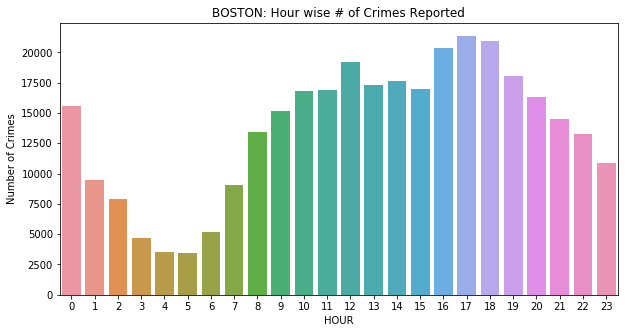
**1.1.1 What is Data Visualization?**

**Data visualization** is the graphic representation of data. It involves producing images that communicate relationships among the represented data to viewers of the images. This communication is achieved through the use of a systematic mapping between graphic marks and data values in the creation of the visualization. This mapping establishes how data values will be represented visually, determining how and to what extent a property of a graphic mark, such as size or color, will change to reflect change in the value of a datum.

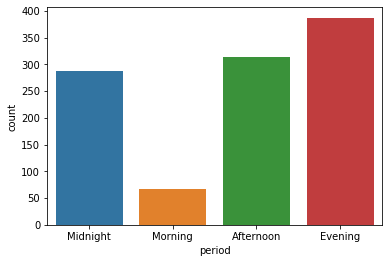
****

**The Figure above shows the total crimes in each period of hour.**

**So, this is data Visualization.**

* + 1. **What is Data Analysis?**

**Data analysis** is a process of inspecting, cleansing, transforming and modelling data with the goal of discovering useful information, informing conclusion and supporting decision-making. Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names, and is used in different business, science, and social science domains. In today’s business word, data analysis plays a role in making decisions more scientific and helping businesses operate more effectively.

**1.1.2 What is Data Analysis?**

**This figure Shows the total number of cases occurred at a point**

**of time in a day but, if we analyse the graph then we conclude that**

**most cases are found during Evening and least cases at Morning.**

**1.2 Why PYTHON?**

**Python** is the preferred programming language for **data scientists**. They need an easy-to-use language that has decent library availability and great community participation. Projects that have inactive communities are usually less likely to maintain or update their platforms, which is not the case with Python.

Python has long been known as a simple programming language to pick up, from a syntax point of view, anyway. Python also has an active community with a vast selection of libraries and resources. The result? You have a programming platform that makes sense to use with emerging technologies like **machine learning** and **data science**.

Professionals working with data science applications don’t want to be bogged down with complicated programming requirements. They want to use programming languages like Python and Ruby to perform tasks hassle-free.

**Ruby** is excellent for performing tasks such as data cleaning and **data munging**, along with other data pre-processing tasks. However, it doesn’t feature as many machine learning libraries as Python. This gives Python the edge when it comes to data science and machine learning

Python also enables developers to roll out programs and get prototypes running, making the development process much faster. Once a project is on its way to becoming an analytical tool or application, it can be ported to more **sophisticated languages** such as Java or C if necessary.

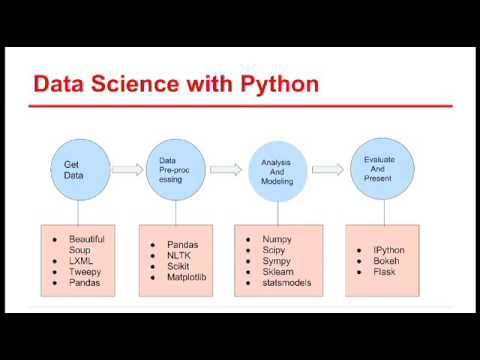
Newer data scientists gravitate toward Python because of its ease of use, which makes it accessible. So popular in fact, a staggering 48 percent of data scientists

1.2 why PYTHON?

With five or fewer years’ experience rated Python their preferred programming language.

This number tapers off as the experience level increases and the analytics become more intensive. Python has proven itself to be an excellent starting point for data scientists.

Data science involves extrapolating useful information from massive stores of statistics, registers, and data. These data are usually unsorted and difficult to correlate with any meaningful accuracy. Machine learning can make connections between disparate datasets but requires serious computational sophistry and power.

Python fills this need by being a general-purpose programming language. It allows you to create **CSV** output for easy data reading in a spreadsheet. Alternatively, more complicated file outputs that can be ingested by machine learning **clusters for computation.**

**This figure shows that how well-suited libraries are designed**

**for the process of Data science and Machine Learning.**

**Chapter 2**

**Python Libraries for Data Science**

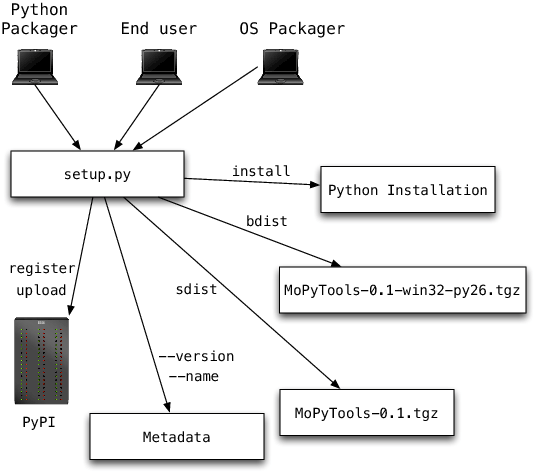
**2.1 Python Libraries**

While The Python Language Reference describes the exact syntax and semantics of the Python language, this library reference manual describes the standard library that is distributed with Python. It also describes some of the optional components that are commonly included in Python distributions.

Python’s standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs.

The Python installers for the Windows platform usually include the entire standard library and often also include many additional components. For Unix-like operating systems Python is normally provided as a collection of packages, so it may be necessary to use the packaging tools provided with the operating system to obtain some or all of the optional components.

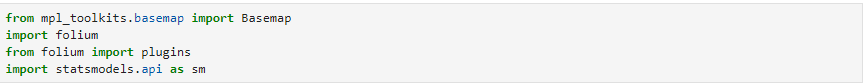
In addition to the standard library, there is a growing collection of several thousand components (from individual programs and modules to packages and entire application development frameworks), available from the Python Package Index.

****

**Working of Python Libraries**

**2.1.2 Libraries use in My Project**





**Press Shift+Enter to execute the *jupyter lab* cell.**

**2.2 Python Libraries use**

**2.2.1 NumPy**

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

* a powerful N-dimensional array object
* sophisticated (broadcasting) functions
* tools for integrating C/C++ and Fortran code
* useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

NumPy is licensed under the BSD license, enabling reuse with few restrictions**.**

**2.2.2 Pandas**

**Pandas** is a Python package providing fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, **real world** data analysis in Python. Additionally, it has the broader goal of becoming **the most powerful and flexible open source data analysis / manipulation tool available in any language**. It is already well on its way toward this goal.

pandas is well suited for many different kinds of data:

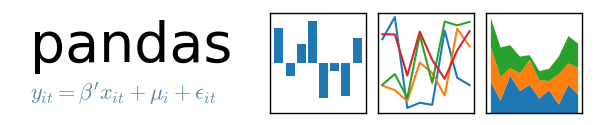
* Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
* Ordered and unordered (not necessarily fixed-frequency) time series data.
* Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels
* Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a pandas data structure

The two primary data structures of pandas, Series (1-dimensional) and DataFrame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. For R users, DataFrame provides everything that R’s data.frame provides and much more. pandas is built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.

Here are just a few of the things that pandas does well:

* Easy handling of **missing data** (represented as NaN) in floating point as well as non-floating point data
* Size mutability: columns can be **inserted and deleted** from DataFrame and higher dimensional objects
* Automatic and explicit **data alignment**: objects can be explicitly aligned to a set of labels, or the user can simply ignore the labels and let Series, DataFrame, etc. automatically align the data for you in computations
* Powerful, flexible **group by** functionality to perform split-apply-combine operations on data sets, for both aggregating and transforming data
* Make it **easy to convert** ragged, differently-indexed data in other Python and NumPy data structures into DataFrame objects
* Intelligent label-based **slicing**, **fancy indexing**, and **subsetting** of large data sets
* Intuitive **merging** and **joining** data sets
* Flexible **reshaping** and pivoting of data sets
* **Hierarchical** labeling of axes (possible to have multiple labels per tick)
* Robust IO tools for loading data from **flat files** (CSV and delimited), Excel files, databases, and saving / loading data from the ultrafast **HDF5 format**
* **Time series**-specific functionality: date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging, etc.

Many of these principles are here to address the shortcomings frequently experienced using other languages / scientific research environments. For data scientists, working with data is typically divided into multiple stages: munging and cleaning data, analyzing / modeling it, then organizing the results of the analysis into a form suitable for plotting or tabular display. pandas is the ideal tool for all of these tasks.

****

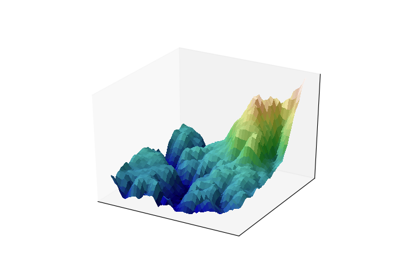
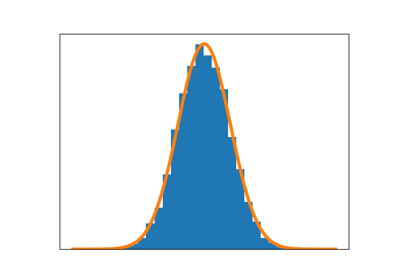
**2.3 Visualization libraries:**

**2.3.1 Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

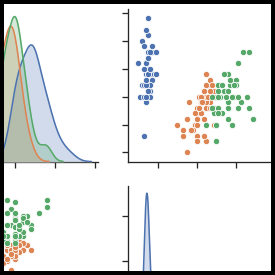
[](https://matplotlib.org/tutorials/introductory/sample_plots.html)**[](https://matplotlib.org/tutorials/introductory/sample_plots.html)**

**2.3.2 Seaborn**

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

For a brief introduction to the ideas behind the library, you can read the introductory notes. Visit the installation page to see how you can download the package. You can browse the example gallery to see what you can do with seaborn, and then check out the tutorial and API reference to find out how.

To see the code or report a bug, please visit the github repository. General support issues are most at home on stackoverflow, where there is a seaborn tags.

****

**2.3.3 Folium**

folium builds on the data wrangling strengths of the Python ecosystem and the mapping strengths of the leaflet.js library. Manipulate your data in Python, then visualize it in on a Leaflet map via folium.

* Concepts

folium makes it easy to visualize data that’s been manipulated in Python on an interactive leaflet map. It enables both the binding of data to a map for choropleth visualizations as well as passing rich vector/raster/HTML visualizations as markers on the map.

The library has a number of built-in tilesets from OpenStreetMap, Mapbox, and Stamen, and supports custom tilesets with Mapbox or Cloudmade API keys. folium supports both Image, Video, GeoJSON and TopoJSON overlays.

**2.3.4 Basemap**

The matplotlib basemap toolkit is a library for plotting 2D data on maps in Python. It is similar in functionality to the matlab mapping toolbox, the IDL mapping facilities, GrADS, or the Generic Mapping Tools. PyNGL and CDAT are other libraries that provide similar capabilities in Python.

Basemap does not do any plotting on it’s own, but provides the facilities to transform coordinates to one of 25 different map projections (using the PROJ.4 C library). Matplotlib is then used to plot contours, images, vectors, lines or points in the transformed coordinates. Shoreline, river and political boundary datasets (from Generic Mapping Tools) are provided, along with methods for plotting them. The GEOS library is used internally to clip the coastline and polticial boundary features to the desired map projection region.

Basemap is geared toward the needs of earth scientists, particularly oceanographers and meteorologists. Jeff Whitaker originally wrote Basemap to help in his research (climate and weather forecasting), since at the time CDAT was the only other tool in python for plotting data on map projections. Over the years, the capabilities of Basemap have evolved as scientists in other disciplines (such as biology, geology and geophysics) requested and contributed new features.

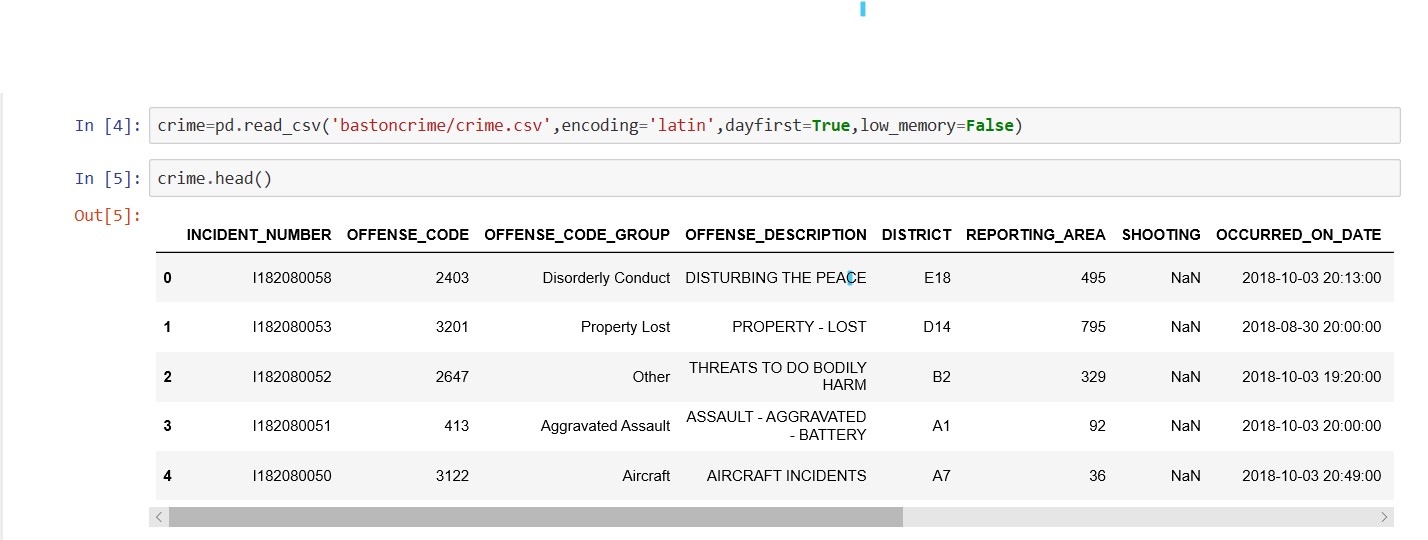
Data set link from kaggle.com

https://www.kaggle.com/datasets/sourinroy/boston-crime-dataset-updated-july-2020

**Chapter 3**

**Analysis $ Visualization**

**3.1 Reading data using Pandas**



In above script

* **‘pd.read\_csv’** – Using pandas library to reading data from crime.csv dataset (Which was download from Kaggle.com).
* **‘encoding’**- Function of pandas library to converting the crime.csv dataset encoding into ‘latin’ from it any original encoding of the dataset.
* **‘dayfirs**t’ – use to read the day or time data first(When its value is set to True).
* **‘low-memory’** – It is use to define that the data or memory of dataset is large

(When its value is set to True).

**3.1.1 Different ways to read data using Pandas**

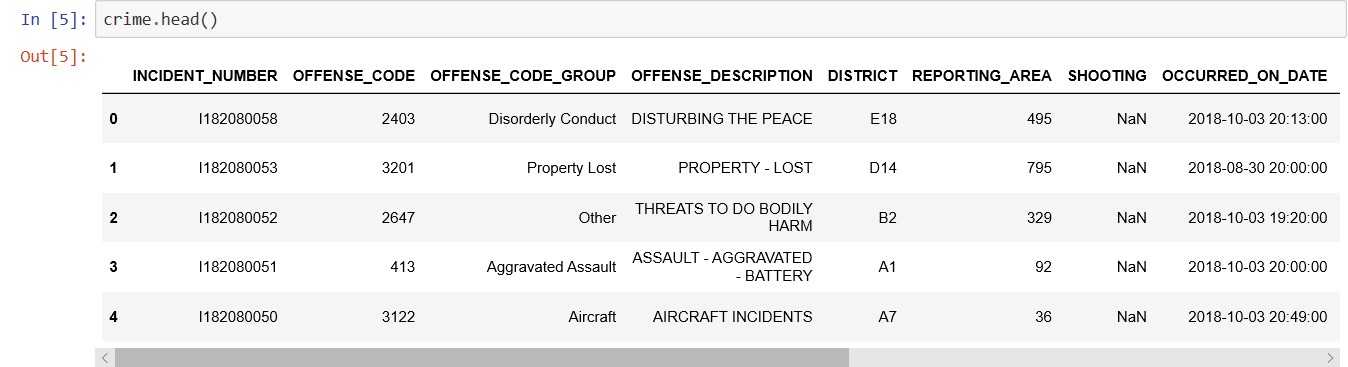
There is a number of pandas commands to read other data formats:

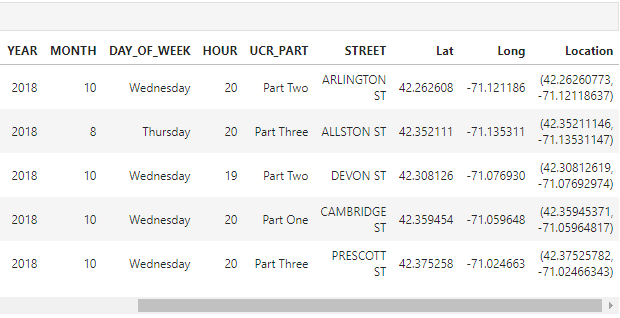
pd.read\_excel('myfile.xlsx',sheet\_name='Sheet1', index\_col=None, na\_values=['NA']).

pd.read\_stata('myfile.dta').

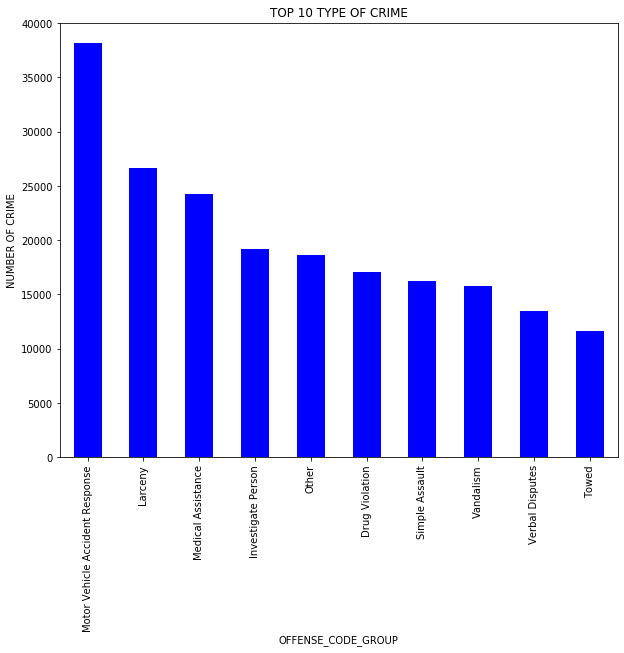
pd.read\_sas('myfile.sas7bdat').

pd.read\_hdf('myfile.h5','df').

**3.2 Sample of Data Set**

** Sample of dataset**

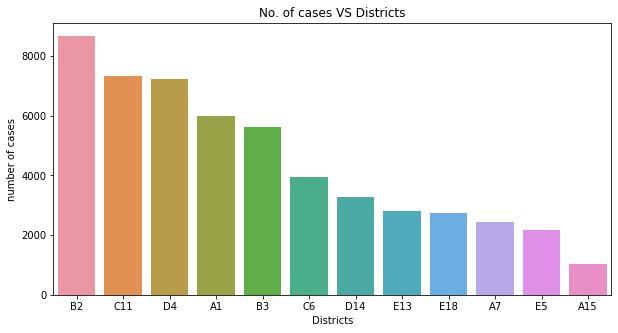
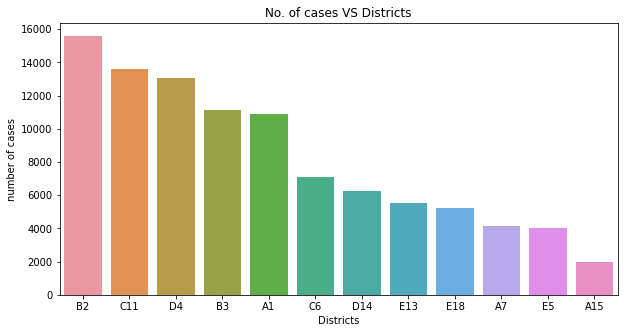
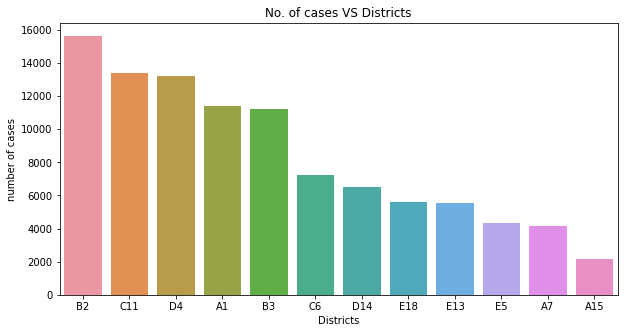
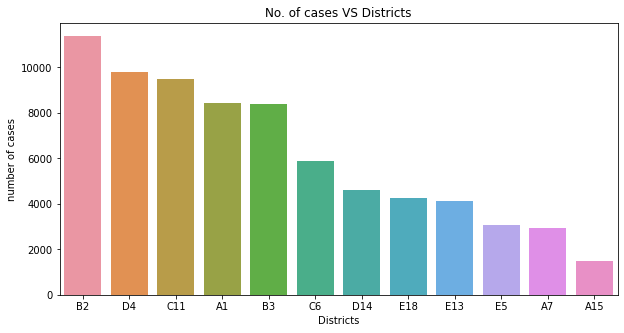
**3.2 Visualizations**

* **Top 10 crimes in Boston**

**Analysis:** From above graph we can see that most of the cases are from **Motor Vehicle Accident Response.**

**Code: **

* **Districts with their cases in each year.**

****

2018

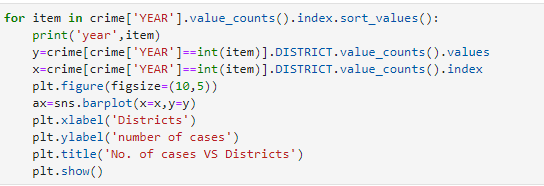
2017

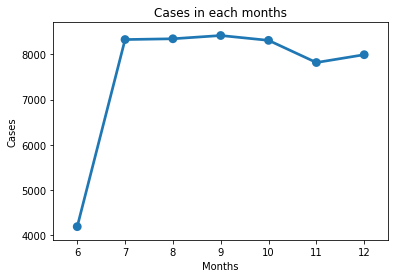
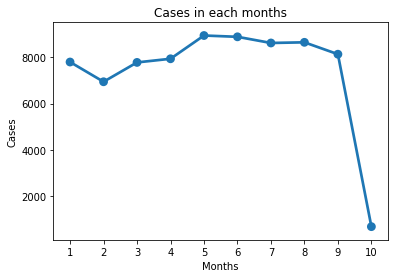
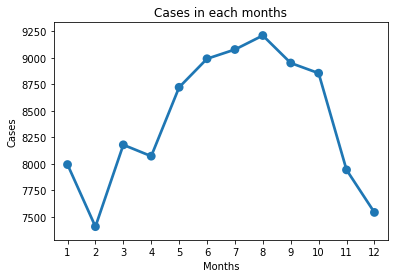
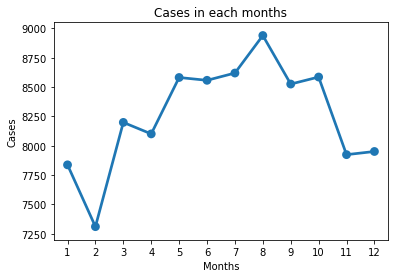
2016

2015

**Analysis:** From above graph we can see that most of the cases are from **District B2** in every year.

**Code:**

****

* ******Months with their cases in each year.**

2015

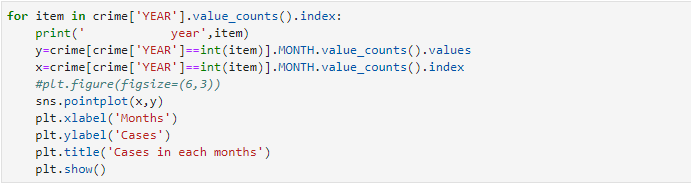
2018

2016

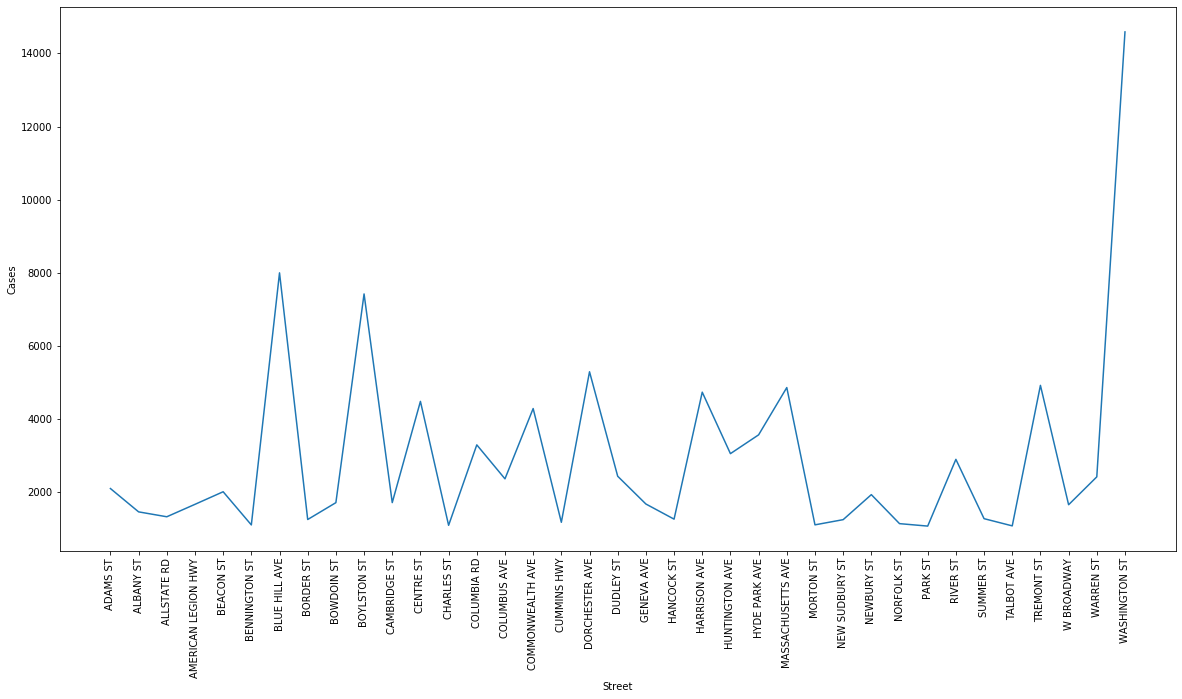
2017

**Analysis:** From above graph we can analyse that most of the cases in Year 2017 and 2016 are from **August,** but in 2018 and 2015 most cases are from May and **September.**

**Code:**

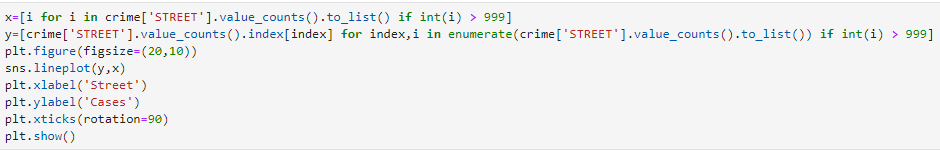
****

* **Streets with their cases.**

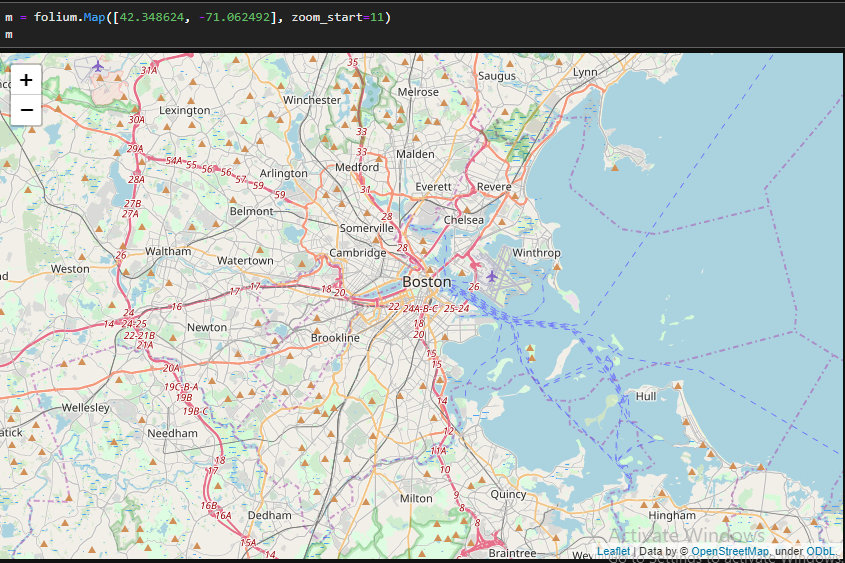


**Analysis:** From above graph we can analyse that most of the cases are from Street **WASHINGTON ST.**

**Code:**

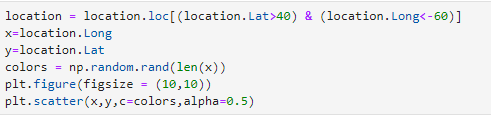
****

* **Streets with their cases.**

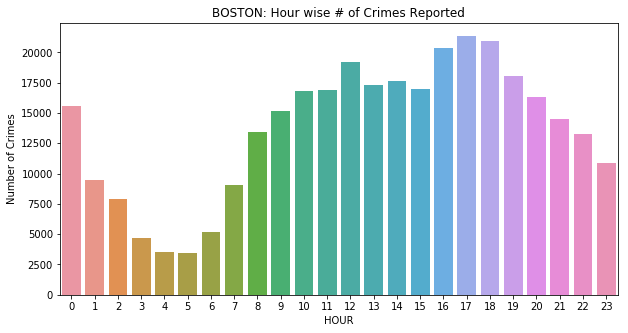
****

**Analysis:** From above graph we can analyse the Boston By Latitude and Longitude and also by seeing the real map.

**Code:**

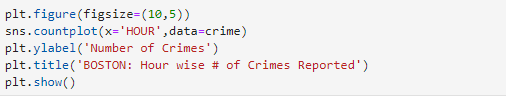
****

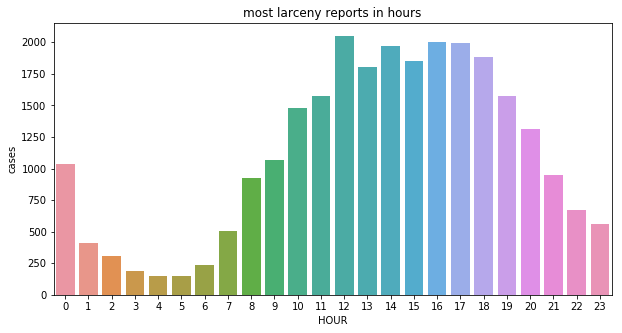
* **Hours with their cases.**



**Analysis:** From above graph we can analyse that most of the crimes Occurred in 18 O’clock.

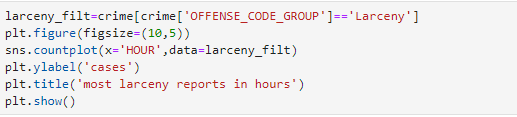
**Code:**

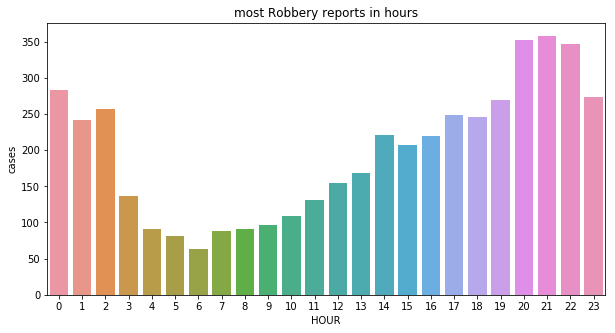


* **Most larceny cases.**

**Analysis:** From above graph we can analyse that During 12:00 O'clock Larceny crimes are most Executed.

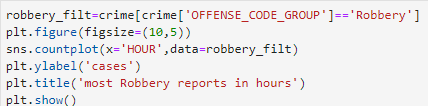
**Code:**

****

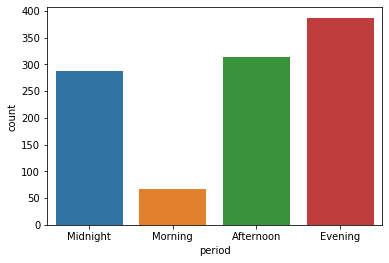
*  **Most Robbery cases.**

**Analysis:** From above graph we can analyse that from 20:00 to 22:00 O'clock Robbery crimes are mostly Executed.

**Code:**

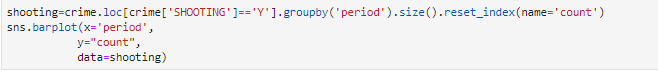
****

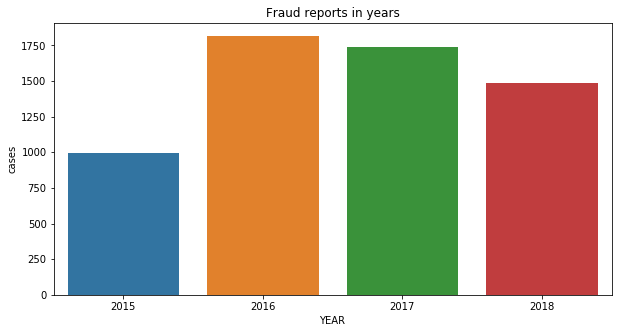
* **Occurrence of crimes in Morning, Afternoon, Evening, Midnight.**



**Analysis:** From above graph we can analyse that most Shootings cases occurs in Evening and then Afternoon and then Midnight and least at Morning.

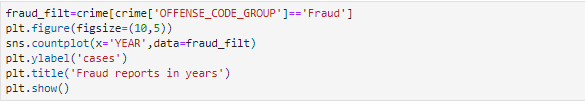
**Code:**

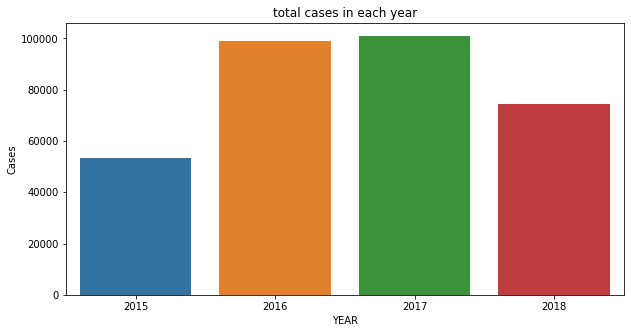
****

* **Cases of Fraud report.**

**Analysis:** From above graph we can analyse that Most Fraud cases occurs in 2016 and then 2017 and then 2018 and 2015 is the least.

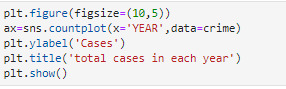
**Code:**

****

* **Total cases in each year.**

**Analysis:** From above graph we can analyse that most cases are from 2017 and least from 2015.

**Code:**



**Conclusion**

In this project graphical visualization on the crime data is done and found that the most of the cases that were reported for **Motor Vehicle Accident report.** It is also found that during **mid night** and **evening** the crime rate is increased in compare to other timings. At **Washington street** most crimes are committed and District **B2** is also the district with most crimes that are committed in 2015, 2016, 2017 and 2018.

**Bibliography**

Following are referring to create this project reports.

1. Google
2. OpenCV Tutorial
3. Python Tutorial
4. Anaconda3
5. Tutorial Point