Exploratory Data Analysis- Crimes In Boston

Junior Data Scientist: Ohood Alsohaime

Introductin

The aim of this study is to examine how crimes have changed over the years, whether it is possible to predict where and when a crime will be committed, and the distribution of crimes across the city. Crimes in Boston data were used in the study. Data contains information about the crime such as date, location, crime group, crime code.

Features:

INCIDENT_NUMBER: The id of the crime committed. It is unique value for each crime.

OFFENSE_CODE: It shows code of crime types.

OFFENSE_CODE_GROUP: General crime types.

OFFENSE_DESCRIPTION: Detailed explanation of the crime.

DISTRICT: District name where the crime occurred.

REPORTING_AREA: Area number that crime reported.

SHOOTING: It shows with 'Y', if the crime included shooting.

OCCURRED_ON_DATE: the date& time that crime occured.

YEAR: the year that crime occured. (2015,2016,2017,2018)

MONTH: the month that crime occured.

DAY OF WEEK: the week that crime occured.

HOUR: the hour that crime occured.

UCR_PART: Uniform Crime Reporting Offence types. Part 1 contains the most dangerous and important crimes.

STREET: the street where crime occured.

LAT: the latitude where the crime occurred.

LONG: the longitude where the crime occurred.

LOCATION: the location where the crime occurred.(include latitude and longitude)

Import Packages and Libraries

In [1]:

```
import pandas as pd   # data processing, CSV file I/O (e.g. pd.read_csv)
import numpy as np    # linear algebra
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.pyplot import figure, show
from wordcloud import WordCloud
import plotly.graph_objects as go
# i installed (Folium+wordcloud) module from Anaconda prompt = pip install folium
import folium
from folium import Choropleth, Circle, Marker
from folium.plugins import HeatMap, MarkerCluster
from sqlalchemy import create_engine
import os  # accessing directory structur
import math
import time
from matplotlib import cm
import pylab as pl
import sqlite3
%matplotlib inline
```

Reading DataSet

In [2]:

```
#Read our Dataset that we get from Kaggle website
df = pd.read_csv('crime.csv')
```

In [3]:

```
#print the first rows of my DataFrame
df.head(3)
```

Out[3]:

	INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	DI
0	I182070945	619	Larceny	LARCENY ALL OTHERS	
1	I182070943	1402	Vandalism	VANDALISM	
2	I182070941	3410	Towed	TOWED MOTOR VEHICLE	

In [4]:

```
# we can see there are some null values
# columns with numerical values are type int64, while text/string values are object
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 319073 entries, 0 to 319072
Data columns (total 17 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   INCIDENT_NUMBER      319073 non-null  object 
 1   OFFENSE_CODE         319073 non-null  int64  
 2   OFFENSE_CODE_GROUP   319073 non-null  object 
 3   OFFENSE_DESCRIPTION  319073 non-null  object 
 4   DISTRICT             317308 non-null  object 
 5   REPORTING_AREA       319073 non-null  object 
 6   SHOOTING             1019 non-null    object 
 7   OCCURRED_ON_DATE     319073 non-null  object 
 8   YEAR                 319073 non-null  int64  
 9   MONTH                319073 non-null  int64  
 10  DAY_OF_WEEK          319073 non-null  object 
 11  HOUR                 319073 non-null  int64  
 12  UCR_PART             318983 non-null  object 
 13  STREET               308202 non-null  object 
 14  Lat                  299074 non-null  float64
 15  Long                 299074 non-null  float64
 16  Location             319073 non-null  object 
dtypes: float64(2), int64(4), object(11)
memory usage: 41.4+ MB
```

In [5]:

```
df.isna().sum() #count the missing values,we found column(SHOOTING)is null ,that means we s
```

Out[5]:

```
INCIDENT_NUMBER          0
OFFENSE_CODE             0
OFFENSE_CODE_GROUP       0
OFFENSE_DESCRIPTION      0
DISTRICT              1765
REPORTING_AREA           0
SHOOTING            318054
OCCURRED_ON_DATE         0
YEAR                     0
MONTH                    0
DAY_OF_WEEK              0
HOUR                     0
UCR_PART                90
STREET               10871
Lat                  19999
Long                 19999
Location                 0
dtype: int64
```

Handling missing values

In [6]:

```
# drop some columns beacuse its unnecessery or its values are missed
crime_df = df.drop(['SHOOTING','OFFENSE_CODE'], axis=1)
crime_df.head()
```

Out[6]:

	INCIDENT_NUMBER	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	DISTRICT	REPORTII
0	I182070945	Larceny	LARCENY ALL OTHERS	D14	
1	I182070943	Vandalism	VANDALISM	C11	
2	I182070941	Towed	TOWED MOTOR VEHICLE	D4	
3	I182070940	Investigate Property	INVESTIGATE PROPERTY	D4	
4	I182070938	Investigate Property	INVESTIGATE PROPERTY	B3	

In [7]:

```
#droping missing values in all Raw that contain Null value
crime_df.dropna(how='any',inplace=True)
#using subset to drop only a null valuue in each column
#Or crime_df.dropna(subset=['STREET'],inplace=True)
#Orcrime_df.dropna(subset=['Lat'],inplace=True)
#ORcrime_df.dropna(subset=['Long'],inplace=True)
#make sure to drop duplicated data
crime_df.drop_duplicates(subset="INCIDENT_NUMBER", inplace=True)
```

In [8]:

```
#detect missing values
crime_df.isna().sum()
```

Out[8]:

```
INCIDENT_NUMBER        0
OFFENSE_CODE_GROUP     0
OFFENSE_DESCRIPTION    0
DISTRICT               0
REPORTING_AREA         0
OCCURRED_ON_DATE       0
YEAR                   0
MONTH                  0
DAY_OF_WEEK            0
HOUR                   0
UCR_PART               0
STREET                 0
Lat                    0
Long                   0
Location               0
dtype: int64
```

In [9]:

```
#Number of unique entries were found for each columns.
crime_df.apply(pd.Series.nunique)
```

Out[9]:

```
INCIDENT_NUMBER        263198
OFFENSE_CODE_GROUP         63
OFFENSE_DESCRIPTION       207
DISTRICT                   12
REPORTING_AREA            879
OCCURRED_ON_DATE       218564
YEAR                        4
MONTH                      12
DAY_OF_WEEK                 7
HOUR                       24
UCR_PART                    4
STREET                   3872
Lat                     17763
Long                    17761
Location                17776
dtype: int64
```

Data Manipulation

In [10]:

```
#call the colums, and rename it to be more readable
crime_df.columns
```

Out[10]:

```
Index(['INCIDENT_NUMBER', 'OFFENSE_CODE_GROUP', 'OFFENSE_DESCRIPTION',
       'DISTRICT', 'REPORTING_AREA', 'OCCURRED_ON_DATE', 'YEAR', 'MONTH',
       'DAY_OF_WEEK', 'HOUR', 'UCR_PART', 'STREET', 'Lat', 'Long', 'Locatio
n'],
      dtype='object')
```

In [11]:

```
crime_df.rename(columns={'INCIDENT_NUMBER':'Offender_id', 'OFFENSE_CODE_GROUP':"Group", 'OF
       'DISTRICT':"District", 'REPORTING_AREA':"Reporting_Area", 'OCCURRED_ON_DATE':"Date",
       'DAY_OF_WEEK':"Day", 'HOUR':"Hour", 'UCR_PART':"UCR_Part", 'STREET':"Street", 'Lat':
```

In [12]:

```
#Returns the unique values for the column and we see we don't need to use Strip()
# there isn't a space before each string in this data
crime_df.Group.unique()
```

Out[12]:

```
array(['Larceny', 'Vandalism', 'Towed', 'Investigate Property',
       'Motor Vehicle Accident Response', 'Auto Theft', 'Verbal Disputes',
       'Robbery', 'Fire Related Reports', 'Other', 'Property Lost',
       'Assembly or Gathering Violations', 'Larceny From Motor Vehicle',
       'Medical Assistance', 'Residential Burglary', 'Simple Assault',
       'Violations', 'Harassment', 'Ballistics', 'Property Found',
       'Police Service Incidents', 'Missing Person Reported',
       'Investigate Person', 'Fraud', 'Drug Violation',
       'Aggravated Assault', 'License Plate Related Incidents',
       'Other Burglary', 'Warrant Arrests', 'Disorderly Conduct',
       'Counterfeiting', 'Liquor Violation', 'Firearm Discovery',
       'Landlord/Tenant Disputes', 'Auto Theft Recovery', 'Service',
       'Operating Under the Influence', 'Confidence Games',
       'Restraining Order Violations', 'Firearm Violations',
       'Missing Person Located', 'License Violation',
       'Commercial Burglary', 'Search Warrants',
       'Recovered Stolen Property', 'Offenses Against Child / Family',
       'Prostitution', 'Bomb Hoax', 'Evading Fare',
       'Property Related Damage', 'Harbor Related Incidents',
       'Prisoner Related Incidents', 'Homicide', 'Embezzlement',
       'Explosives', 'Arson', 'Criminal Harassment',
       'Phone Call Complaints', 'Aircraft', 'Biological Threat',
       'Manslaughter', 'Gambling', 'Burglary - No Property Taken'],
      dtype=object)
```

In [13]:

```
#Check the data type for every columns, just to modify  column's type that we need it to do
crime_df.dtypes
```

Out[13]:

```
Offender_id        object
Group              object
Description        object
District           object
Reporting_Area     object
Date               object
Year                int64
Month               int64
Day                object
Hour                int64
UCR_Part           object
Street             object
Lati              float64
Longi             float64
Location           object
dtype: object
```

In [14]:

```
#Calling head for First Row to see the column with its value So,
#we should change Date,Year,Month,Day and Hour to Date Type
crime_df
```

Out[14]:

	Offender_id	Group	Description	District	Reporting_Area	Date	Year	Mo
0	I182070945	Larceny	LARCENY ALL OTHERS	D14	808	2018-09-02 13:00:00	2018	
1	I182070943	Vandalism	VANDALISM	C11	347	2018-08-21 00:00:00	2018	
2	I182070941	Towed	TOWED MOTOR VEHICLE	D4	151	2018-09-03 19:27:00	2018	
3	I182070940	Investigate Property	INVESTIGATE PROPERTY	D4	272	2018-09-03 21:16:00	2018	
4	I182070938	Investigate Property	INVESTIGATE PROPERTY	B3	421	2018-09-03 21:05:00	2018	
...	...	...	...	...	...	...	...	
319066	I060168073-00	Drug Violation	DRUGS - POSS CLASS D - INTENT MFR DIST DISP	E13	912	2018-01-27 14:01:00	2018	
319068	I050310906-00	Warrant Arrests	WARRANT ARREST	D4	285	2016-06-05 17:25:00	2016	
319069	I030217815-08	Homicide	MURDER, NON-NEGLIGIENT MANSLAUGHTER	E18	520	2015-07-09 13:38:00	2015	
319071	I010370257-00	Warrant Arrests	WARRANT ARREST	E13	569	2016-05-31 19:35:00	2016	
319072	142052550	Warrant Arrests	WARRANT ARREST	D4	903	2015-06-22 00:12:00	2015	

263198 rows × 15 columns

In [15]:

```
# #Number of unique crimes by year, sorted in descending order
crime_years=crime_df.groupby('Year')['Group']
crime_years.first()
```

Out[15]:

```
Year
2015    Harassment
2016         Fraud
2017         Fraud
2018       Larceny
Name: Group, dtype: object
```

In [16]:

```
#Number of unique crimes by year, sorted in descending order

crime_df_2015=crime_years.get_group(2015).value_counts
crime_df_2016=crime_years.get_group(2016).value_counts()
crime_df_2017=crime_years.get_group(2017).value_counts()
crime_df_2018=crime_years.get_group(2018).value_counts()
```

Data Visulization

Visualization is required all the time while working upon a dataset in Machine Learning.

Answer for Question2- What types of crimes are the most common?

In [17]:

```
#Top Ten crimes that happned during 4 years ago

#Check how many counts per unique value of OFFENSE_CODE_GROUP
top=crime_df['Group'].value_counts().head(10)
top
```

Out[17]:

```
Motor Vehicle Accident Response    29033
Larceny                            23630
Medical Assistance                 21887
Investigate Person                 17260
Other                              13248
Vandalism                          13172
Simple Assault                     12770
Verbal Disputes                    12428
Towed                              10317
Drug Violation                     10243
Name: Group, dtype: int64
```

In [18]:

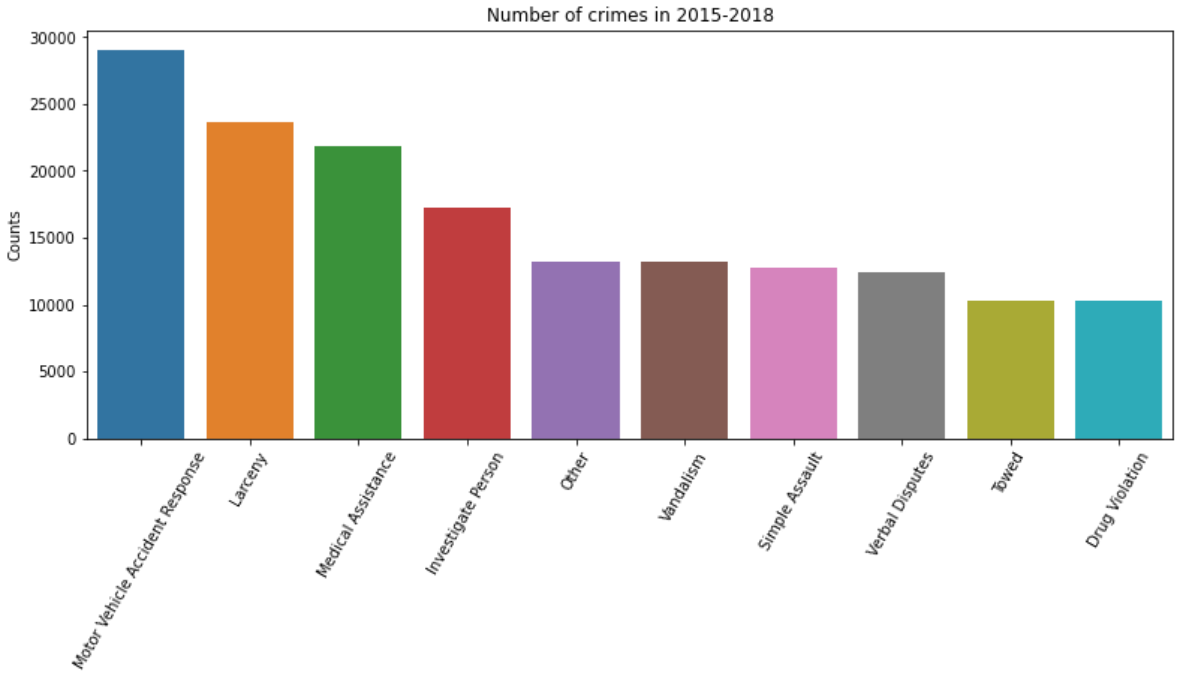
```
# plot the Top Ten common crimes over the years:20115,2016,2017,2017
# Set the width and height of the figure
plt.figure(figsize=(13,5))

# Add title
plt.title("Number of crimes in 2015-2018")

sns.barplot(x=top.index,y=top)

#Rotate x-labels, otherwise it's utterly hectic
plt.xticks(rotation=60)

# Add label for vertical axis
plt.ylabel("Counts");
```



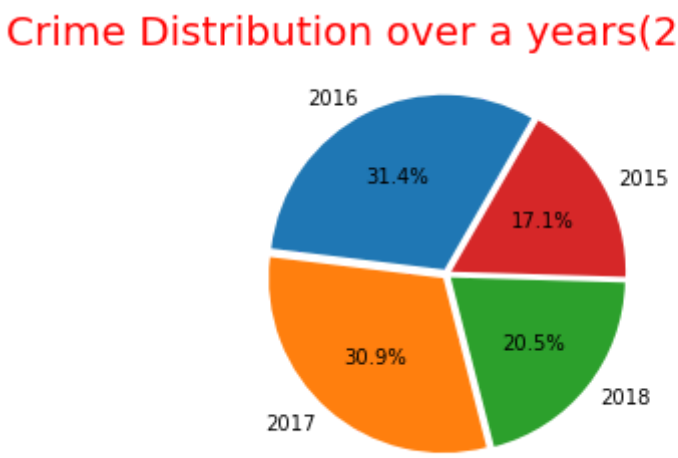
Apparently Boston seems to be a dangerous city to drive in.

Answer for Question2- part 1: Does the frequencies of crime change over years?

In [19]:

```
plt.title('Crime Distribution over a years(2015:2018)',fontsize=20,color = 'r')
explode = (0.03, 0.03, 0.03,0.03)
labels = ['2016','2017','2018','2015']
years = crime_df['Year'].value_counts()
plt.pie(years, explode=explode,startangle=60, labels=labels,autopct='%0.01f%%')

plt.show()
```



When we look above at the distribution of the crime over the years, we see that there were more crimes in 2016 and 2017 than in other years. Let's look in more detail to understand the reason for this. We may have missing data.

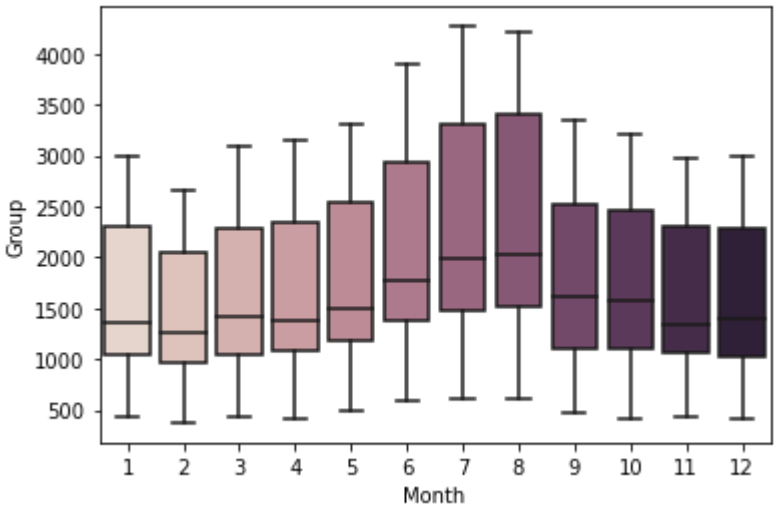
Answer for Question2- part 2: Does the frequencies of crime change over the months?

The box-plot clarify the most crimes happend over months

Looking at the next (box_plot), month variable in the regional distribution, the crime rate is 6., 7. and 8. months

In [20]:

```
grouped = crime_df.groupby(['Month','District']).count()
sns.boxplot(x ="Month", y = "Group", data = grouped.reset_index(), palette="ch:.100");
```



In another way, we can create a new columns to show (Season, Day& Night)

Seasons We have years and months in our data, but we create the seasons column to see seasonality.

In [21]:

```
def getSeason(month):
    if (month == 12 or month == 1 or month == 2):
       return "Winter"
    elif(month == 3 or month == 4 or month == 5):
       return "Spring"
    elif(month ==6 or month==7 or month == 8):
       return "Summer"
    else:
       return "Fall"
```

In [22]:

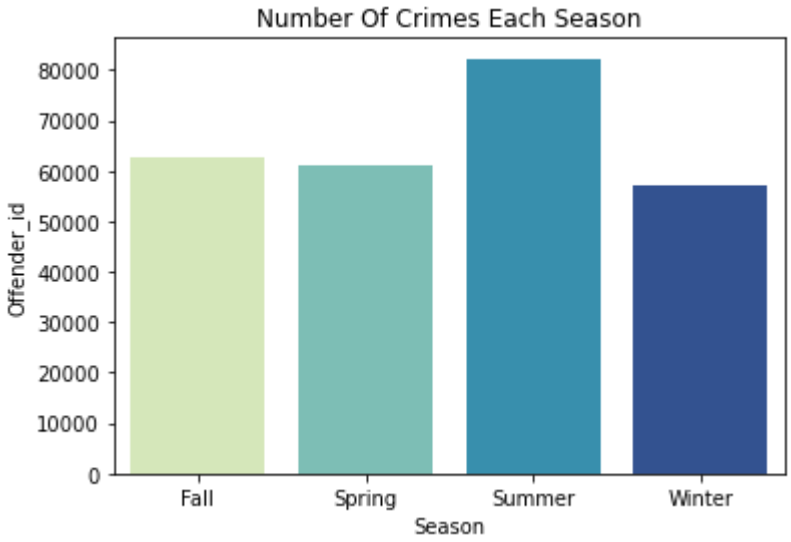
```
crime_df['Season'] =crime_df.Month.apply(getSeason)
crime_df.head(3)
```

Out[22]:

	Offender_id	Group	Description	District	Reporting_Area	Date	Year	Month	Da
0	I182070945	Larceny	LARCENY ALL OTHERS	D14	808	2018-09-02 13:00:00	2018	9	Sunda
1	I182070943	Vandalism	VANDALISM	C11	347	2018-08-21 00:00:00	2018	8	Tuesda
2	I182070941	Towed	TOWED MOTOR VEHICLE	D4	151	2018-09-03 19:27:00	2018	9	Monda

In [23]:

```
season_counts = crime_df.groupby('Season')['Offender_id'].count().to_frame().reset_index()
ax = sns.barplot(x = 'Season' , y = "Offender_id",data = season_counts, palette='YlGnBu')
plt.title('Number Of Crimes Each Season');
```



According to the graph above, we see an increase in crime during the summer season. Is there seasonality in crimes? The answer is in the Crimes by Month of Year graph. During the summer months, data were entered for all years, so it is normal to see more crime in these months. This does not mean seasonality.

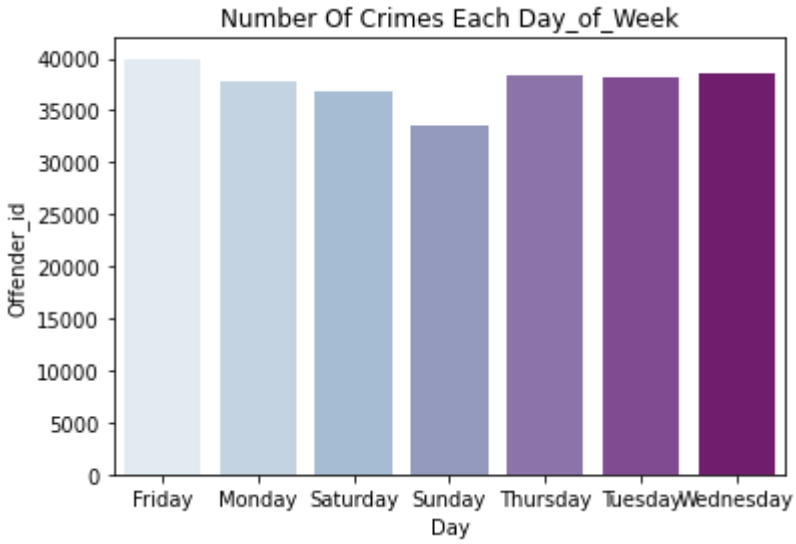
Answer for Question2- part 2: Number Of Crimes Each Day_of_Week:

Does the frequencies of crime change over the day?

In [24]:

```
day_counts = crime_df.groupby('Day').count()['Offender_id'].to_frame().reset_index()
ax = sns.barplot(x = 'Day' , y="Offender_id", data = day_counts, palette='BuPu')
plt.title('Number Of Crimes Each Day_of_Week')
print(day_counts)
```

```
         Day  Offender_id
0     Friday        39967
1     Monday        37829
2   Saturday        36825
3     Sunday        33599
4   Thursday        38312
5    Tuesday        38140
6  Wednesday        38526
```



we noticed from above more crimes were committed on Fridays, but there is not much difference between the numbers.

Answer for Question 3:Which time of the day the most of crimes committed?

Plotly Backage - 2D Histogram of a Bivariate Normal Distribution

A 2D histogram, also known as a density heatmap, is the 2-dimensional generalization of a histogram which resembles a heatmap but is computed by grouping a set of points specified by their x and y coordinates into bins, and applying an aggregation function such as count or sum (if z is provided) to compute the color of the tile representing the bin.

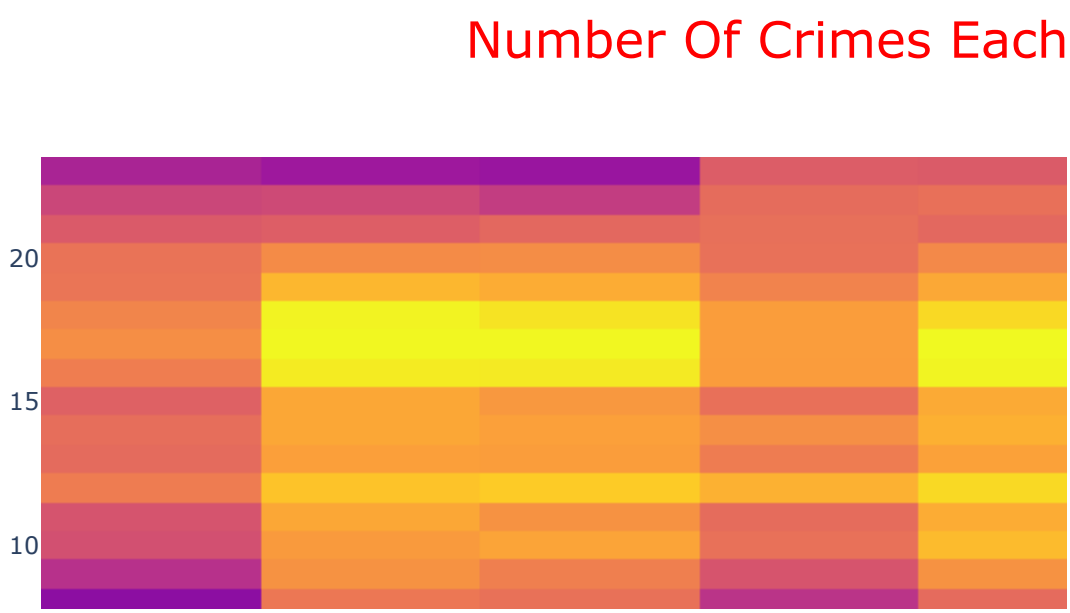
In [25]:

```
x = crime_df.Day
y = crime_df.Hour

fig = go.Figure(go.Histogram2d(x=x,y=y))

#I tried many times to add a title to this figure and finally i collected these methods fro
#I'm very happy when i see the results are easily coming True with me :)

fig.update_layout(title_text='Number Of Crimes Each Hour',title_font_color='Red',title_font
fig.show()
```



More details, Display a Time of day or night do most crimes take place?

In [26]:

```
#Create a function to display Most crimes happened in the morning or evening
def day_night(h):
    if h>=12:
        return "Day-Evening"
    else:
        return "night-Morning"
```

In [41]:

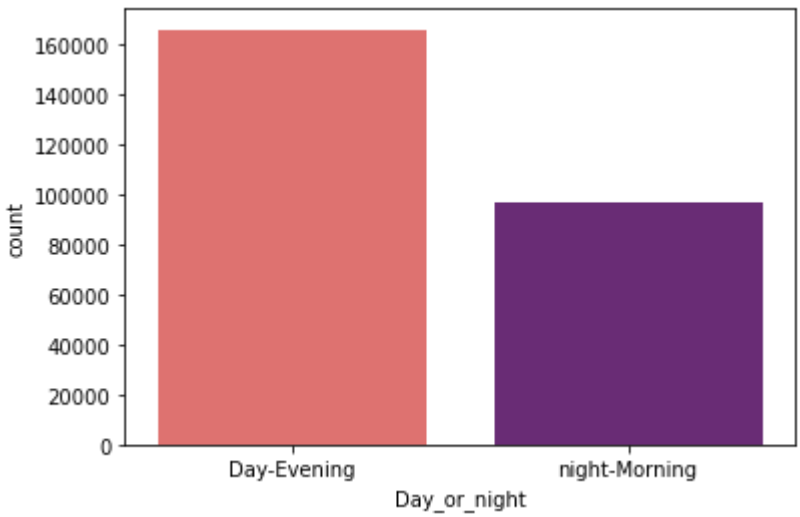
```
#crime_df['Night'].loc[crime_df['Day_']==0]=1
crime_df['Day_or_night'] = crime_df.Hour.apply(day_night)
crime_df.head(3)
```

Out[41]:

District	Reporting_Area	Date	Year	Month	Day	Hour	UCR_Part	Street	Lat
D14	808	2018-09-02 13:00:00	2018	9	Sunday	13	Part One	LINCOLN ST	42.35779
C11	347	2018-08-21 00:00:00	2018	8	Tuesday	0	Part Two	HECLA ST	42.30682
D4	151	2018-09-03 19:27:00	2018	9	Monday	19	Part Three	CAZENOVE ST	42.34658

In [28]:

```
sns.countplot(x=crime_df['Day_or_night'],palette='magma_r');
```



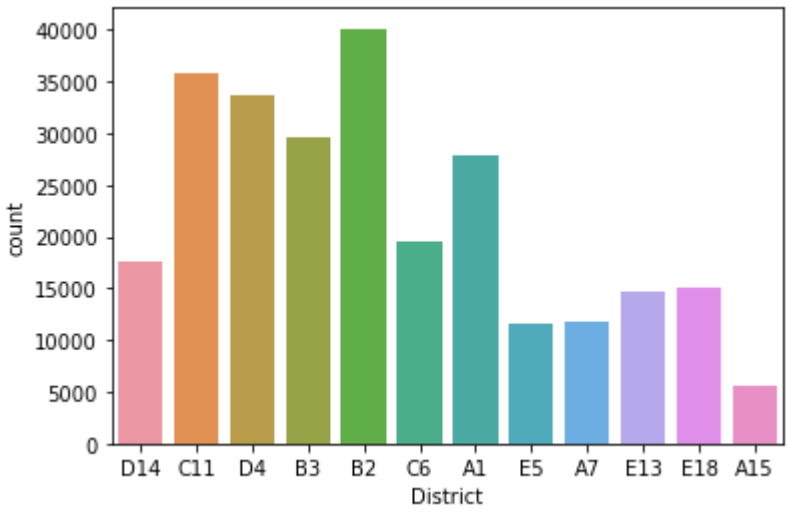
Answer for Question4: Where are different types of crime most likely to occur?

This count-plot shows us the most dangerous places

In this section, we'll try to find which features have district patterns in which crime is higher or lower.

In [29]:

```
sns.countplot(data=crime_df, x='District');
```



We can see that district B2 has the highest crime rate, and district A15 has the lowest crime rate.

In addition to - Plotting interactive map in python using Folium library

Folium is a python package that combines all the spectrum of tools python offers to manipulate data with the leaflet javascript library to create rich and interactive maps.

In [30]:

```
# i need all crimes that occurerd in 2018 to determine it on the map
df = crime_df[crime_df['Year'] == 2018]
```

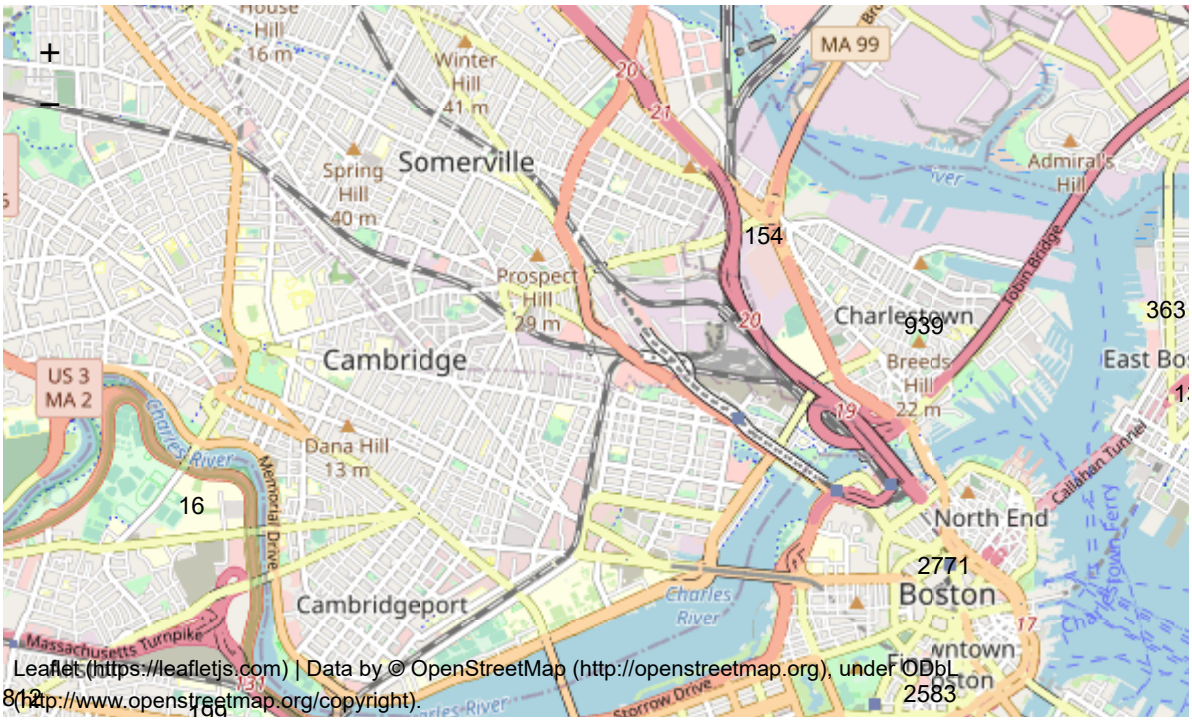
In [31]:

```
#df.lat = pd.to_numeric(df.Lati)
#df.long = pd.to_numeric(df.Longi)

m = folium.Map(location=[42.361145, -71.057083], zoom_start=13)

#Add points to the map
cluster = MarkerCluster()
for idx, row in df.iterrows():
    if not math.isnan(row['Longi']) and not math.isnan(row['Lati']):
        cluster.add_child(Marker([row['Lati'], row['Longi']]))
m.add_child(cluster)
```

Out[31]:



WordCloud Backage

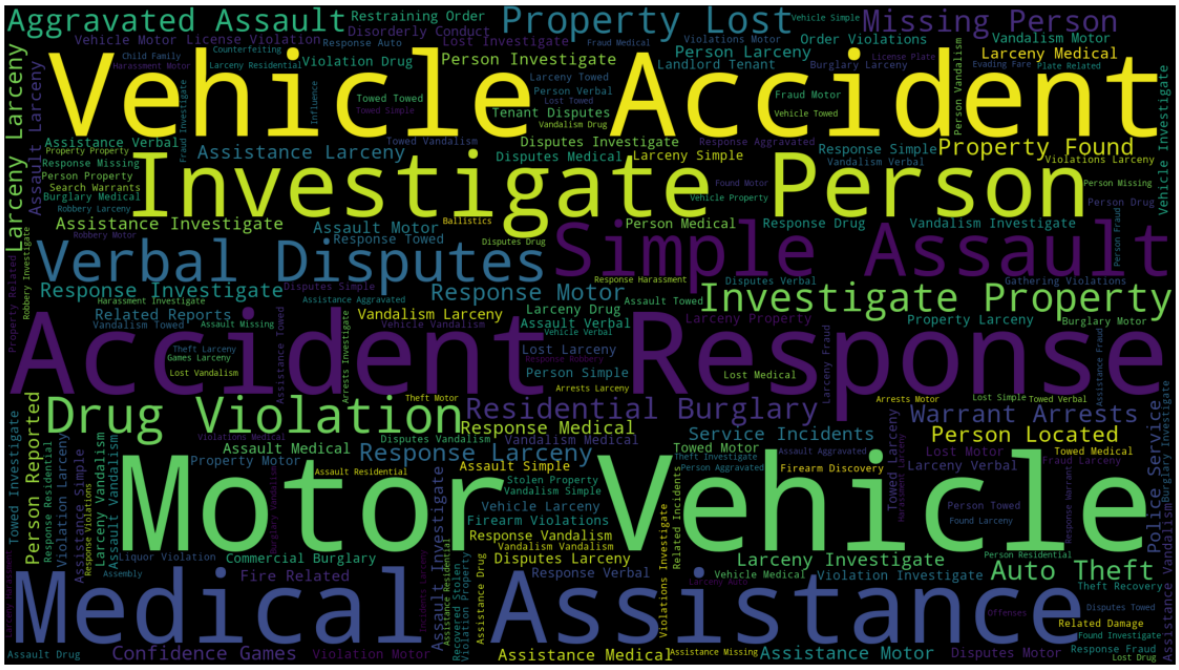
Python offers an inbuilt library called “WordCloud” which helps to generate Word cloud.

We can install this library by using the following command in => anaconda prompt:

! pip install wordcloud

In [32]:

```
plt.figure(figsize=(20,15))
wordcloud = WordCloud(
                          background_color='Black',
                          width=1920,
                          height=1080
                         ).generate(" ".join(crime_df.Group))
plt.imshow(wordcloud)
plt.axis('off')
plt.savefig('graph.png')
plt.show()
```



Conclusions

In [33]:

```
#calculate the highest cirems that occured in a Street,Year,Month, day and time
max_street_crime = crime_df['Street'].value_counts().index[0]
max_year_crime = crime_df['Year'].value_counts().index[0]
max_hour_crime = crime_df['Hour'].value_counts().index[0]
max_month_crime = crime_df['Month'].value_counts().index[0]
max_day_crime = crime_df['Day'].value_counts().index[0]

Month = ['January','February','March','April','May','June','July',
         'August','September','October','November','December']

print('Street with higher occurrence of crimes:', max_street_crime)
print('Year with highest crimes occurrence:', max_year_crime)
print('Hour with highest crimes occurrence:', max_hour_crime)
print('Month with highest crime occurrence:', Month[max_month_crime-1], max_month_crime)
print('Day with highest crimes occurrence:', max_day_crime)
```

```
Street with higher occurrence of crimes: WASHINGTON ST
Year with highest crimes occurrence: 2017
Hour with highest crimes occurrence: 17
Month with highest crime occurrence: August 8
Day with highest crimes occurrence: Friday
```

Washington St gets the prize to "The No 1 Street with more crimes 2015-2018, Boston". Probably because it is the longest street in Boston. Many motor vehicle accidents must happen in Washington St, as well as larceny, and drug violation in a less extent.

New Feature (Discover - Offenders's personal_Information from SQL- DataBase)

In this step I tried to discover the whole information about the offenders that caused the crimes and I suggest i have a dataBase for all citizens and residents information , so i'll match the offenders_ID with a dataBase.Personal_Id to bring all offeneders datails like his name,location and other info.To do this step i have to read how to convert a:

SQLite Database with Pandas

An SQLite database can be read directly into Python Pandas (a data analysis library). In this article we'll demonstrate loading data from an SQLite database table into a Python Pandas Data Frame. We'll also briefly cover the creation of the sqlite database table using Python.

1st step-pandas.DataFrame.to_sql

DataFrame.to_sql(name, con, schema=None, if_exists='fail', index=True, index_label=None, chunksize=None, dtype=None, method=None) [source]¶ Write records stored in a DataFrame to a SQL database.

Databases supported by SQLAlchemy [1] are supported. Tables can be newly created, appended to, or overwritten.

In [40]:

```
# First of all,we should import two these modules:
#(from sqlalchemy import create_engine)+(#import pandas as pd)

#Create an in-memory SQLite database
engine1 = create_engine("sqlite:///", echo=False)
sqlite_connection = engine1.connect()

#convert the crime dataFram to Sql
df.to_sql('crime_df', con=engine1)
engine1.execute("SELECT * FROM crime_Df limit 10;").fetchall()
```

Out[40]:

```
[(0, 'I182070945', 'Larceny', 'LARCENY ALL OTHERS', 'D14', '808', '2018-09-0
2 13:00:00', 2018, 9, 'Sunday', 13, 'Part One', 'LINCOLN ST', 42.35779134, -
71.13937053, '(42.35779134, -71.13937053)', 'Fall', 'Day-Evening'),
 (1, 'I182070943', 'Vandalism', 'VANDALISM', 'C11', '347', '2018-08-21 00:0
0:00', 2018, 8, 'Tuesday', 0, 'Part Two', 'HECLA ST', 42.30682138, -71.06030
035, '(42.30682138, -71.06030035)', 'Summer', 'night-Morning'),
 (2, 'I182070941', 'Towed', 'TOWED MOTOR VEHICLE', 'D4', '151', '2018-09-03
19:27:00', 2018, 9, 'Monday', 19, 'Part Three', 'CAZENOVE ST', 42.34658879,
-71.07242943, '(42.34658879, -71.07242943)', 'Fall', 'Day-Evening'),
 (3, 'I182070940', 'Investigate Property', 'INVESTIGATE PROPERTY', 'D4', '27
2', '2018-09-03 21:16:00', 2018, 9, 'Monday', 21, 'Part Three', 'NEWCOMB S
T', 42.33418175, -71.07866441, '(42.33418175, -71.07866441)', 'Fall', 'Day-E
vening'),
 (4, 'I182070938', 'Investigate Property', 'INVESTIGATE PROPERTY', 'B3', '42
1', '2018-09-03 21:05:00', 2018, 9, 'Monday', 21, 'Part Three', 'DELHI ST',
42.27536542, -71.09036101, '(42.27536542, -71.09036101)', 'Fall', 'Day-Eveni
ng'),
 (5, 'I182070936', 'Motor Vehicle Accident Response', 'M/V ACCIDENT INVOLVIN
G PEDESTRIAN - INJURY', 'C11', '398', '2018-09-03 21:09:00', 2018, 9, 'Monda
y', 21, 'Part Three', 'TALBOT AVE', 42.29019621, -71.07159012, '(42.2901962
1, -71.07159012)', 'Fall', 'Day-Evening'),
 (6, 'I182070933', 'Auto Theft', 'AUTO THEFT', 'B2', '330', '2018-09-03 21:2
5:00', 2018, 9, 'Monday', 21, 'Part One', 'NORMANDY ST', 42.30607218, -71.08
27326, '(42.30607218, -71.08273260)', 'Fall', 'Day-Evening'),
 (7, 'I182070932', 'Verbal Disputes', 'VERBAL DISPUTE', 'B2', '584', '2018-0
9-03 20:39:37', 2018, 9, 'Monday', 20, 'Part Three', 'LAWN ST', 42.32701648,
-71.10555088, '(42.32701648, -71.10555088)', 'Fall', 'Day-Evening'),
 (8, 'I182070931', 'Robbery', 'ROBBERY - STREET', 'C6', '177', '2018-09-03 2
0:48:00', 2018, 9, 'Monday', 20, 'Part One', 'MASSACHUSETTS AVE', 42.3315214
8, -71.07085307, '(42.33152148, -71.07085307)', 'Fall', 'Day-Evening'),
 (9, 'I182070929', 'Verbal Disputes', 'VERBAL DISPUTE', 'C11', '364', '2018-
09-03 20:38:00', 2018, 9, 'Monday', 20, 'Part Three', 'LESLIE ST', 42.295146
64, -71.05860832, '(42.29514664, -71.05860832)', 'Fall', 'Day-Evening')]
```

2nd step-Connect Python to a Database

Connect this notebook to the chinook.db database using create_engine

In [35]:

```
engine2 = create_engine("sqlite:///chinook.db")

# List the table names in the database
all_tables = engine2.table_names()
all_tables
```

```
<ipython-input-35-283bdcff0b33>:4: SADeprecationWarning:

The Engine.table_names() method is deprecated and will be removed in a futur
e release.  Please refer to Inspector.get_table_names(). (deprecated since:
1.4)
```

Out[35]:

```
['Citizen_info', 'citizen_info2', 'sqlite_sequence']
```

* To explore the dataBase

Write a SQL query using pd.read_sql() to show all of the data in the Citizen_info table

In [36]:

```
citizen_id = pd.read_sql('SELECT * FROM Citizen_info;', engine2)
citizen_id
```

Out[36]:

	Personal_id	LastName	FirstName	Title	ReportsTo	BirthDate	HireDate	Address	
0	\tI182070945	John	Andrew	General Manager	3	1962-02-18 00:00:00	2002-08-14 00:00:00	11120 Jasper Ave NW	Edm
1	\tI182070943	Edwards	Nancy	Sales Manager	1	1958-12-08 00:00:00	2002-05-01 00:00:00	825 8 Ave SW	Ca
2	\tI182070933	Peacock	Jane	Sales Support Agent	2	1973-08-29 00:00:00	2002-04-01 00:00:00	1111 6 Ave SW	Ca

In [37]:

```
# if (offender_id) match with (Personal_ID),it will bring all personal info related to the
```

In [38]:

```
df.to_sql('crime_df', con=engine1, if_exists='replace',
          index_label='Incident_Id')
engine2.execute("SELECT * FROM Citizen_info").fetchall()

#engine2.execute("SELECT * FROM Citizen_info2").fetchall()
```

Out[38]:

```
[('\tI182070945', 'John', 'Andrew', 'General Manager', '3', '1962-02-18 00:0
0:00', '2002-08-14 00:00:00', '11120 Jasper Ave NW', 'Edmonton', 'AB', 'Cana
da', 'T5K 2N1', '+1 (780) 428-9482', '+1 (780) 428-3457', 'andrew@chinookcor
p.com'),
 ('\tI182070943', 'Edwards', 'Nancy', 'Sales Manager', '1', '1958-12-08 00:0
0:00', '2002-05-01 00:00:00', '825 8 Ave SW', 'Calgary', 'AB', 'Brookline',
'T2P 2T3', '+1 (403) 262-3443', '+1 (403) 262-3322', 'nancy@chinookcorp.co
m'),
 ('\tI182070933', 'Peacock', 'Jane', 'Sales Support Agent', '2', '1973-08-29
00:00:00', '2002-04-01 00:00:00', '1111 6 Ave SW', 'Calgary', 'AB', 'US', 'T
2P 5M5', '+1 (403) 262-3443', '+1 (403) 262-6712', 'jane@chinookcorp.com')]
```