**DATA PRE-PROCESSING AND VISUALIZATION**

**INTRODUCTION**

Exploratory Data Analysis or (EDA) is understanding the data sets by summarizing their main characteristics often plotting them visually. This step is very important especially when we arrive at modeling the data in order to apply Machine learning. Plotting in EDA consists of Histograms, Box plot, Scatter plot and many more. It often takes much time to explore the data. Through the process of EDA, we can ask to define the problem statement or definition on our data set which is very important**.**

**How to perform Exploratory Data Analysis ?**

This is one such question that everyone is keen on knowing the answer. Well, the answer is it depends on the data set that we are working. There is no one method or common methods in order to perform EDA, whereas in this tutorial you can understand some common methods and plots that would be used in the EDA process.

**What data are we exploring today ?**

Since I am a huge fan of cars, I got a very beautiful data-set of cars from Kaggle.To give a piece of brief information about the data set this data contains more of 10, 000 rows and more than 10 columns which contains features of the car such as Engine Fuel Type, Engine HP, Transmission Type, highway MPG, city MPG and many more. we will explore the data and make it ready for modeling.

**Why EDA is important?**

The main purpose of EDA is to detect any errors, outliers as well as to understand different patterns in the data. It allows Analysts to understand the data better before making any assumptions. The outcomes of EDA helps businesses to know their customers, expand their business and take decisions accordingly.

**TASKS**

1. Get maximum insights from a data set
2. Uncover underlying structure
3. Extract important variables from the dataset
4. Detect outliers and anomalies(if any)
5. Test underlying assumptions
6. Determine the optimal factor settings

**Packages need to import**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib as plt

**DATA CLEANING**

This is about using a combination of Python to perform data preproccesing and some exploratory statistical analysis on an untidy dataset. Blogs and articles for upcoming data scientists often harp on the need for using real-world data which is messy and the importance of learning data cleaning and preprocessing for this profession. However, we mostly find blogs and tutorials with standard datasets which skip the pre-processing step. Similarly, real-world data analysis may require a combination of EDA and statistical analysis. For that purpose, it may require that data scientists use both Python and R and switch between them depending upon the micro-task at hand.

I share some steps of data processing and exploratory analysis which came across as part of an experiment. I have introduced a toy dataset for this purpose but the structure and messyness of the data is similar to what I encountered. Imagine a retail company with five stores in different geographic locations. Each store has two billing counters. The company is trialing a new product which they have only put at the billing counter. The cashiers are supposed to pitch the item to the customers during billing. The cashiers also ask each customer three questions about the product and then ask if they want to buy that product. The questions are about three attributes of the product and the customers have to answer ‘Yes’ — if they like the attribute or ‘No’ — if they don’t like the attribute. Thus we have 4 columns in our dataset for each copy of the product — 3 for the attributes and 1 for recording whether customers end up buying the item or not. All values are categorical — “Yes” or “No”.

1**. Importing the required libraries for EDA**

Below are the libraries that are used in order to perform EDA (Exploratory

data analysis).



**2. Loading the data into the data frame.**

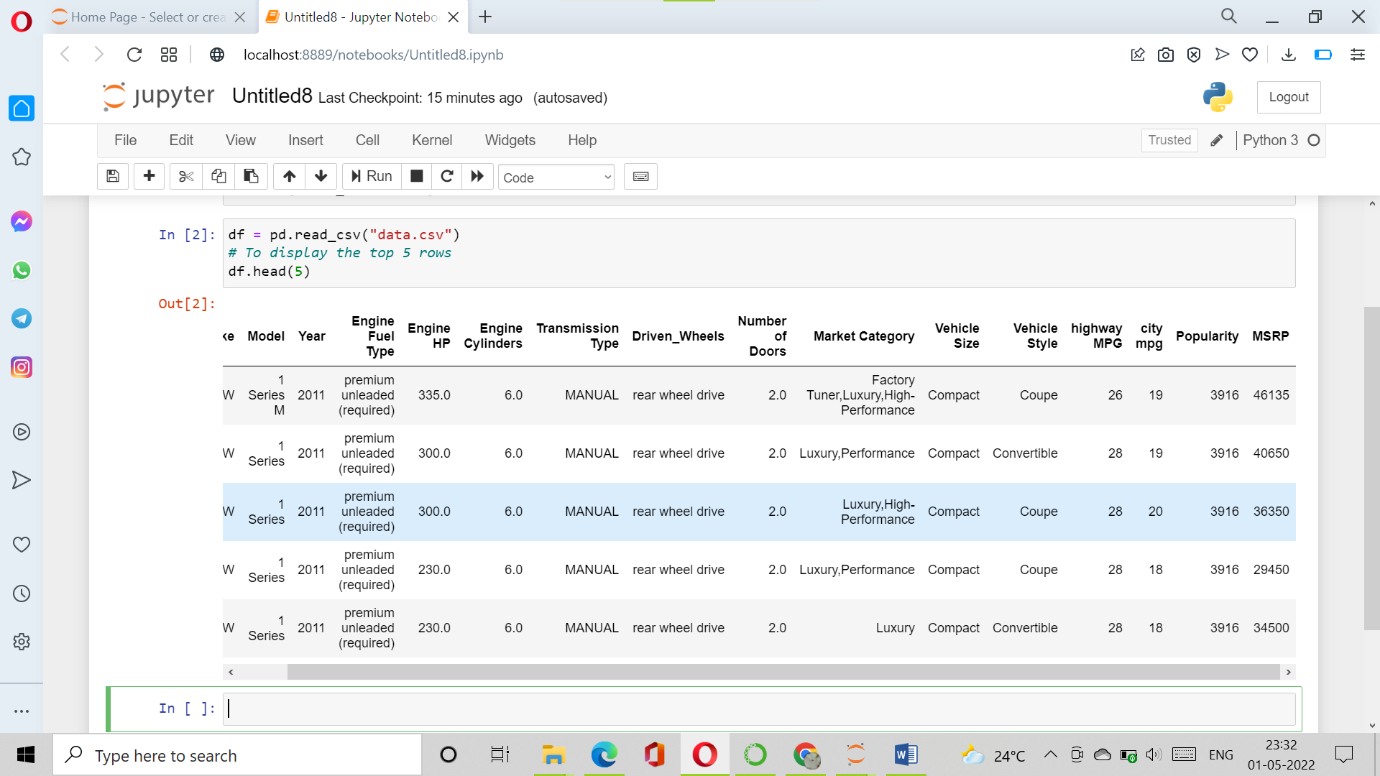
Loading the data into the pandas data frame is certainly one of the most important steps in EDA, as we can see that the value from the data set is comma-separated. So all we have to do is to just read the CSV into a data frame and pandas data frame does the job for us.

To get or load the dataset into the notebook, all I did was one trivial step. In Google Colab at the left-hand side of the notebook, you will find a > (greater than symbol). When you click that you will find a tab with three options, you just have to select Files. Then you can easily upload your file with the help of the Upload option. No need to mount to the google drive or use any specific libraries just upload the data set and your job is done. One thing to remember in this step is that uploaded files will get deleted when this runtime is recycled. This is how I got the data set into the notebook.

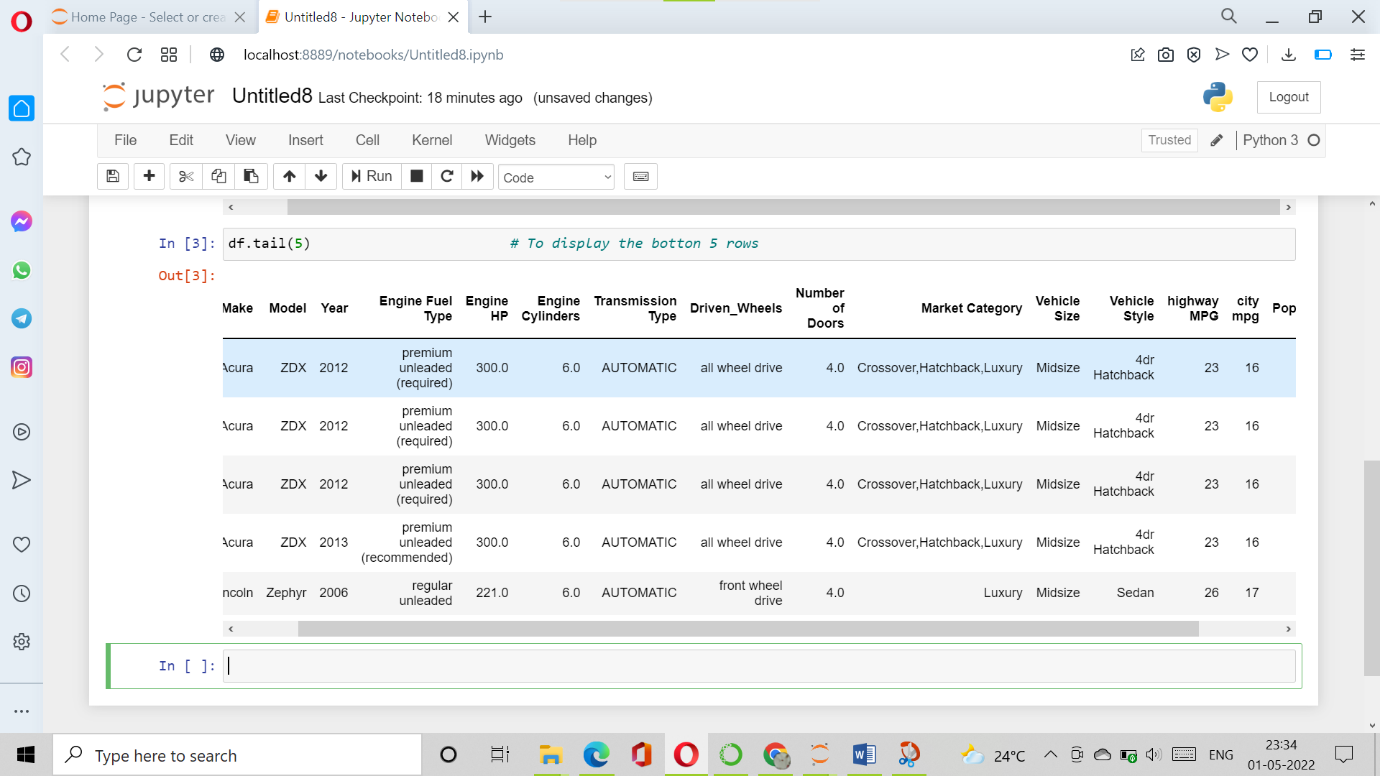
df = pd.read\_csv("data.csv")

*# To display the top 5 rows*

df.head(5)



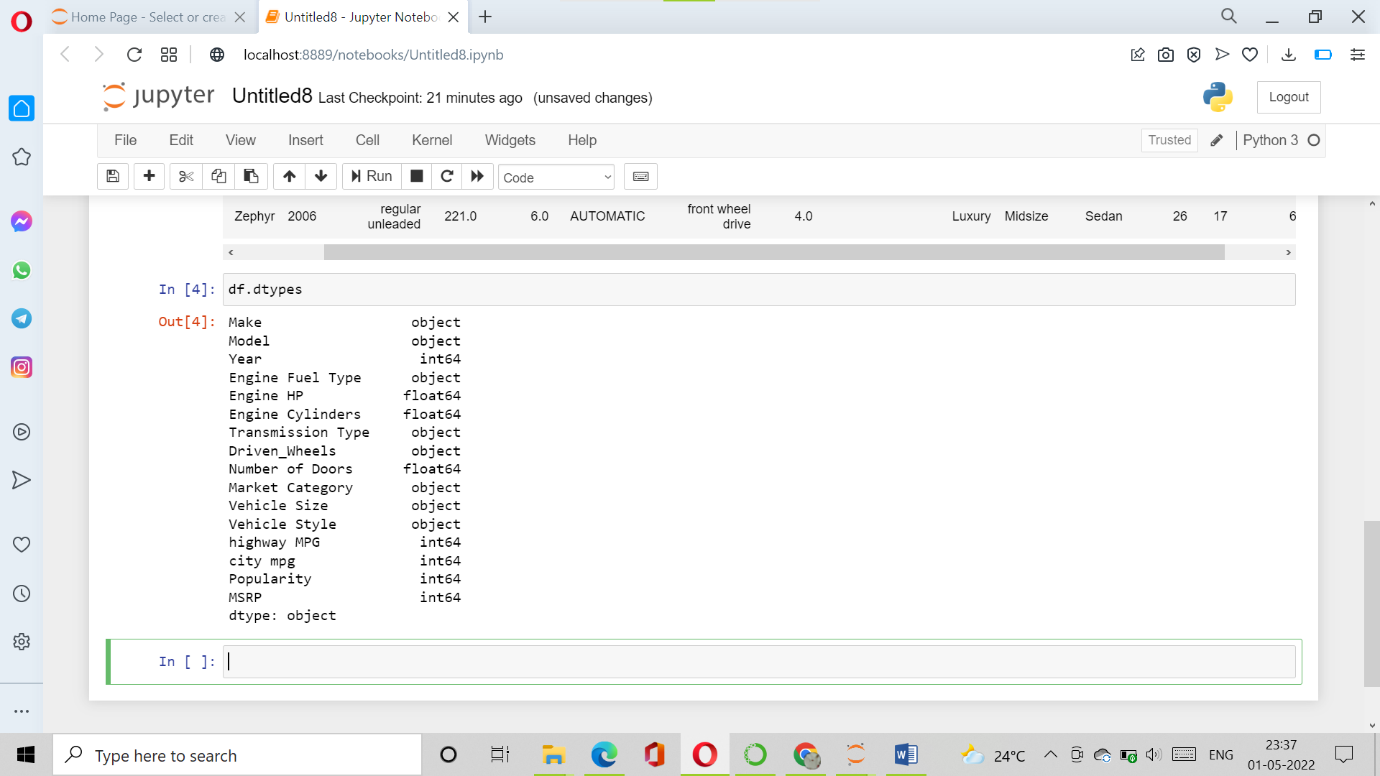
df.tail(5) *# To display the botton 5 rows*



## 3. Checking the types of data

Here we check for the datatypes because sometimes the MSRP or the price of the car would be stored as a string, if in that case, we have to convert that string to the integer data only then we can plot the data via a graph. Here, in this case, the data is already in integer format so nothing to worry.

df.dtypes

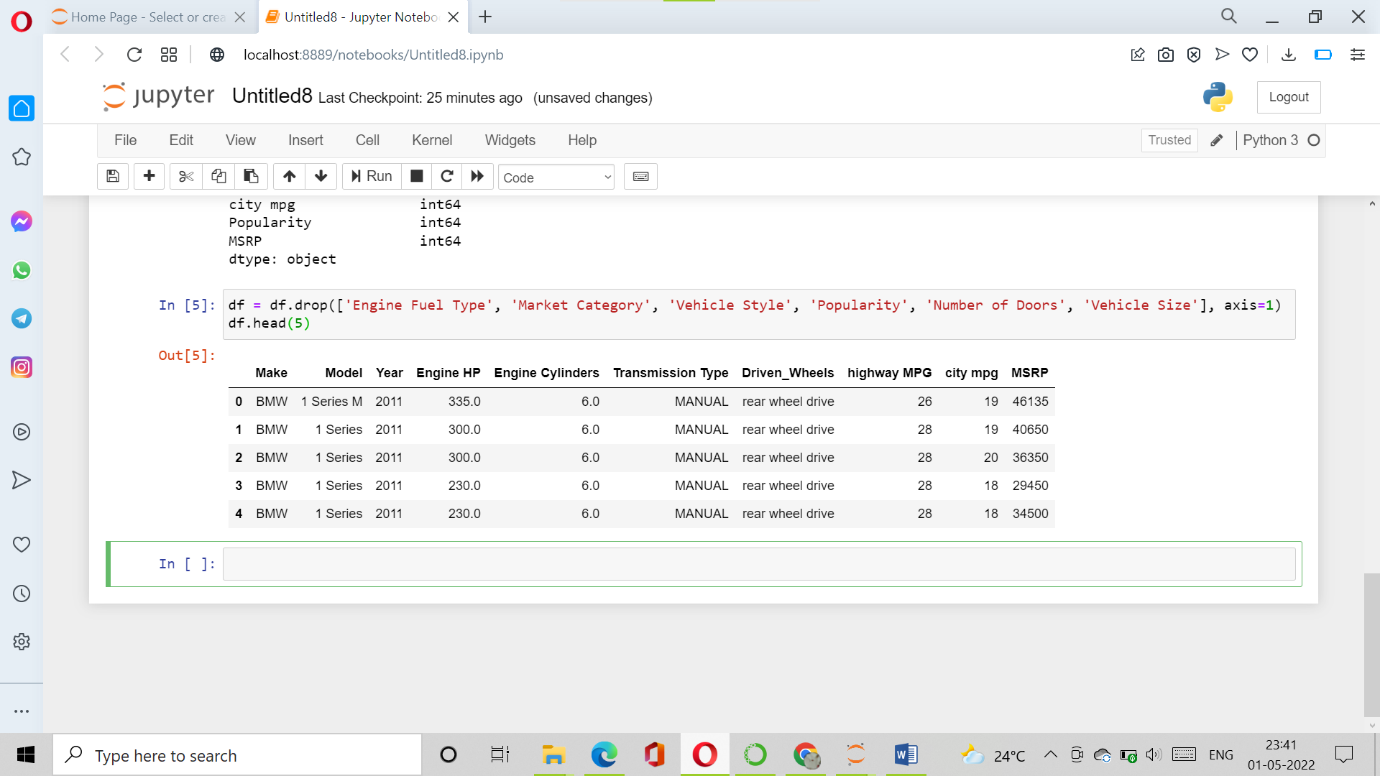


## 4. Dropping irrelevant columns

This step is certainly needed in every EDA because sometimes there would be many columns that we never use in such cases dropping is the only solution. In this case, the columns such as Engine Fuel Type, Market Category, Vehicle style, Popularity, Number of doors, Vehicle Size doesn't make any sense to me so I just dropped for this instance.

df = df.drop(['Engine Fuel Type', 'Market Category', 'Vehicle Style', 'Popularity', 'Number of Doors', 'Vehicle Size'], axis=1)

df.head(5)

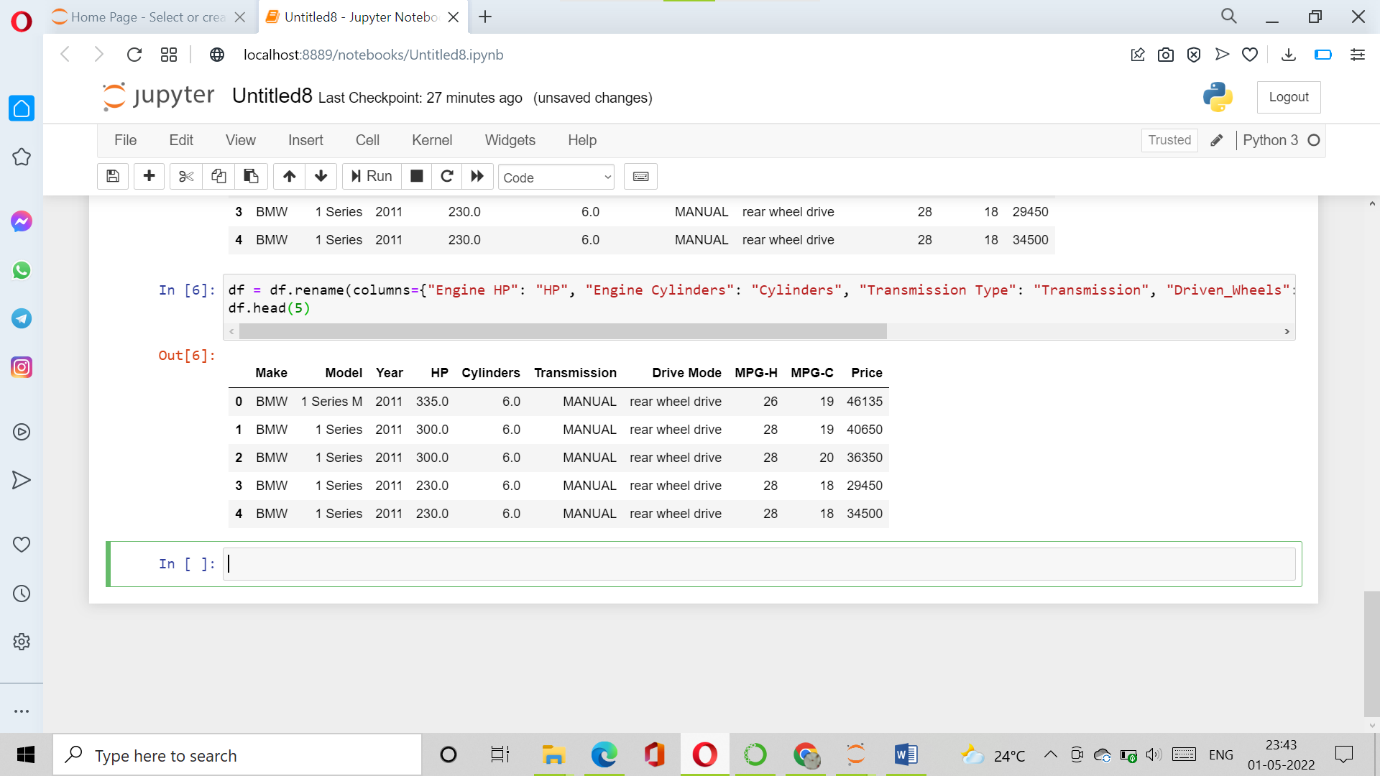


## 5. Renaming the columns

In this instance, most of the column names are very confusing to read, so I just tweaked their column names. This is a good approach it improves the readability of the data set.

df = df.rename(columns={"Engine HP": "HP", "Engine Cylinders": "Cylinders", "Transmission Type": "Transmission", "Driven\_Wheels": "Drive Mode","highway MPG": "MPG-H", "city mpg": "MPG-C", "MSRP": "Price" })

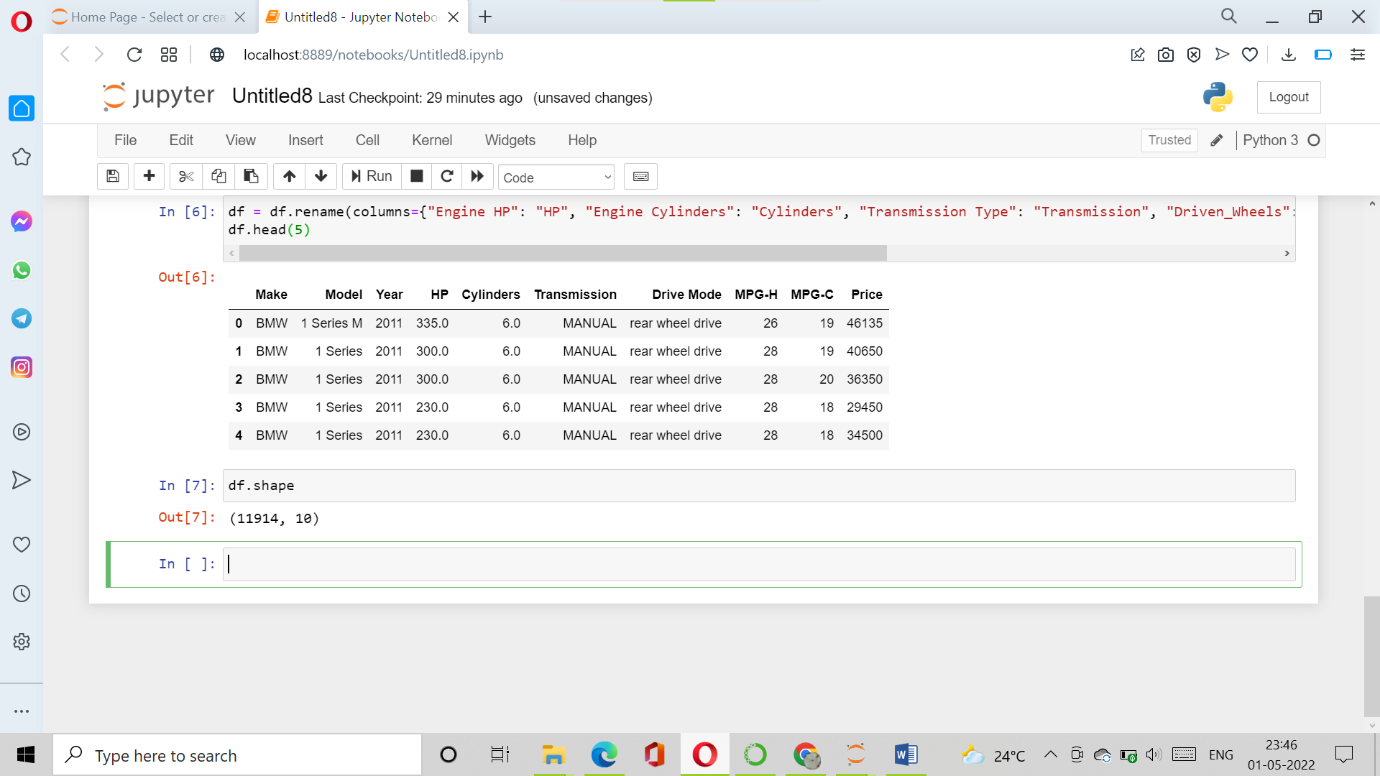
df.head(5)



## 6. Dropping the duplicate rows

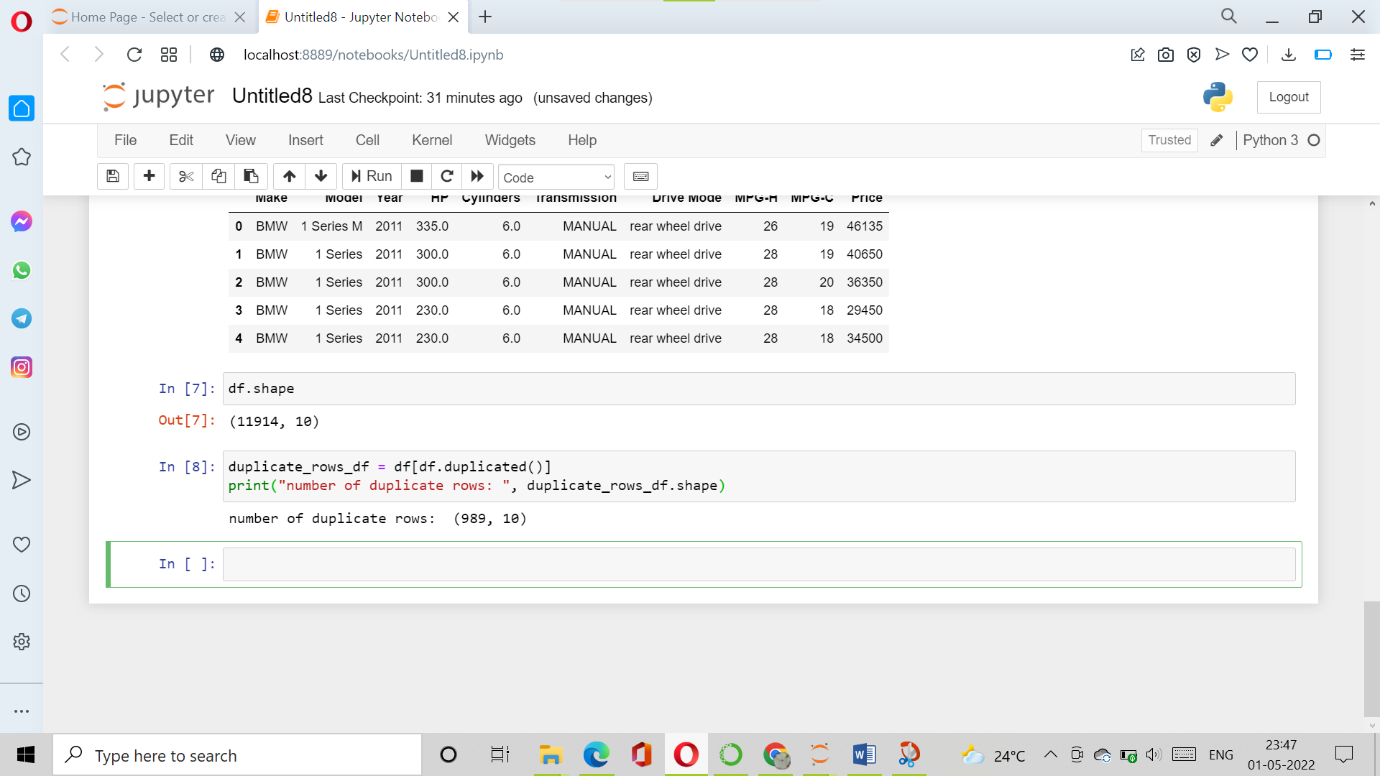
This is often a handy thing to do because a huge data set as in this case contains more than 10, 000 rows often have some duplicate data which might be disturbing, so here I remove all the duplicate value from the data-set. For example prior to removing I had 11914 rows of data but after removing the duplicates 10925 data meaning that I had 989 of duplicate data.

df.shape



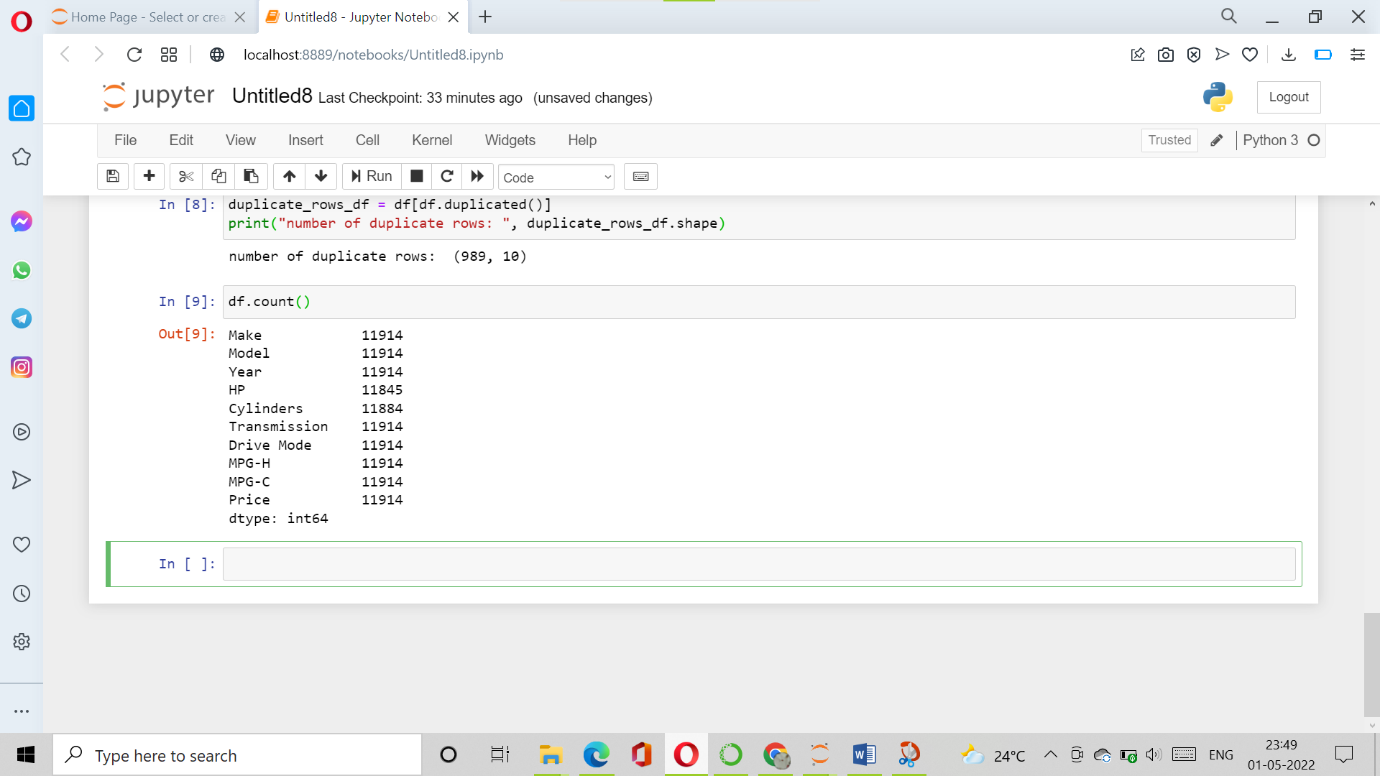
duplicate\_rows\_df = df[df.duplicated()]

print("number of duplicate rows: ", duplicate\_rows\_df.shape)



Now let us remove the duplicate data because it's ok to remove them.

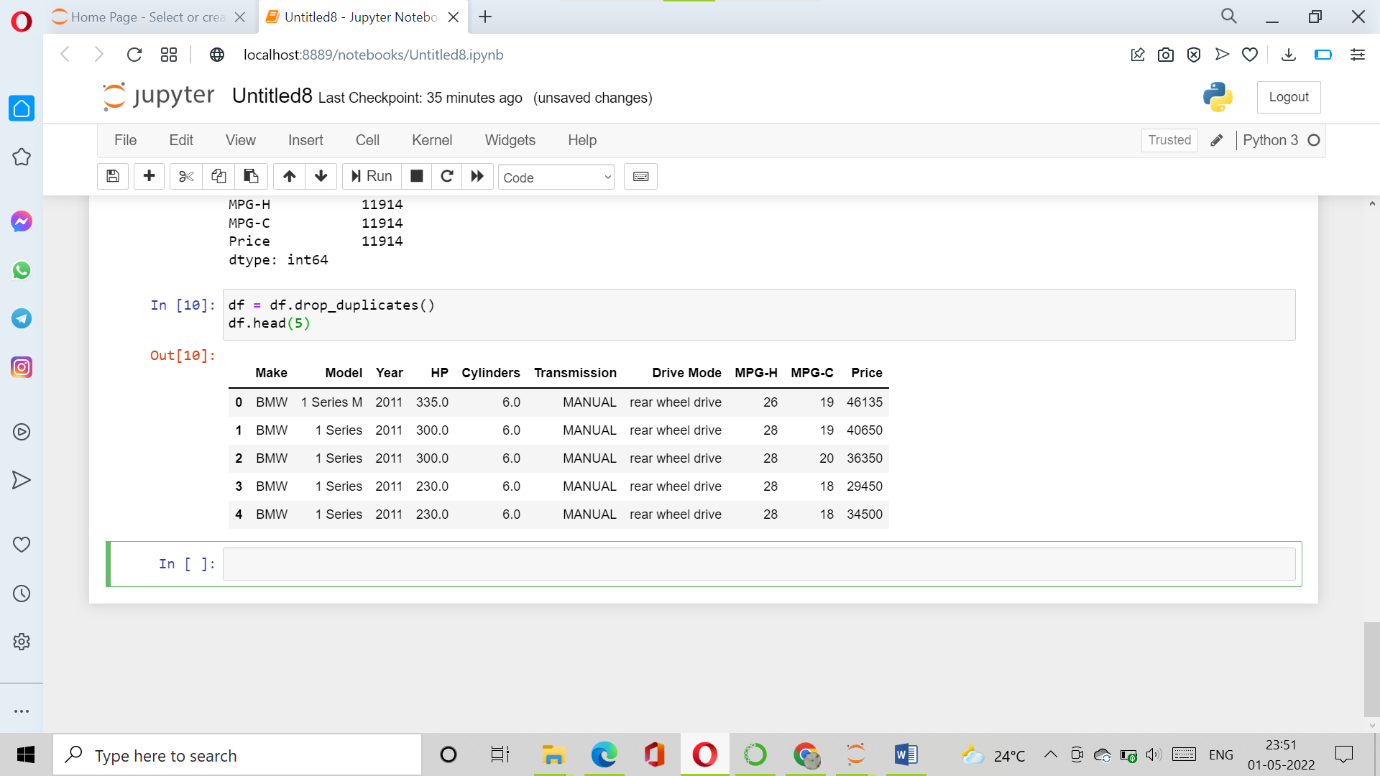
df.count()



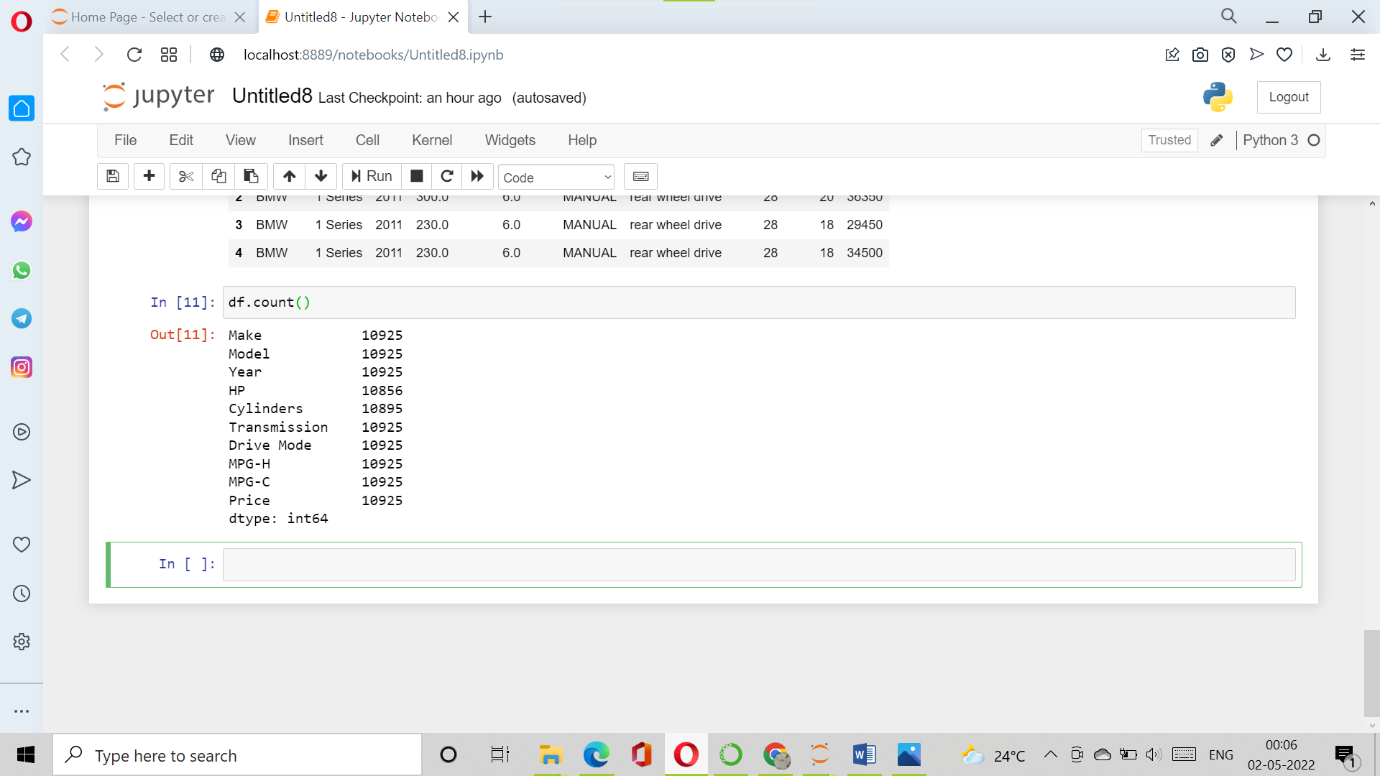
So seen above there are 11914 rows and we are removing 989 rows of duplicate data.

df = df.drop\_duplicates()

df.head(5)



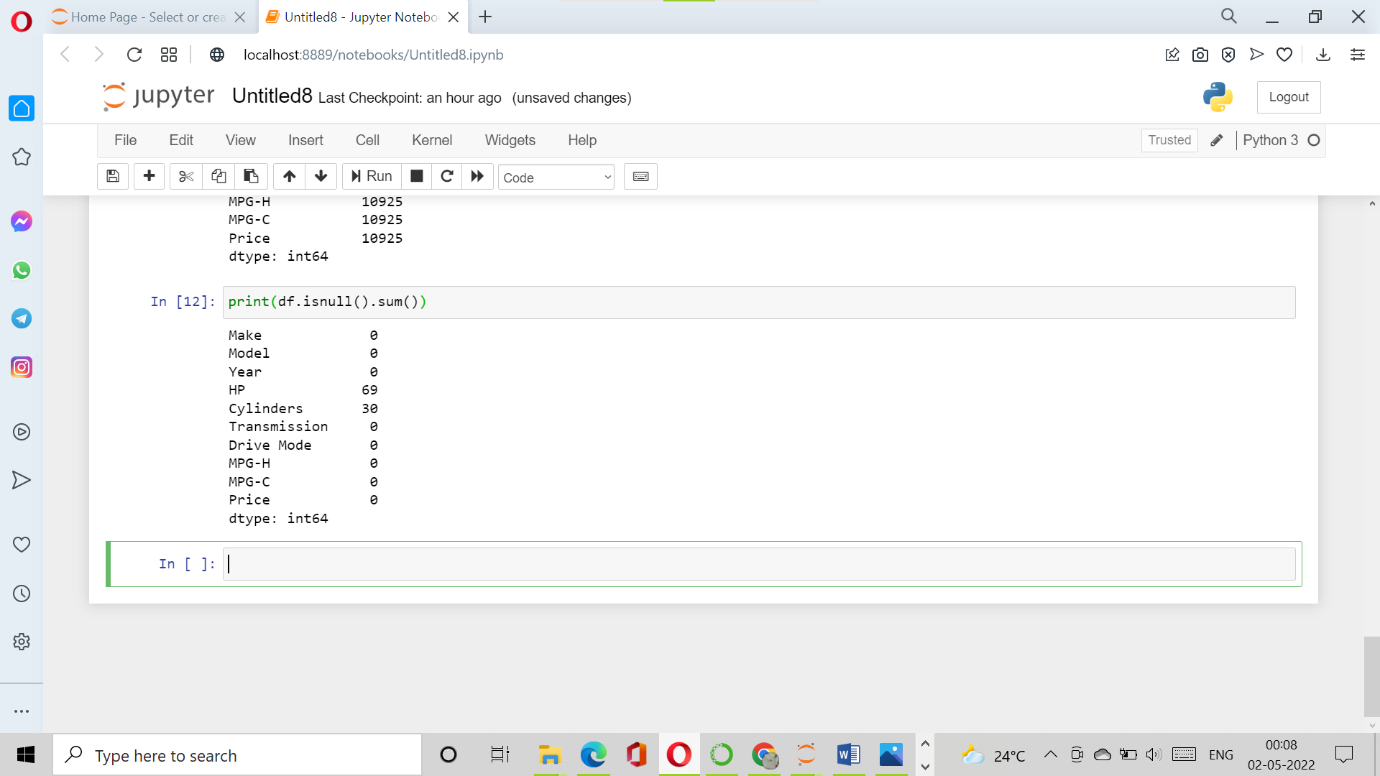
df.count()



## 7. Dropping the missing or null values.

This is mostly similar to the previous step but in here all the missing values are detected and are dropped later. Now, this is not a good approach to do so, because many people just replace the missing values with the mean or the average of that column, but in this case, I just dropped that missing values. This is because there is nearly 100 missing value compared to 10, 000 values this is a small number and this is negligible so I just dropped those values.

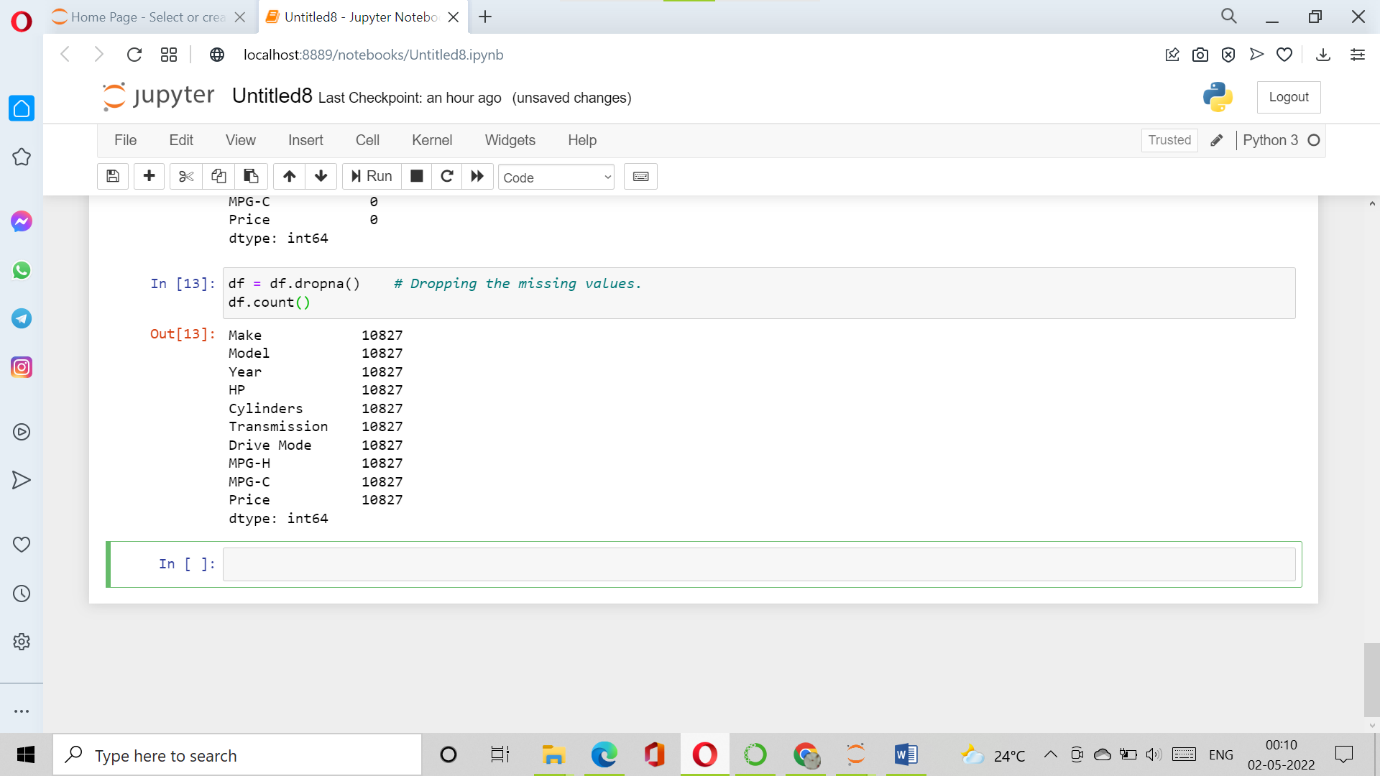
print(df.isnull().sum())



This is the reason in the above step while counting both Cylinders and Horsepower (HP) had 10856 and 10895 over 10925 rows.

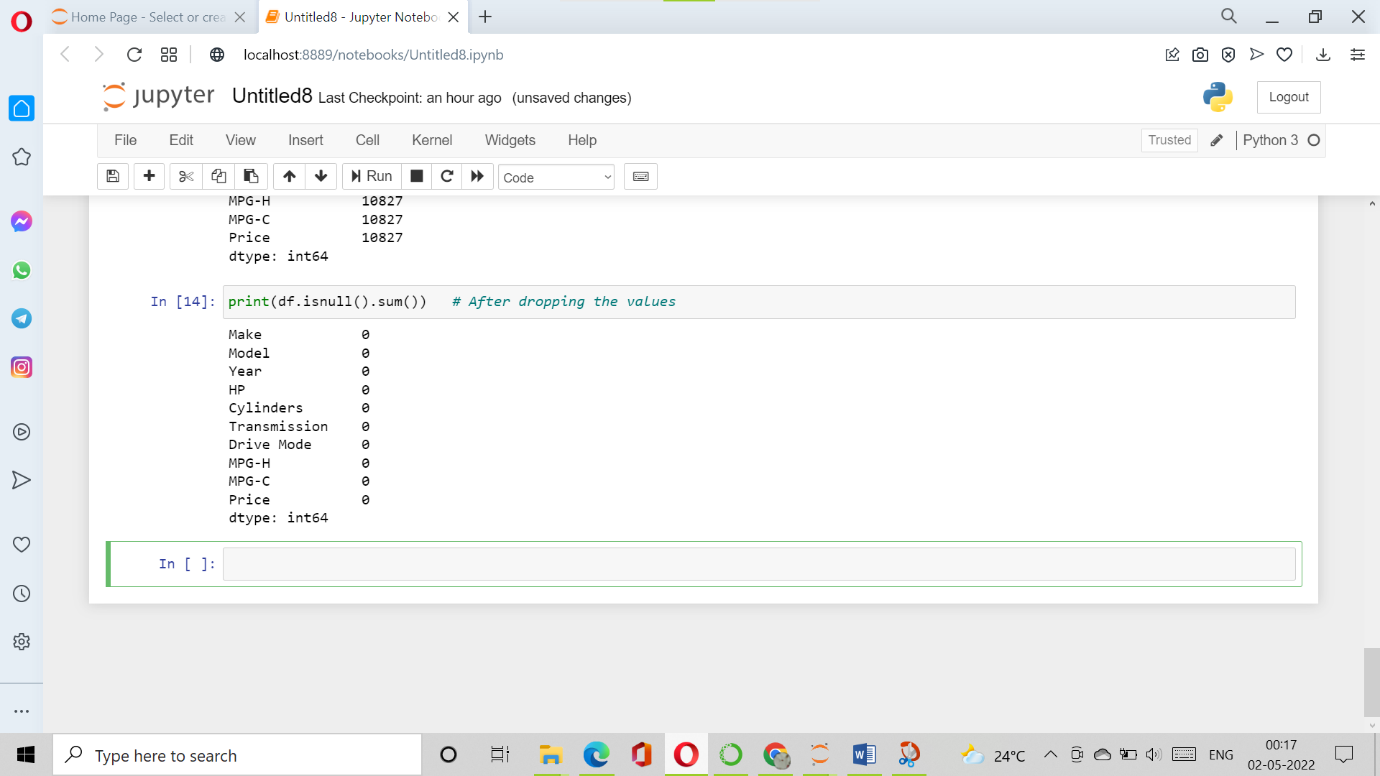
df = df.dropna() *# Dropping the missing values.*

df.count()



Now i have removed all the rows which contain the Null or N/A values (Cylinders and Horsepower (HP)).

print(df.isnull().sum()) *# After dropping the values*



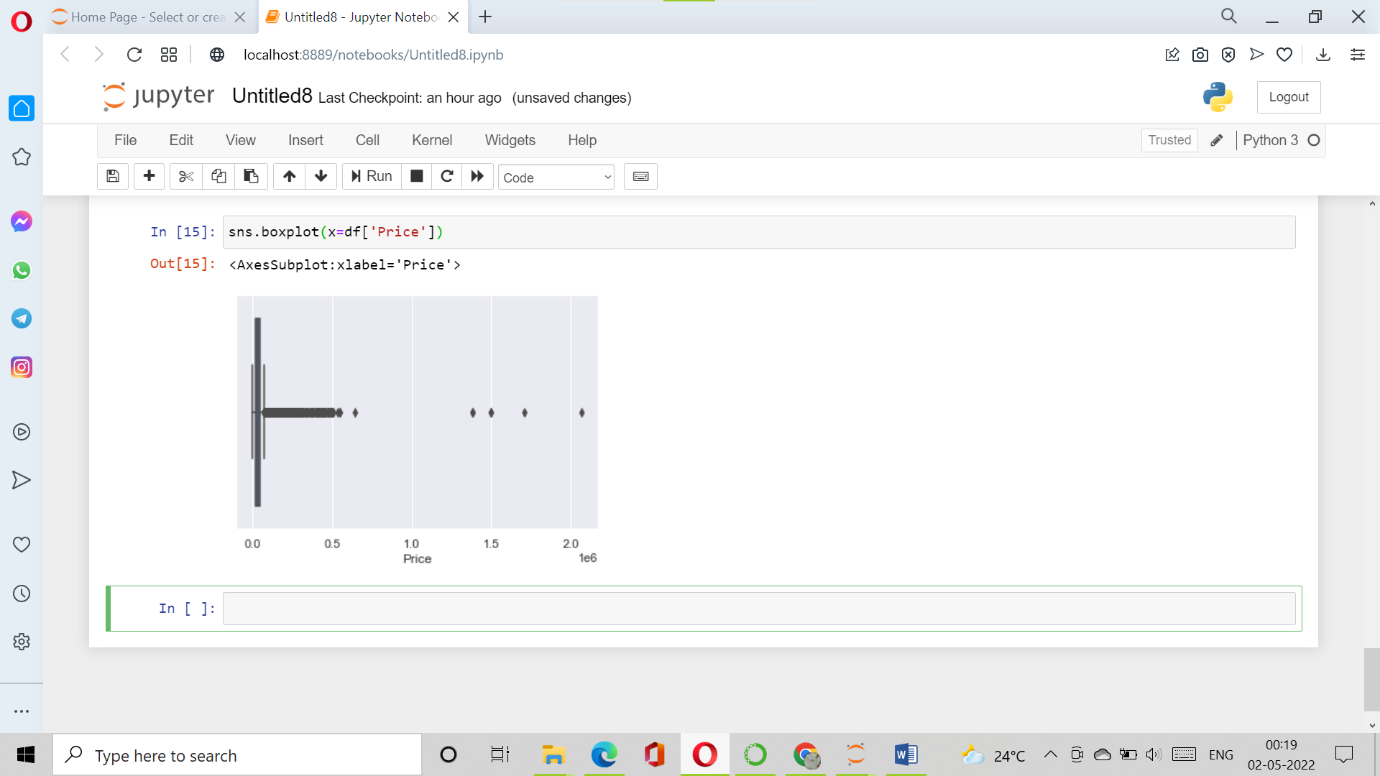
**Visualization of Data**

## 8. Detecting Outliers

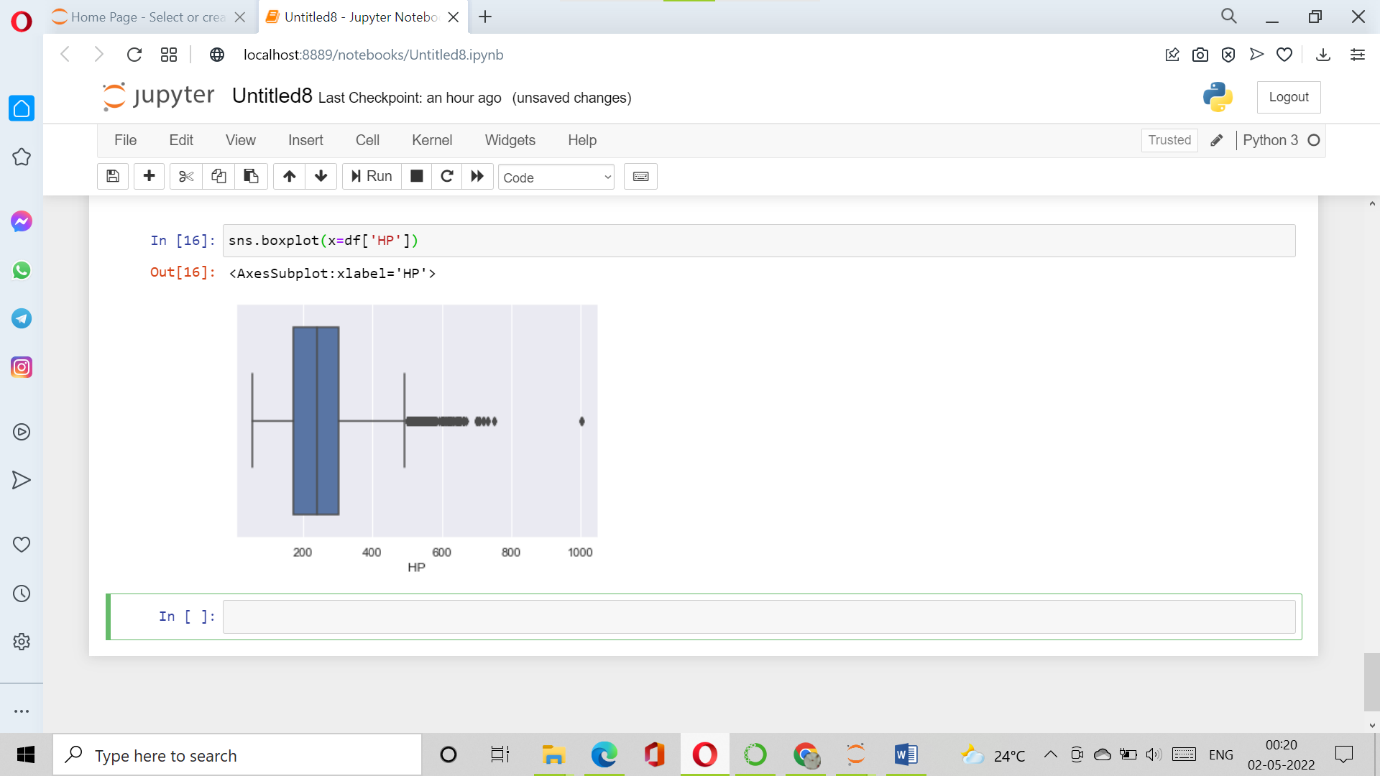
An outlier is a point or set of points that are different from other points. Sometimes they can be very high or very low. It's often a good idea to detect and remove the outliers. Because outliers are one of the primary reasons for resulting in a less accurate model. Hence it's a good idea to remove them. The outlier detection and removing that I am going to perform is called IQR score technique. Often outliers can be seen with visualizations using a box plot. Shown below are the box plot of MSRP, Cylinders, Horsepower and EngineSize. Herein all the plots, you can find some points are outside the box they are none other than outliers. The technique of finding and removing outlier that I am performing in this assignment is taken help of a tutorial from[towards data science](https://towardsdatascience.com/ways-to-detect-and-remove-the-outliers-404d16608dba).

sns.boxplot(x=df['Price'])

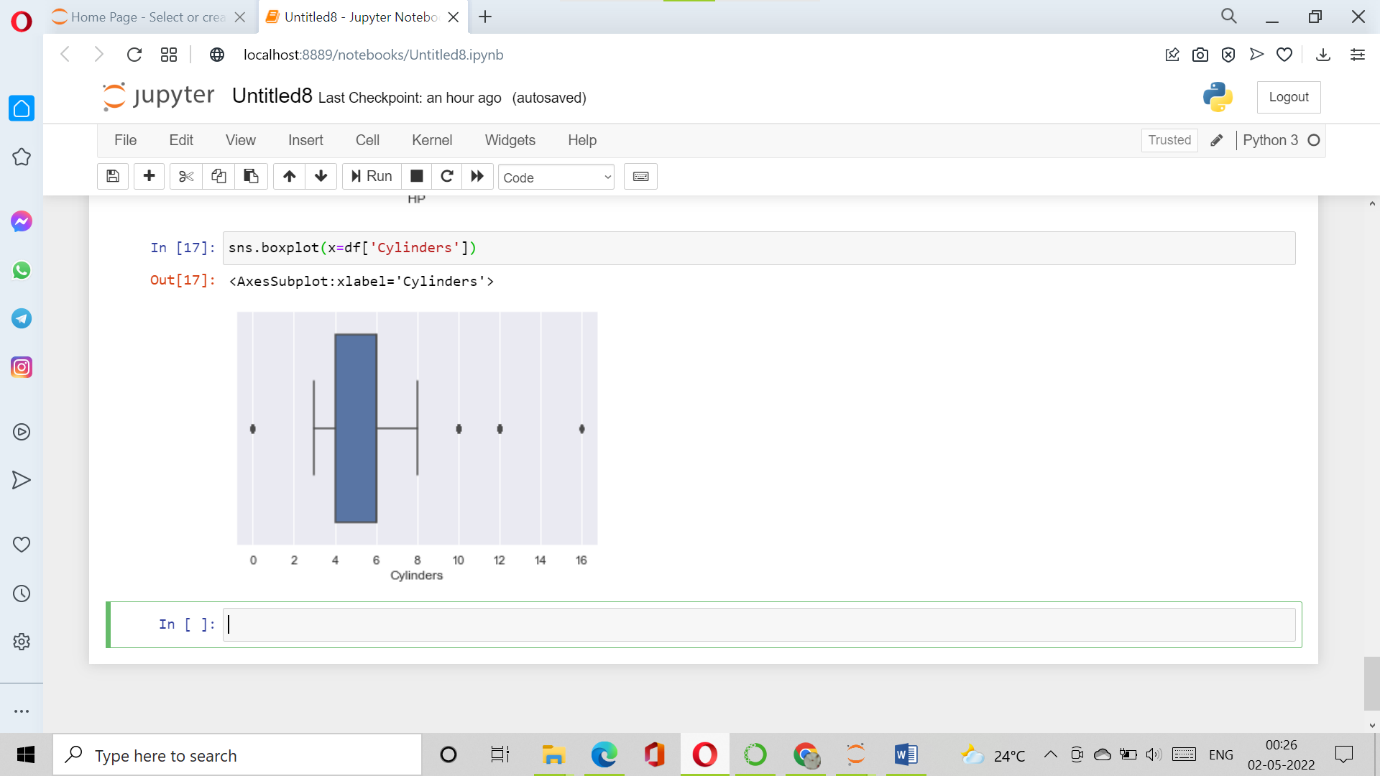
<matplotlib.axes.\_subplots.AxesSubplot at 0x7f89e9c87a90>



sns.boxplot(x=df['HP'])



sns.boxplot(x=df['Cylinders'])

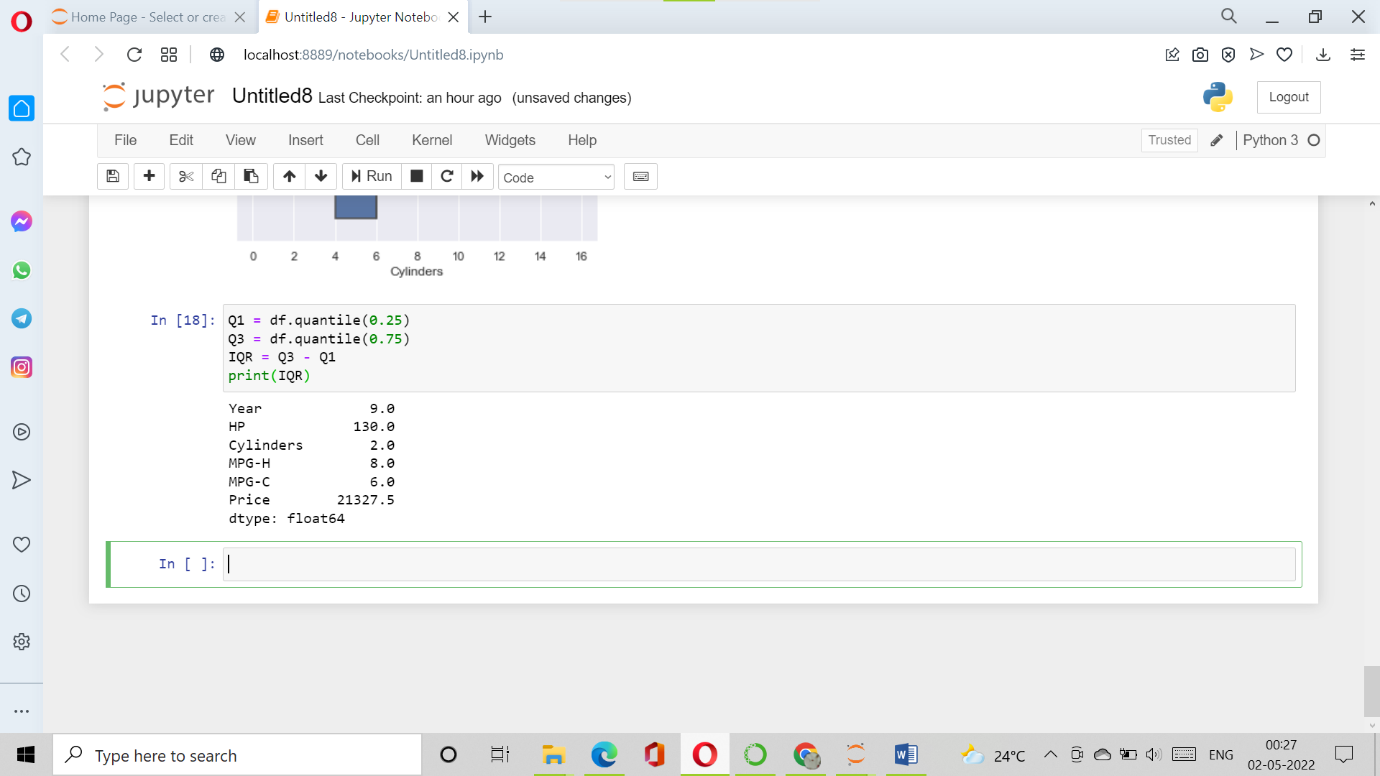


Q1 = df.quantile(0.25)

Q3 = df.quantile(0.75)

IQR = Q3 - Q1

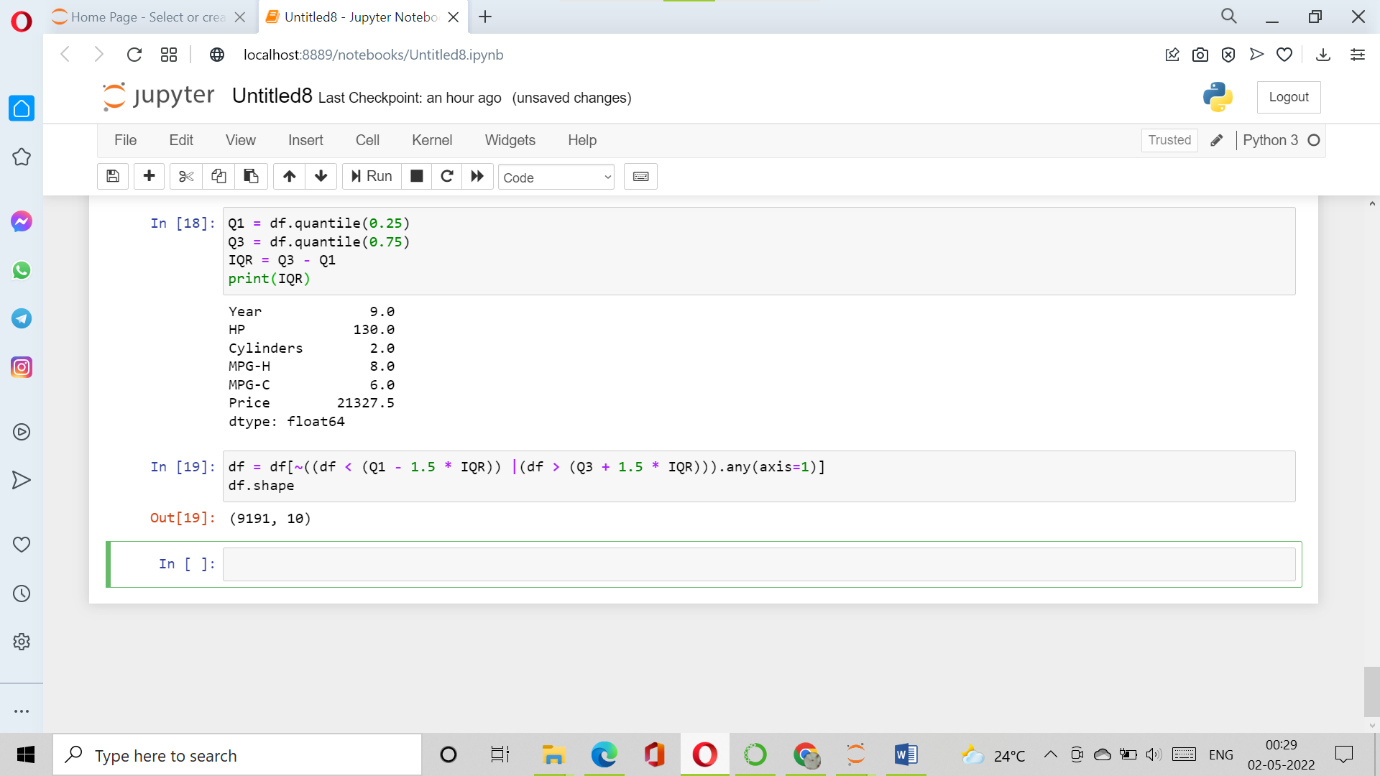
print(IQR)



it's just important to know how to use this technique in order to remove the outliers.

df = df[~((df < (Q1 - 1.5 \* IQR)) |(df > (Q3 + 1.5 \* IQR))).any(axis=1)]

df.shape



As seen above there were around 1600 rows were outliers. But we cannot completely remove the outliers because even after we use the above technique there maybe 1–2 outlier unremoved but that ok because there were more than 100 outliers. Something is better than nothing.

## 9. Plot different features against one another (scatter), against frequency (histogram)

### **Histogram**

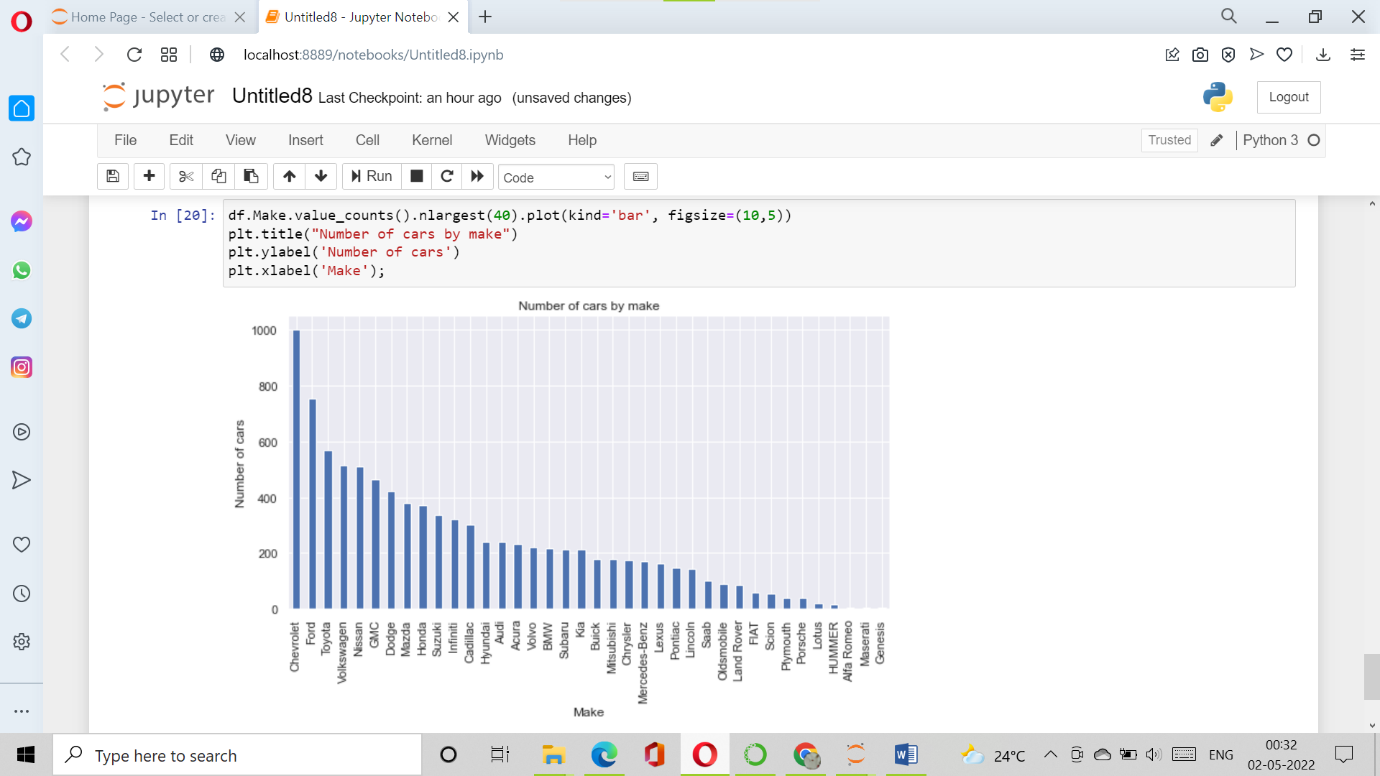
Histogram refers to the frequency of occurrence of variables in an interval. In this case, there are mainly 10 different types of car manufacturing companies, but it is often important to know who has the most number of cars. To do this histogram is one of the trivial solutions which lets us know the total number of car manufactured by a different company.

df.Make.value\_counts().nlargest(40).plot(kind='bar', figsize=(10,5))

plt.title("Number of cars by make")

plt.ylabel('Number of cars')

plt.xlabel('Make');



### **Heat Maps**

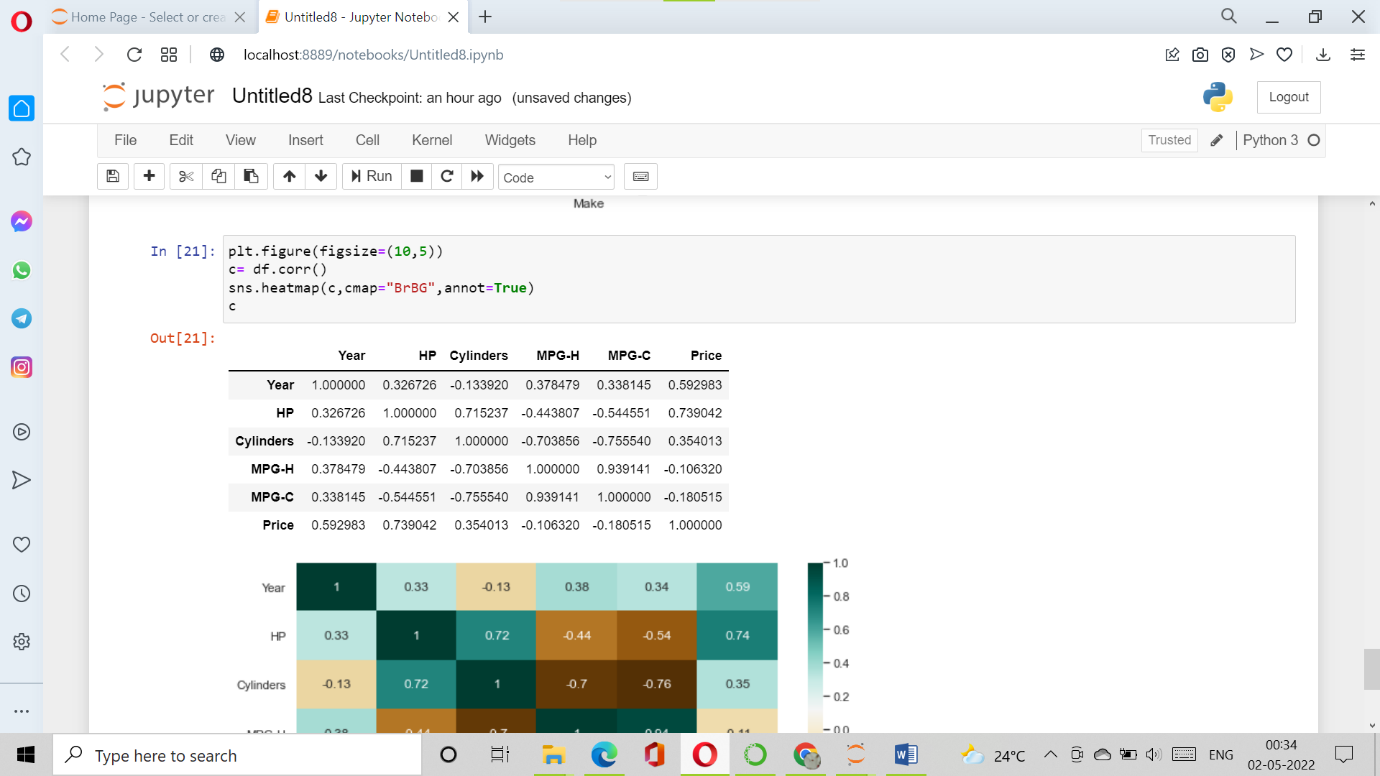
Heat Maps is a type of plot which is necessary when we need to find the dependent variables. One of the best way to find the relationship between the features can be done using heat maps. In the below heat map we know that the price feature depends mainly on the Engine Size, Horsepower, and Cylinders.

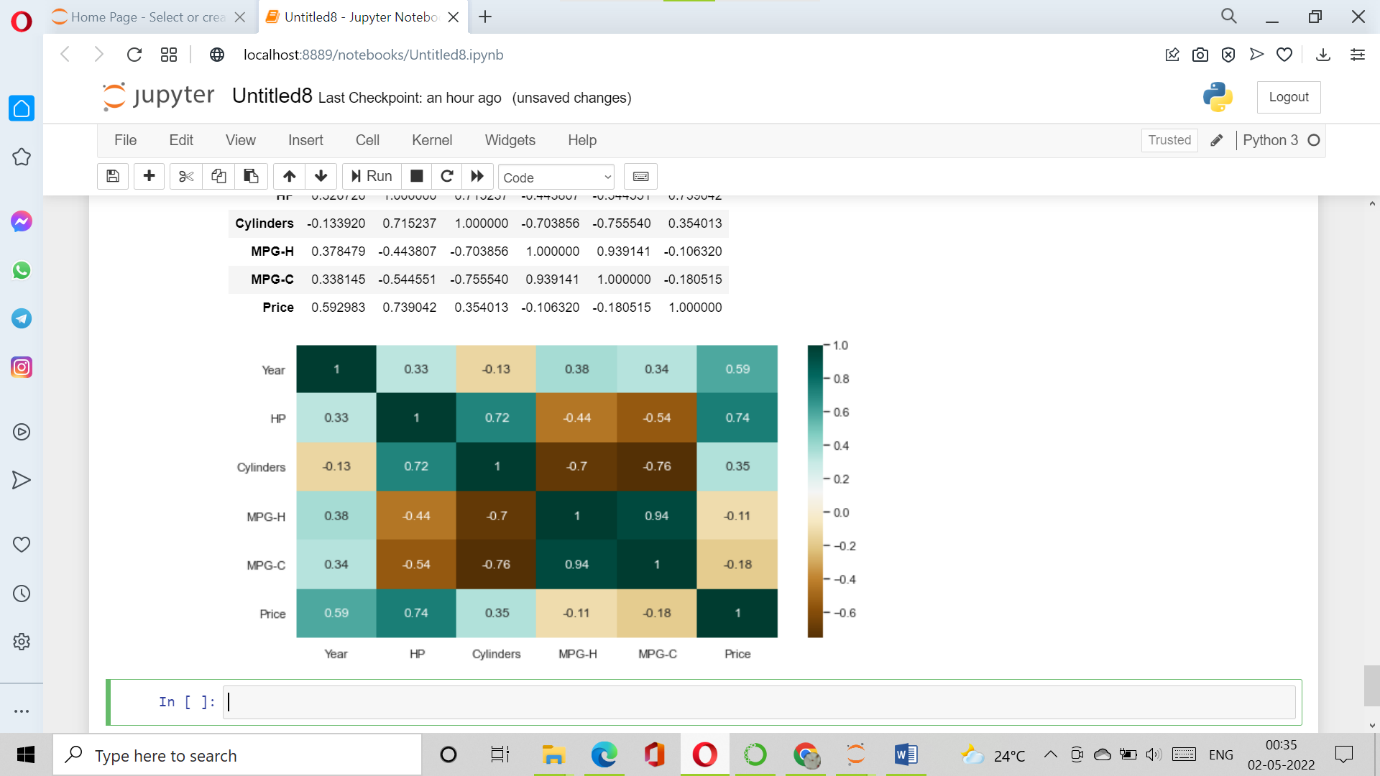
plt.figure(figsize=(10,5))

c= df.corr()

sns.heatmap(c,cmap="BrBG",annot=True)

c





### **Scatterplot**

We generally use scatter plots to find the correlation between two variables. Here the scatter plots are plotted between Horsepower and Price and we can see the plot below. With the plot given below, we can easily draw a trend line. These features provide a good scattering of points.

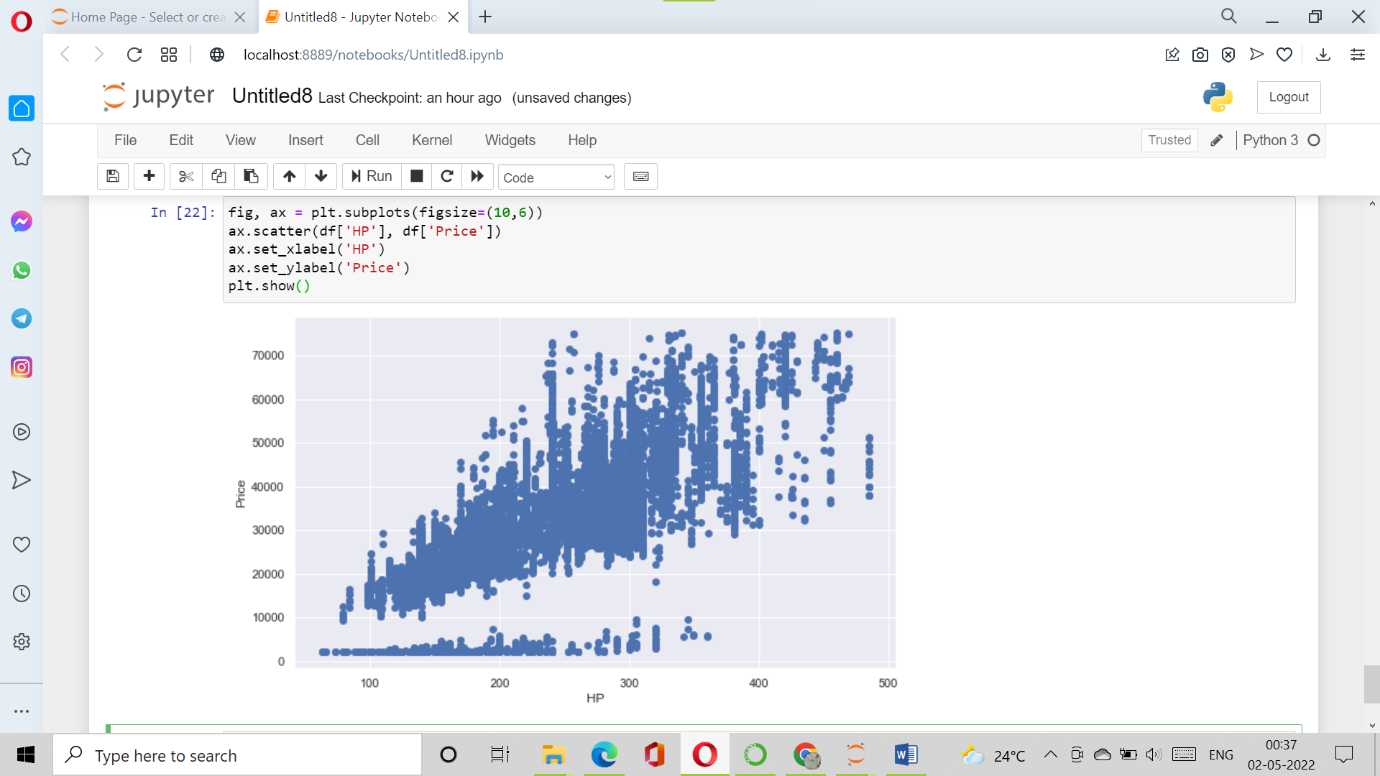
fig, ax = plt.subplots(figsize=(10,6))

ax.scatter(df['HP'], df['Price'])

ax.set\_xlabel('HP')

ax.set\_ylabel('Price')

plt.show()



**Hence the above are some of the steps involved in Exploratory data analysis, these are some general steps that you must follow in order to perform EDA. There are many more yet to come but for now, this is more than enough idea as to how to perform a good EDA given any data sets.**