, we explore and analyze a dataset containing **emergency 911 calls in Montgomery County, PA**. The dataset provides information about the **reasons for the calls**, the **location**, and the **timestamp** of each call. **The goal** of this project is to gain insights into the patterns, trends, and distribution of emergency calls and understand the primary reasons for these calls.

Dataset Description:

The dataset used in this project is sourced from Kaggle. It contains **663522 rows**, and **9 columns** of records of emergency 911 calls. Each record includes information such as the reason for the call, the township, the ZIP code, and the timestamp.

Tools and Libraries Used:

To perform the data analysis and visualization tasks, I utilized the following tools and libraries:

-**Python** programming language

-**Pandas** library for data manipulation and analysis

-**Matplotlib** library for creating visualizations

-**Seaborn** library for enhanced data visualization

-**PyCharm** as the development environment

Project Workflow:

1.Data Loading and Initial Exploration:

- Read the dataset using Pandas and examine its structure.
- Check for missing values, **data types**, and basic statistics of the dataset.
- Gain an **understanding of the columns** and their meanings.

2.Data Preparation and Feature Engineering:

- Extract relevant information from the timestamp column to create new columns for hour, month, and day of the week.
- Perform data transformations and **cleaning**, if required, to ensure data quality.

3.Exploratory Data Analysis:

- Analyze the **distribution** of emergency calls across different ZIP codes and townships.
- Identify the top ZIP codes and townships with the **highest** number of calls.
- Determine the **most common reasons** for emergency calls.
- **Visualize** the above insights using bar plots, count plots, and other appropriate visualizations.

4.Temporal Analysis:

- Explore the trends in emergency calls over time, such as by month, day of the week, and hour.
- Create line plots and heatmaps to **visualize** the temporal **patterns**.
- Investigate the variations in call volumes for different reasons over time.

Data Analysis and Visualizations:

In this section, I present the **code snippets and visualizations** that reveal key insights from the emergency 911 calls dataset. The code demonstrates data manipulation, **exploratory analysis**, and visualization techniques using Python and libraries such as Pandas, Matplotlib, and Seaborn.

Importing Libraries and Reading the Dataset

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

This code snippet imports the following libraries:

pandas as pd for data manipulation.

matplotlib.pyplot as plt for creating plots.

seaborn as sns for enhanced visualizations.

```
sns.set(style="whitegrid", palette="Set2")
sns.set_context("notebook", font_scale=1.2)
```

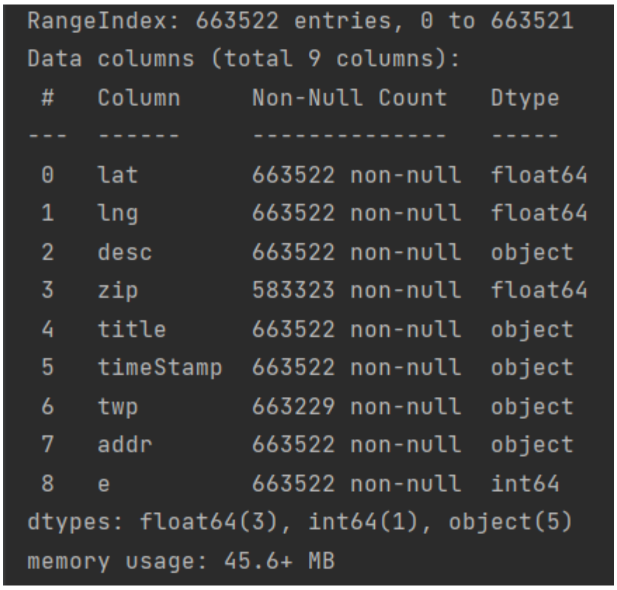
Set styles for each graph.

```
df = pd.read_csv(r"C:\Users\Barif\OneDrive\שולחן העבודה\הכל פה
גבו\Programming\data analytics\Projects\911 Calls\911.csv")
```

Read the CSV file into a DataFrame.

```
print(df.info())
```

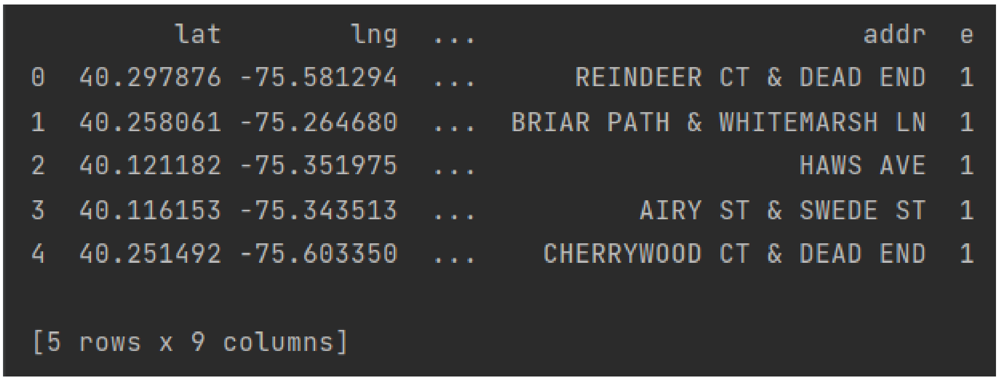
Display information about the DataFrame.



```
RangeIndex: 663522 entries, 0 to 663521
Data columns (total 9 columns):
 #   Column     Non-Null Count   Dtype  
---  ------     --------------   -----  
 0   lat        663522 non-null  float64
 1   lng        663522 non-null  float64
 2   desc       663522 non-null  object 
 3   zip        583323 non-null  float64
 4   title      663522 non-null  object 
 5   timeStamp  663522 non-null  object 
 6   twp        663229 non-null  object 
 7   addr       663522 non-null  object 
 8   e          663522 non-null  int64  
dtypes: float64(3), int64(1), object(5)
memory usage: 45.6+ MB
```

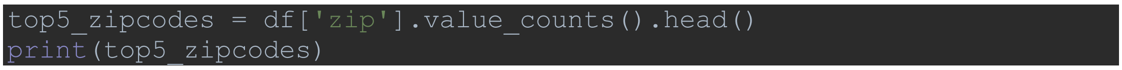
```
print(df.head())
```

Display the first few rows of the DataFrame.



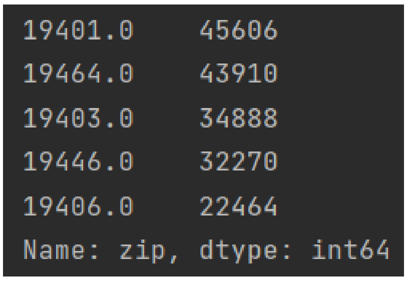
```
         lat        lng  ...                        addr  e
0  40.297876 -75.581294  ...      REINDEER CT & DEAD END  1
1  40.258061 -75.264680  ...  BRIAR PATH & WHITEMARSH LN  1
2  40.121182 -75.351975  ...                    HAWS AVE  1
3  40.116153 -75.343513  ...          AIRY ST & SWEDE ST  1
4  40.251492 -75.603350  ...    CHERRYWOOD CT & DEAD END  1

[5 rows x 9 columns]
```

Analysis of Call Frequencies by Zip Code and Township

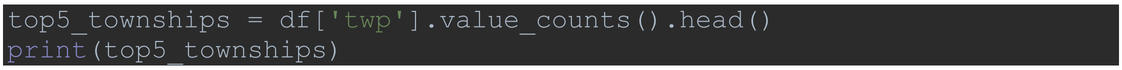


```
top5_zipcodes = df['zip'].value_counts().head()
print(top5_zipcodes)
```

Top 5 zip codes with the highest **call frequencies.**

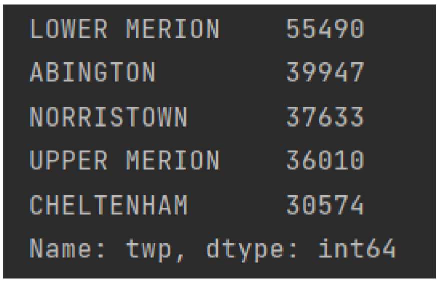


```
19401.0    45606
19464.0    43910
19403.0    34888
19446.0    32270
19406.0    22464
Name: zip, dtype: int64
```



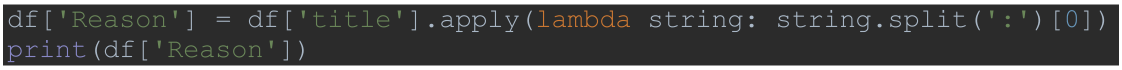
```
top5_townships = df['twp'].value_counts().head()
print(top5_townships)
```

Top 5 townships with the highest **call frequencies.**



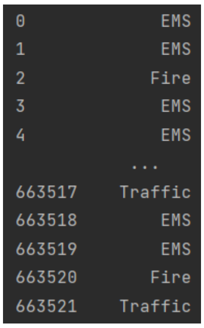
```
LOWER MERION    55490
ABINGTON        39947
NORRISTOWN      37633
UPPER MERION    36010
CHELTENHAM      30574
Name: twp, dtype: int64
```

Analysis of Reasons for Emergency Calls

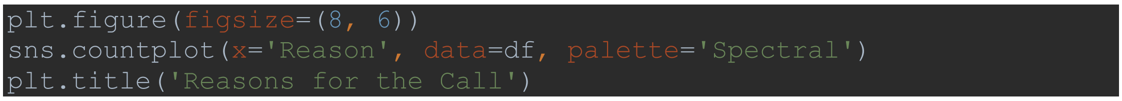


```
df['Reason'] = df['title'].apply(lambda string: string.split(':')[0])
print(df['Reason'])
```

Extract the reason for the call **from the title** column.

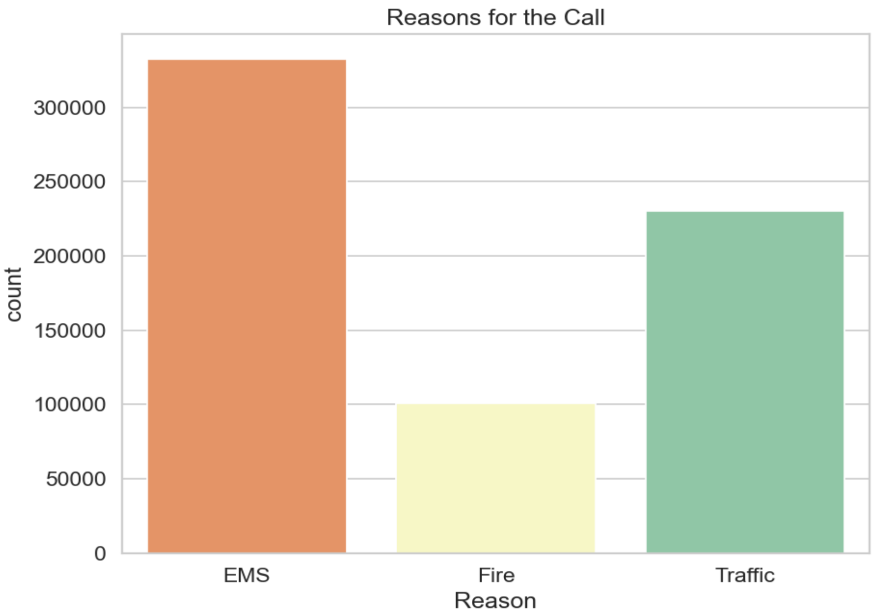


```
0             EMS
1             EMS
2            Fire
3             EMS
4             EMS
           ...
663517    Traffic
663518        EMS
663519        EMS
663520       Fire
663521    Traffic
```

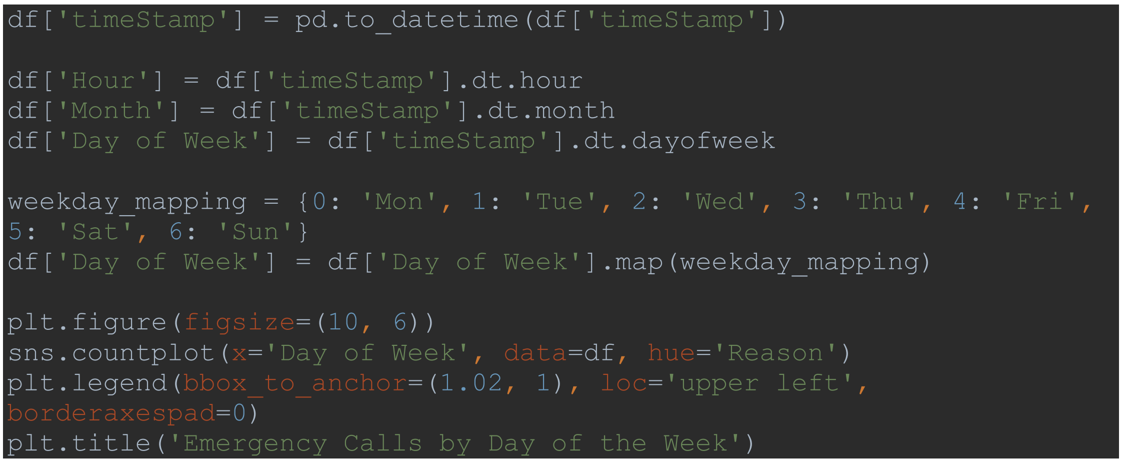


```
plt.figure(figsize=(8, 6))
sns.countplot(x='Reason', data=df, palette='Spectral')
plt.title('Reasons for the Call')
```

Create a count plot to visualize the reasons for the call.



Analysis of Emergency Calls by Day of the Week



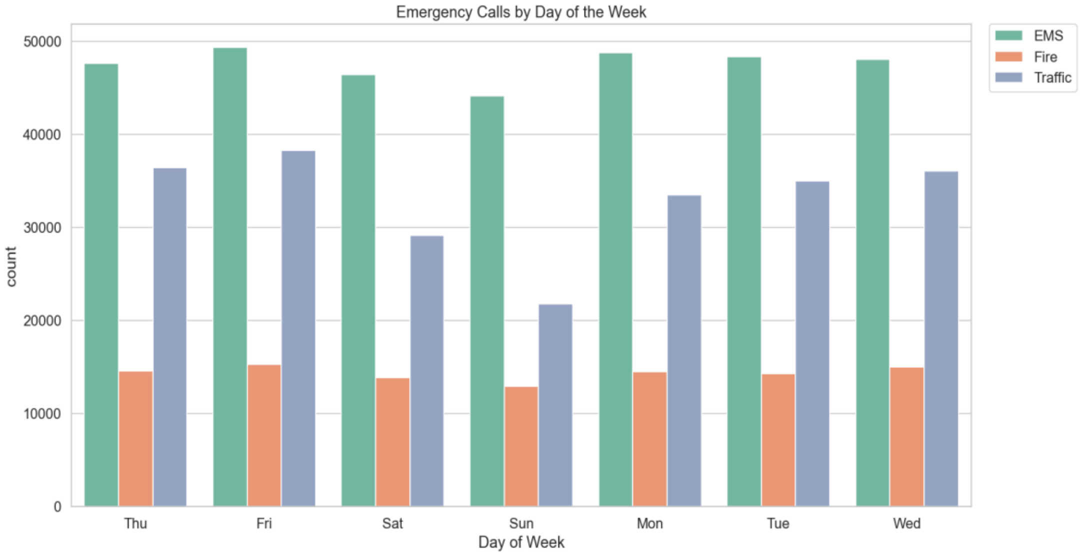
```
df['timeStamp'] = pd.to_datetime(df['timeStamp'])

df['Hour'] = df['timeStamp'].dt.hour
df['Month'] = df['timeStamp'].dt.month
df['Day of Week'] = df['timeStamp'].dt.dayofweek

weekday_mapping = {0: 'Mon', 1: 'Tue', 2: 'Wed', 3: 'Thu', 4: 'Fri',
5: 'Sat', 6: 'Sun'}
df['Day of Week'] = df['Day of Week'].map(weekday_mapping)

plt.figure(figsize=(10, 6))
sns.countplot(x='Day of Week', data=df, hue='Reason')
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left',
borderaxespad=0)
plt.title('Emergency Calls by Day of the Week')
```

Convert the **timeStamp** column **to datetime type** to extract specific times.

Add new columns for hour, month, and day of the week.

Map day of the week numbers to their corresponding names.

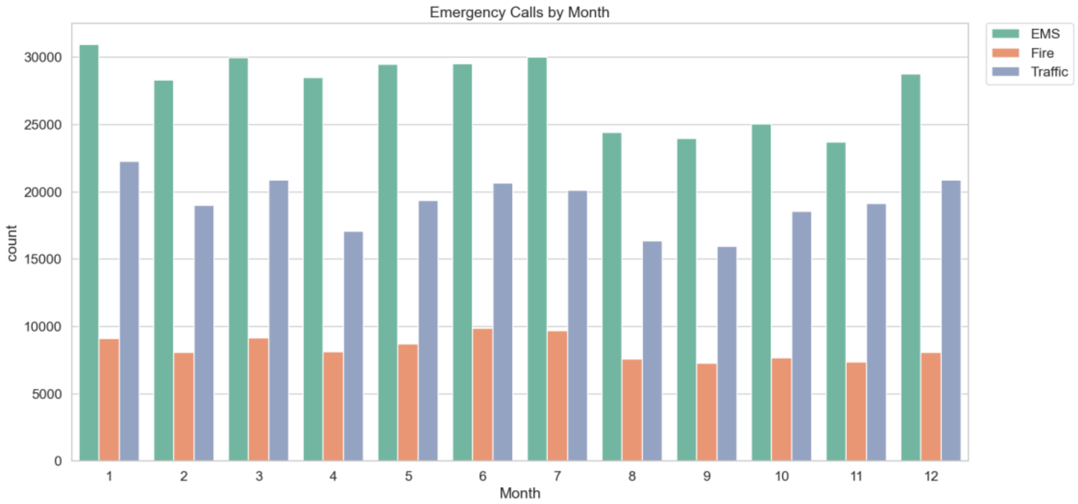
Create a count plot to visualize the emergency calls **by day of the week.**



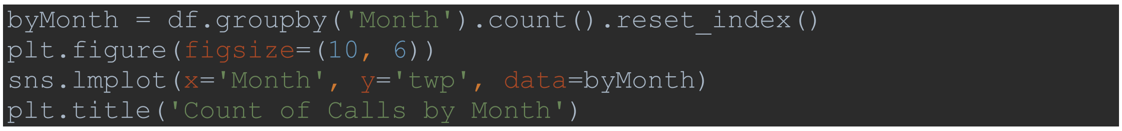
Analysis of Emergency Calls Kinds by Month

```
plt.figure(figsize=(10, 6))
sns.countplot(x='Month', data=df, hue='Reason')
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left',
borderaxespad=0)
plt.title('Emergency Calls by Month')
```

Create a count plot to visualize the emergency calls **by month.**



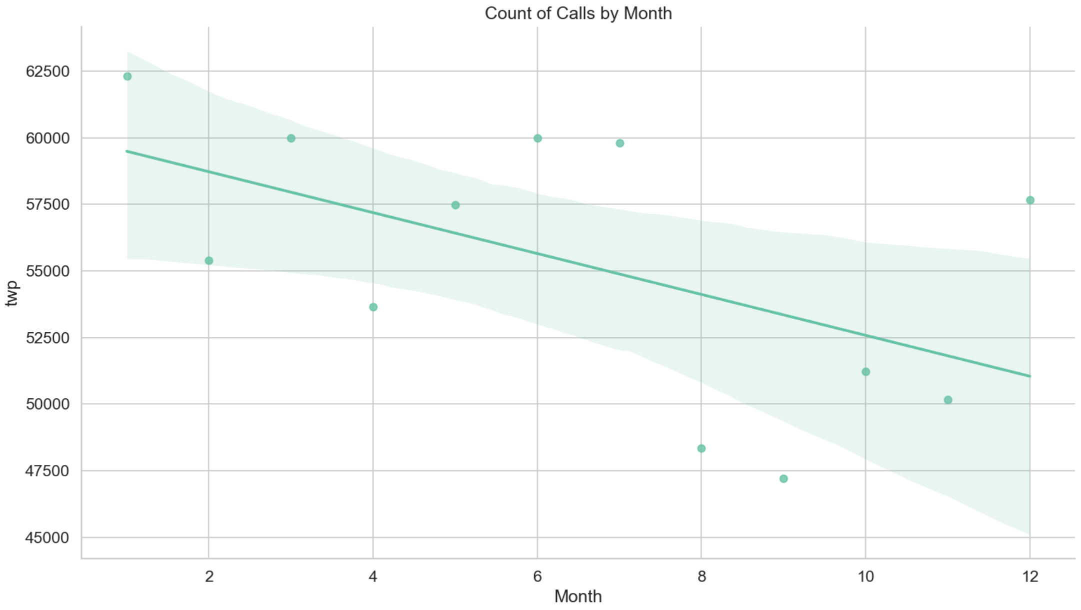
Analysis of Emergency Calls by Month



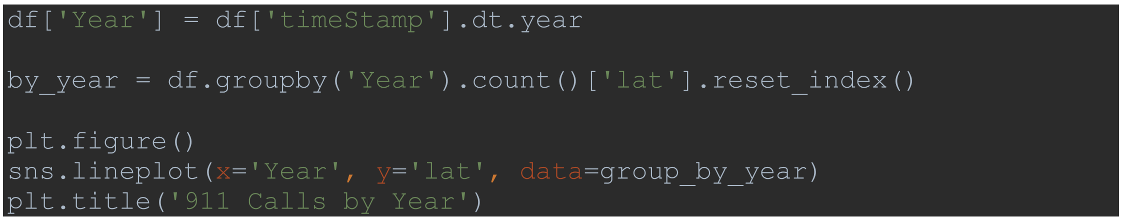
```
byMonth = df.groupby('Month').count().reset_index()
plt.figure(figsize=(10, 6))
sns.lmplot(x='Month', y='twp', data=byMonth)
plt.title('Count of Calls by Month')
```

Grouped the DataFrame **by month** and counted the calls.

Created a **linear regression** plot to visualize the count of calls **by month.**



Analysis of Emergency Calls by Year

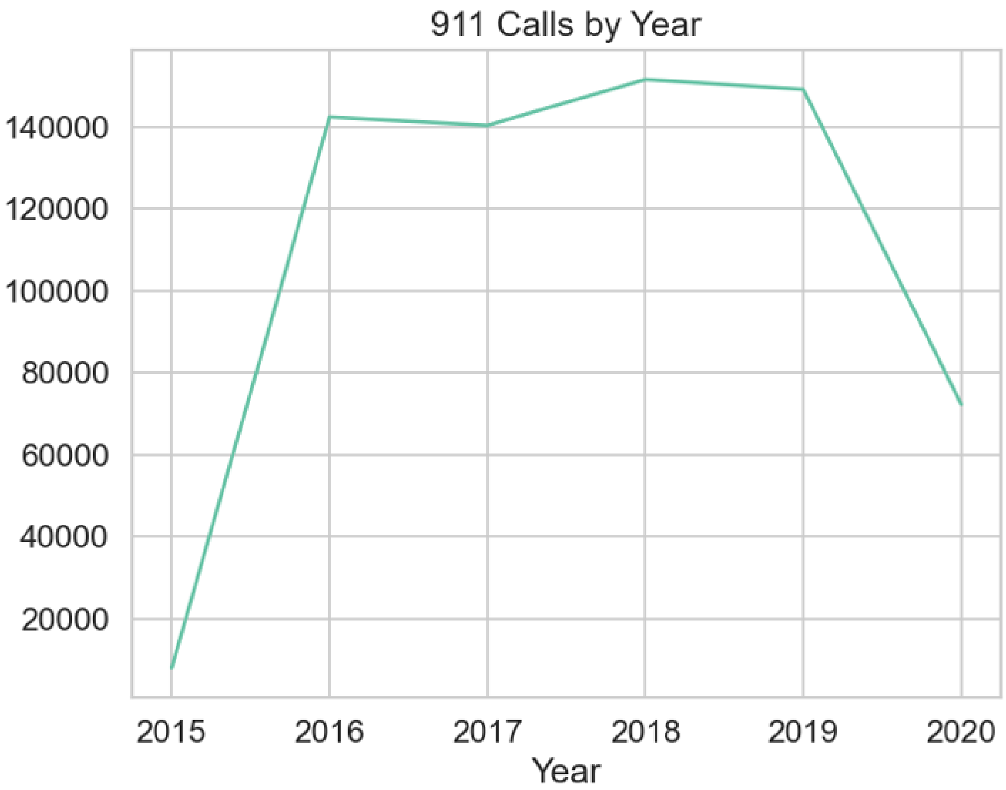


```
df['Year'] = df['timeStamp'].dt.year

by_year = df.groupby('Year').count()['lat'].reset_index()

plt.figure()
sns.lineplot(x='Year', y='lat', data=group_by_year)
plt.title('911 Calls by Year')
```

Add new column ‘Year’ with a **date data type.**

Group the DataFrame **by year** and count the calls.

Create a line plot to visualize the 911 calls **by year.**

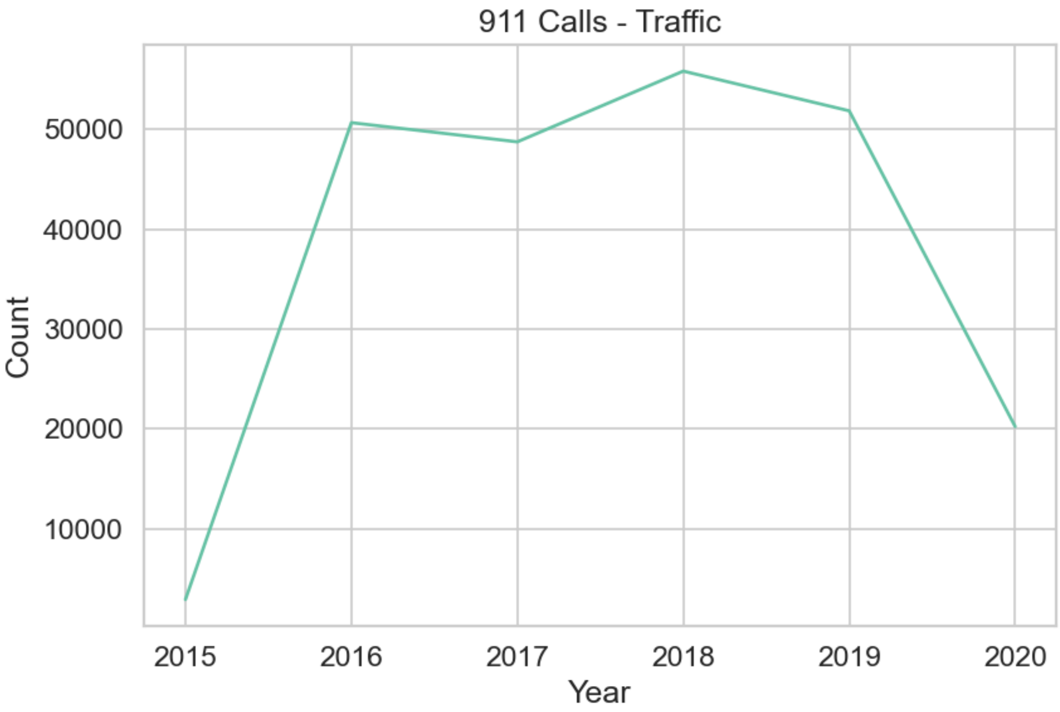


Analysis of Traffic-Related Emergency Calls by Year

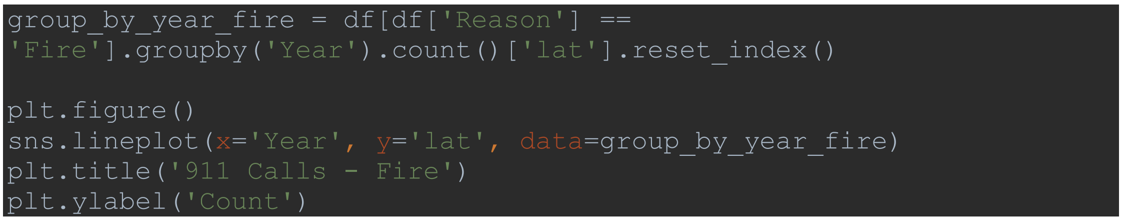
```
group_by_year_traffic = df[df['Reason'] ==
'Traffic'].groupby('Year').count()['lat'].reset_index()

plt.figure()
plt.title('911 Calls - Traffic')
sns.lineplot(x='Year', y='lat', data=group_by_year_traffic)
plt.ylabel('Count')
```

Group the DataFrame by date and count the calls for the **'Traffic'** reason.

Create a line plot to visualize the 911 calls for the 'Traffic' reason **by year.**



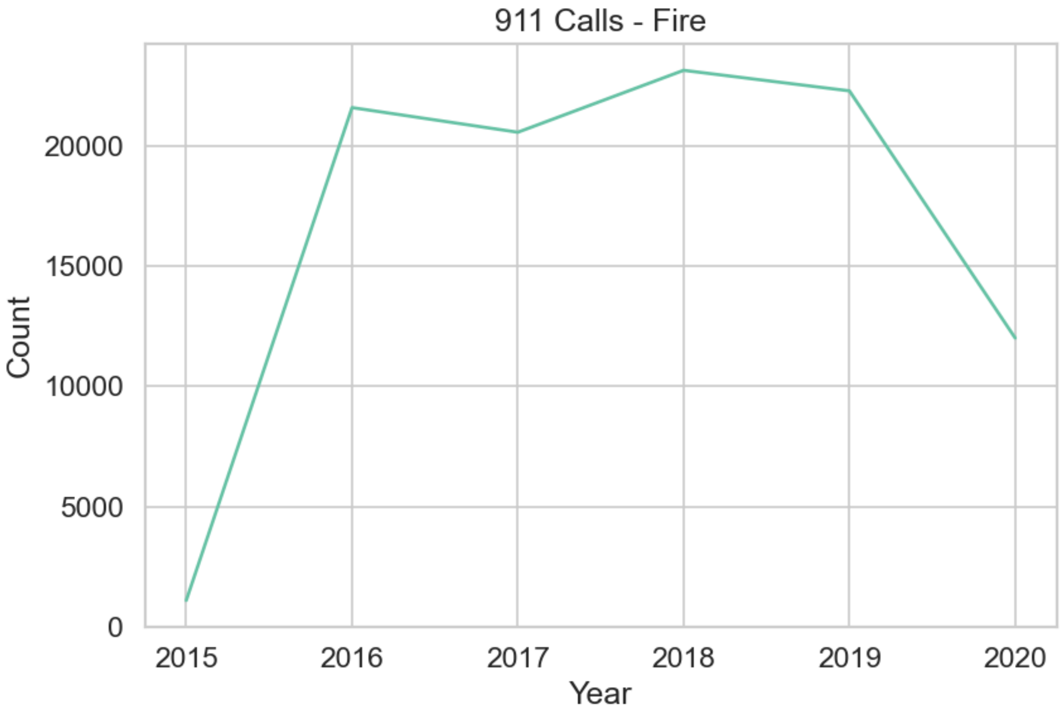
Analysis of Fire-Related Emergency Calls by Year



```
group_by_year_fire = df[df['Reason'] ==
'Fire'].groupby('Year').count()['lat'].reset_index()

plt.figure()
sns.lineplot(x='Year', y='lat', data=group_by_year_fire)
plt.title('911 Calls - Fire')
plt.ylabel('Count')
```

Group the DataFrame by year and count the calls for the **'Traffic'** reason.

Create a line plot to visualize the 911 calls for the 'Fire' reason **by year.**

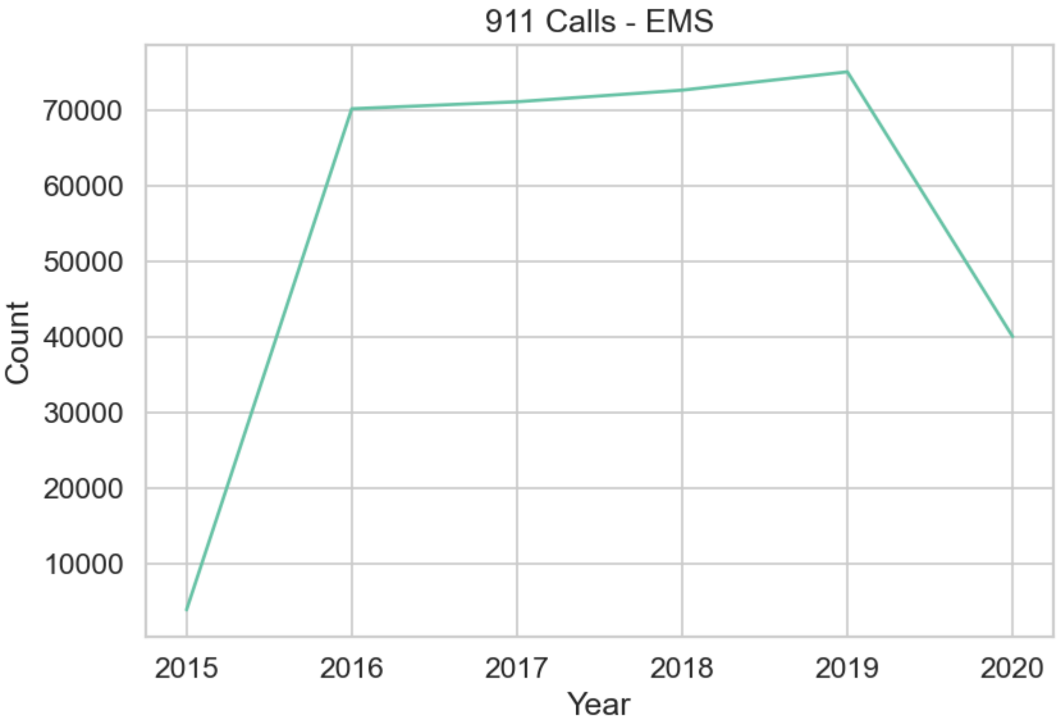


Analysis of EMS-Related Emergency Calls by Year

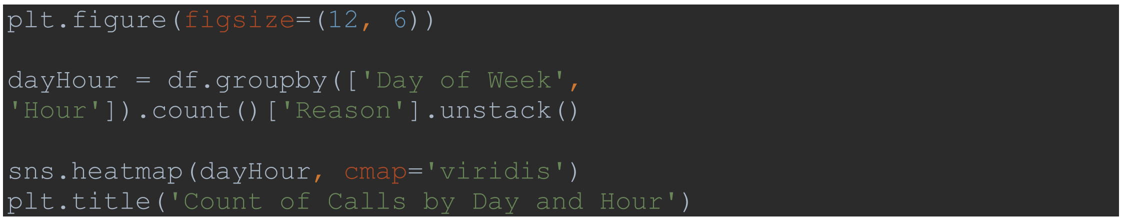
```
group_by_date_ems = df[df['Reason'] ==
'EMS'].groupby('Year').count()['lat'].reset_index()

plt.figure()
sns.lineplot(x='Year', y='lat', data=group_by_date_ems)
plt.title('911 Calls - EMS')
plt.ylabel('Count')
```

Group the DataFrame by year and count the calls for the **'EMS'** reason.

Create a line plot to visualize the 911 calls for the 'EMS' reason **by year.**

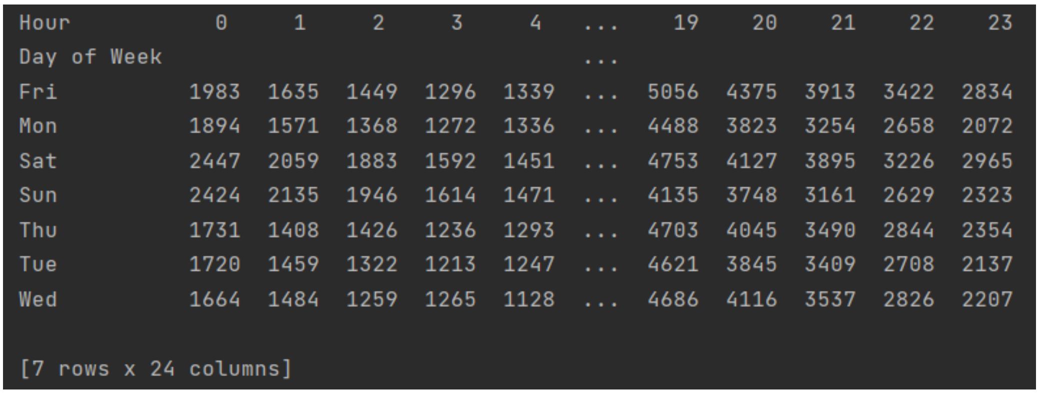


Analysis of Emergency Calls by Day and Hour

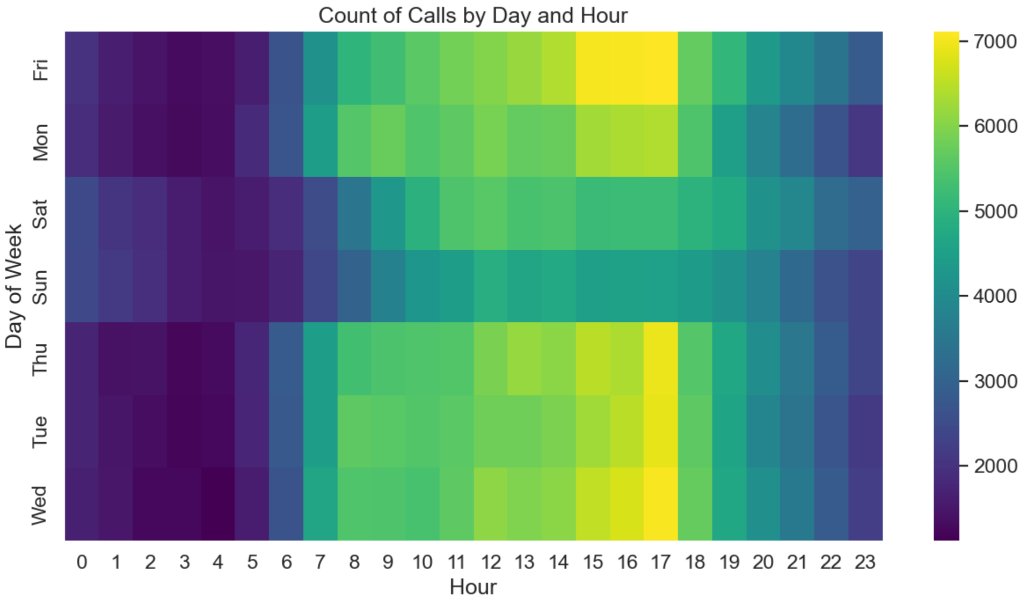


```
plt.figure(figsize=(12, 6))

dayHour = df.groupby(['Day of Week',
'Hour']).count()['Reason'].unstack()

sns.heatmap(dayHour, cmap='viridis')
plt.title('Count of Calls by Day and Hour')
```

Group the DataFrame by **day of the week and hour** and count the calls for each combination.
Create a heatmap to visualize the count of calls by day and hour.



```
Hour            0     1     2     3     4   ...    19    20    21    22    23
Day of Week                                 ...
Fri          1983  1635  1449  1296  1339   ...  5056  4375  3913  3422  2834
Mon          1894  1571  1368  1272  1336   ...  4488  3823  3254  2658  2072
Sat          2447  2059  1883  1592  1451   ...  4753  4127  3895  3226  2965
Sun          2424  2135  1946  1614  1471   ...  4135  3748  3161  2629  2323
Thu          1731  1408  1426  1236  1293   ...  4703  4045  3490  2844  2354
Tue          1720  1459  1322  1213  1247   ...  4621  3845  3409  2708  2137
Wed          1664  1484  1259  1265  1128   ...  4686  4116  3537  2826  2207

[7 rows x 24 columns]
```


Count of Calls by Day and Hour
Day of Week
Fri
Mon
Sat
Sun
Thu
Tue
Wed
Hour
