Boston Crime Dataset - 1st Draft

In [56]:
```
# Importing pandas and numpy libraries
import pandas as pd
import numpy as np

# Read csv file into a pandas dataframe.
crime = pd.read_csv('C:/crime.csv',encoding = 'unicode_escape')

# Using set_option(),changing the default number of rows and columns to be dis
played.
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)

# Taking a look at the first few rows of the dataset.
crime.head()
```

```
C:\Anaconda\lib\site-packages\IPython\core\interactiveshell.py:2785: DtypeWar
ning: Columns (6) have mixed types. Specify dtype option on import or set low
_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[56]:

	INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRI
0	I182080058	2403	Disorderly Conduct	DISTURBING THE P
1	I182080053	3201	Property Lost	PROPERTY - LOST
2	I182080052	2647	Other	THREATS TO DO BO HARM
3	I182080051	413	Aggravated Assault	ASSAULT - AGGRAV - BATTERY
4	I182080050	3122	Aircraft	AIRCRAFT INCIDEN

In [57]:
```
crime.columns            # Printing all the column names of the dataset
```

Out[57]:
```
Index(['INCIDENT_NUMBER', 'OFFENSE_CODE', 'OFFENSE_CODE_GROUP',
       'OFFENSE_DESCRIPTION', 'DISTRICT', 'REPORTING_AREA', 'SHOOTING',
       'OCCURRED_ON_DATE', 'YEAR', 'MONTH', 'DAY_OF_WEEK', 'HOUR', 'UCR_PAR
T',
       'STREET', 'Lat', 'Long', 'Location'],
      dtype='object')
```

Data Cleaning Process:

```
In [58]: crime.isnull().sum()      # sum of the missing values in each column
```

```
Out[58]: INCIDENT_NUMBER             0
         OFFENSE_CODE                0
         OFFENSE_CODE_GROUP          0
         OFFENSE_DESCRIPTION         0
         DISTRICT                 1774
         REPORTING_AREA              0
         SHOOTING               326765
         OCCURRED_ON_DATE            0
         YEAR                        0
         MONTH                       0
         DAY_OF_WEEK                 0
         HOUR                        0
         UCR_PART                   93
         STREET                  10977
         Lat                     20632
         Long                    20632
         Location                    0
         dtype: int64
```

```
In [59]: # Removing the columns which are insignificant
         for column in crime:
             if(crime[column].count() < 100000):
                 crime.drop([column], axis = 1, inplace = True)
         crime.head()
```

Out[59]:

	INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRI
0	I182080058	2403	Disorderly Conduct	DISTURBING THE P
1	I182080053	3201	Property Lost	PROPERTY - LOST
2	I182080052	2647	Other	THREATS TO DO BC HARM
3	I182080051	413	Aggravated Assault	ASSAULT - AGGRAV - BATTERY
4	I182080050	3122	Aircraft	AIRCRAFT INCIDEN

In [60]:
```
crime.isnull().sum()
```

Out[60]:
```
INCIDENT_NUMBER            0
OFFENSE_CODE               0
OFFENSE_CODE_GROUP         0
OFFENSE_DESCRIPTION        0
DISTRICT                1774
REPORTING_AREA             0
OCCURRED_ON_DATE           0
YEAR                       0
MONTH                      0
DAY_OF_WEEK                0
HOUR                       0
UCR_PART                  93
STREET                 10977
Lat                    20632
Long                   20632
Location                   0
dtype: int64
```

In [61]:
```
# Filling the columns with fillna method
crime.fillna({
   'UCR_PART': 'N/A',
    'DISTRICT': 'N/A',
    'STREET': 'N/A',
    'Lat' : 'N/A',
    'Long' : 'N/A'
}, inplace= True)
crime.head()
```

Out[61]:

	INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRI
0	I182080058	2403	Disorderly Conduct	DISTURBING THE P
1	I182080053	3201	Property Lost	PROPERTY - LOST
2	I182080052	2647	Other	THREATS TO DO BO HARM
3	I182080051	413	Aggravated Assault	ASSAULT - AGGRAV - BATTERY
4	I182080050	3122	Aircraft	AIRCRAFT INCIDEN

In [62]:
```
# Deleting 'REPORTING_AREA' variable as it is insignificant for the data analy
sis.
del crime['REPORTING_AREA']
crime.head()
```

Out[62]:

	INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRII
0	I182080058	2403	Disorderly Conduct	DISTURBING THE P
1	I182080053	3201	Property Lost	PROPERTY - LOST
2	I182080052	2647	Other	THREATS TO DO BO HARM
3	I182080051	413	Aggravated Assault	ASSAULT - AGGRAV - BATTERY
4	I182080050	3122	Aircraft	AIRCRAFT INCIDEN

In [63]:
```
crime.isnull().sum()
```

Out[63]:
```
INCIDENT_NUMBER        0
OFFENSE_CODE           0
OFFENSE_CODE_GROUP     0
OFFENSE_DESCRIPTION    0
DISTRICT               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
```

The above output shows that all the Null values have been taken care of.

In [64]:
```
#  Specifying 'INCIDENT_NUMBER' column to use as index.
crime.set_index('INCIDENT_NUMBER', inplace = True)
crime.head()
```

Out[64]:

	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTI
INCIDENT_NUMBER			
I182080058	2403	Disorderly Conduct	DISTURBING THE PEA
I182080053	3201	Property Lost	PROPERTY - LOST
I182080052	2647	Other	THREATS TO DO BOD HARM
I182080051	413	Aggravated Assault	ASSAULT - AGGRAVAT - BATTERY
I182080050	3122	Aircraft	AIRCRAFT INCIDENTS

In [65]:
```
# Renaming few column names
crime.rename(columns={'INCIDENT_NUMBER': 'Incident_Number', 'OFFENSE_CODE': 'O
ffense_Code', 'OFFENSE_CODE_GROUP':'Offense_Code_Group', 'OFFENSE_DESCRIPTION'
: 'Offense_Description', 'DISTRICT': 'District', 'OCCURRED_ON_DATE':'Occured_o
n_Date', 'YEAR': 'Year', 'MONTH':'Month', 'DAY_OF_WEEK':'Day_of_Week', 'HOUR':
'Hour', 'STREET':'Street'}, inplace=True)
crime.head()
```

Out[65]:

	Offense_Code	Offense_Code_Group	Offense_Description	District
INCIDENT_NUMBER				
I182080058	2403	Disorderly Conduct	DISTURBING THE PEACE	E18
I182080053	3201	Property Lost	PROPERTY - LOST	D14
I182080052	2647	Other	THREATS TO DO BODILY HARM	B2
I182080051	413	Aggravated Assault	ASSAULT - AGGRAVATED - BATTERY	A1
I182080050	3122	Aircraft	AIRCRAFT INCIDENTS	A7

In [66]:
```
# Checking for Duplicate Values
crime.duplicated().sum()
```

Out[66]:
```
968
```

In [67]:
```
# Removing all the Duplicate Values from the dataset.
crime.drop_duplicates(keep=False, inplace=True)
crime.head()
```

Out[67]:

	Offense_Code	Offense_Code_Group	Offense_Description	District
INCIDENT_NUMBER				
I182080058	2403	Disorderly Conduct	DISTURBING THE PEACE	E18
I182080053	3201	Property Lost	PROPERTY - LOST	D14
I182080052	2647	Other	THREATS TO DO BODILY HARM	B2
I182080051	413	Aggravated Assault	ASSAULT - AGGRAVATED - BATTERY	A1
I182080050	3122	Aircraft	AIRCRAFT INCIDENTS	A7

Now that the Data is cleaned, We can perform Data Visualization process to answer all the important questions.

In [68]:
```
# Importing datetime, matplotlib and seaborn libraries.
import os
import csv
from datetime import datetime
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
from matplotlib import cm
import seaborn as sns
```

In [69]:
```
#Functions for visulization.

def create_list_number_crime(name_column, list_unique):
    # list_unique = df[name_column].unique()

    i = 0

    list_number = list()

    while i < len(list_unique):
        list_number.append(len(crime.loc[crime[name_column] == list_unique[i
]]))
        i += 1

    return list_unique, list_number
```

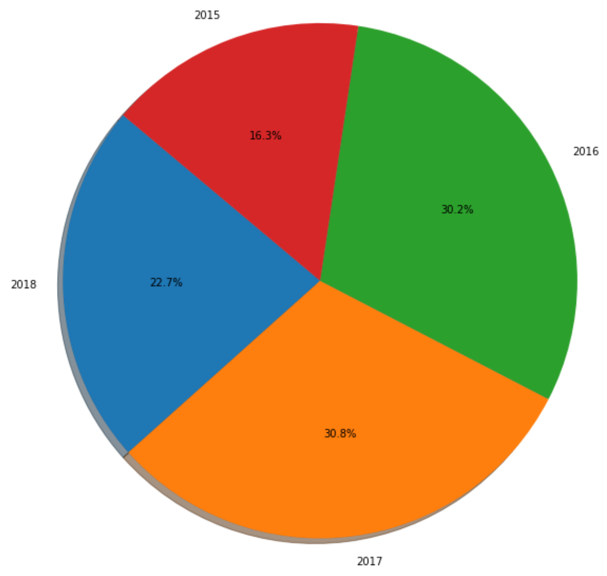
In [70]:
```
# pie_plot def  function
def pie_plot(list_number, list_unique):
    plt.figure(figsize=(20,10))
    plt.pie(list_unique,
        labels=list_number,
        autopct='%1.1f%%',
        shadow=True,
        startangle=140)

    plt.axis('equal')
    plt.show()
    return 0
```

In [71]:
```
# bar_code
def bar_chart(list_number, list_unique):
    objects = list_unique
    y_pos = np.arange(len(objects))
    performance = list_number

    plt.figure(figsize=(20,10))
    plt.bar(y_pos, performance, align='center', alpha=0.5)
    plt.xticks(y_pos, objects)
    plt.ylabel('Number')
    plt.show()

    return 0
```

Q. How the Crime has Changed Over Years ?

A. Total number of crime for each year using Pie Graph

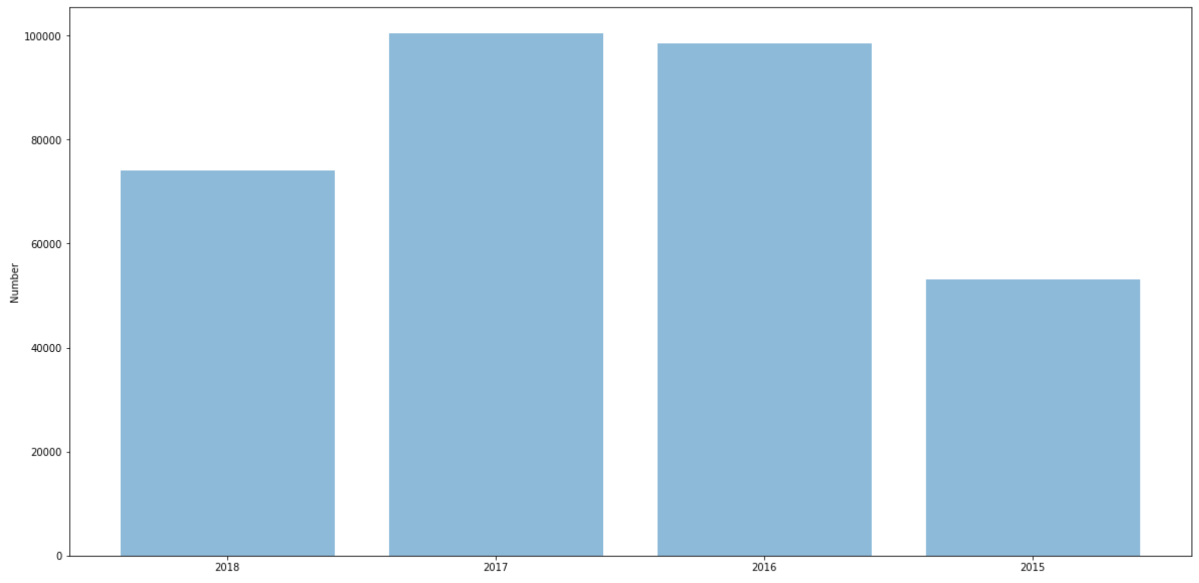
In [72]:
```
list_unique_year, list_number_year = create_list_number_crime('Year',crime['Ye
ar'].unique())

pie_plot(list_unique_year, list_number_year)
```



Out[72]:
```
0
```

Total number of crime for each year using Bar Plot.

In [73]:
```
bar_chart(list_number_year,list_unique_year)
```



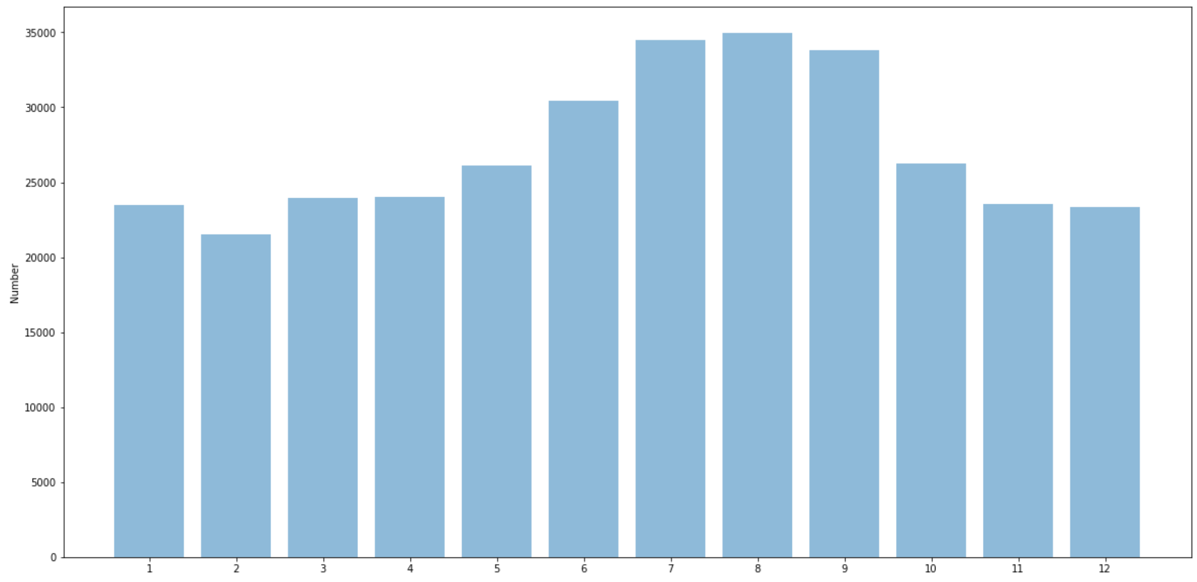
Out[73]:
```
0
```

After analyzing the Pie and Bar charts of crime committed each year, we can conclude that crime has occured most in the year 2017. But in the year 2018, crimes in Boston has decreased by a good margin which indicates a positive sign.

B. Total number of Crime: Month Wise

```
In [ ]: list_unique_month, list_number_month = create_list_number_crime('Month',list(r
        ange(1,13)))
```

```
In [75]: bar_chart(list_number_month,list_unique_month)
```

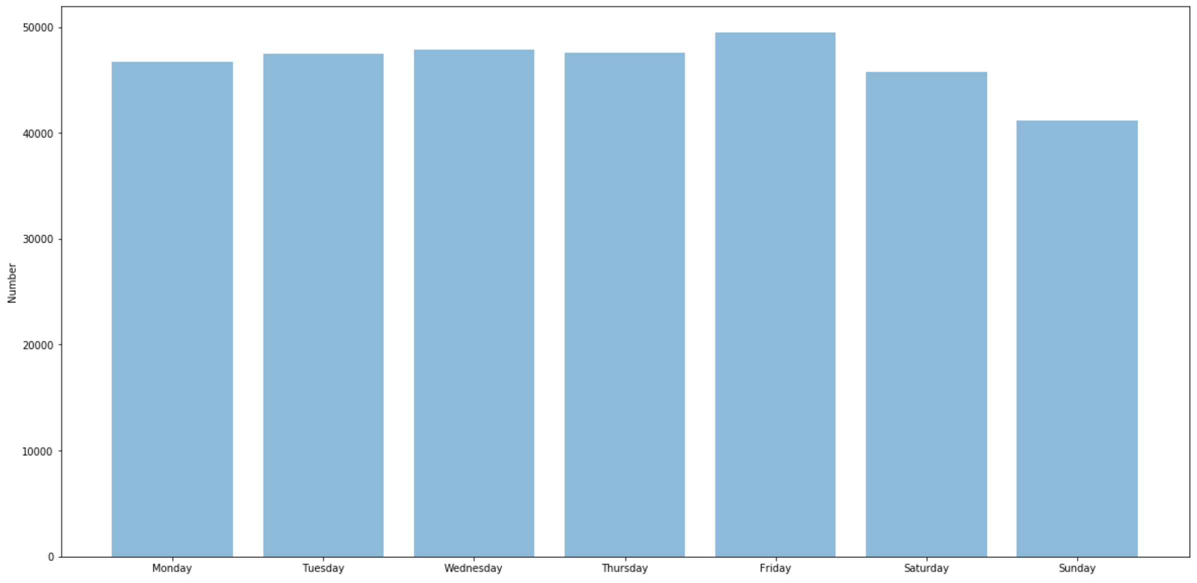


```
Out[75]: 0
```

After observing the above bar plot of crimes occuring month wise, we can conclude that the most number of crimes have occured in the month of 'August'.

C. Total number of crime for each day of week

In [76]:
```
#5.4. Total number crime for each day of week

day_of_week = ('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturd
ay', 'Sunday')

list_unique_day, list_number_day = create_list_number_crime('Day_of_Week',day_
of_week)

#pie_plot(list_unique_day,list_number_day)

bar_chart(list_number_day,list_unique_day)
```



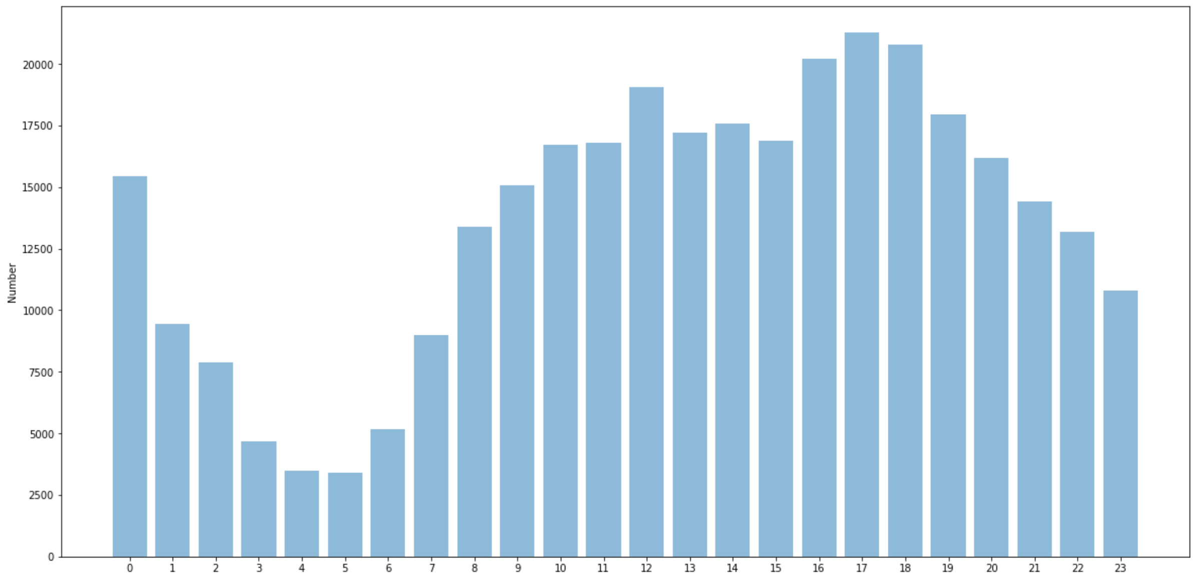
Out[76]:
```
0
```

The above bar plot shows that crimes occur most on Friday.

D. Total number of crime for each hour

In [77]:
```
list_unique_hour, list_number_hour = create_list_number_crime('Hour',list(rang
e(0,24)))
```

```
In [78]: bar_chart(list_number_hour,list_unique_hour)
```



```
Out[78]: 0
```

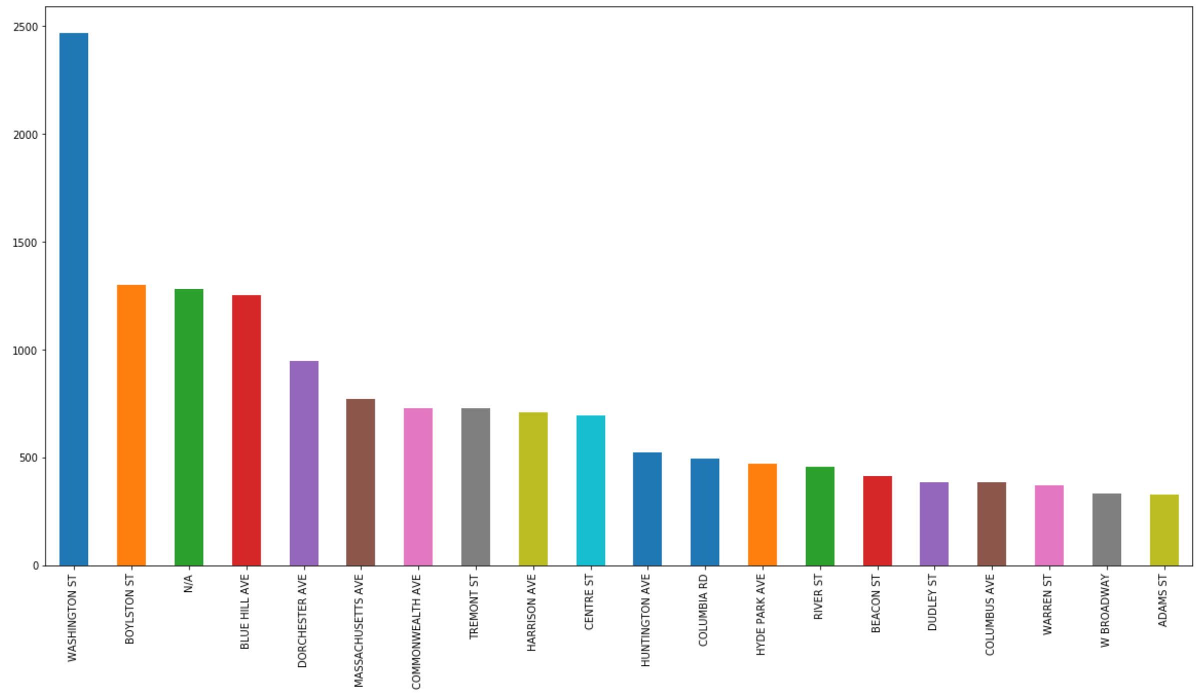
We can clearly observe that most of the crimes have occured in the 17th hour of the day followed by 18th which means Evening Time is clearly a bit unsafe.

Q. Which are the Most Safe & Unsafe Streets in Boston Between the period of 2015 - 2018 ?

E. Bar plot of crimes commited on the streets of Boston

a. in the year 2015:

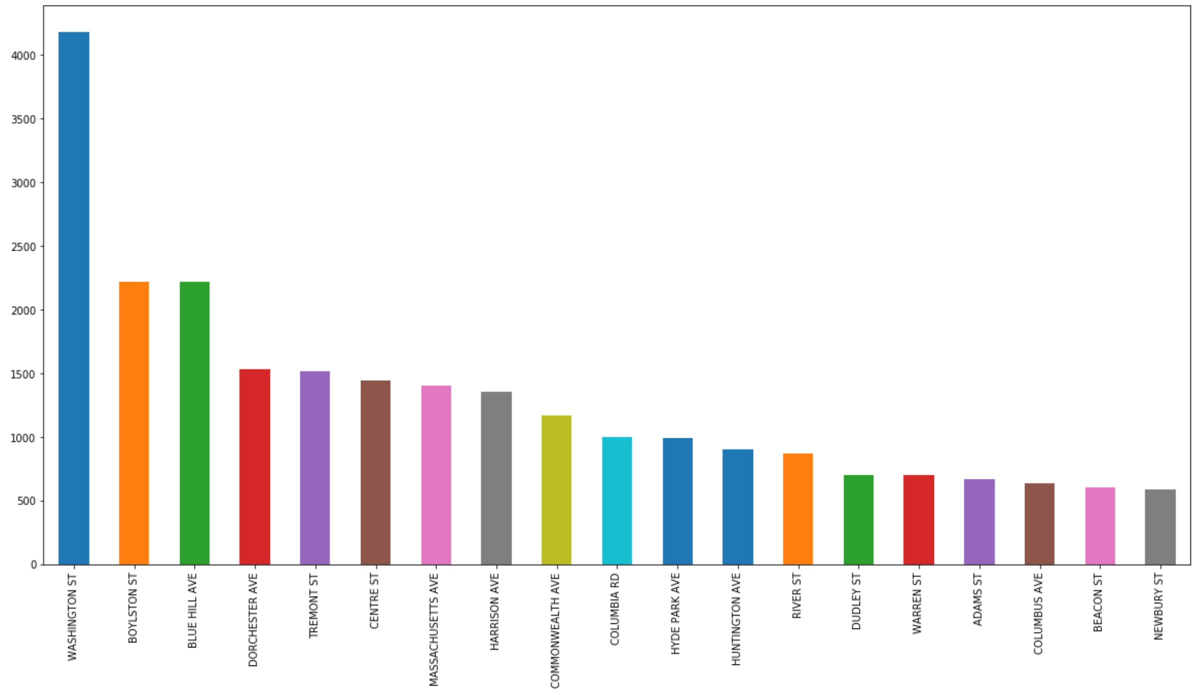
In [79]:
```
plt.figure(figsize=(20,10))
crime['Street'].loc[crime['Year']==2015].value_counts()[:20].plot.bar()
plt.show()
```



Bar plot of crimes commited on the streets of Boston

b. in the year 2016:

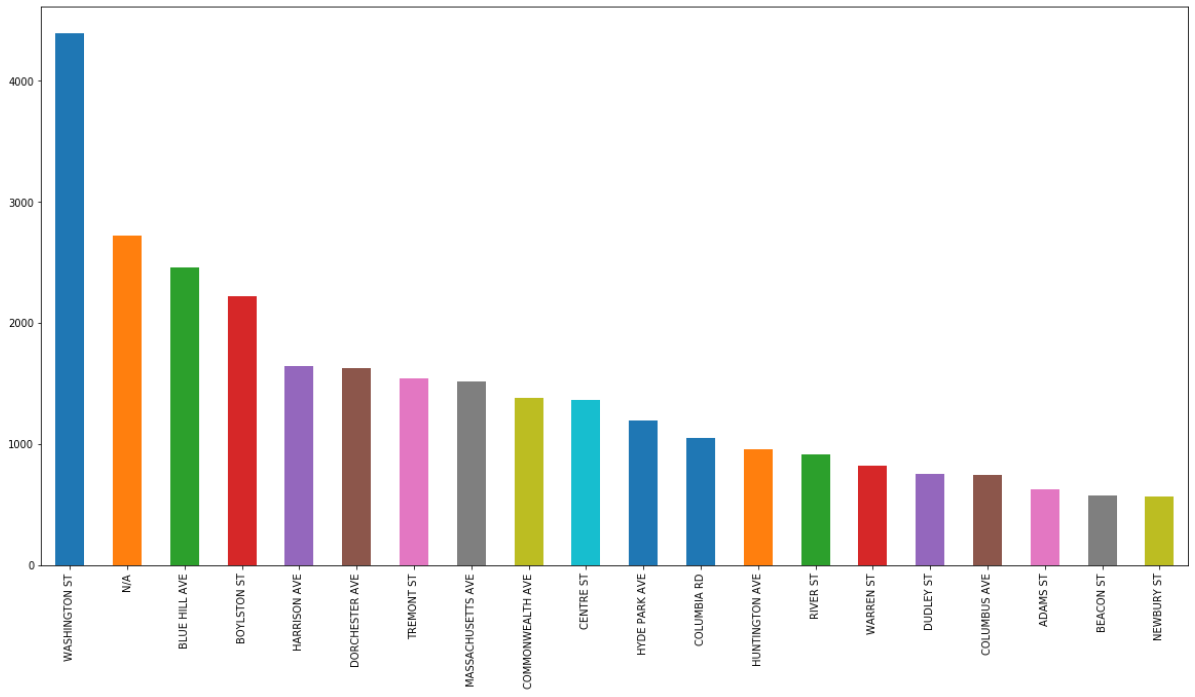
In [80]:
```
plt.figure(figsize=(20,10))
crime['Street'].loc[crime['Year']==2016].value_counts()[1:20].plot.bar()
plt.show()
```



Bar plot of crimes commited on the streets of Boston

c. in the year 2017:

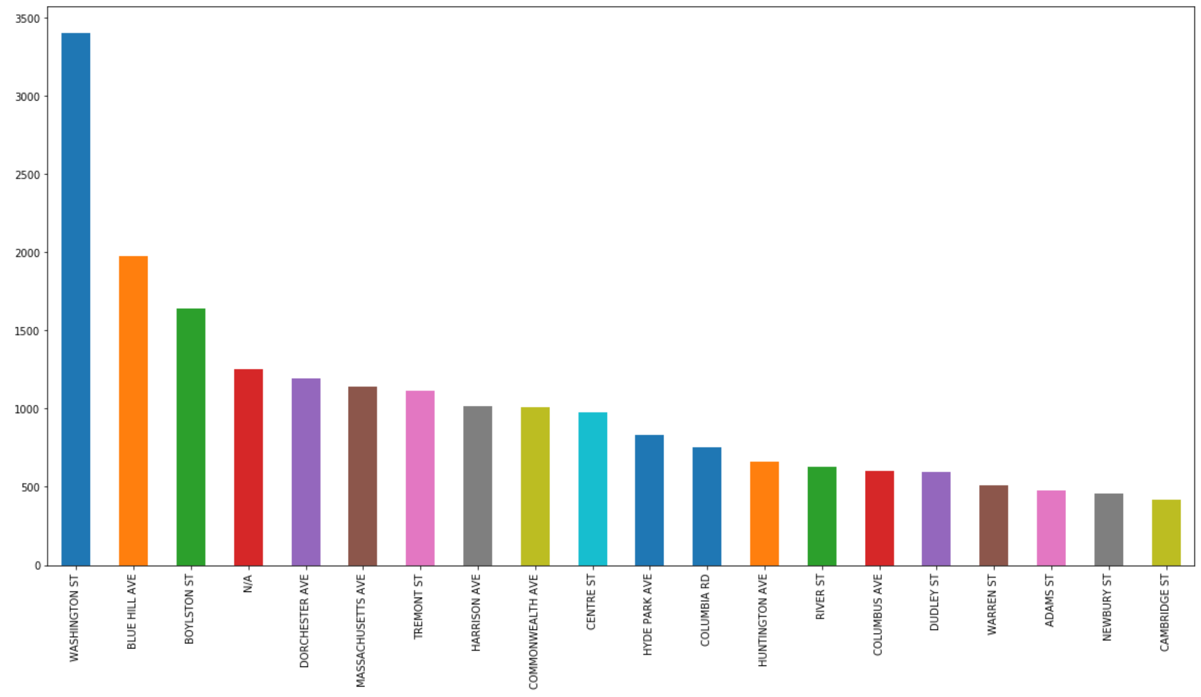
```
In [81]: plt.figure(figsize=(20,10))
         crime['Street'].loc[crime['Year']==2017].value_counts()[:20].plot.bar()
         plt.show()
```



Bar plot of crimes committed on the streets of Boston

d. in the year 2018:

In [82]:
```
plt.figure(figsize=(20,10))
crime['Street'].loc[crime['Year']==2018].value_counts()[:20].plot.bar()
plt.show()
```



After observing the bar plot of crimes committed on the streets of Boston in all the four years (2015-2018), it

can be observed that 'Washington St' is the street where maximum number of crimes has occured every year

and the Safest streest in Boston are "ADAMS ST" , "NEWBURY ST" , "CAMBRIDGE ST".

F. Bar plot of crimes occuring at Day or Night:

In [83]:
```
# Creating New Variable: Day
crime['Day'] = 0
```

In [84]:
```
# Creating New Variable: Night
crime['Night'] = 0
```

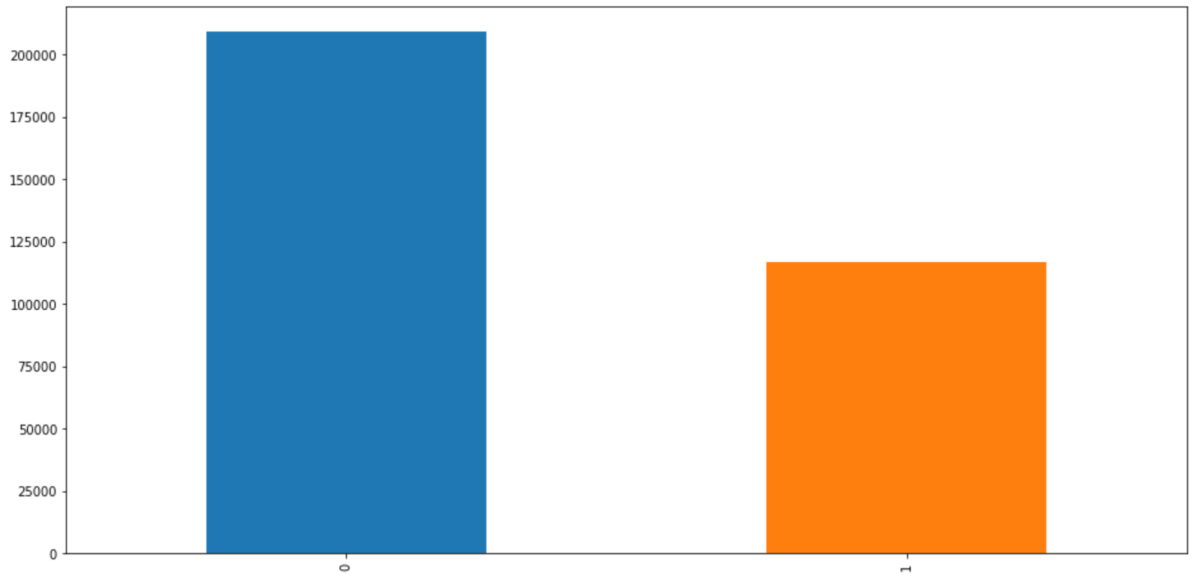
In [85]:
```
# Day or night for 1st month
crime['Day'].loc[(crime['Month'] == 1) & (crime['Hour'] >= 6) & (crime['Hour']
 <= 18)] = 1

# Day or night for 2nd month
crime['Day'].loc[(crime['Month'] == 2) & (crime['Hour'] >= 6) & (crime['Hour']
 <= 18)] = 1

# Day or night for 3rd month
crime['Day'].loc[(crime['Month'] == 3) & (crime['Hour'] >= 6) & (crime['Hour']
 <= 18)] = 1

# Day or night for 4th month
crime['Day'].loc[(crime['Month'] == 4) & (crime['Hour'] >= 6) & (crime['Hour']
 <= 18)] = 1

# Day or night for 5th month
crime['Day'].loc[(crime['Month'] == 5) & (crime['Hour'] >= 6) & (crime['Hour']
 <= 18)] = 1

# Day or night for 6th month
crime['Day'].loc[(crime['Month'] == 6) & (crime['Hour'] >= 6) & (crime['Hour']
 <= 18)] = 1

# Day or night for 7th month
crime['Day'].loc[(crime['Month'] == 7) & (crime['Hour'] >= 6) & (crime['Hour']
 <= 18)] = 1

# Day or night for 8th month
crime['Day'].loc[(crime['Month'] == 8) & (crime['Hour'] >= 6) & (crime['Hour']
 <= 18)] = 1

# Day or night for 9th month
crime['Day'].loc[(crime['Month'] == 9) & (crime['Hour'] >= 6) & (crime['Hour']
 <= 18)] = 1

# Day or night for 10th month
crime['Day'].loc[(crime['Month'] == 10) & (crime['Hour'] >= 6) & (crime['Hour'
] <= 18)] = 1

# Day or night for 11th month
crime['Day'].loc[(crime['Month'] == 11) & (crime['Hour'] >= 6) & (crime['Hour'
] <= 18)] = 1

# Day or night for 12th month
crime['Day'].loc[(crime['Month'] == 12) & (crime['Hour'] >= 6) & (crime['Hour'
] <= 18)] = 1
```

```
C:\Anaconda\lib\site-packages\pandas\core\indexing.py:189: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  self._setitem_with_indexer(indexer, value)
```

In [86]:
```
crime['Night'].loc[crime['Day']==0]=1
```

```
C:\Anaconda\lib\site-packages\pandas\core\indexing.py:189: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  self._setitem_with_indexer(indexer, value)
```

In [87]:
```
plt.figure(figsize=(16,8))
crime['Night'].value_counts().plot.bar()
plt.show()
```



In the above Bar Plot, '0' : 'Night' and '1' : represents 'Day'. We can interpret that majority of the Crimes in Boston occur at night.

Modeling of Dataset

Multi classification

In [ ]:
```
Y - OFFENSE_CODE_GROUP

X - 'DISTRICT','MONTH','DAY_OF_WEEK', 'HOUR','Lat','Long', 'OFFENSE_CODE_GROU
P','Day','Night'
```

```
In [88]: crime['Offense_Code_Group'].value_counts().head(15)
```

```
Out[88]: Motor Vehicle Accident Response    38106
         Larceny                            26552
         Medical Assistance                 24099
         Investigate Person                 19114
         Other                              18542
         Drug Violation                     16813
         Simple Assault                     16239
         Vandalism                          15692
         Verbal Disputes                    13478
         Towed                              11591
         Investigate Property               11331
         Larceny From Motor Vehicle         11073
         Property Lost                      10067
         Warrant Arrests                     8408
         Aggravated Assault                  8000
         Name: Offense_Code_Group, dtype: int64
```

```
In [89]: list_offense_code_group = ('Motor Vehicle Accident Response',
                                     'Larceny',
                                     'Medical Assistance',
                                     'Investigate Person',
                                     'Other',
                                     'Drug Violation',
                                     'Simple Assault',
                                     'Vandalism',
                                     'Verbal Disputes',
                                     'Towed',
                                     'Investigate Property',
                                     'Larceny From Motor Vehicle'
                                     'Property Lost'
                                     'Warrant Arrests'
                                     'Aggravated Assault'
                                    )
```

```
In [90]: crime_model = pd.DataFrame()
```

```
In [91]: i = 0

         while i < len(list_offense_code_group):

             crime_model = crime_model.append(crime.loc[crime['Offense_Code_Group'] ==
         list_offense_code_group[i]])

             i+=1
```

```
In [92]: list_column = ['District','Month','Day_of_Week',
                        'Hour','Lat','Long', 'Offense_Code_Group','Day','Night']
```

```
In [93]: crime_model = crime_model[list_column]
```

In [94]:
```
# DISTRICT

crime_model['District'] = crime_model['District'].map({
    'B3':1,
    'E18':2,
    'B2':3,
    'E5':4,
    'C6':5,
    'D14':6,
    'E13':7,
    'C11':8,
    'D4':9,
    'A7':10,
    'A1':11,
    'A15':12
})

crime_model['District'].unique()
```

Out[94]:
```
array([nan,  6.,  5.,  8.,  3.,  2., 11., 10.,  1.,  7.,  9.,  4., 12.])
```

In [95]:
```
# MONTH

crime_model['Month'].unique()
```

Out[95]:
```
array([10,  9,  8,  5,  7,  6,  3,  4, 11, 12,  1,  2], dtype=int64)
```

In [96]:
```
# DAY_OF_WEEK

crime_model['Day_of_Week'] = crime_model['Day_of_Week'].map({
    'Tuesday':2,
    'Saturday':6,
    'Monday':1,
    'Sunday':7,
    'Thursday':4,
    'Wednesday':3,
    'Friday':5
})

crime_model['Day_of_Week'].unique()
```

Out[96]:
```
array([3, 2, 1, 7, 4, 5, 6], dtype=int64)
```

In [97]:
```
# HOUR

crime_model['Hour'].unique()
```

Out[97]:
```
array([20, 19, 15, 16, 14,  9, 17, 11, 22,  8,  7,  0, 23, 21, 10, 18, 12,
        2,  6, 13,  5,  4,  3,  1], dtype=int64)
```

In [98]:
```
# Lat, Long

crime_model[['Lat', 'Long']].head()
```

Out[98]:

	Lat	Long
INCIDENT_NUMBER		
I182080048	42.3207	-71.0568
I182080043	42.3443	-71.1578
I182080038	42.316	-71.0904
I182080030	42.33	-71.0385
I182079979	42.3109	-71.0577

In [99]:
```
crime_model.fillna(0, inplace = True)
```

In [100]:
```
x = crime_model[['District','Month','Day_of_Week','Hour','Lat','Long','Day','N
ight']]
```

In [101]:
```
y = crime_model['Offense_Code_Group']
```

In [102]:
```
y.unique()
```

Out[102]:
```
array(['Motor Vehicle Accident Response', 'Larceny', 'Medical Assistance',
       'Investigate Person', 'Other', 'Drug Violation', 'Simple Assault',
       'Vandalism', 'Verbal Disputes', 'Towed', 'Investigate Property'],
      dtype=object)
```

In [103]:
```
y = y.map({
    'Motor Vehicle Accident Response':1,
    'Larceny':2,
    'Medical Assistance':3,
    'Investigate Person':4,
    'Other':5,
    'Drug Violation':6,
    'Simple Assault':7,
    'Vandalism':8,
    'Verbal Disputes':9,
    'Towed':10,
    'Investigate Property':11,
    'Larceny From Motor Vehicle':12
})
```

Split data into Training set and Test set for further Model Prediction.

In [104]:
```
# Split dataframe into random train and test subsets
from sklearn.cross_validation import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(
    x,
    y,
    test_size = 0.1,
    random_state=42
)

print(X_train.shape, Y_train.shape)
print(X_test.shape, Y_test.shape)
```

```
(190401, 8) (190401,)
(21156, 8) (21156,)
```