LEGALST 123 Problem Set 1

written by Kloe Yap

1 Background

San Francisco is among the top California counties for its rate of property and violent crime (California Criminal Justice Statistics Center, 2023). With a population of roughly 900,000 persons and close to 2000 sworn police officers the city has experienced growing crime. In this problem set, I am exploring the crime patterns in San Francisco, whether some crimes are more likely to occur in some neighboorhoods over others, and the effects of policing strategies.

In [1]:
```
# import necessary libraries
import pandas as pd
import geopandas as gpd
import requests
import io
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import folium
from branca.colormap import linear
import branca.colormap
import folium.plugins # The Folium Javascript Map Library
from folium.plugins import HeatMap
from IPython.display import display, HTML
```

```
/opt/conda/lib/python3.9/site-packages/geopandas/_compat.py:111: UserWarnin
g: The Shapely GEOS version (3.10.3-CAPI-1.16.1) is incompatible with the G
EOS version PyGEOS was compiled with (3.10.4-CAPI-1.16.2). Conversions betw
een both will be slow.
  warnings.warn(
```

2 Get the Data

In [2]:
```
r = requests.get("https://data.sfgov.org/api/views/wg3w-h783/rows.csv?access
incident_reports = pd.read_csv(io.StringIO(r.text), usecols=["Incident Numbe
                                                             "Incident Time",
                                                             "Report Datetime
                                                             "Analysis Neighb

incident_reports
```

Out[2]:

	Incident Datetime	Incident Date	Incident Time	Incident Year	Incident Day of Week	Report Datetime	Incident Number	I Ca
0	2021/07/25 12:00:00 AM	2021/07/25	00:00	2021	Sunday	2021/07/25 01:41:00 PM	216105573	
1	2022/06/28 11:58:00 PM	2022/06/28	23:58	2022	Tuesday	2022/06/28 11:58:00 PM	220264913	O
2	2022/03/11 10:30:00 AM	2022/03/11	10:30	2022	Friday	2022/03/11 08:03:00 PM	226040232	P
3	2021/05/15 05:47:00 PM	2021/05/15	17:47	2021	Saturday	2021/05/15 05:47:00 PM	210183345	Rec
4	2022/06/28 05:22:00 PM	2022/06/28	17:22	2022	Tuesday	2022/06/28 05:22:00 PM	220361741	Rec
...	...	...	...	...	...	...	...	
701676	2023/01/25 12:00:00 AM	2023/01/25	00:00	2023	Wednesday	2023/02/23 12:38:00 PM	230132742	M M
701677	2023/02/23 06:05:00 PM	2023/02/23	18:05	2023	Thursday	2023/02/23 08:15:00 PM	230134061	
701678	2022/10/31 12:00:00 AM	2022/10/31	00:00	2022	Monday	2023/02/22 02:05:00 PM	230130348	
701679	2022/02/23 01:10:00 PM	2022/02/23	13:10	2022	Wednesday	2023/02/23 06:51:00 PM	230133944	
701680	2023/02/23 08:00:00 AM	2023/02/23	08:00	2023	Thursday	2023/02/23 03:54:00 PM	230133392	Sus

701681 rows × 12 columns

When getting the data from the database, I have started doing a little data cleaning. I made sure to only get the columns I needed for my analysis to keep the storage low and lesser columns made it easier to look at the data as a whole.

In [3]:
```
len(incident_reports.dropna(subset=["Latitude", "Longitude"]))
```

Out[3]: 664456

In the cell above, I wanted to check the difference it would make if I dropped the cases where the latitude and longitude were NaN values. There were not that many, but I

decided to continue working without dropping those incidents because they are still incidents that have been reported. I wanted to continue working with the original data.

In [4]:
```
police_districts = gpd.read_file('https://data.sfgov.org/resource/q52f-skbd.
police_districts.head()
```

Out[4]:

	shape_area	shape_leng	company	shape_le_1	district	geometry
0	91344142.1925	87550.2751419	B	100231.353916	SOUTHERN	MULTIPOLYGON (((-122.39186 37.79425, -122.3917...
1	201384622.317	163013.798332	C	144143.480351	BAYVIEW	MULTIPOLYGON (((-122.38098 37.76480, -122.3810...
2	80623839.7922	40152.783389	D	40518.8342346	MISSION	MULTIPOLYGON (((-122.40954 37.76932, -122.4086...
3	82781685.5603	56493.858208	E	50608.3103205	NORTHERN	MULTIPOLYGON (((-122.43379 37.80793, -122.4337...
4	11072154.5623	12424.2689691	J	18796.7841847	TENDERLOIN	MULTIPOLYGON (((-122.40217 37.78626, -122.4171...

In [5]:
```
analysis_neighborhoods = gpd.read_file('https://data.sfgov.org/resource/xfcw
analysis_neighborhoods.head()
```

Out[5]:

	nhood	geometry
0	Western Addition	MULTIPOLYGON (((-122.42144 37.78557, -122.4213...
1	West of Twin Peaks	MULTIPOLYGON (((-122.46104 37.75096, -122.4605...
2	Visitacion Valley	MULTIPOLYGON (((-122.40385 37.71883, -122.4035...
3	Twin Peaks	MULTIPOLYGON (((-122.44695 37.75655, -122.4459...
4	South of Market	MULTIPOLYGON (((-122.40371 37.78404, -122.4027...

Cleaning the Data

A little more data cleaning in addition to above...

The first step I took to clean my data was to sort the incident dates by chronological order. This would later on help me with my graphs and line plots in which I wanted to be timely ordered.

Next, I wanted to fill the NaN values in the "Analysis Neighborhood" column with "No Neighbohood Recorded". This was just for the purpose of cleaner data.

Lastly, I dropped all incidents where the "Police District" column was "Out of SF". We are focusing on the incidents occurring in San Francisco, so this data is not relevant to our analysis. They are incidents where SFPD issued a warrant or responded to a case, but it did not happen in San Francisco.

In [6]:
```
# sort the dates from 2018-present
incident_reports.sort_values(by="Incident Date", inplace=True)

# column "Analysis Neighborhood"
incident_reports["Analysis Neighborhood"] = incident_reports["Analysis Neigh

# column "Police Districts" df[~df.C.str.contains("XYZ")]
incident_reports = incident_reports[~incident_reports["Police District"].str

# incident_reports.head(5)
incident_reports.head()
```

Out[6]:

	Incident Datetime	Incident Date	Incident Time	Incident Year	Incident Day of Week	Report Datetime	Incident Number	Inciden Categor
184717	2018/01/01 02:28:00 AM	2018/01/01	02:28	2018	Monday	2018/01/01 02:31:00 AM	180000348	Assau
272495	2018/01/01 01:10:00 AM	2018/01/01	01:10	2018	Monday	2018/01/01 10:35:00 PM	186000390	Larcen Thef
231542	2018/01/01 05:30:00 PM	2018/01/01	17:30	2018	Monday	2018/01/01 07:42:00 PM	180002247	Larcen Thef
263479	2018/01/01 04:00:00 PM	2018/01/01	16:00	2018	Monday	2018/01/16 10:46:00 AM	180040902	Los Propert
211811	2018/01/01 04:00:00 PM	2018/01/01	16:00	2018	Monday	2018/01/01 09:05:00 PM	186002681	Larcen Thef

3 Citywide Crime Trends

Problem 3.1

Problem 3.1 A: Provide citywide annual counts of reported incidents in a table.

In [7]:
```
pd.DataFrame(incident_reports.value_counts(["Incident Year"])).reset_index()
```

Out[7]:

	Incident Year	count
0	2018	148736
1	2019	144137
2	2022	128922
3	2021	124678
4	2020	114697
5	2023	19799

The dataframe above provides the citywide annual counts of reported incidents, with 2018 leading with the most incident reports out of the last 5 full years.

Problem 3.1 B: Provide citywide counts by month of reported incidents in a table.

In [8]:
```
incident_reports["Month"] = incident_reports["Incident Date"].str.split("/")
```

```
/tmp/ipykernel_63/180279736.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  incident_reports["Month"] = incident_reports["Incident Date"].str.split
("/").str[1]
```

In [9]:
```
incident_reports["Month"] = incident_reports["Month"].astype(int).replace([1
incident_reports.head();
```

```
/tmp/ipykernel_63/1156100260.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  incident_reports["Month"] = incident_reports["Month"].astype(int).replace
([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12], ['Jan', 'Feb', 'Mar', 'Apr', 'Ma
y', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'])
```

In [10]:
```
pd.DataFrame(incident_reports.value_counts("Month")).rename({0 : "count"}, a
```

Out[10]:

	Month	count
0	Jan	67105
1	Feb	60459
2	Oct	58708
3	Aug	58030
4	Jul	57908
5	Sep	56610
6	Dec	55044
7	Nov	54658
8	Jun	54095
9	May	54046
10	Mar	53416
11	Apr	50890

January leads all the months with the most reported incidents, followed by February and October.

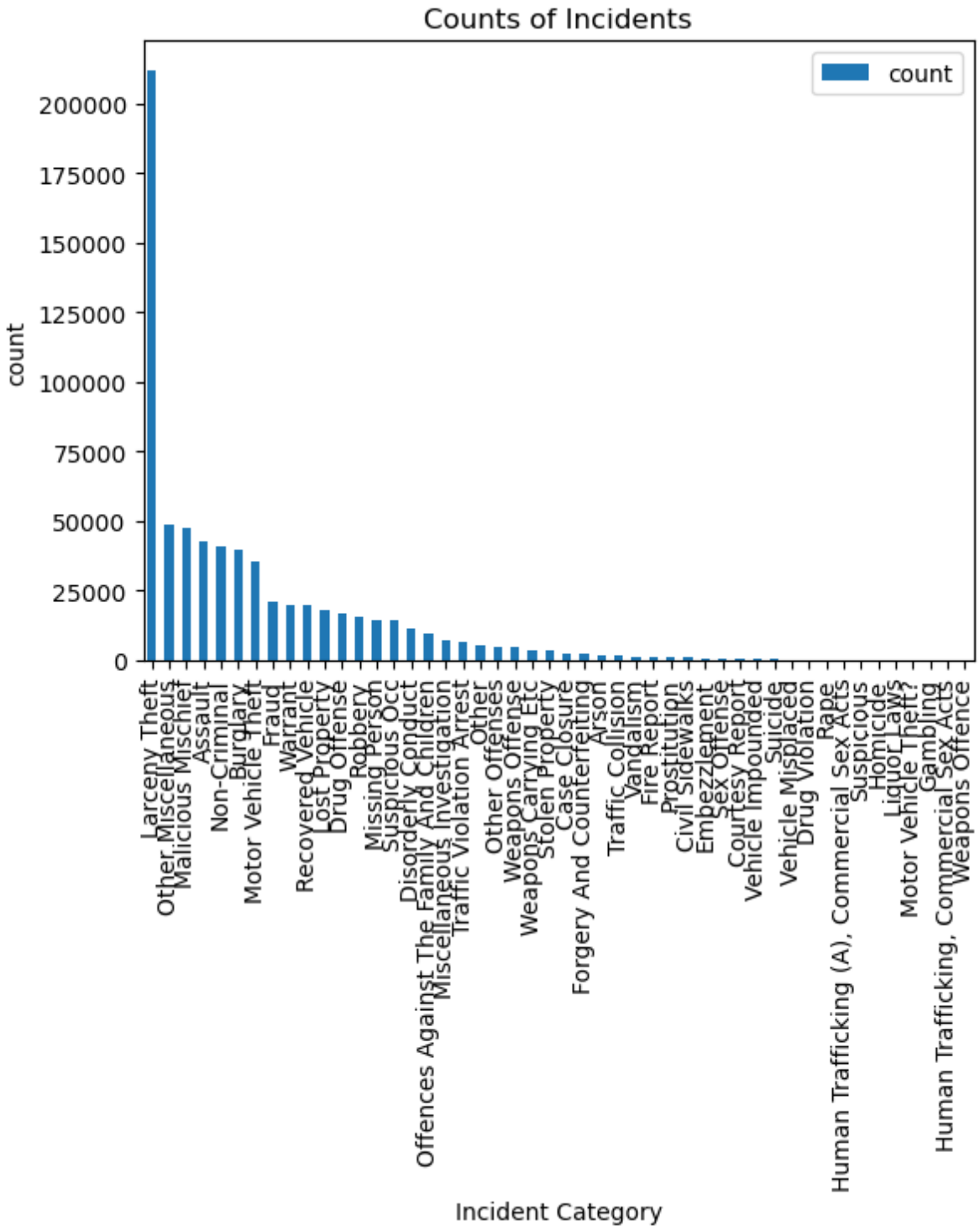
Problem 3.1 C: Calculate and plot the cumulative distribution of incidents by category.

In [11]:
```
table_cols = incident_reports.columns
report_type_data = pd.DataFrame(incident_reports.value_counts("Incident Cate
report_type_data.head()
```

Out[11]:

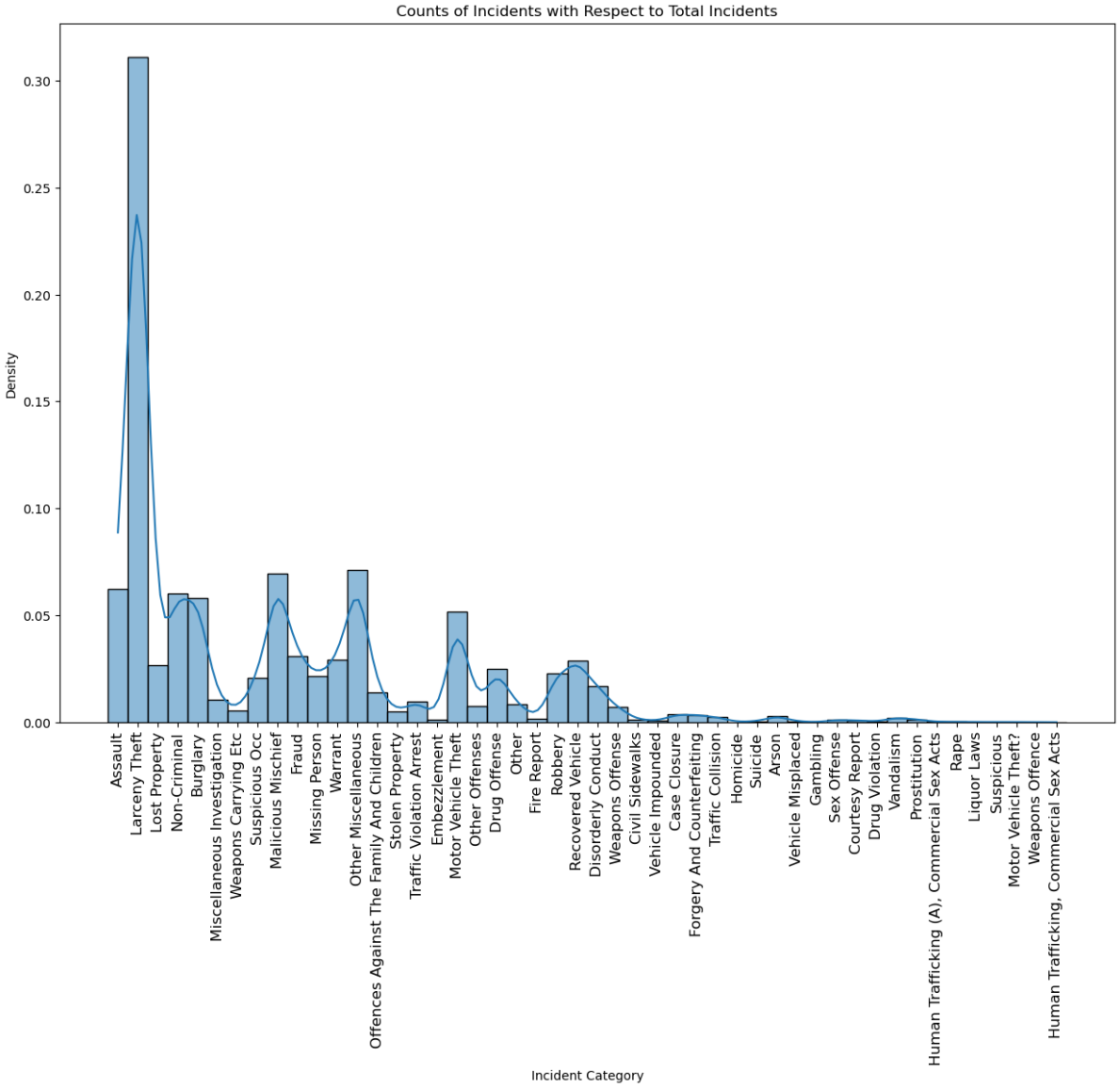
	Incident Category	count
0	Larceny Theft	211783
1	Other Miscellaneous	48582
2	Malicious Mischief	47343
3	Assault	42413
4	Non-Criminal	40940

In [12]:
```
report_type_data.plot.bar(x="Incident Category", y="count")
plt.ylabel("count")
plt.title("Counts of Incidents");
```



In [13]:
```
plt.figure(figsize=(15,10))
sns.histplot(data=incident_reports, x="Incident Category", kde=True, stat='c
plt.xticks(rotation=90, fontsize = 'large');
plt.title('Counts of Incidents with Respect to Total Incidents')
```

Out[13]:
```
Text(0.5, 1.0, 'Counts of Incidents with Respect to Total Incidents')
```



This first bar plot above contains the count data for each incident category. In SF, larceny theft leads all the other incidents by a wide range at over 200,000 incidents reported. Then, in the 2nd bar plot, I am plotting the density. This gives us the measurement with respect to the total number of incidents reported.

Problem 3.1 D: Identify the top ten most frequent incidents.

In [14]:
```
top_ten = incident_reports.value_counts("Incident Category").head(10)
pd.DataFrame(top_ten).reset_index().rename(columns = {0: "count"})
```

Out[14]:

	Incident Category	count
0	Larceny Theft	211783
1	Other Miscellaneous	48582
2	Malicious Mischief	47343
3	Assault	42413
4	Non-Criminal	40940
5	Burglary	39412
6	Motor Vehicle Theft	35330
7	Fraud	21157
8	Warrant	20004
9	Recovered Vehicle	19613

These are the top 10 most frequent incidents being reported in San Francisco. Larceny theft is the mostly frequently reported incident type, followed by "Other Miscellaneous" and "Malicious Mischief".

Problem 3.1 E: Provide a single visualization of the monthly counts of incidents for the ten top categories (make sure your audience can understand it).

In [15]:
```
incident_categories = incident_reports["Incident Category"].unique()
top_ten_categories = top_ten.index
```

In [16]:
```
incident_reports["Month and Year"] = incident_reports["Month"] + " " + incic
# incident_reports["Month and Year"].value_counts()
top_ten_filter = incident_reports[incident_reports["Incident Category"].isir
monthly_counts = top_ten_filter.pivot_table(index="Month and Year", columns=
                                            aggfunc='size', fill_value=0,
monthly_counts
```

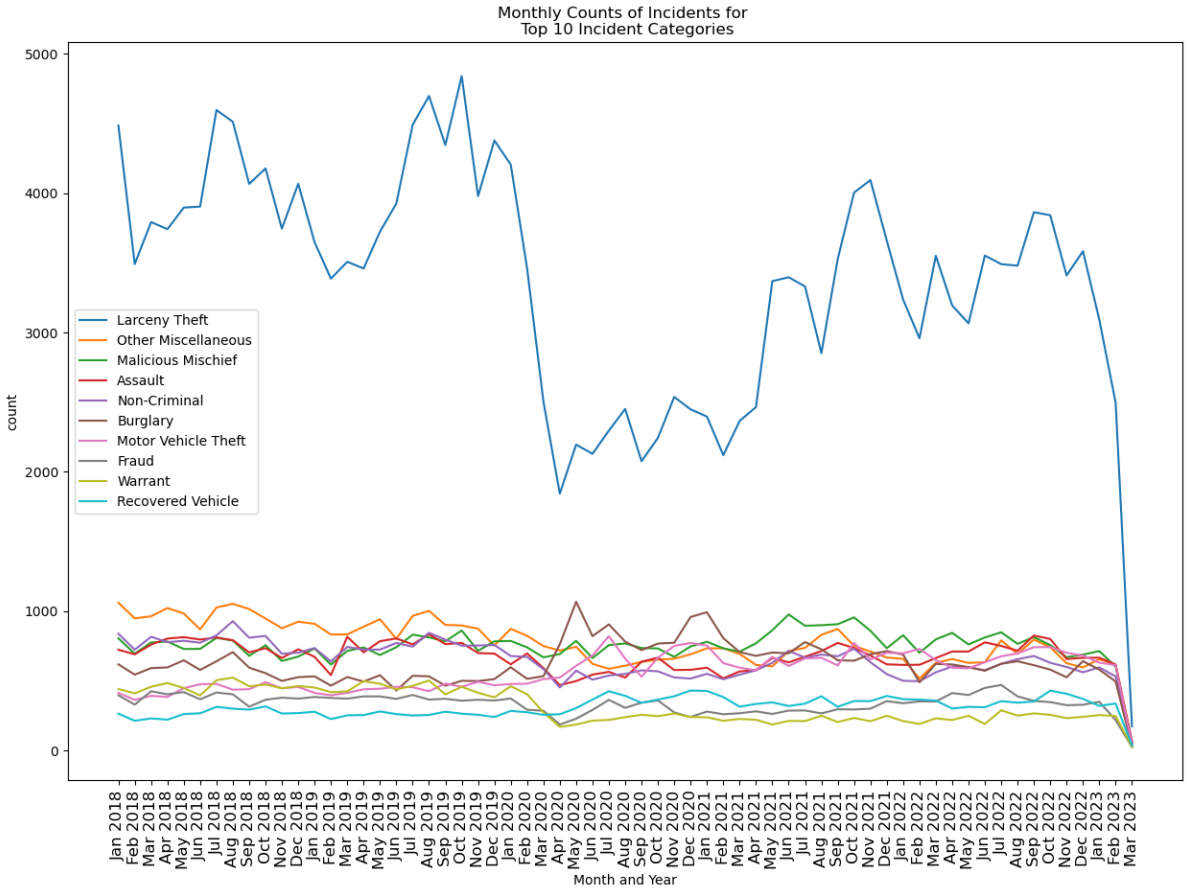
```
/tmp/ipykernel_63/3546192096.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  incident_reports["Month and Year"] = incident_reports["Month"] + " " + in
cident_reports["Incident Year"].astype("str")
```

Out[16]:

Incident Category	Assault	Burglary	Fraud	Larceny Theft	Malicious Mischief	Motor Vehicle Theft	Non-Criminal	Other Miscellaneous	Re
Month and Year									
Jan 2018	723	618	396	4485	805	413	838	1060	
Feb 2018	690	544	329	3491	694	363	723	948	
Mar 2018	760	591	425	3792	775	392	816	963	
Apr 2018	803	596	403	3741	782	384	777	1022	
May 2018	813	647	419	3896	728	447	787	983	
...	...	...	...	...	...	...	...	...	
Nov 2022	656	525	325	3409	669	701	599	626	
Dec 2022	665	641	328	3583	688	681	561	596	
Jan 2023	666	580	349	3092	713	630	596	655	
Feb 2023	617	495	218	2495	602	613	531	614	
Mar 2023	63	28	25	173	51	38	59	56	

63 rows × 10 columns

In [17]:
```
plt.figure(figsize=(15,10))
for cat in top_ten_categories:
    sns.lineplot(data = monthly_counts, x=monthly_counts.index, y= monthly_c
plt.xticks(rotation=90, fontsize = 'large');
plt.title("Monthly Counts of Incidents for \n Top 10 Incident Categories")
plt.ylabel("count");
```



The line plot above shows the number of each type of incident for the top 10 incident categories and its change over time. Specifically, this line plot combines data from each month and we can see when the counts of each incident type increases or decreases in respect to time. What we can see is that there is a huge dip around March 2020 for most of the incidents and we know that this is when lockdown started for COVID-19, so there was less going on.

Besides larceny theft, the other incident types have pretty consistent counts throughout each month. Burglary does increase around May 2020, but remains consistent throughout.

Problem 3.1 F: Provide a summary report (in no more than 200 words) of the descriptive statistics that highlights what (in your judgment) your audience needs to know.

The charts generated above revealed some interesting findings, including that January had the highest number of incident reports compared to all other months. While this may not be an outlier, it still stands out and raises questions about why January saw more incidents. One possibility is that the start of a new year and the active holiday season may have contributed to the increase in crime and incidents during that month.

Larceny theft emerged as the most frequent type of crime in the police incident report dataset, and its line on the line plot was significantly higher than the other incident

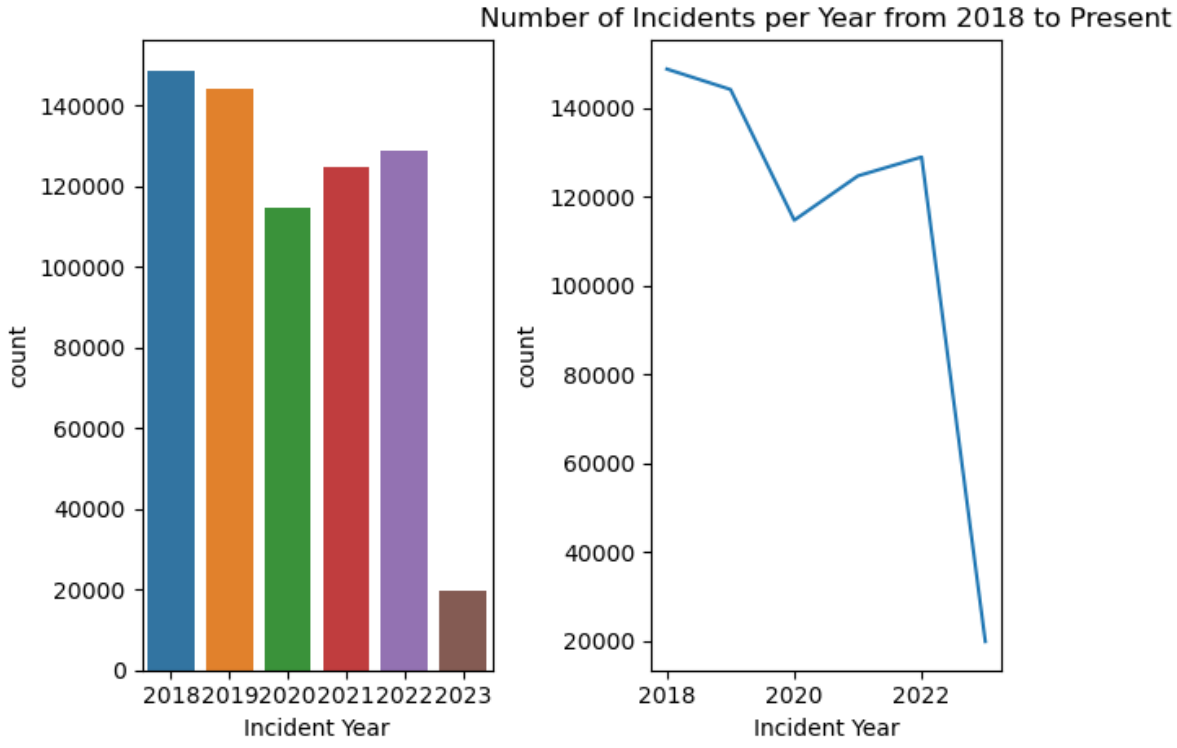
types. The peaks in the line suggest that the timing of the month and year had a significant impact on the number of reported incidents. Interestingly, there was a dip in the number of incidents reported in March 2020, likely due to the COVID-19 pandemic and stay-at-home orders.

Out of all the full years that have passed (2018, 2019, 2020, 2021, 2022), 2018 had the most incident reports. Although it had the largest count, it is not considered an outlier. 2019 only had ~4000 less reports, so this could have occurred by chance, but could also be caused by changes police behavior, policy changes, economic factors, and much much more.

Problem 3.2

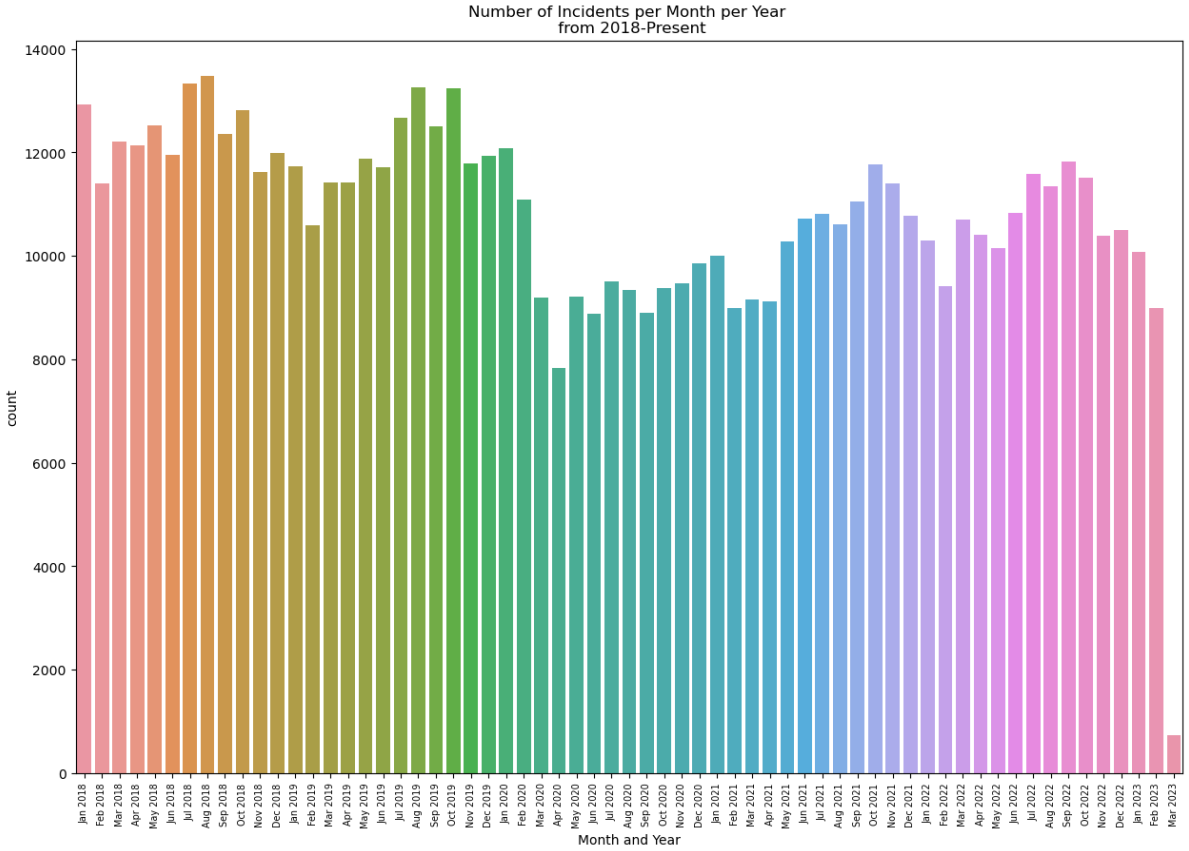
Plot the number of incidents per year from 2018 to present, then plot the number of incidents per month per year from 2018 to present. What do the two plots you just produced tell you about the incident reports?

In [18]:
```
fig, ax =plt.subplots(1,2)
num_2018_present = incident_reports.groupby("Incident Year")[["Incident Year
sns.barplot(data = num_2018_present, x="Incident Year", y="count", ax=ax[0])
sns.lineplot(data = num_2018_present, x="Incident Year", y="count", ax=ax[1]
plt.title("Number of Incidents per Year from 2018 to Present");
plt.tight_layout();
```



In [19]:
```
plt.figure(figsize=(15,10))
num_month_year_count = incident_reports.groupby([("Month and Year")], sort=F
sns.barplot(data = num_month_year_count, x="Month and Year", y="count")
```

```
plt.xticks(rotation=90, fontsize = 'x-small');
plt.title("Number of Incidents per Month per Year \n from 2018-Present");
```



The two plots I plotted above showed me how the number of incident reports changed over time. In the first bar plot I created, it shows the number of incident reports reported every year from 2018 to present, with 2018 having the most incident reports and 2020 having the least. 2023 just started, so it cannot count as the least. To support the first 2 plots is the 2nd one I plotted right above. The second bar plot plots the number of incidents per month per year and as expected, in the year where there was the most incidents reported (2018), the height of the bars are also taller for every month in that year.

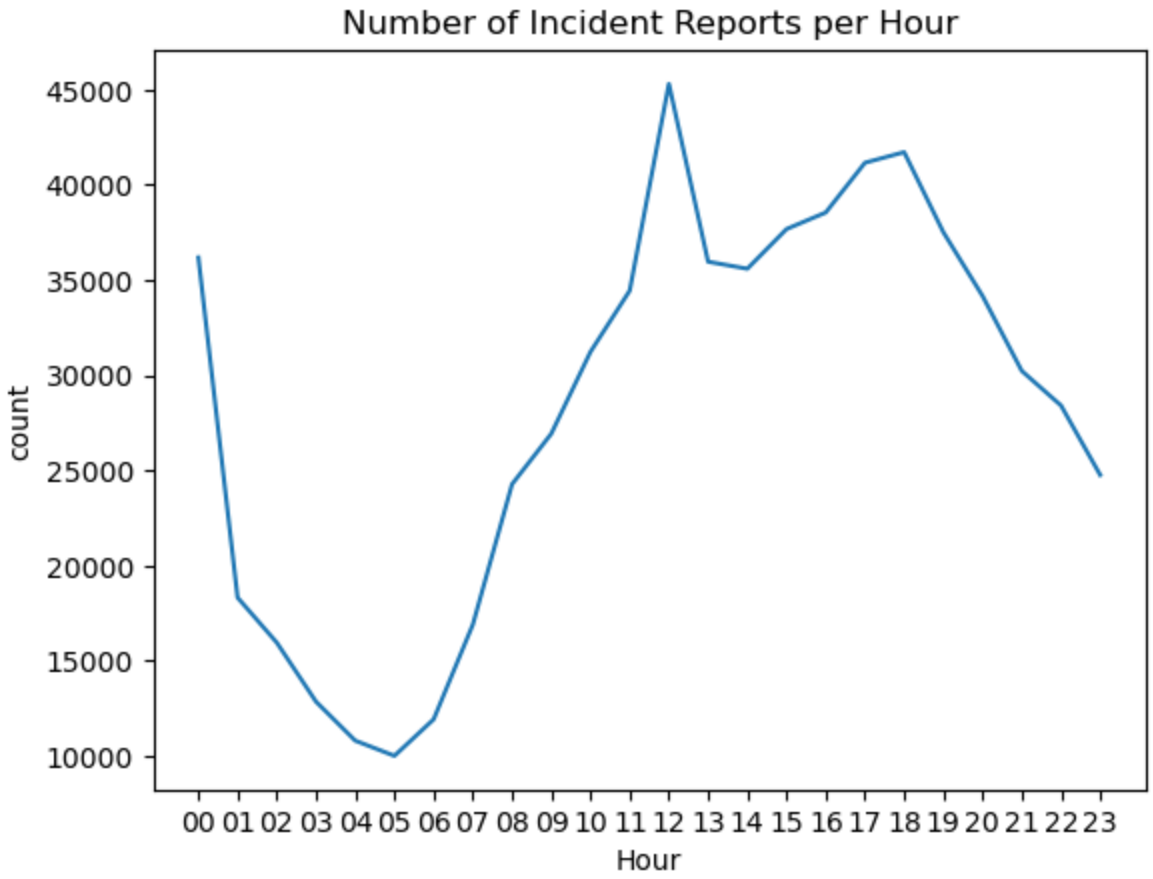
Problem 3.3

Summarize the data over other time measures, day of week and time of day. Is there a relationship between day of week, time, and whether an incident occurs? Provide at least one visualization to help readers understand the data at this level.

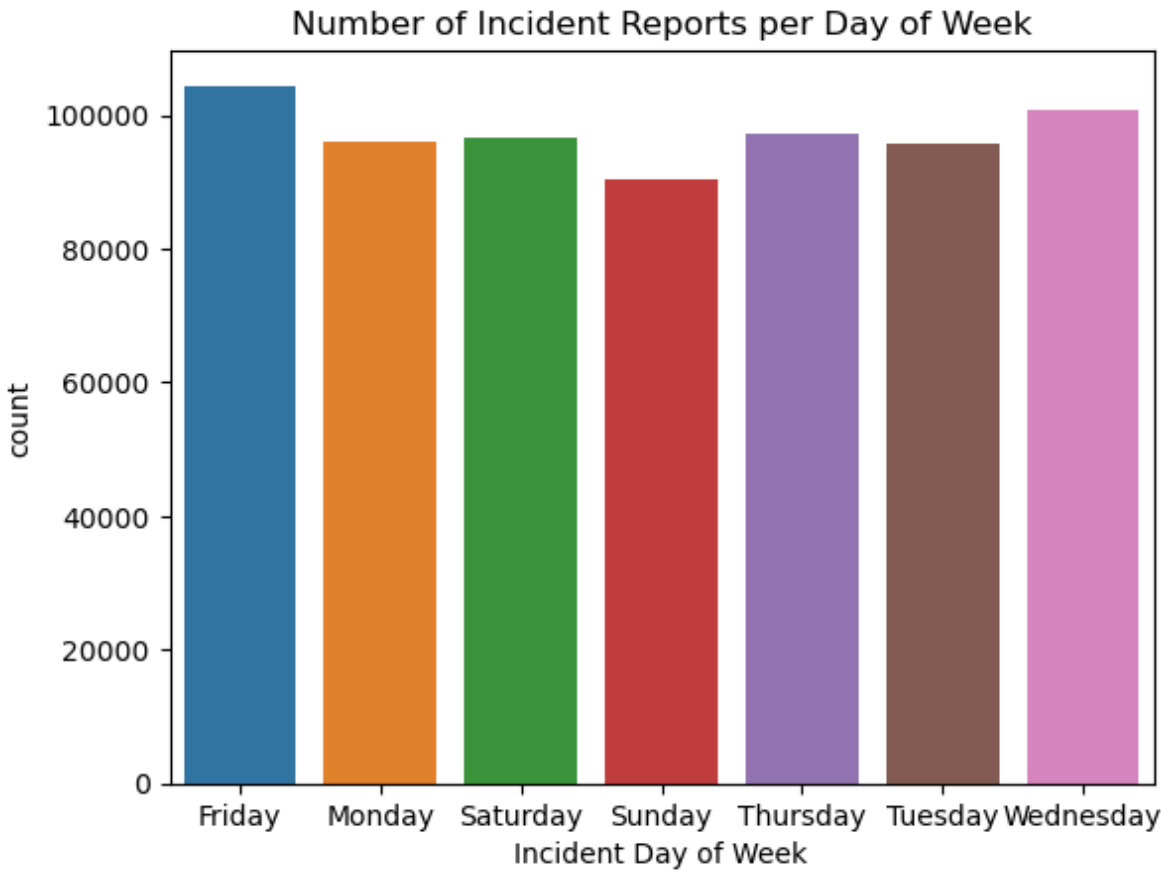
In [20]:
```
incident_reports["Hour"] = incident_reports["Incident Time"].replace(r":\d\c
```

```
/tmp/ipykernel_63/413280398.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  incident_reports["Hour"] = incident_reports["Incident Time"].replace
(r":\d\d", "", regex=True)
```

In [21]:
```
inc_per_hour = incident_reports.groupby("Hour")[["Hour"]].agg('size').reset_
sns.lineplot(data = inc_per_hour, x="Hour", y="count");
plt.title("Number of Incident Reports per Hour");
```



In [22]:
```
inc_per_day = incident_reports.groupby("Incident Day of Week")[["Incident Da
sns.barplot(data = inc_per_day, x="Incident Day of Week", y="count");
plt.title("Number of Incident Reports per Day of Week");
```



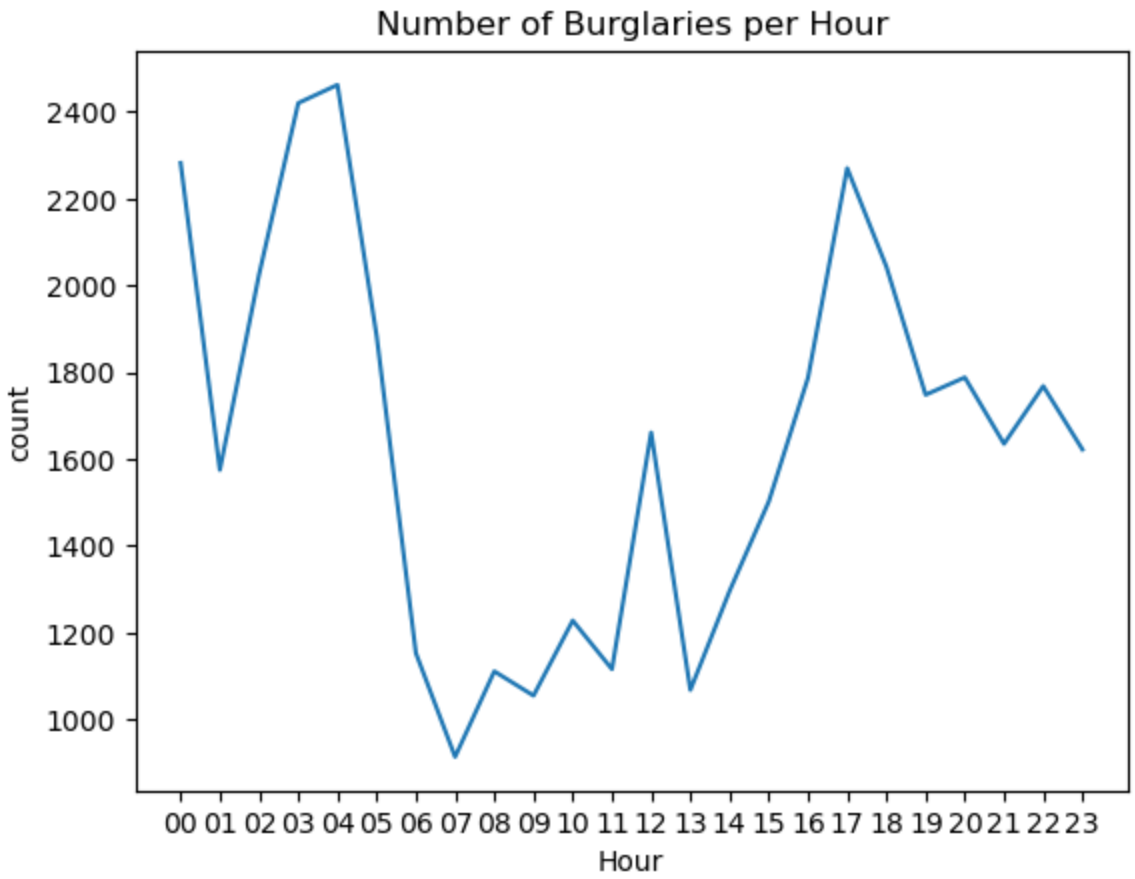
In the very first plot above, the line plot shows the number of incident reports for every hour of the day. The line plot starts off on the high at 12 AM, then slowly decreases throughout the morning. This is as expected because during the extremely early morning hours, most people are asleep, so there should be less incident reports. However, as the day goes on, the number of incident reports increases, peaking in the afternoon, 12 PM. There seems to be a relationship between the time of the day and the number of incident reports because we can see from the line plot that during the day there are lots of reports, but when the night comes and people are more likely in their homes, there are less reports.

However, there doesn't seem to be a clear relationship between the day of the week and the number of incident reports. The bar plot is roughly normal with no single day being extremely higher or lower than the others.

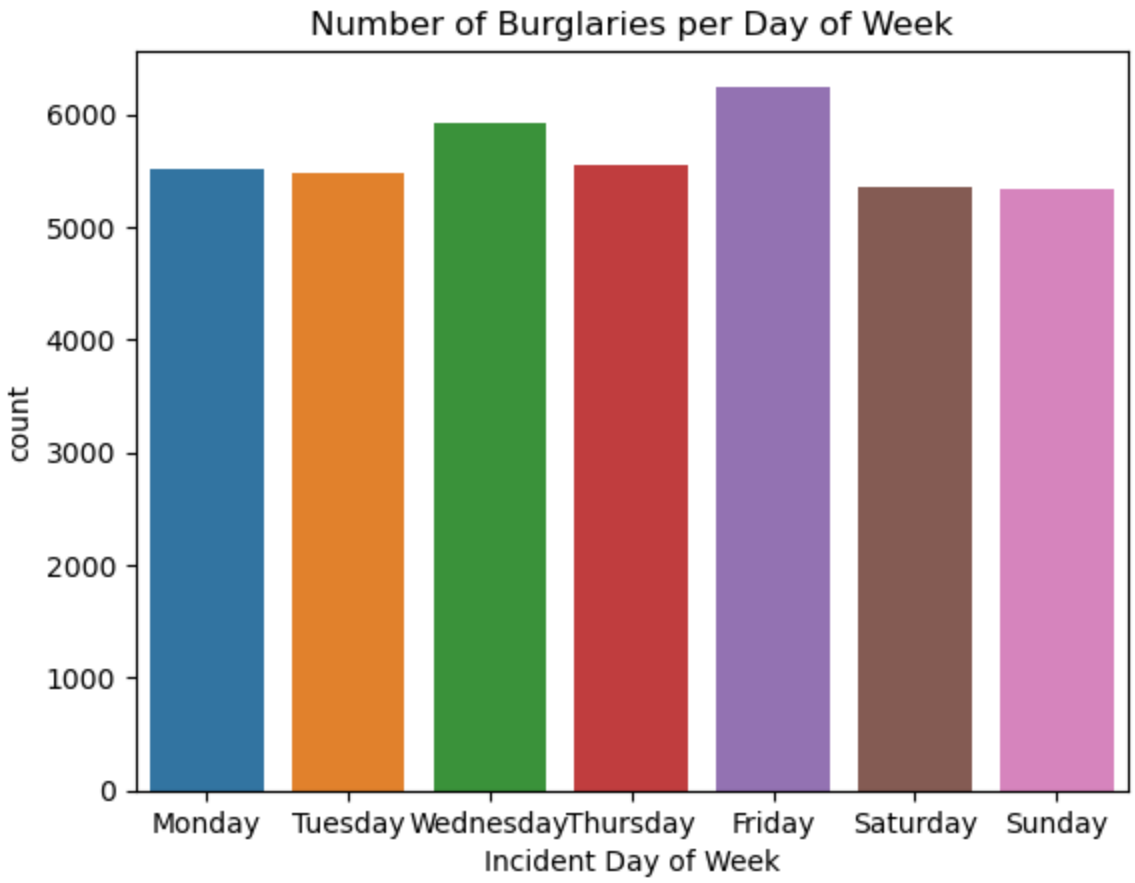
Problem 3.4

Is there a relationship between day/time and particular types of incidents? What about time of year and particular types of incidents?

In [23]:
```
burglary_per_hr = incident_reports.groupby(["Hour", "Incident Category"])[["
burglary_per_hr = burglary_per_hr[burglary_per_hr["Incident Category"] == "B
sns.lineplot(data=burglary_per_hr, x="Hour", y="count")
plt.title("Number of Burglaries per Hour");
```

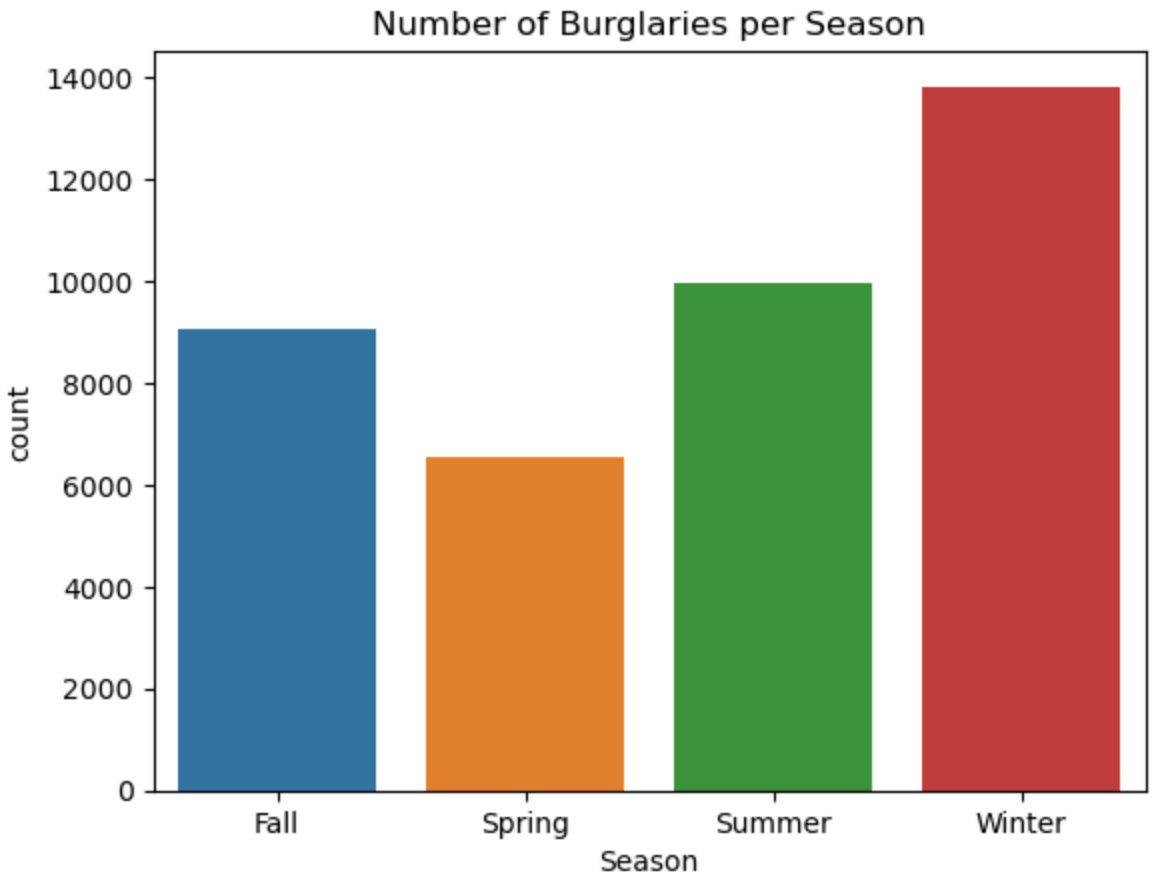


In [24]:
```
burglary_per_day = incident_reports.pivot_table(index="Incident Day of Week"
sns.barplot(data=burglary_per_day, x="Incident Day of Week", y="count")
plt.title("Number of Burglaries per Day of Week");
```



In [25]:
```
def assign_season(month):
    if (month == "Mar") | (month == "April") | (month == "May"):
        return "Spring"
    elif (month == "Jun") | (month == "Jul") | (month == "Aug"):
        return "Summer"
    elif (month == "Sep") | (month == "Oct") | (month == "Nov"):
        return "Fall"
    else:
        return "Winter"
incident_reports["Season"] = incident_reports["Month"].apply(lambda s : assi
burglary_tbl = incident_reports[incident_reports["Incident Category"] == "Bu
seasons = burglary_tbl.groupby("Season")[["Season"]].count()
seasons = seasons.rename(columns = {"Season" : "count"}).reset_index()
sns.barplot(data=seasons, x="Season", y="count")
plt.title("Number of Burglaries per Season");
# number of incidents per months
```

```
/tmp/ipykernel_63/2222379133.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  incident_reports["Season"] = incident_reports["Month"].apply(lambda s : a
ssign_season(s))
```



The incident type shown in the two plots above is focused on burglaries only. I chose burglaries because I expect there to be a relationship between the time of the day and the number of burglaries. As part of intuition, I expect that during the night hours, there should be more burglaries than daylight hours. This is evident in the line plot where we can see that there are two peaks around 3-4 AM and 5 PM, then a smaller one at 12 PM. During midnight hours, stores are closed, so thieves take advantage; hence, the higher numbers during those hours.

There doesn't seem to be much of a relationship between the day of the week and the number of burglaries. The bar plot is uniformly shaped, with Friday being the leading day with the most burglaries. However, since the counts are so close, I don't think the day of the week had much to do with the number of burglaries.

In my third plot, I wanted to see which season had the highest count of burglaries. Looking at the bar chart, the winter season has the highest count with about 14,000 burglaries reported. My assumption for this is that in the winter, many people are on vacation and thieves are aware of this, so they focus on the properties where owners are inactive in their homes.

Problem 3.5

What police districts experience the most crime? Do different police districts experience different types of crimes at different rates, or is the distribution of crime spatially consistent across police districts?

In [26]:
```
non_crime = np.array(["Recovered Vehicle", "Non-Criminal", "Fire Report", "S
                      "Vehicle Misplaced", "Liquor Laws", "Gambling", "Case C
                      "Warrant", "Lost Property", "Courtesy Report", "Civil S
crimesdf = incident_reports[~incident_reports["Incident Category"].isin(non_
crimesdf;
```

In [27]:
```
pd.DataFrame(crimesdf["Police District"].value_counts()).rename(columns = {"
```

Out[27]:

	count
Central	89844
Northern	82474
Mission	71914
Southern	68026
Tenderloin	52370
Bayview	49732
Ingleside	43575
Taraval	40464
Richmond	37311
Park	25607

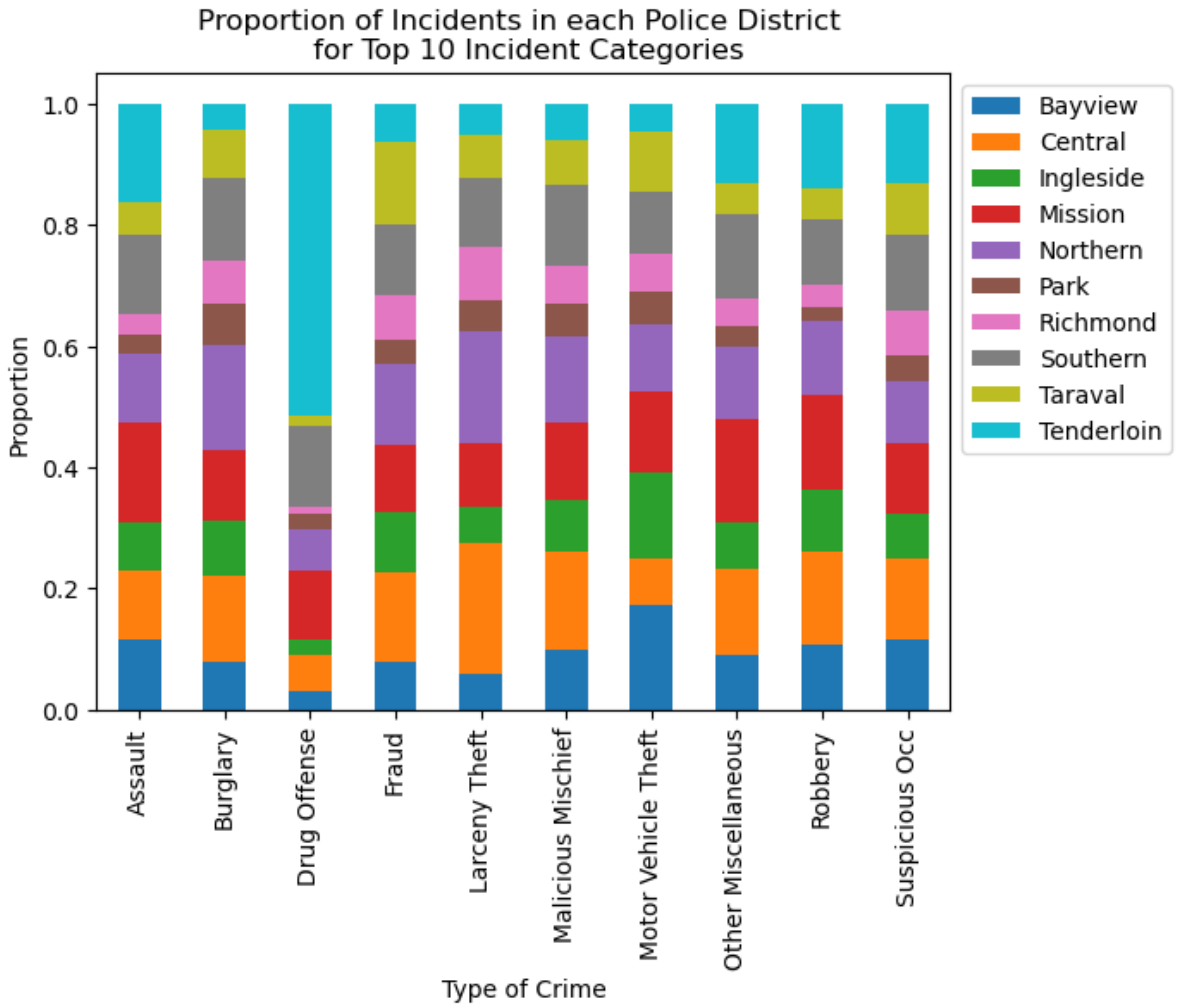
The Central police district experiences the most crime, followed by the Northern and Mission districts.

In [28]:
```
top_10 = crimesdf["Incident Category"].value_counts().head(10).index
top_10
```

Out[28]:
```
Index(['Larceny Theft', 'Other Miscellaneous', 'Malicious Mischief', 'Assau
lt',
       'Burglary', 'Motor Vehicle Theft', 'Fraud', 'Drug Offense', 'Robber
y',
       'Suspicious Occ'],
      dtype='object')
```

In [29]:
```
top_ten_tbl = crimesdf[crimesdf["Incident Category"].isin(top_10)]
district_prop = top_ten_tbl.pivot_table(index="Police District", columns="Ir
district_prop = district_prop / district_prop.sum()

district_prop = district_prop.T
district_prop = district_prop.reset_index().rename_axis(index=None, columns=
district_prop
district_prop.plot.bar(x="Incident Category", stacked=True)
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.xlabel("Type of Crime")
```

```
plt.ylabel("Proportion")
plt.title("Proportion of Incidents in each Police District \n for Top 10 Inc
```

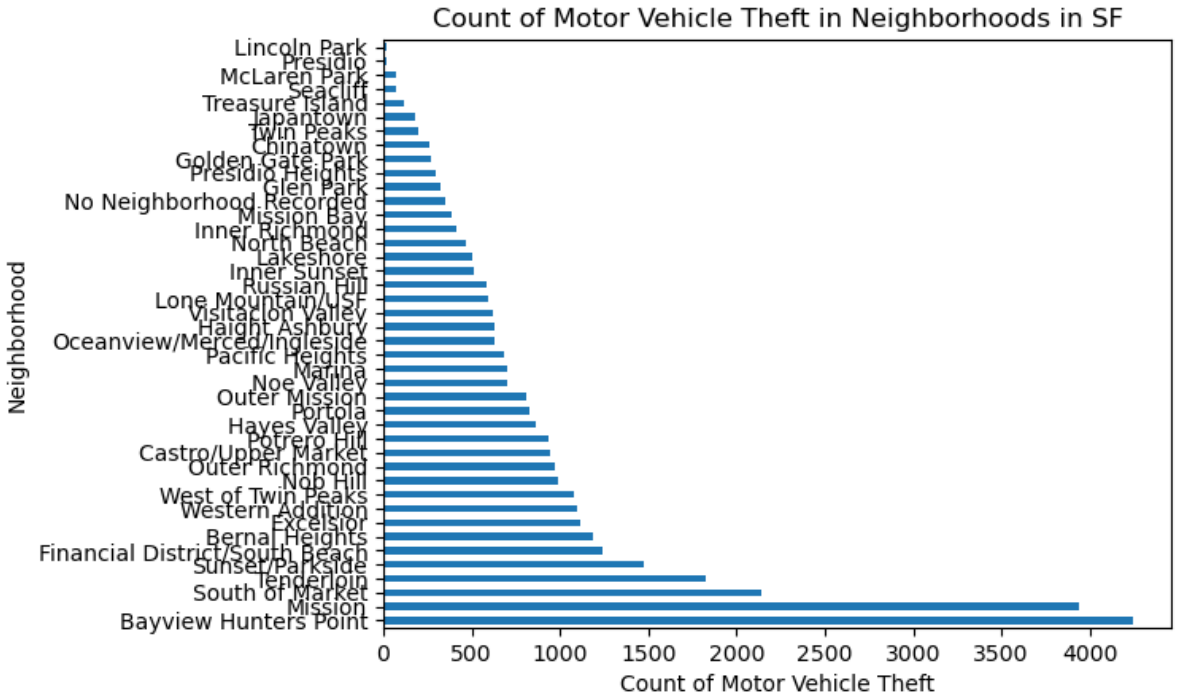


In the stacked bar chart shown above, I visualized the proportion of the top ten **crime** categories across each police district in San Francisco. For instance, in the case of burglary, approximately 10% of the incidents were reported in the Bayview District. From the chart, it is clear that the distribution of crime is roughly spatially consistent across the city's police district, without any one of the districts standing out as having a disproportionate amount of any particular type of incident except for drug offenses. Drug offenses is the only type of crime that is not evenly distributed across the district and as we can see from the stacked bar chart, about 50% of this offense comes from Tenderloin. Nonetheless, each incident type occurs in all of the police districts. These findings suggest that the occurrence of various types of incidents is not **strongly** correlated with the police district location in San Francisco.

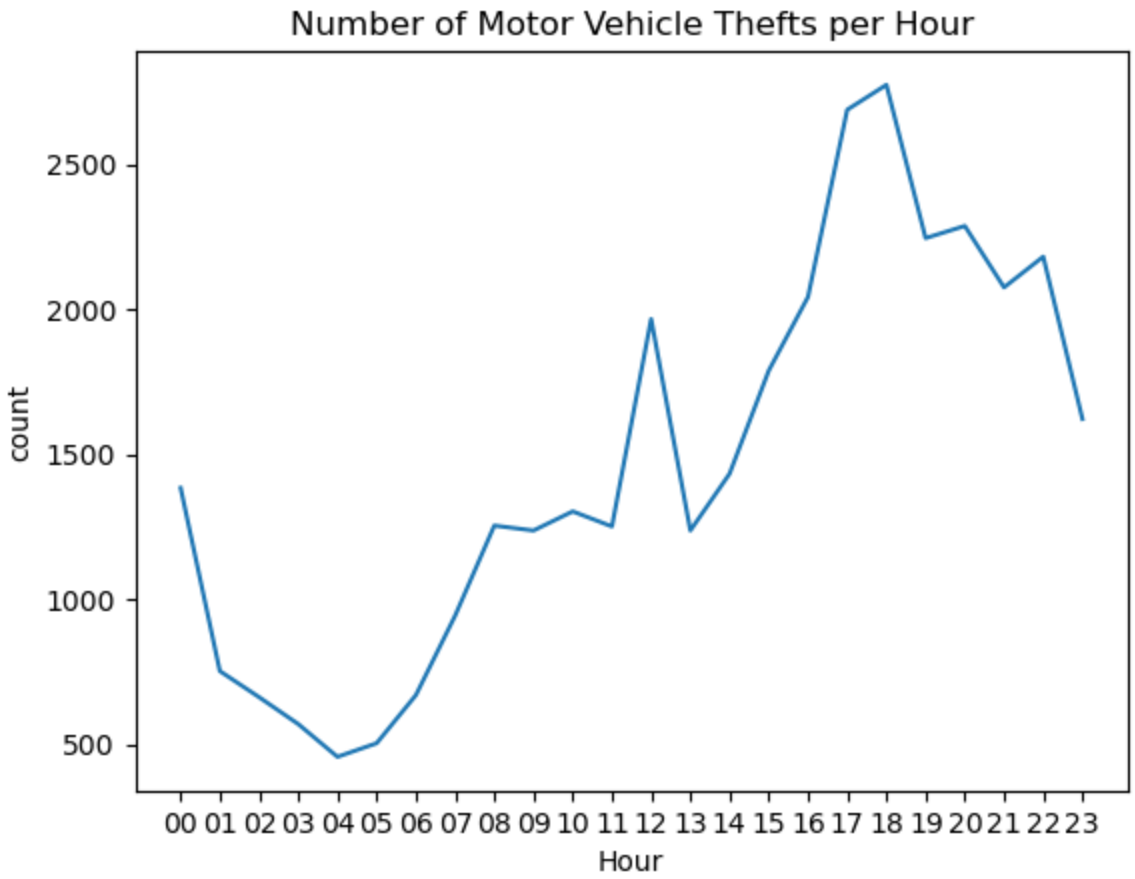
Problem 3.6

Discuss one other interesting finding from your data in a few sentences, and use a visualization.

In [30]:
```
nhood_motor = incident_reports[["Analysis Neighborhood", "Incident Category"
nhood_motor = nhood_motor[nhood_motor["Incident Category"] == "Motor Vehicle
nhood_motor["Analysis Neighborhood"].value_counts().plot(kind="barh")
plt.ylabel("Neighborhood")
plt.xlabel("Count of Motor Vehicle Theft")
plt.title("Count of Motor Vehicle Theft in Neighborhoods in SF");
```



In [31]:
```
motor_per_hr = nhood_motor.groupby("Hour")[["Hour"]].size().reset_index().re
sns.lineplot(data=motor_per_hr, x="Hour", y="count");
plt.title("Number of Motor Vehicle Thefts per Hour");
```



In the bar plot I created above, I wanted to see which neighborhood in SF had the most motor vehicle thefts. Bayview Hunters Point and Mission neighborhoods lead with the most motor vehicle thefts compared to any other neighborhood. Without much information, it is difficult to conclude the cause of this, but some assumptions I made are the lack of space in these densely populated areas of San Francisco, so more cars are forced to be parked on the streets and the neighborhoods containing more housing, so people park their cars overnight on the streets. Additionally, I made a line plot and it shows that the thefts occur later in the day, a strong indicator that thieves target vehicles when it gets dark.

4 Geographic Distribution of Crime

Problem 4.1

Plot individual incidents in 2019 as points on a map of San Francisco, delimiting the 10 police districts. You'll want to use Folium (as in the labs) and Geopandas, which extends DataFrames from pandas to GeoDataFrames, which include geographic information.

In [32]:
```
total_incidents = incident_reports.shape[0]
np.random.seed(10)
```

```
# function to determine the proportion of data each district is in
def calc_prop_district(district):
    return incident_reports[incident_reports["Police District"] == district]
# for district in incident_reports["Police District"].unique():
#     print(district, "proportion of data:", incident_reports[incident_repor

inc2019 = incident_reports[incident_reports["Incident Year"] == 2019].dropna
points = 1000

# returns a df with the # of rows by proportion to the actual data
def num_to_sample(district):
    return inc2019[inc2019["Police District"] == district].sample(round(calc
# mission_df = inc2019[inc2019["Police District"] == "Central"].sample(round
# mission_df

mapping_df = pd.DataFrame()
for district in incident_reports["Police District"].unique():
    mapping_df = mapping_df.append(num_to_sample(district))

mapping_df
```

Out[32]:

	Incident Datetime	Incident Date	Incident Time	Incident Year	Incident Day of Week	Report Datetime	Incident Number	
510323	2019/12/19 02:50:00 AM	2019/12/19	02:50	2019	Thursday	2019/12/19 03:01:00 AM	190951474	
328220	2019/02/02 08:45:00 PM	2019/02/02	20:45	2019	Saturday	2019/02/03 12:03:00 PM	190084611	Lar
471560	2019/10/30 09:32:00 AM	2019/10/30	09:32	2019	Wednesday	2019/10/30 10:05:00 AM	190820415	Mis
367530	2019/05/18 11:00:00 PM	2019/05/18	23:00	2019	Saturday	2019/05/23 09:51:00 AM	190368170	Lar
416943	2019/04/24 08:20:00 AM	2019/04/24	08:20	2019	Wednesday	2019/04/24 09:55:00 AM	196082071	Lar
...	...	...	...	...	...	...	...	
511322	2019/11/13 08:00:00 PM	2019/11/13	20:00	2019	Wednesday	2019/11/19 08:52:00 PM	196252428	Lar
546674	2019/11/12 11:45:00 PM	2019/11/12	23:45	2019	Tuesday	2019/11/12 11:45:00 PM	190858416	
525683	2019/10/27 04:00:00 PM	2019/10/27	16:00	2019	Sunday	2019/10/29 06:30:00 PM	190819139	Mot
328975	2019/01/01 10:00:00 PM	2019/01/01	22:00	2019	Tuesday	2019/01/23 07:39:00 AM	196018509	Lar
324523	2019/02/19 03:30:00 PM	2019/02/19	15:30	2019	Tuesday	2019/02/20 10:15:00 AM	190126699	Mot

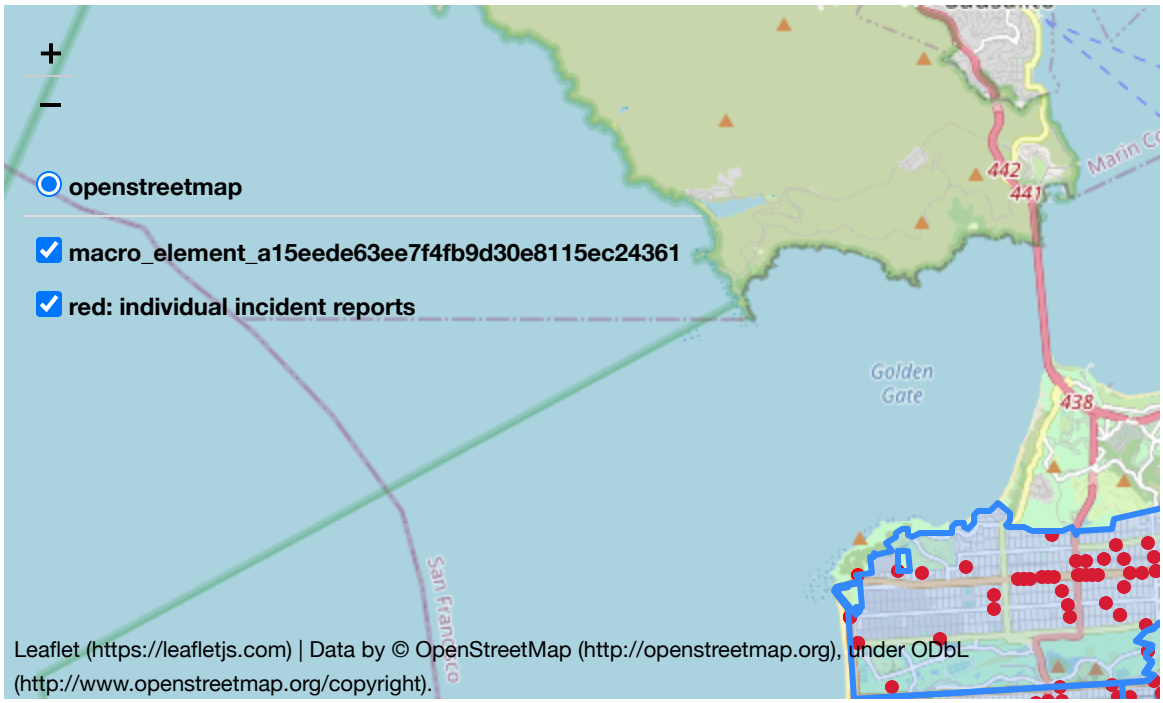
998 rows × 16 columns

The way I shrunk the number of data points to plot on the map because Folium cannot take over 600,000 points is through stratified sampling. With this strategy, I calculate the proportion of incident reports from each of the 10 districts. Using the proportion value returned, I multiplied it by the number of points I wanted to plot which was 2000. This gives me the amount I want to sample from each police district. This way, I am essentially representing each police district the same way it was represented in the data as a whole.

In [33]:
```
sf_coords = (37.76, -122.45)
sfmap = folium.Map(sf_coords, zoom_start=12)
sfmap;
```

```
# inc2019 = incident_reports[incident_reports["Incident Year"] == 2019].samp
# inc2019 = inc2019.dropna(subset=["Latitude", "Longitude"])
# folium.CircleMarker([37.775161, -122.403636], radius=2, color="red", weigh
# for i, row in inc2019.iterrows():
    # folium.Marker(location=[row['Latitude'], row['Longitude']]).add_to(sfm
mapping_df.apply(lambda row:folium.CircleMarker(location=[row["Latitude"],
                                                  row["Longitude"]], radius=
         axis=1)
folium.GeoJson(police_districts).add_to(sfmap);

# source: https://stackoverflow.com/questions/37466683/create-a-legend-on-a-
lgd_txt = '<span style="color: {col};">{txt}</span>'

for idx, color in enumerate( ['red']):
    fg = folium.FeatureGroup(name= lgd_txt.format( txt= color + ": individua
    sfmap.add_child(fg)

folium.map.LayerControl('topleft', collapsed= False).add_to(sfmap)

sfmap
```

Out[33]:



The map above shows where most of the incident reports are clustered. Each red point in the map represents an incident report from the dataset. The incidents cluster in the Tenderloin and Central District. However, in relative to the size of the districts, Tenderloin receives a good proportion of incident reports.

4.1 A: Over what spatial unit does crime seem to cluster?

In [34]:

```
# inc2019 = incident_reports[incident_reports["Incident Year"] == 2019]
# inc2019 = inc2019.dropna(subset=["Latitude", "Longitude"])
```

In [35]:
```
non_crime = np.array(["Recovered Vehicle", "Non-Criminal", "Fire Report", "S
                      "Vehicle Misplaced", "Liquor Laws", "Gambling", "Case C
                      "Warrant", "Lost Property", "Courtesy Report", "Civil S
```

In the array above, I picked out the categories that are not considered a crime. I picked these specifc ones in the array non_crime because they either do not involve violation of laws, like fire report, or do not have an intention to commit crime.

In [36]:
```
crimes_2019 = incident_reports[incident_reports["Incident Year"] == 2019].dr
crimes_2019 = crimes_2019[~crimes_2019["Incident Category"].isin(non_crime)]
crimes_2019.head(5)
```

Out[36]:

	Incident Datetime	Incident Date	Incident Time	Incident Year	Incident Day of Week	Report Datetime	Incident Number	Inc Cat
385348	2019/01/01 12:00:00 PM	2019/01/01	12:00	2019	Tuesday	2019/01/01 02:00:00 PM	190001245	Motor V
405447	2019/01/01 09:24:00 PM	2019/01/01	21:24	2019	Tuesday	2019/01/01 09:26:00 PM	190002265	Vic
340107	2019/01/01 02:39:00 AM	2019/01/01	02:39	2019	Tuesday	2019/01/01 02:39:00 AM	190000407	Miscella
347047	2019/01/01 06:00:00 PM	2019/01/01	18:00	2019	Tuesday	2019/01/02 10:07:00 AM	196000675	Larceny
346314	2019/01/01 09:26:00 PM	2019/01/01	21:26	2019	Tuesday	2019/01/01 09:29:00 PM	190002287	Larceny

In [37]:
```
crimes = crimes_2019["Incident Category"].unique()
crimes
```

Out[37]:
```
array(['Motor Vehicle Theft', 'Traffic Violation Arrest',
       'Other Miscellaneous', 'Larceny Theft', 'Fraud', 'Suspicious Occ',
       'Malicious Mischief', 'Drug Offense', 'Disorderly Conduct',
       'Burglary', 'Stolen Property', 'Assault',
       'Offences Against The Family And Children', 'Weapons Offense',
       'Other Offenses', 'Motor Vehicle Theft?', 'Robbery',
       'Forgery And Counterfeiting', 'Other', 'Weapons Carrying Etc',
       'Embezzlement', 'Arson', 'Miscellaneous Investigation',
       'Traffic Collision', 'Rape', 'Vandalism', 'Vehicle Impounded',
       'Sex Offense', 'Drug Violation', 'Prostitution',
       'Human Trafficking (A), Commercial Sex Acts', 'Weapons Offence',
       'Homicide', 'Human Trafficking, Commercial Sex Acts'], dtype=object)
```

Below is how I performed a stratified sample to shrink my dataset once again to 1000 points. I used the proportion of each incident type from the whole dataset, then

multiplied it to 1000. I, then, multiplied the proportion to 1000 and that number is the number I want to sample from using the filtered data.

In [38]:
```
total_crimes = crimes_2019.shape[0]

np.random.seed(10)
# function to determine the proportion of data each district is in
def calc_prop_crime(crime):
    return crimes_2019[crimes_2019["Incident Category"] == crime].shape[0] /

pointss = 1000

# returns a df with the # of rows by proportion to the actual data
def sampled_per_crime_df(crime):
    return crimes_2019[crimes_2019["Incident Category"] == crime].sample(rou

mapping_crimes_df = pd.DataFrame()
for crime in crimes_2019["Incident Category"].unique():
    mapping_crimes_df = mapping_crimes_df.append(sampled_per_crime_df(crime)

mapping_crimes_df.head()
```

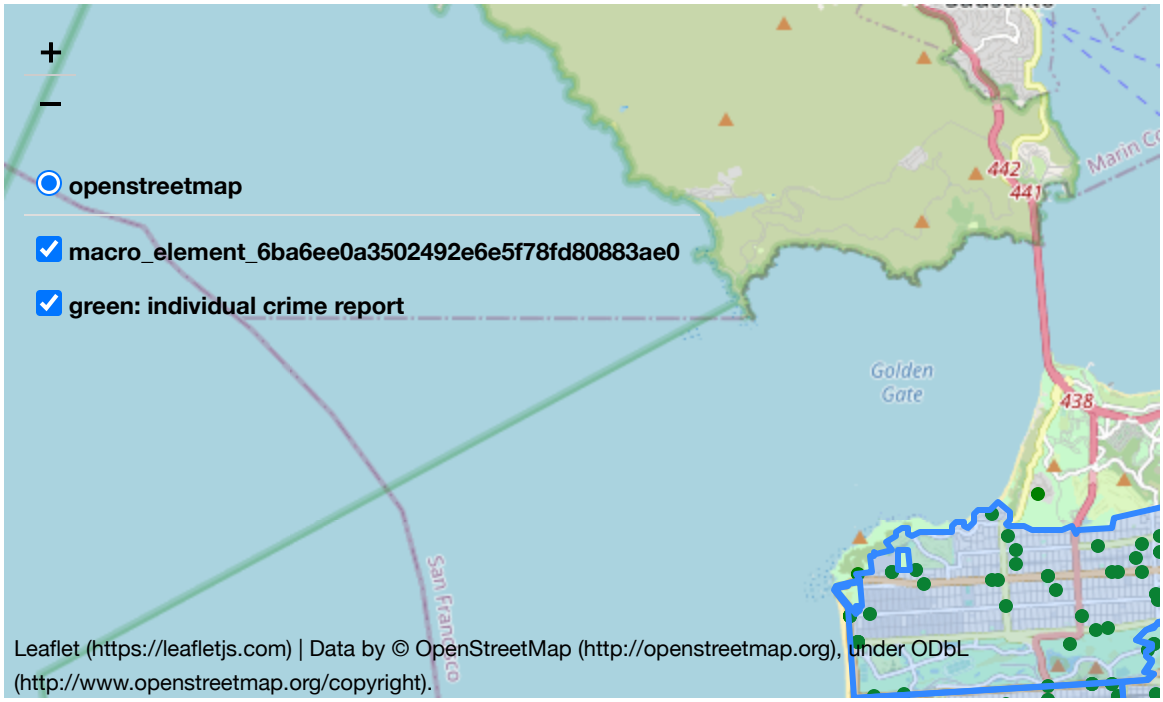
Out[38]:

	Incident Datetime	Incident Date	Incident Time	Incident Year	Incident Day of Week	Report Datetime	Incident Number	Inc Cat
374545	2019/03/18 06:30:00 PM	2019/03/18	18:30	2019	Monday	2019/03/19 10:42:00 AM	190196090	V
501159	2019/12/21 12:00:00 PM	2019/12/21	12:00	2019	Saturday	2019/12/24 03:55:00 PM	190965726	V
477349	2019/11/11 03:00:00 PM	2019/11/11	15:00	2019	Monday	2019/11/11 04:42:00 PM	190854464	V
424972	2019/10/09 06:55:00 PM	2019/10/09	18:55	2019	Wednesday	2019/10/09 11:04:00 PM	190762168	V
419792	2019/03/30 03:00:00 PM	2019/03/30	15:00	2019	Saturday	2019/04/01 02:13:00 PM	190229942	V

In [39]:
```
sfmap_crime = folium.Map(sf_coords, zoom_start=12)
sfmap_crime;

mapping_crimes_df.apply(lambda row:folium.CircleMarker(location=[row["Latitu
                                                    row["Longitude"]], radius=
         axis=1)
folium.GeoJson(police_districts).add_to(sfmap_crime);

# source: https://stackoverflow.com/questions/37466683/create-a-legend-on-a-
lgd_txt = '<span style="color: {col};">{txt}</span>'
```

```
for idx, color in enumerate( ['green']):
    fg = folium.FeatureGroup(name= lgd_txt.format( txt= color + ": individua
    sfmap_crime.add_child(fg)

folium.map.LayerControl('topleft', collapsed= False).add_to(sfmap_crime)

sfmap_crime
```

Out[39]:



Similar to where most of the incident reports are geographically located, most of the crime reports are also clustered around and in the Tenderloin district.

4.1 B: Shade the points by type of crime and analyze whether certain areas experience certain types of crime more often.

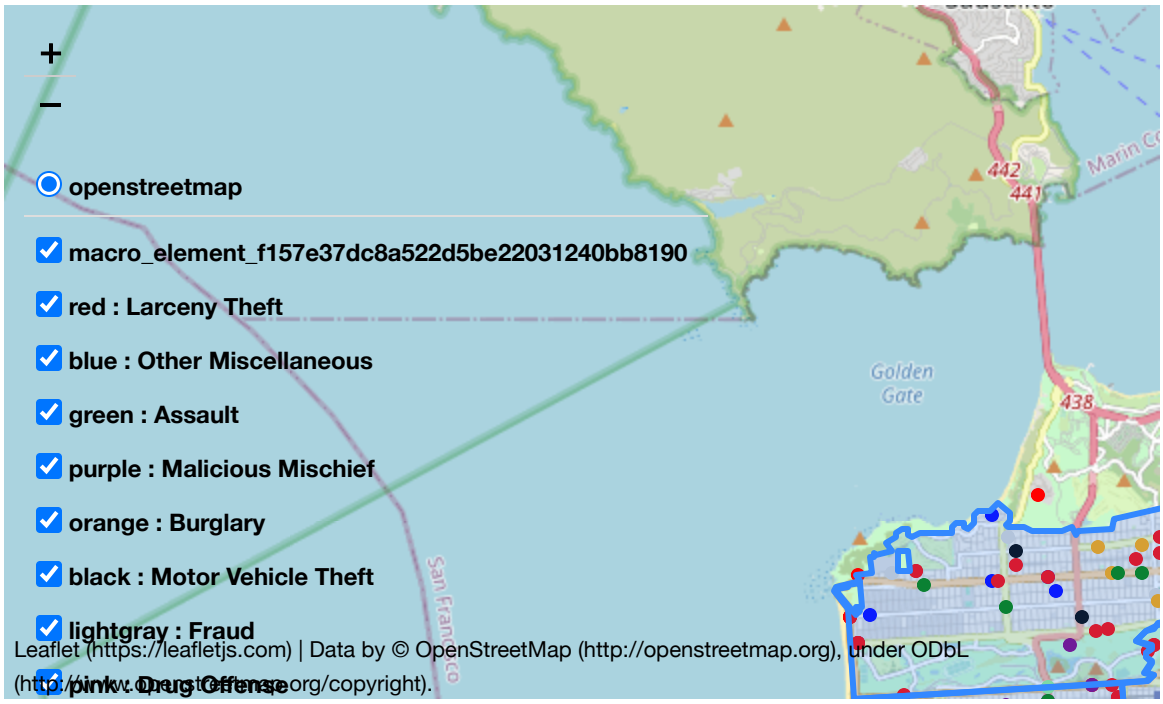
In [40]:
```
top_10_crimes = crimes_2019["Incident Category"].value_counts().head(10).inc
top_10_crimes
```

Out[40]:
```
Index(['Larceny Theft', 'Other Miscellaneous', 'Assault', 'Malicious Mischi
ef',
       'Burglary', 'Motor Vehicle Theft', 'Fraud', 'Drug Offense', 'Robber
y',
       'Suspicious Occ'],
      dtype='object')
```

In [41]:
```
# map for crimes categorized by colors
sfmap_crime_split = folium.Map(sf_coords, zoom_start=12)
sfmap_crime_split;

def assign_cluster(category):
    counter = 0
    for i in top_10_crimes:
        if (category == top_10_crimes[counter]):
            value = counter
```

```
        counter += 1
    return value

# assign each category a number
mapping_crimes_df_copy = mapping_crimes_df.copy()
mapping_crimes_df_copy = mapping_crimes_df_copy[mapping_crimes_df_copy["Inci
mapping_crimes_df_copy["cluster"] = mapping_crimes_df_copy["Incident Categor
# mapping_crimes_df = mapping_crimes_df[mapping_crimes_df["cluster"] < 19]
# colors = np.array(['red', 'blue', 'green', 'purple', 'orange', 'darkred',
#  'darkblue', 'darkgreen', 'cadetblue', 'darkpurple', 'white', 'pink',
#  'lightblue', 'lightgreen', 'gray', 'black', 'lightgray'])

colors = np.array(['red', 'blue', 'green','purple', 'orange','black', 'light
'lightblue', 'lightgreen'])
# folium.GeoJson(police_districts).add_to(sfmap_crime_split);
mapping_crimes_df_copy.apply(lambda row:folium.CircleMarker(location=[row["L
                                                    row["Longitude"]], radius=
         axis=1)

folium.GeoJson(police_districts).add_to(sfmap_crime_split);

lgd_txt = '<span style="color: {col};">{txt}</span>'
c = 0
for idx, color in enumerate( ['red', 'blue', 'green','purple', 'orange','bla
'lightblue', 'lightgreen']):
    fg = folium.FeatureGroup(name= lgd_txt.format(txt= color + " : " + top_1
    c += 1
    sfmap_crime_split.add_child(fg)

folium.map.LayerControl('topleft', collapsed= False).add_to(sfmap_crime_spli

sfmap_crime_split
```

Out[41]:



Utilizing the map above that distinctively classifies each of the top 10 crimes to a specific color, it is clear that the color red is the dominating color. Red represents larceny theft and it is widely spread out within every police district. It does seem to crowd around the tip of Central district, but Central district is big and it is not the only type of crime taken place there since we see multiple colors in that area.

However, the pink dots are mostly just centered around one area, with that being Tenderloin. Drug Offense is denoted by the color pink and there is a cluster of them in and surrounding the Tenderloin district. This observation supports the claim that certain crimes do occur more at specific districts.

Another prominent color in this map is blue that represents "Other Miscellaneous", but because "Other Miscellaneous" is broad and can contain many a wide range of crimes, it is difficult to conclude that it occurs in a specific district. It is also pretty widely spread throughout SF, but also more clustered around Central, Tenderloin, Mission, Northern, and Southern.

4.1 C: Propose social scientific explanations for the patterns that you find

The map above and the clustering of points in certain areas drives me to analyze why there are more crimes in certain districts.

The first one that stands out to me is Central district. Central district houses tourist attractions such as Union Square, Chinatown, Fisherman's Wharf, and the Embarcadero Center, in addition to the Financial District. With this being known, it can be assumed that Central district is the busiest district in SF during all times of the day/year/week. With this being known, Central district becomes the location where crimes, such as larceny theft, are most likely to occur because there are more people walking and this contributes to being easier targets.

As I mentioned previously, the pink dots are clustered in the Tenderloin district. Possible factors of this are the socioeconomic conditions such as the high povery rates at Tenderloin compared to other districts. Due to this, there is also a higher population of homelesses, so the likelihood of drug offenses to occur there is more than the other districts. Knowing this fact about Tenderloin and its history of illegal drug possession/sales, the amount of surveillance police emphasize in the district must be more than others as well. Hence, more police can easily lead to more incidents.

Problem 4.2

Merge the incidents data with the GeoJSON file which contains the information on the boundaries of “analysis neighborhoods” in San Francisco.

In [42]:
```
ir_gdf = gpd.GeoDataFrame(
    incident_reports, geometry=gpd.points_from_xy(incident_reports.Longitude
```

```
ir_nhood_join = gpd.sjoin(ir_gdf, analysis_neighborhoods, how="left")
ir_nhood_join.head(5)
```

Out[42]:

	Incident Datetime	Incident Date	Incident Time	Incident Year	Incident Day of Week	Report Datetime	Incident Number	Inciden Categor
184717	2018/01/01 02:28:00 AM	2018/01/01	02:28	2018	Monday	2018/01/01 02:31:00 AM	180000348	Assau
272495	2018/01/01 01:10:00 AM	2018/01/01	01:10	2018	Monday	2018/01/01 10:35:00 PM	186000390	Larcen Thef
231542	2018/01/01 05:30:00 PM	2018/01/01	17:30	2018	Monday	2018/01/01 07:42:00 PM	180002247	Larcen Thef
263479	2018/01/01 04:00:00 PM	2018/01/01	16:00	2018	Monday	2018/01/16 10:46:00 AM	180040902	Los Propert
211811	2018/01/01 04:00:00 PM	2018/01/01	16:00	2018	Monday	2018/01/01 09:05:00 PM	186002681	Larcen Thef

5 Mapping Incidents

Problem 5.1

Construct a choropleth map, coloring in each neighborhood by how many incidents it had in 2019. Then, construct several additional maps that explore differences by day of week, time of year, time of day, etc.

In [43]:
```
ir_nhood_join.head()
```

Out[43]:

	Incident Datetime	Incident Date	Incident Time	Incident Year	Incident Day of Week	Report Datetime	Incident Number	Incidem Categor
184717	2018/01/01 02:28:00 AM	2018/01/01	02:28	2018	Monday	2018/01/01 02:31:00 AM	180000348	Assau
272495	2018/01/01 01:10:00 AM	2018/01/01	01:10	2018	Monday	2018/01/01 10:35:00 PM	186000390	Larcen Thef
231542	2018/01/01 05:30:00 PM	2018/01/01	17:30	2018	Monday	2018/01/01 07:42:00 PM	180002247	Larcen Thef
263479	2018/01/01 04:00:00 PM	2018/01/01	16:00	2018	Monday	2018/01/16 10:46:00 AM	180040902	Los Propert
211811	2018/01/01 04:00:00 PM	2018/01/01	16:00	2018	Monday	2018/01/01 09:05:00 PM	186002681	Larcen Thef

In [44]:
```
ir_nhood_join = ir_nhood_join.dropna(subset=["nhood"])
ir_nhood_join_2019 = ir_nhood_join[ir_nhood_join["Incident Year"] == 2019]
nhood_counts = ir_nhood_join_2019.groupby("nhood")[["nhood"]].count().rename
nhood_dict = nhood_counts.set_index("nhood")["count"]
ir_nhood_join_2019.head()
```

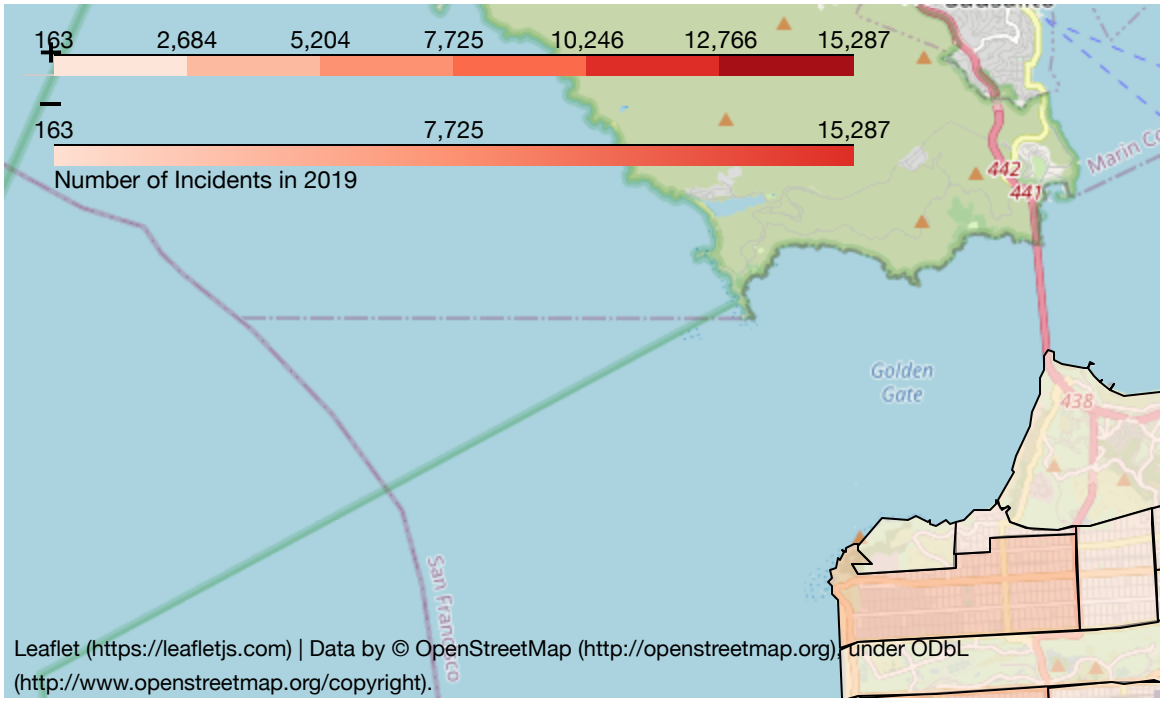
Out[44]:

	Incident Datetime	Incident Date	Incident Time	Incident Year	Incident Day of Week	Report Datetime	Incident Number	Inc Cate
385348	2019/01/01 12:00:00 PM	2019/01/01	12:00	2019	Tuesday	2019/01/01 02:00:00 PM	190001245	Motor Ve
399141	2019/01/01 12:11:00 PM	2019/01/01	12:11	2019	Tuesday	2019/01/01 12:13:00 PM	190001007	Wa
385611	2019/01/01 12:20:00 PM	2019/01/01	12:20	2019	Tuesday	2019/01/01 07:36:00 PM	196000261	Lost Pro
405447	2019/01/01 09:24:00 PM	2019/01/01	21:24	2019	Tuesday	2019/01/01 09:26:00 PM	190002265	T Vio A
340107	2019/01/01 02:39:00 AM	2019/01/01	02:39	2019	Tuesday	2019/01/01 02:39:00 AM	190000407	Miscellar

In [45]:
```
colormap1 = linear.Reds_03.scale(
    min(nhood_counts["count"]),
    max(nhood_counts["count"]))
```

In [46]:
```
sfmap_nhood = folium.Map((37.76, -122.45), zoom_start=12)

folium.Choropleth(
    geo_data=analysis_neighborhoods,
    data=nhood_counts,
    columns=['nhood', 'count'],
    key_on='properties.nhood',
    fill_color='Reds'
).add_to(sfmap_nhood)

colormap1.caption = 'Number of Incidents in 2019'
colormap1.add_to(sfmap_nhood)

sfmap_nhood
```

Out[46]:



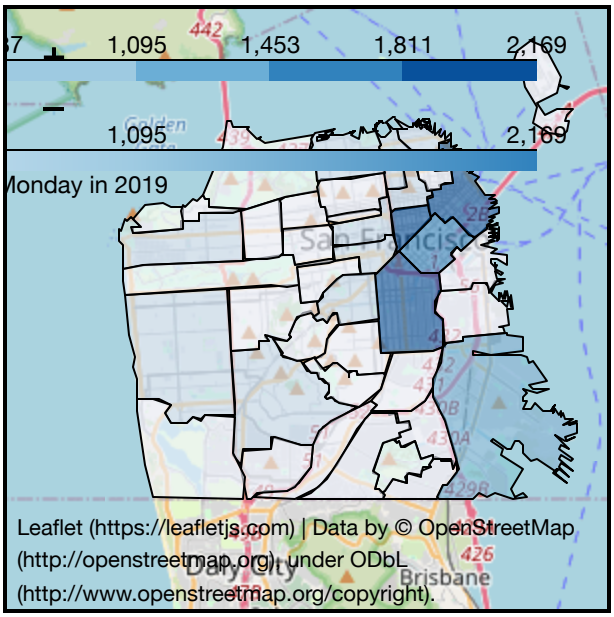
In [47]:
```
ir_nhood_join_2019_copy = ir_nhood_join_2019.copy()
def week_choro_maps(day_of_week, color):
    day = ir_nhood_join_2019_copy[ir_nhood_join_2019_copy["Incident Day of W
    nhood_counts = day.groupby("nhood")[["nhood"]].count().rename(columns =

    day_nhood = folium.Map((37.76, -122.45), zoom_start=11);

    folium.Choropleth(
        geo_data=analysis_neighborhoods,
        data=nhood_counts,
        columns=['nhood', 'count'],
        key_on='properties.nhood',
        fill_color=color
    ).add_to(day_nhood)
    return day_nhood, nhood_counts

def add_scale(colormap, day, daymap):
    colormap.caption = "Number of Incidents on " + day + " in 2019"
    colormap.add_to(daymap)
```

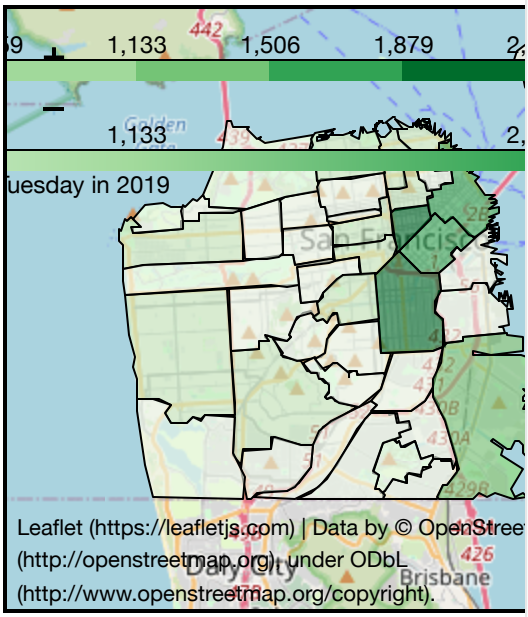
```
mon_nhood = week_choro_maps("Monday", "Blues")[0]
mon_cm = linear.Blues_03.scale(
    min(week_choro_maps("Monday", "Blues")[1]["count"]),
    max(week_choro_maps("Monday", "Blues")[1]["count"]))
add_scale(mon_cm, "Monday", mon_nhood)

tues_nhood = week_choro_maps("Tuesday", "Greens")[0]
tues_cm = linear.Greens_03.scale(
    min(week_choro_maps("Tuesday", "Greens")[1]["count"]),
    max(week_choro_maps("Tuesday", "Greens")[1]["count"]))
add_scale(tues_cm, "Tuesday", tues_nhood)


wed_nhood = week_choro_maps("Wednesday", "Purples")[0]
wed_cm = linear.Purples_03.scale(
    min(week_choro_maps("Wednesday", "Purples")[1]["count"]),
    max(week_choro_maps("Wednesday", "Purples")[1]["count"]))
add_scale(wed_cm, "Wednesday", wed_nhood)


thurs_nhood = week_choro_maps("Thursday", "Greys")[0]
thurs_cm = linear.Greys_03.scale(
    min(week_choro_maps("Thursday", "Greys")[1]["count"]),
    max(week_choro_maps("Thursday", "Greys")[1]["count"]))
add_scale(thurs_cm, "Thursday", thurs_nhood)

fri_nhood = week_choro_maps("Friday", "YlOrRd")[0]
fri_cm = linear.YlOrRd_03.scale(
    min(week_choro_maps("Friday", "YlOrRd")[1]["count"]),
    max(week_choro_maps("Friday", "YlOrRd")[1]["count"]))
add_scale(fri_cm, "Friday", fri_nhood)
fri_nhood

sat_nhood = week_choro_maps("Saturday", "Oranges")[0]
sat_cm = linear.Oranges_03.scale(
    min(week_choro_maps("Saturday", "Oranges")[1]["count"]),
    max(week_choro_maps("Saturday", "Oranges")[1]["count"]))
add_scale(sat_cm, "Saturday", sat_nhood)


sun_nhood = week_choro_maps("Sunday", "PuBuGn")[0]
sun_cm = linear.PuBuGn_03.scale(
    min(week_choro_maps("Sunday", "PuBuGn")[1]["count"]),
    max(week_choro_maps("Sunday", "PuBuGn")[1]["count"]))
add_scale(sun_cm, "Sunday", sun_nhood)
```

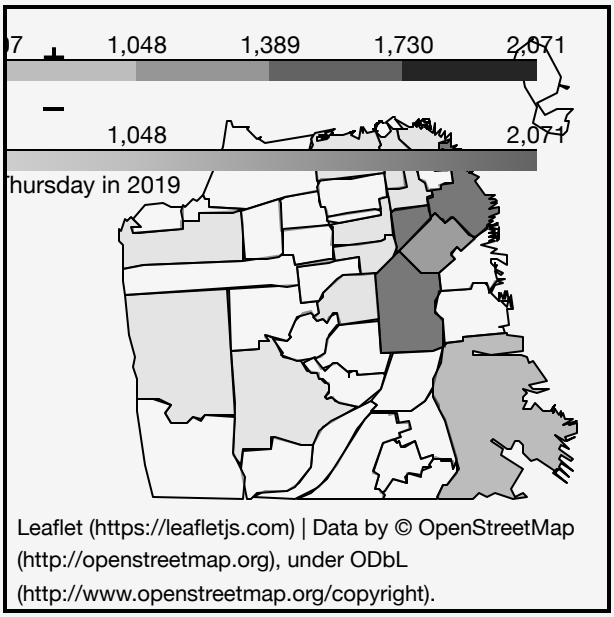
In [48]:

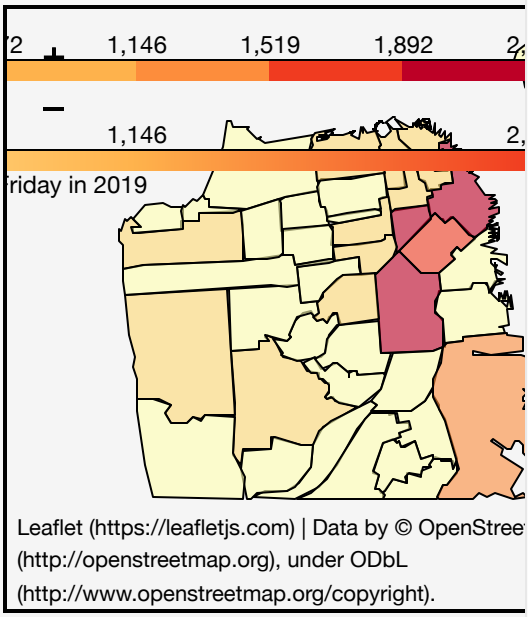
```
# source: https://stackoverflow.com/questions/57943687/showing-two-folium-ma
# inspired from the source above, modified to fit my own needs

htmlmap = HTML('<table><tr>'
               '<td><iframe srcdoc="{}" style="width: {}px; height: {}px; bo
               '<td><iframe srcdoc="{}" style="width: {}px; height: {}px; bo
               '<td><iframe srcdoc="{}" style="width: {}px; height: {}px; bo
               '</tr><tr>'
               '<td><iframe srcdoc="{}" style="width: {}px; height: {}px; bo
```

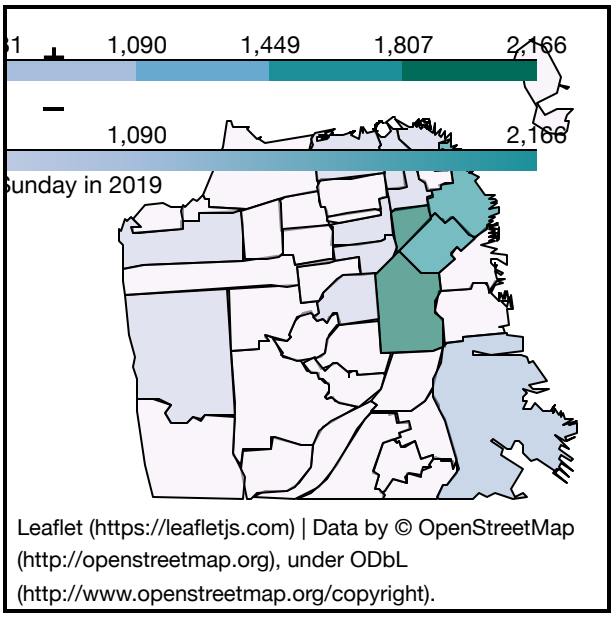
```
                '<td><iframe srcdoc="{}" style="width: {}px; height: {}px; bo
                '<td><iframe srcdoc="{}" style="width: {}px; height: {}px; bo
                '</tr><tr>'
                '<td><iframe srcdoc="{}" style="width: {}px; height: {}px; bo
                '</tr></table>'
            .format(mon_nhood.get_root().render().replace('"', '&quot;'), 300
                    tues_nhood.get_root().render().replace('"', '&quot;'), 30
                    wed_nhood.get_root().render().replace('"', '&quot;'), 300
                    thurs_nhood.get_root().render().replace('"', '&quot;'), 3
                    fri_nhood.get_root().render().replace('"', '&quot;'), 300
                    sat_nhood.get_root().render().replace('"', '&quot;'), 300
                    sun_nhood.get_root().render().replace('"', '&quot;'), 300
display(htmlmap)
```


1,095
1,453
1,811
2,169
1,095
2,169
Monday in 2019
Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).


1,133
1,506
1,879
1,133
uesday in 2019
Leaflet (https://leafletjs.com) | Data by © OpenStree (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).


1,048
1,389
1,730
2,071
1,048
2,071
hursday in 2019
Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).


1,146
1,519
1,892
1,146
riday in 2019
Leaflet (https://leafletjs.com) | Data by © OpenStree (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

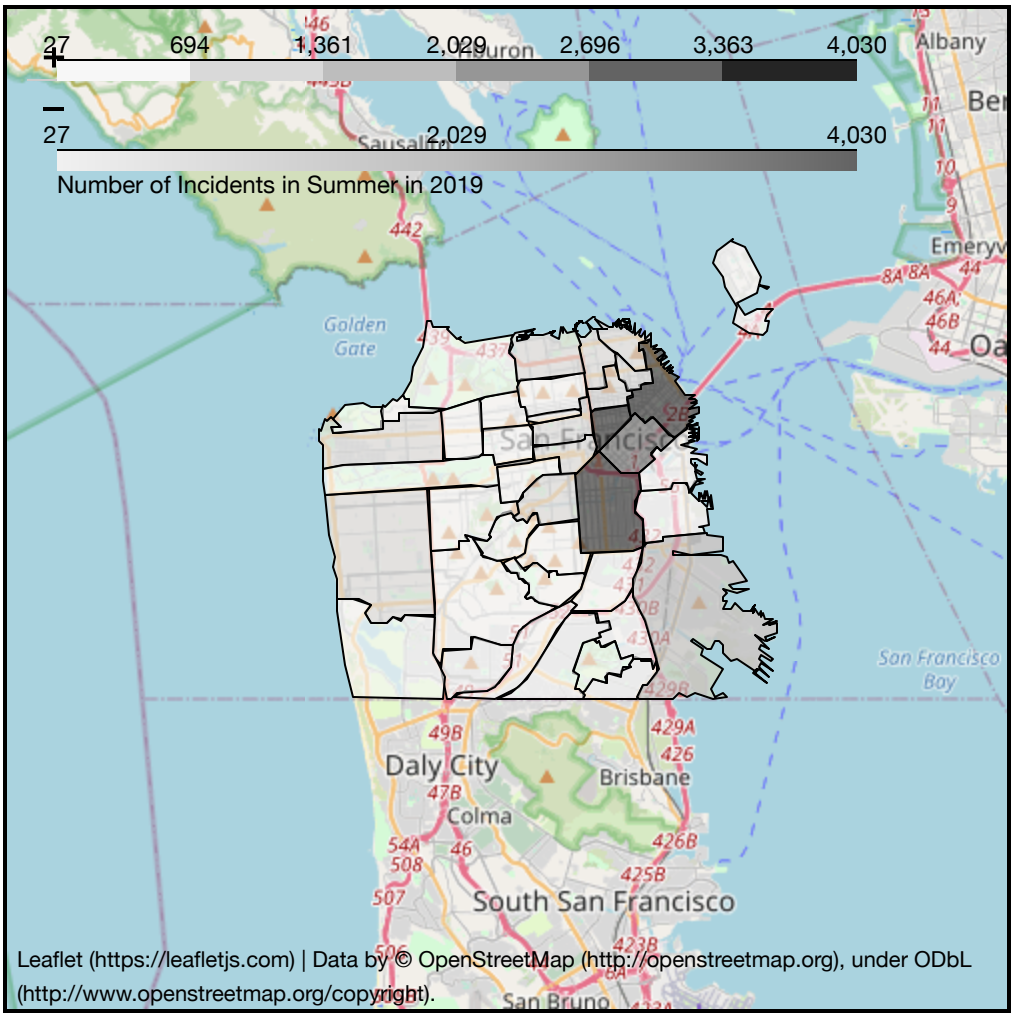

1,090
1,449
1,807
2,166
1,090
2,166
unday in 2019
Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

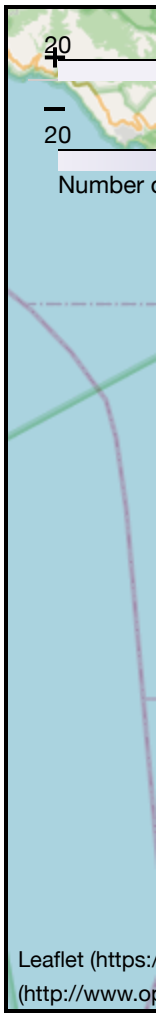
The 7 maps above show the choropleth maps for every single day of the week. The intent was for me to analyze whether the day of the week had an impact on the number of incident reports in each police district. There is a slight difference that we can see, but only with close analyzation. For example on Monday, Mission and Financial District neighborhoods have a similar of incident reports, but on Tuesday, the Financial District neighborhood does not have as many reports as Mission. This shows to me that the day of the week can have an effect of the amount of incident reports, but more research and information would be needed to determine that.

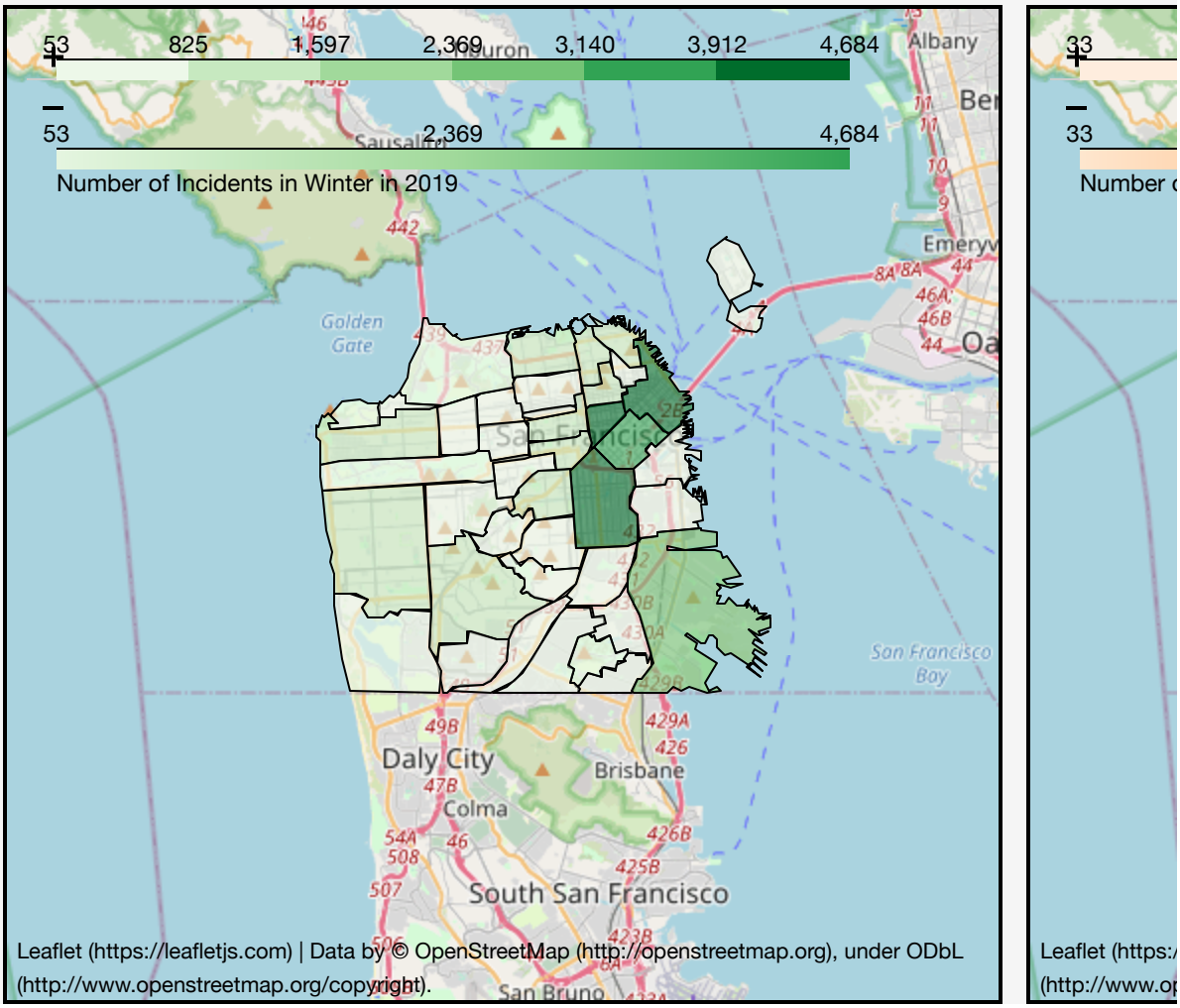
In [49]:
```
# future thought: use probability (how likely does this crime happen at this
```

In [50]:
```
def time_choro_maps(column, variable, color):
    nhood_filtered = ir_nhood_join_2019_copy[ir_nhood_join_2019_copy[column]
    filtered_counts = nhood_filtered.groupby("nhood")[["nhood"]].count().rer

    filteredmap = folium.Map((37.76, -122.45), zoom_start=11);

    folium.Choropleth(
        geo_data=analysis_neighborhoods,
        data=filtered_counts,
        columns=['nhood', 'count'],
        key_on='properties.nhood',
        fill_color=color
    ).add_to(filteredmap)
    return filteredmap, filtered_counts

def add_scale(colormap, var, maps):
    colormap.caption = "Number of Incidents in " + var + " in 2019"
    colormap.add_to(maps)

sum_nhood = time_choro_maps("Season", "Summer", "Greys")[0]
sum_cm = linear.Greys_03.scale(
    min(time_choro_maps("Season","Summer", "Greys")[1]["count"]),
    max(time_choro_maps("Season", "Summer", "Greys")[1]["count"]))
add_scale(sum_cm, "Summer", sum_nhood)

spr_nhood = time_choro_maps("Season", "Spring", "Purples")[0]
spr_cm = linear.Purples_03.scale(
    min(time_choro_maps("Season", "Spring", "Purples")[1]["count"]),
    max(time_choro_maps("Season", "Spring", "Purples")[1]["count"]))
add_scale(spr_cm, "Spring", spr_nhood)

win_nhood = time_choro_maps("Season", "Winter", "Greens")[0]
win_cm = linear.Greens_03.scale(
    min(time_choro_maps("Season", "Winter", "Greens")[1]["count"]),
    max(time_choro_maps("Season", "Winter", "Greens")[1]["count"]))
add_scale(win_cm, "Winter", win_nhood)

fall_nhood = time_choro_maps("Season", "Fall", "Oranges")[0]
fall_cm = linear.Oranges_03.scale(
    min(time_choro_maps("Season", "Fall", "Oranges")[1]["count"]),
    max(time_choro_maps("Season", "Fall", "Oranges")[1]["count"]))
```

```
add_scale(fall_cm, "Fall", fall_nhood)

# source: https://stackoverflow.com/questions/57943687/showing-two-folium-ma
# inspired from the source above, modified to fit my own needs

htmlmapszn = HTML('<table><tr>'
                  '<td><iframe srcdoc="{}" style="width: {}px; height: {}px; bo
                  '<td><iframe srcdoc="{}" style="width: {}px; height: {}px; bo
                  '</tr><tr>'
                  '<td><iframe srcdoc="{}" style="width: {}px; height: {}px; bo
                  '<td><iframe srcdoc="{}" style="width: {}px; height: {}px; bo
                  '</tr><tr>'
                  '</tr></table>'
              .format(sum_nhood.get_root().render().replace('"', '&quot;'), 500
                      spr_nhood.get_root().render().replace('"', '&quot;'), 500
                      win_nhood.get_root().render().replace('"', '&quot;'), 500
                      fall_nhood.get_root().render().replace('"', '&quot;'), 50
display(htmlmapszn)
```


27
694
1,361
2,029
2,696
3,363
4,030
Number of Incidents in Summer in 2019
Albany
Sausalito
Golden Gate
San Francisco
San Francisco Bay
Daly City
Brisbane
Colma
South San Francisco
San Bruno
Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).


20



In the four maps I created above, I wanted to examine if the seasons affected where incident reports were concentrated. There is no change from the first choropleth map I created. The incident reports are still clustered in the same neighborhoods as the first choropleth map I created and no matter what season it is, there seems to be no change. However, similar to the maps created for the day of the week, there are small changes based on the season. In the summer, Mission and Financial District have roughly the same amount of incident reports, but looking at Spring, Financial District experiences less reports. Interestingly, the only time there is a difference is in Spring.

Problem 5.2

Do you notice any patterns? Are there particular neighborhoods where crime concentrates more heavily? Propose a short explanation for what you find.

The areas shaded with the darkest red are the areas with the highest count of crimes. The two leading neighbhoods are Tenderloin and Mission district. Following them are Financial District/South Beach, South of Market, and Bayview Hunters Point neighborhoods. Crimes may concentrate in these areas for multiple reasons including their population and their location. These neighborhoods are highly populated areas in

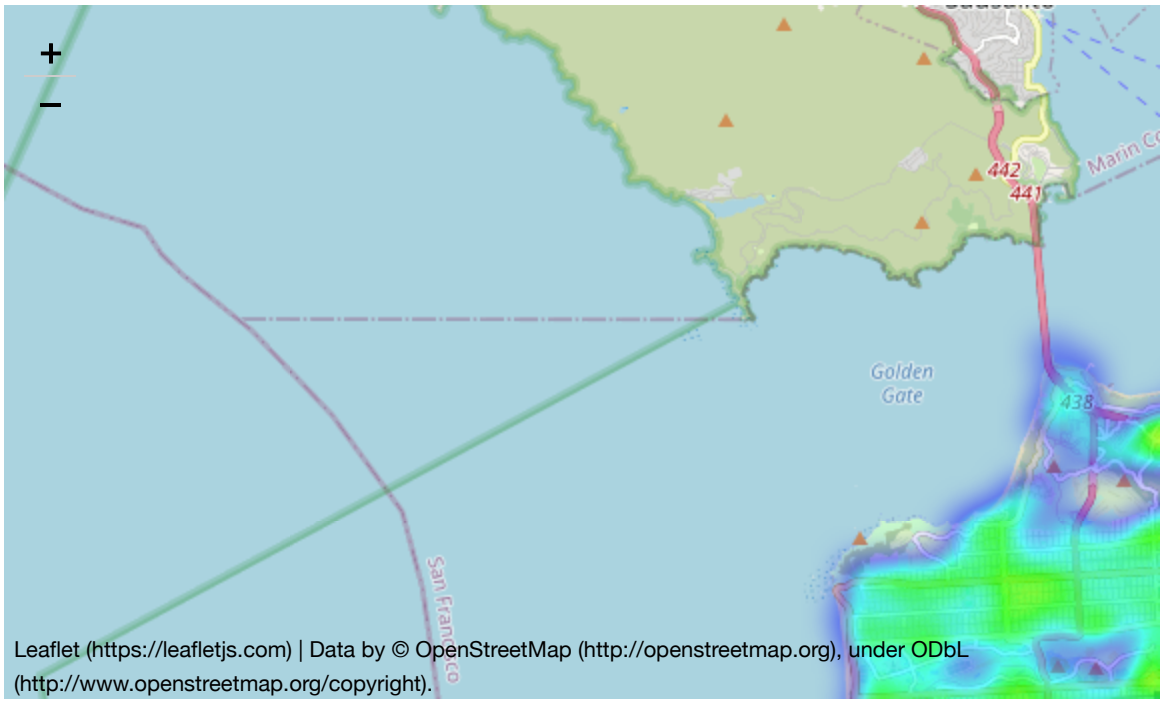
San Francisco, so the higher the population, the higher the chance of incidents occuring. Additionally, a common theme between these neighborhoods are their locations. They are all very close to each other and all geographically located on the right side of the map. Crime can spread easily because criminals travel from one place to another to commit their offenses. They are also near the bay where there are bridges and the location may make escaping easier.

Problem 5.3

Construct a heat map of crime. How does the heat map compare to the choropleth map? Are neighborhoods a reasonably good proxy for the actual concentration of crime?

In [51]:
```
sf_heatmap = folium.Map((37.76, -122.45), zoom_start=12)
ir_nhood_join = ir_nhood_join.dropna(subset=["Latitude", "Longitude"])
lat = ir_nhood_join["Latitude"].values
lon = ir_nhood_join["Longitude"].values
call_locs = np.vstack((lat, lon)).transpose().tolist()
heatmap = HeatMap(call_locs, radius = 10)
sf_heatmap.add_child(heatmap)
sf_heatmap
```

Out[51]:



Unlike the choropleth map, the heat map does not how the number of incident reports per neighborhood. The heat map focuses on where the incidents are mostly clustered around and examining the map above, we can see that there is a yellow spot located where the Tenderloin district is while the rest of the map is green. This indicates where the high density of crimes is located which was not as clear in the choropleth map. In the choropleth map, there were 3 districts that had the same color, but in the heat map there is a clear distinction of where the crimes are highly concentrated.

Neighborhoods are a good proxy for the actual concentration of crime as proven by the choropleth and heat maps. We are able to visualize which areas of San Francisco experience the most crimes and this could be helpful in making decisions such as distribution of police officers. Moreover, analyzing where the concentration of incident reports are can help provide insights to policy makers of San Francisco about the population that live in that area and the needs of the neighborhood to improve their safety.

6 Discussion Questions

Problem 6.1

Based on the evidence you just developed, why do you think "hotspots" policing became more popular in the last few decades? What are the pros and cons to this kind of approach? Remember to refer to course readings in your response.

Based on all the maps I created in this problem set, I can conclude that crimes are concentrated around the same areas. Given this information, within the last few decades, hotspot policing became more popular as a practice of preventive policing because with data and analysis of incident reports, the idea of deploying more police officers to these areas would prevent crimes from occurring. By practicing hotspot policing, crime would be reduced effectively since police are more prevalent in areas with higher crime rates and resources are used efficiently instead of deploying a bunch of police to areas where crime rarely occurs.

Some pros of hotspot policing is its idea of deploying police where it is most needed. By ensuring that the areas with more crimes occurring have more police, the community would feel safer. With this method, resources are also distributed efficiently since areas without high concentration of crimes would not need as many police officers and this could possibly bring down the costs for the city as well. Specifically, to SF and the evidence gathered, some areas with high rates of crime include tourist attractions and having more police deployed to those areas brings more awareness to public safety. This brings more tourists to SF, which is great for the economy.

However, the problem with hotspot policing is this strategy is a short-term solution to crime, it creates the ratchet effect, and causes racial profiling. With the knowledge of an area being more susceptible to crime, police officers, who are obviously humans as well, are more likely to be more vigilant, increasing the chances of them falsely or mistakenly scruitize people. Overpolicing these neighborhoods results in an increase of surveillance; hence, more incidents to be reported from those areas. This ties to the idea of the ratchet effect from Harcourt from one of our readings. Because more police are deployed to the ares with high concentration of crime, they are more likely to find more

crimes and this creates the misconception that some of these places are less safe than they actually are, causing a stigma to be created about the area. The problem with this expands to areas without as many police officers because in those areas, people will start taking advantage of lower surveillance and it creates another misconception that the area is safer than it really is. Additionally, the areas with higher rates of crime are often areas with high poverty rates/a larger population of minorities, so hotspot policing can result in minorities under stricter surveillance and more arrests of them. With people from the minority groups having higher incarceration rates due to overpolicing, police are more likely to target them, creating a non-stop cycle of racial bias.

Problem 6.2

Comment on what sorts of incidents get reported in this database. For instance, do you see a lot of reports about things like white collar crime? How do you think incident categories are selected? As data scientists, what kinds of ethical and legal concerns should we be aware of when we construct these sorts of datasets?

The incidents that get reported to this database include various types of crimes, but also incidents that are not crimes, but involve police presence and their expectation to report them. In this dataset, the most of the reported incidents are incidents that are considered street crime where when it occurs, police can physically go to the location and take note of it like theft or assault. Another way I thought about it is they are crimes that can be recorded and witnessed by bystanders. The vast majority of incidents come from that category while very little (only 22006 reports out of a ~700,000 row dataset) are considered white collar crimes. Most of the time, acts of fraud, embezzlement, and more are dealt with at higher levels of law enforcements such as the FTC. When someone is in suspicion of a white collar crime, the police are probably not the first people to be called for.

I think these incident categories are selected based on their definitions, their severity, and the harm the offenses causes for others. Incidents like robbery, larceny theft, and assault cause physical harm to people, while incidents like fraud and embezzlement are financial crimes that can be done over online platforms and less of a chance of physically harming someone.

When we construct these kinds of datasets, we have to be sure we are taking into account of privacy. Information about the criminals, victims, or anyone involved that is sensitive such as specific addresses should be kept confidential and not leaked to the public. The information, we as data scientists are obtaining, should be known by the subjects and the purpose should also be revealed. We also have to be sure these reports are accurately reported and representative of the area of study because important decisions are made based on the information the data provides.

Problem 6.3

What other sorts of data would help improve your analysis?

To improve my analysis, a dataset containing the population of each neighborhood and a dataset with the number of police officers deployed to each of the districts would be helpful. With just the incident reports dataset, we are only working with the number of incidents, but not with respect to the number of people in those areas. Therefore, it is easy to misinterpret the data and assume certain some areas experience lots of crimes, but in reality since their population is larger, they actually have the relatively the same crime rate as a smaller neighborhood. Being able to compare the proportions would help the analysis of the probability of crime to occur in an area. Furthermore, knowing the number of police officers on duty during certain days/times can show the effects of hotspot policing, police behavior, and whether whether having more police officers is actually more beneficial to the communities.

Diverging away from population, data that contains socioeconomic factors in each neighborhood would help improve my analysis. As evidence shows, Tenderloin has a high crime rate and there are many reasons that result in this happening. This ties into the idea of solving the root cause of this problem and figuring out what some of these areas are lacking such as a lack of schools, mental health hospitals, or homeless shelters.

Bonus: Policy Questions

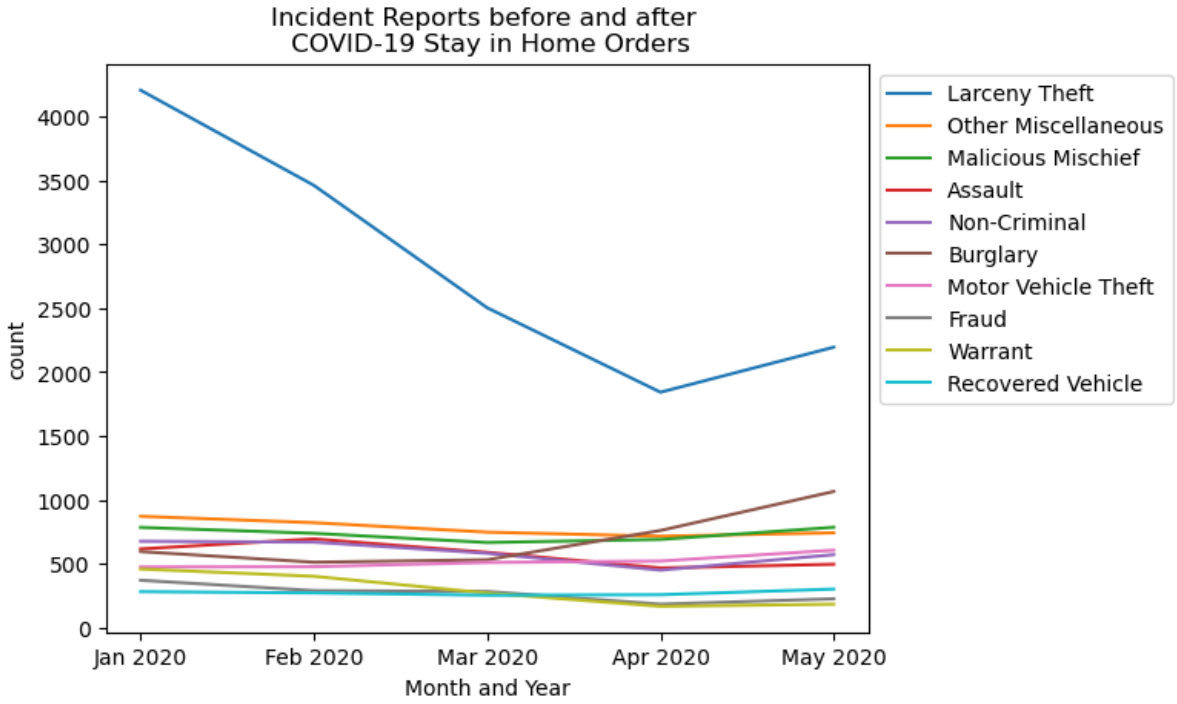
Problem B.1

On March 16, 2020 the first shelter-at-home order started in San Francisco. Residents and most visitors to the City stayed at home. What effect would you expect this policy to have on crime rates? Compare crime rates before and after the first week of the pandemics restriction and comment on the differences you find.

In my analysis above, I mentioned a dip in incident reports when the stay at home orders were enforced in March 2020. I expect the effect of this policy to be a decrease on crime rates because there are less people on the streets, so there are less targets for theft, robbery, and burglary.

In [52]:
```
covid = monthly_counts.reset_index()
months = ["Jan 2020", "Feb 2020", "Mar 2020", "Apr 2020", "May 2020"]
covid = covid.loc[covid["Month and Year"].isin(months)]
```

In [53]:
```
for cat in top_ten_categories:
    sns.lineplot(data = covid, x="Month and Year", y= covid[cat].values, lab

plt.ylabel("count")
plt.title("Incident Reports before and after \n COVID-19 Stay in Home Orders
plt.legend(bbox_to_anchor=(1.0, 1.0));
```



Although the incident reports did decrease in March 2020, it seems to spike back up shortly after. The number of burglaryies, for example, increases in April and May of 2020. Reasonings for this can be because people were losing their jobs because of the pandemic which takes a toll in their mental health and can cause them to be in financial desperation. This causes people to perform actions they probably wouldn't have before and resort to burglary to acquire the things they need. Besides larceny theft and burglary, the other types of incidents did not experience a sharp change in the number of them being reported.

Problem B.2

A few months later, the Emergency Bail Schedule was adopted statewide which allowed pretrial release for most offenders, excluding serious felony and specific misdemeanor offenses. What effect would you expect this policy to have on crime rates and how would you explore the question?

This implementation of this policy can lead to both an increase and decrease in crime rates. With this policy, people are less afraid of the consequences; therefore, more likely to commit crimes they know they will be allowed pretrial release. While waiting for their court hearing, they may continue to commit crimes which is dangerous to the

community and increases crime overall. Additionally, the rates of people not showing up for their court hearing will also increase. On the other hand, it can also decrease crime rates because people can get more support when they are not in jail, lowering the risk of recidivism. This may also lessen the discrimination of lower and higher income people because often times, the people detained pre-trial are people who cannot afford to pay their bails.

To explore this question, I would conduct a pre-post analysis of the policy implementation. The specific time periods I would want to look at is a year before and a year after the policy gets adopted because I don't expect the crime rates to change overnight; I expect a gradual change. I would use multi-linear and discontinuity regression to analyze the impact the policy has and to further my analysis, I would group the crimes into categories such as: street, property, and petty crimes. This is because I anticipate some crimes to increase, while some may decrease.