Data Analytics journal

Name : Vishal Naik
Roll No : 20268
Std : T.Y Bsc Computer Science
Sem 6

INDEX		
Practical 1	Creating your own data set having at least 5 attributes and 50 rows (roll no, name, marks, in 5 subjects) use stastical techniques such as mean, median, mode and Deviation.	16/01/2023
Practical 2	Data cleaning – (dropping irrelevant cols, renaming cols, dropping duplicate rows, dropping missing or null values, replacing values).	23/01/2023
Practical 3	Basic data analysis (head, shape, count, dtypes, tail, Describe).	30/01/2023
Practical 4	Combining data from multimedia	01/03/2023
Practical 5	Web scrapping	06/03/2023
Practical 6	Implentation of K-means algorithm	20/03/2023
Practical 7	Implementation of classification algorithm – decision tree	03/04/2023
Practical 8	Data visualization using box plot, scatter plot, hear maps, histogram.	10/04/2023

Practical 01

Date:16/01/2023

Aim:
Creating your own data set having at least 5 attributes and 50 rows (roll no, name, marks, in 5 subjects) use stastical techniques such as mean, median, mode and Deviation.

Data set:

	A	B	C	D	E	F	G	H
1	Name	Roll no	Inter net of things	Full Stack Development	Android Development	Data Analytics	Project	Total
2	Ramesh 1	1	110	134	134	97	50	525
3	Paresh 2	2	120	102	100	80	24	426
4	Sanjana 3	3	100	130	140	90	40	500
5	Sanali 4	4	85	130	95	60	46	416
6	Sophia 5	5	111	110	100	52	33	406
7	Soham 6	6	121	141	64	4	25	355
8	Soham 7	7	101	100	99	89	24	413
9	Soham 8	8	86	34	75	25	88	308
10	Soham 9	9	112	135	130	30	51	458
11	Soham 10	10	122	103	112	12	25	374
12	Soham 11	11	102	131	140	98	41	512
13	Soham 12	12	87	131	160	81	47	506
14	Soham 13	13	113	111	150	91	34	499
15	Soham 14	14	123	142	158	61	26	510
16	Soham 15	15	103	101	135	53	25	417
17	Soham 16	16	88	35	101	5	89	318
18	Soham 17	17	114	136	141	90	52	533
19	Soham 18	18	124	104	96	26	26	376
20	Soham 19	19	104	132	101	31	42	410
21	Soham 20	20	89	132	65	13	48	347
22	Soham 21	21	115	112	100	99	35	461
23	Soham 22	22	125	143	76	82	27	453
24	Soham 23	23	105	102	131	92	26	456
25	Soham 24	24	90	36	113	62	90	391
26	Soham 25	25	116	137	141	54	53	501
27	Soham 26	26	126	105	161	6	27	425
28	Soham 27	27	106	133	151	91	43	524
29	Soham 28	28	91	133	159	27	49	459
30	Soham 29	29	117	113	136	32	36	434
31	Soham 30	30	127	144	102	14	28	415
32	Soham 31	31	107	103	142	100	27	479
33	Soham 32	32	92	37	97	83	91	400
34	Soham 33	33	118	138	102	93	54	505
35	Soham 34	34	128	106	66	63	28	391
36	Soham 35	35	108	134	101	55	44	442
37	Soham 36	36	93	134	77	7	50	361
38	Soham 37	37	119	114	132	92	37	494
39	Soham 38	38	129	145	114	28	29	445
40	Soham 39	39	109	104	142	33	28	416
41	Soham 40	40	94	38	162	15	92	401
42	Soham 41	41	120	139	152	101	55	567
43	Soham 42	42	130	107	160	84	29	510
44	Soham 43	43	110	135	137	94	45	521
45	Soham 44	44	95	135	103	64	51	448
46	Soham 45	45	121	115	143	56	38	473
47	Soham 46	46	131	146	98	8	30	413
48	Soham 47	47	111	105	103	93	29	441
49	Soham 48	48	96	39	67	29	93	324
50	Soham 49	49	122	140	102	34	56	454
51	Soham 50	50	132	108	78	16	30	364

code:

import numpy as np
import pandas as pd
from scipy import stats
df=pd.read_csv("exel sheet 1.csv")
df.head(5)
print("mean:",df["Inter net of things"].mean())
print("median:",df["Inter net of things"].median())
print("mode:",stats.mode(df["Inter net of things"],keepdims=True))Data cleaning – (dropping irrelevant cols, renaming cols, dropping duplicate rows, dropping missing or null values, replacing values).

output:

```
In [21]: import numpy as np
         import pandas as pd
         from scipy import stats
         df=pd.read_csv("exel sheet 1.csv")
         df.head(5)
         print("mean:",df["Inter net of things"].mean())
         print("median:",df["Inter net of things"].median())
         print("mode:",stats.mode(df["Inter net of things"],keepdims=True))

         mean: 109.96
         median: 111.0
         mode: ModeResult(mode=array([110]), count=array([2]))

In [ ]:
```

Conclusion:
creating own data set and using stastical techniques on it was implemented successfully.

Practical 02

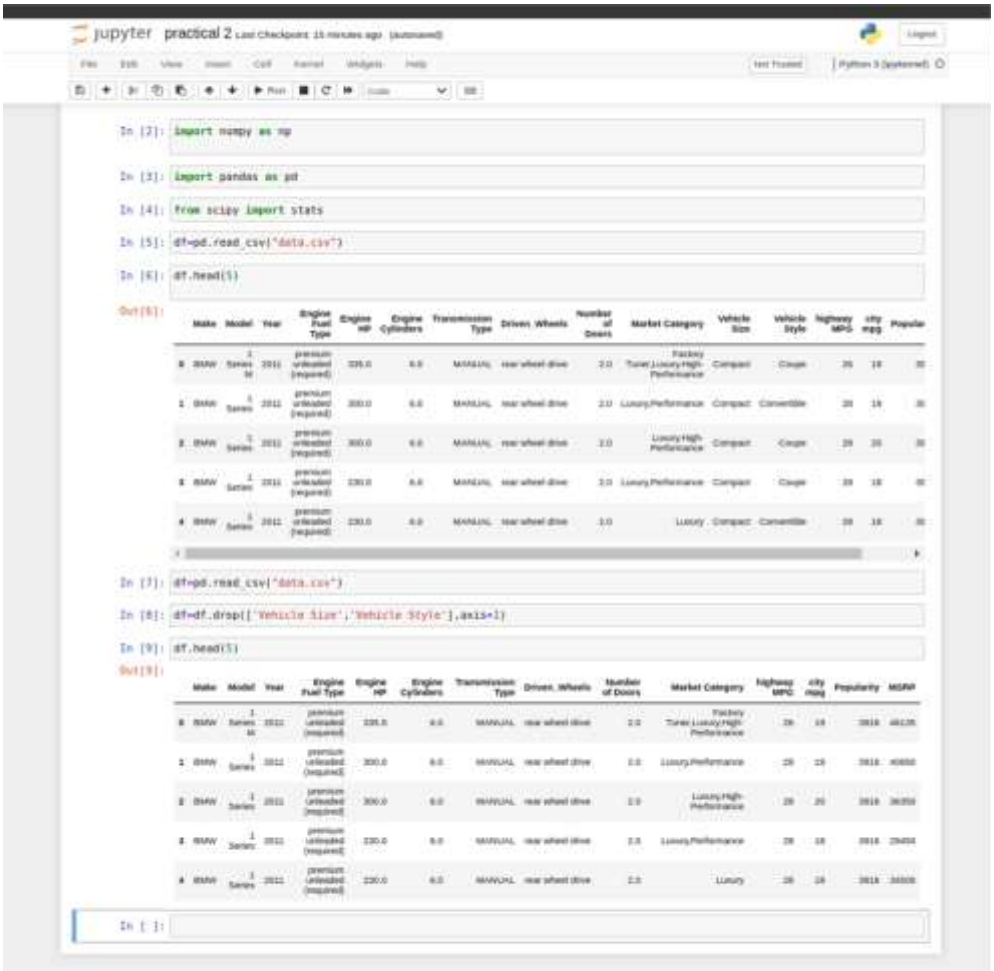
Date:23/01/2023

Aim: Data cleaning – (dropping irrelevant cols, renaming cols, dropping duplicate rows, dropping missing or null values, replacing values).

Code:

(to remove irrelevant colms)

```
import numpy as np
import pandas as pd
from scipy import stats
df=pd.read_csv("data.csv")
df.head(5)

df=pd.read_csv("data.csv")
df=df.drop(['Vehicle Size','Vehicle Style'],axis=1)
df.head(5)
```



(to change column name)
df=pd.read_csv("data.csv")
df=df.rename(columns={"Make":"Type","Year":"Type"})
df.head()

jupyter practical 2 Last Checkpoint: 24 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

In [9]:
```
df.head(5)
```
Out[9]:

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	highway MPG	city mpg	Popularity	MSRP
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Factory Tuner,Luxury,High-Performance	26	19	3916	46135
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	28	19	3916	40650
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,High-Performance	28	20	3916	36350
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	28	18	3916	29450
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury	28	18	3916	34500

In [10]:
```
df=pd.read_csv("data.csv")
```
In [11]:
```
df=df.rename(columns={"Make":"Type","Year":"Type"})
```
In [12]:
```
df.head()
```
Out[12]:

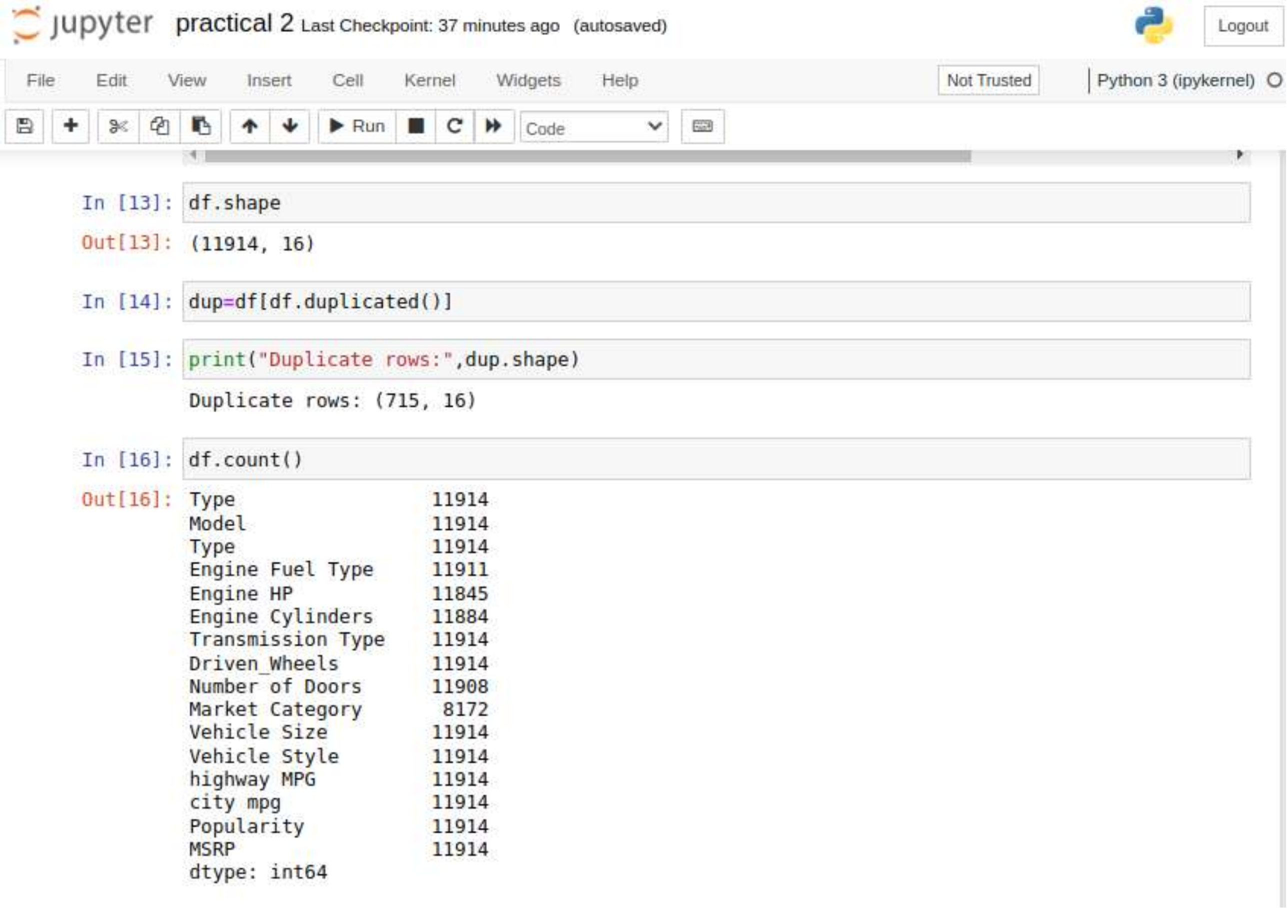
	Type	Model	Type	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size	Vehicle Style	highway MPG	city mpg	Populari
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Factory Tuner,Luxury,High-Performance	Compact	Coupe	26	19	391
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Convertible	28	19	391
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,High-Performance	Compact	Coupe	28	20	391
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Coupe	28	18	391
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury	Compact	Convertible	28	18	391

In [13]:
```
df.shape
```
Out[13]:
```
(11914, 16)
```

df=pd.read_csv("data.csv")
df.shape

code: (to show the number of duplicate rows)

dup=df[df.duplicated()]
print("Duplicate rows:",dup.shape)
df.count()




```
In [13]: df.shape
Out[13]: (11914, 16)

In [14]: dup=df[df.duplicated()]

In [15]: print("Duplicate rows:",dup.shape)
         Duplicate rows: (715, 16)

In [16]: df.count()
Out[16]: Type                 11914
         Model                11914
         Type                 11914
         Engine Fuel Type     11911
         Engine HP            11845
         Engine Cylinders     11884
         Transmission Type    11914
         Driven_Wheels        11914
         Number of Doors      11908
         Market Category       8172
         Vehicle Size         11914
         Vehicle Style        11914
         highway MPG          11914
         city mpg             11914
         Popularity           11914
         MSRP                 11914
         dtype: int64
```

code: (to remove duplicate rows)

df=df.drop_duplicates()
df.head()
df.count()

jupyter practical 2 Last Checkpoint: 40 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

```
In [17]: df=df.drop_duplicates()
```

```
In [18]: df.head()
```

Out[18]:

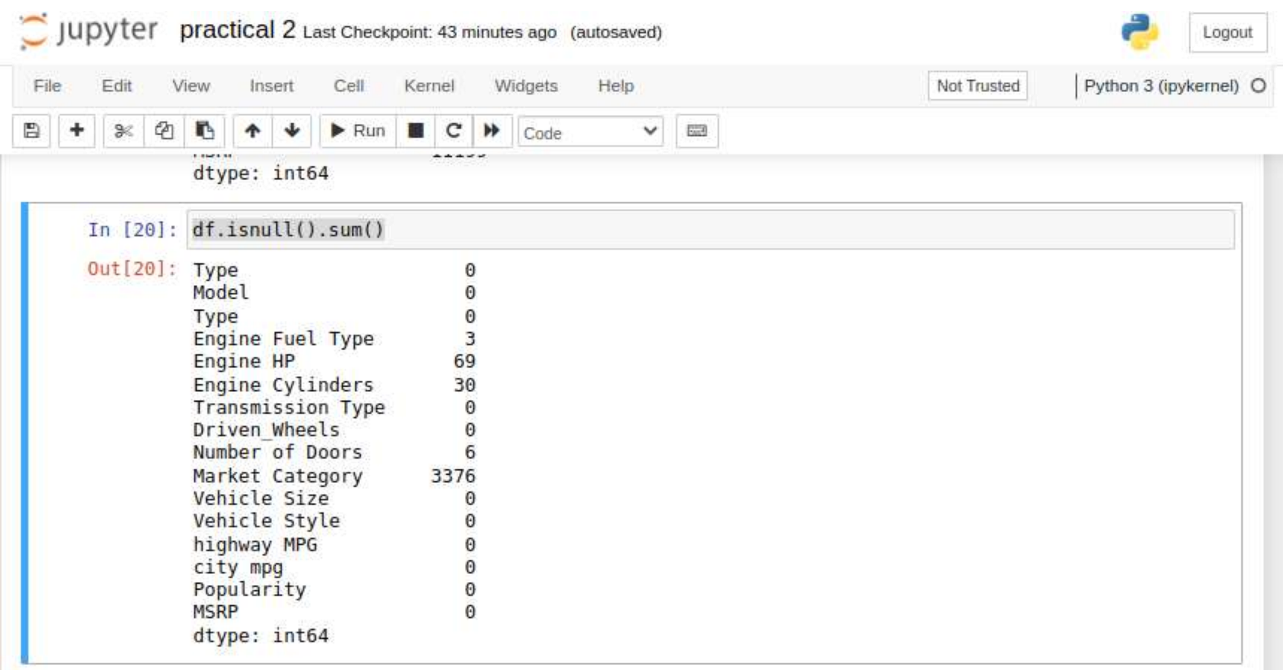
	Type	Model	Type	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size	
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Factory Tuner,Luxury,High-Performance	Compact	
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Co
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,High-Performance	Compact	
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury	Compact	Co

```
In [19]: df.count()
```

```
Out[19]: Type                 11199
Model                11199
Type                 11199
Engine Fuel Type     11196
Engine HP            11130
Engine Cylinders     11169
Transmission Type    11199
Driven_Wheels        11199
Number of Doors      11193
Market Category       7823
Vehicle Size         11199
Vehicle Style        11199
highway MPG          11199
city mpg             11199
Popularity           11199
MSRP                 11199
dtype: int64
```

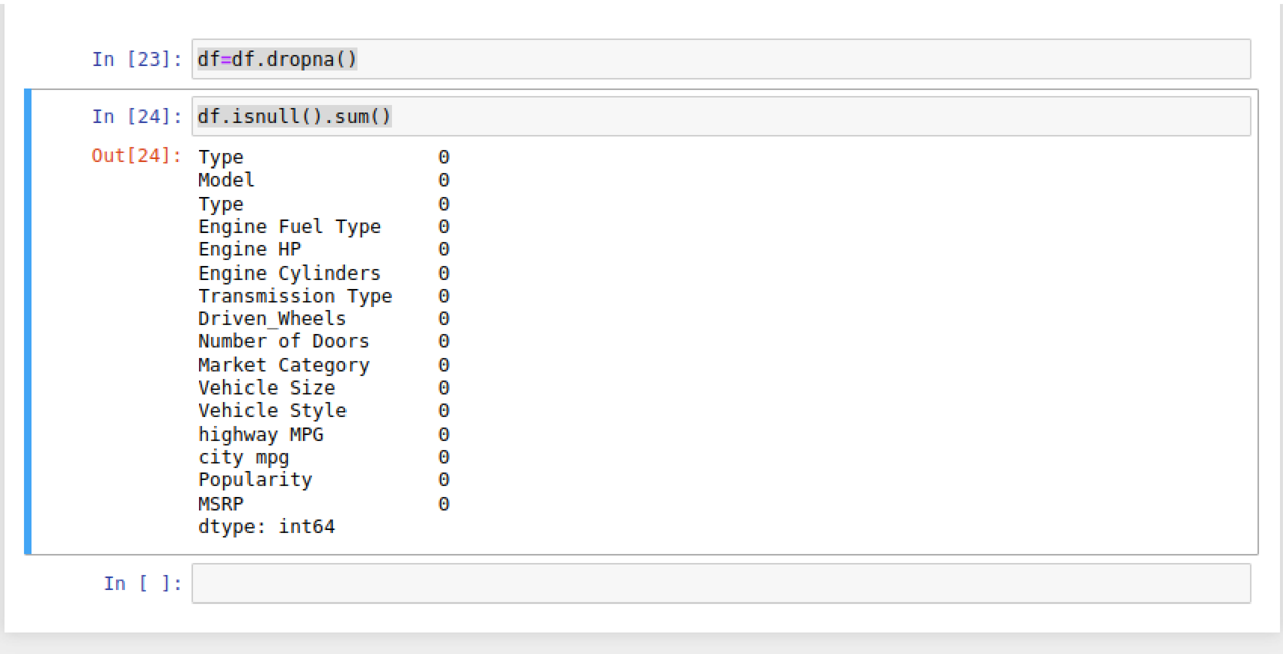
code: (to show number of empty cells in the colomns)

df.isnull().sum()



code:(to drop duplicate cells in the columns)

df=df.dropna()
df.isnull().sum()



code:

```
df1=pd.read_csv("data.csv")
for num, x in enumerate(df1["Market Category"]):
    print(num, x)
```

```
In [18]: for num, x in enumerate(df1["Market Category"]):
             print(num, x)

79 Flex Fuel,Performance
80 Flex Fuel
81 Flex Fuel
82 Flex Fuel
83 Flex Fuel,Performance
84 Flex Fuel
85 Flex Fuel
86 Flex Fuel
87 nan
88 nan
89 Performance
90 Performance
91 nan
92 nan
93 nan
94 nan
95 Performance
96 Performance
97 Performance
98 Performance
```

```
df1["Market Category"].fillna("no category",inplace = True)
for num, x in enumerate(df1["Market Category"]):
    print(num, x)
df1["Market Category"].isnull().sum()
```

```
In [19]: df1["Market Category"].fillna("no category",inplace = True)
         for num, x in enumerate(df1["Market Category"]):
             print(num, x)
         df1["Market Category"].isnull().sum()

79 Flex Fuel,Performance
80 Flex Fuel
81 Flex Fuel
82 Flex Fuel
83 Flex Fuel,Performance
84 Flex Fuel
85 Flex Fuel
86 Flex Fuel
87 no category
88 no category
89 Performance
90 Performance
91 no category
92 no category
93 no category
94 no category
95 Performance
96 Performance
97 Performance
98 Performance
```

Conclusion : Data cleaning was implemented successfully.

Practical 03

Date:30/01/2023

Aim: Basic data analysis (head, shape, count, dtypes, tail, Describe).

Head:

```
In [3]: df=pd.read_csv("data.csv")
        df.head(10)
```

Out[3]:

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size	Vehicle Style	highway MPG	city mpg	Popula
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Factory Tuner,Luxury,High-Performance	Compact	Coupe	26	19	3
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Convertible	28	19	3
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,High-Performance	Compact	Coupe	28	20	3
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Coupe	28	18	3
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury	Compact	Convertible	28	18	3
5	BMW	1 Series	2012	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Coupe	28	18	3
6	BMW	1 Series	2012	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Convertible	26	17	3
7	BMW	1 Series	2012	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,High-Performance	Compact	Coupe	28	20	3
8	BMW	1 Series	2012	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury	Compact	Convertible	28	18	3
9	BMW	1 Series	2013	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury	Compact	Convertible	27	18	3

shape:

```
In [5]: df=pd.read_csv("data.csv")
        df.shape
```

Out[5]: (11914, 16)

count:

```
In [6]: df=pd.read_csv("data.csv")
        df.count()

Out[6]: Make                 11914
        Model                11914
        Year                 11914
        Engine Fuel Type     11911
        Engine HP            11845
        Engine Cylinders     11884
        Transmission Type    11914
        Driven_Wheels        11914
        Number of Doors      11908
        Market Category       8172
        Vehicle Size         11914
        Vehicle Style        11914
        highway MPG          11914
        city mpg             11914
        Popularity           11914
        MSRP                 11914
        dtype: int64
```

dtypes:

```
In [7]: df=pd.read_csv("data.csv")
        df.dtypes

Out[7]: Make                  object
        Model                 object
        Year                   int64
        Engine Fuel Type      object
        Engine HP            float64
        Engine Cylinders     float64
        Transmission Type     object
        Driven_Wheels         object
        Number of Doors      float64
        Market Category       object
        Vehicle Size          object
        Vehicle Style         object
        highway MPG            int64
        city mpg               int64
        Popularity             int64
        MSRP                   int64
        dtype: object
```

tail:

```
In [8]: df=pd.read_csv("data.csv")
        df.tail(8)
```

Out[8]:

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size	Vehicle Style	highw MI
11906	Acura	ZDX	2011	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury	Midsize	4dr Hatchback	
11907	Acura	ZDX	2011	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury	Midsize	4dr Hatchback	
11908	Acura	ZDX	2011	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury	Midsize	4dr Hatchback	
11909	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury	Midsize	4dr Hatchback	
11910	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury	Midsize	4dr Hatchback	
11911	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury	Midsize	4dr Hatchback	
11912	Acura	ZDX	2013	premium unleaded (recommended)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury	Midsize	4dr Hatchback	
11913	Lincoln	Zephyr	2006	regular unleaded	221.0	6.0	AUTOMATIC	front wheel drive	4.0	Luxury	Midsize	Sedan	

Describe:

```
In [9]: df=pd.read_csv("data.csv")
        df.describe()
```

Out[9]:

	Year	Engine HP	Engine Cylinders	Number of Doors	highway MPG	city mpg	Popularity	MSRP
count	11914.000000	11845.00000	11884.000000	11908.000000	11914.000000	11914.000000	11914.000000	1.191400e+04
mean	2010.384338	249.38607	5.628829	3.436093	26.637485	19.733255	1554.911197	4.059474e+04
std	7.579740	109.19187	1.780559	0.881315	8.863001	8.987798	1441.855347	6.010910e+04
min	1990.000000	55.00000	0.000000	2.000000	12.000000	7.000000	2.000000	2.000000e+03
25%	2007.000000	170.00000	4.000000	2.000000	22.000000	16.000000	549.000000	2.100000e+04
50%	2015.000000	227.00000	6.000000	4.000000	26.000000	18.000000	1385.000000	2.999500e+04
75%	2016.000000	300.00000	6.000000	4.000000	30.000000	22.000000	2009.000000	4.223125e+04
max	2017.000000	1001.00000	16.000000	4.000000	354.000000	137.000000	5657.000000	2.065902e+06

Conclusion: Basic data analysis was carried out successfully

Practical 04

Date:01/03/2023

Aim: Combining data from multiple.
Sources (https://www.dataquest.io/blog/pandas-concatenation-tutorial/)

Code:

import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

north_america =
pd.read_csv('north_america_2000_2010.csv',index_col=0)
south_america =
pd.read_csv('south_america_2000_2010.csv',index_col=0)

north_america

In [11]: north_america

Out[11]:

Country	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010
Canada	1779.0	1771.0	1754.0	1740.0	1760.0	1747	1745.0	1741.0	1735	1701.0	1703.0
Mexico	2311.2	2285.2	2271.2	2276.5	2270.6	2281	2280.6	2261.4	2258	2250.2	2242.4
USA	1836.0	1814.0	1810.0	1800.0	1802.0	1799	1800.0	1798.0	1792	1767.0	1778.0

south_america

In [12]: south_america

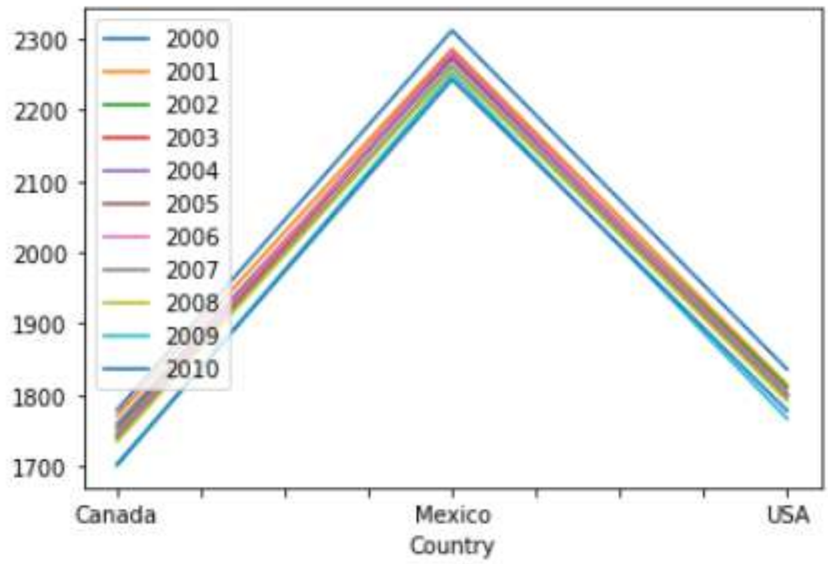
Out[12]:

Country	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010
Chile	2263	2242	2250	2235	2232	2157	2165	2128	2095	2074	2069.6

north_america.plot()

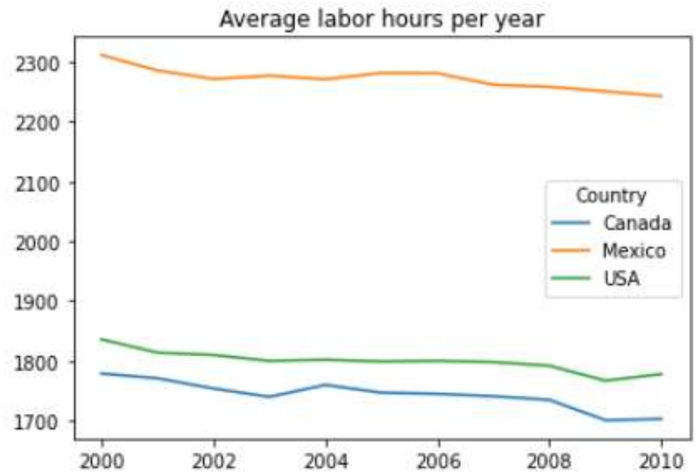
```
In [13]: north_america.plot()
```

Out[13]: <AxesSubplot:xlabel='Country'>



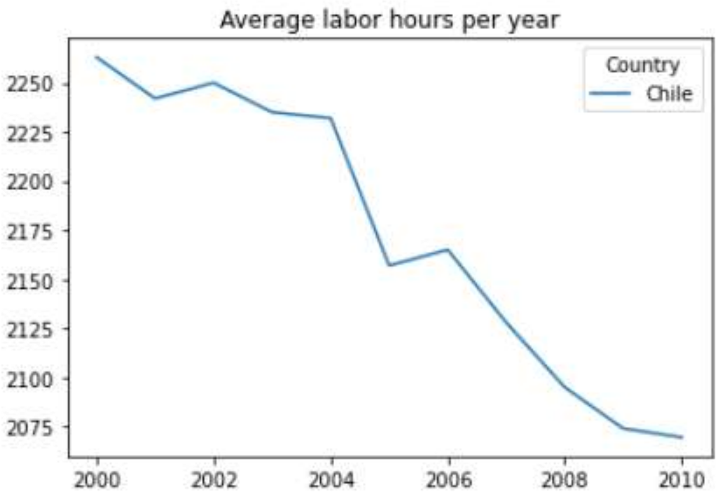
north_america.transpose().plot(title='Average labor hours per year')
plt.show()

```
In [14]: north_america.transpose().plot(title='Average labor hours per year')
         plt.show()
```



south_america.transpose().plot(title='Average labor hours per year')
plt.show()

```
In [15]: south_america.transpose().plot(title='Average labor hours per year')
         plt.show()
```



result = pd.concat([north_america,south_america])
result

```
In [16]: result = pd.concat([north_america,south_america])
         result
```

Out[16]:

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010
Country											
Canada	1779.0	1771.0	1754.0	1740.0	1760.0	1747	1745.0	1741.0	1735	1701.0	1703.0
Mexico	2311.2	2285.2	2271.2	2276.5	2270.6	2281	2280.6	2261.4	2258	2250.2	2242.4
USA	1836.0	1814.0	1810.0	1800.0	1802.0	1799	1800.0	1798.0	1792	1767.0	1778.0
Chile	2263.0	2242.0	2250.0	2235.0	2232.0	2157	2165.0	2128.0	2095	2074.0	2069.6

```
americas_df = [result]
for year in range(2011,2016):
    filename = 'americas_{}.csv'.format(year)
    df = pd.read_csv(filename,index_col=0)
    americas_df.append(df)
americas_df[1]
```

```
In [17]: americas_df = [result]
         for year in range(2011,2016):
             filename = 'americas_{}.csv'.format(year)
             df = pd.read_csv(filename,index_col=0)
             americas_df.append(df)
         americas_df[1]
```

Out[17]:

	2011
Country	
Canada	1700.0
Chile	2047.4
Mexico	2250.2
USA	1786.0

```
result = pd.concat(americas_df,axis=1)
result.index.names = ['Country']
result
```

```
In [18]: result = pd.concat(americas_df,axis=1)
         result.index.names = ['Country']
         result
```

Out[18]:

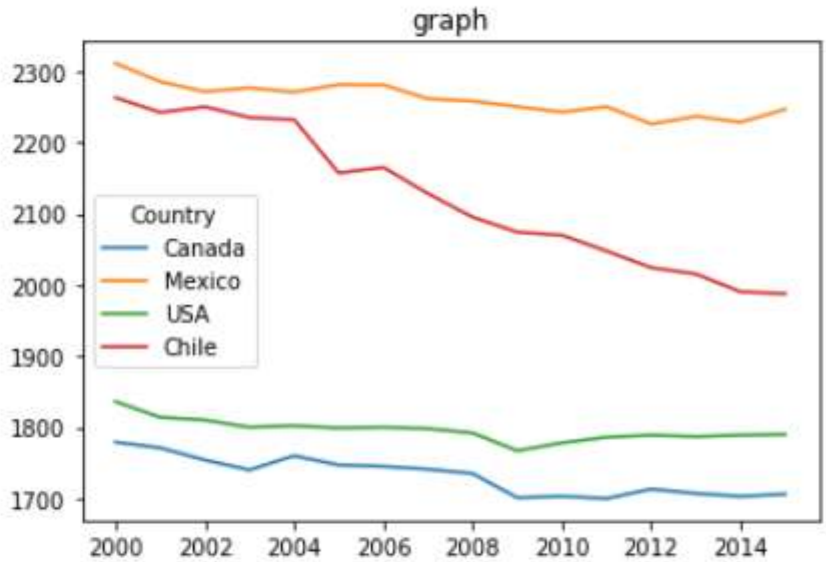
	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
Country																
Canada	1779.0	1771.0	1754.0	1740.0	1760.0	1747	1745.0	1741.0	1735	1701.0	1703.0	1700.0	1713.0	1707.0	1703.0	1706.0
Mexico	2311.2	2285.2	2271.2	2276.5	2270.6	2281	2280.6	2261.4	2258	2250.2	2242.4	2250.2	2225.8	2236.6	2228.4	2246.4
USA	1836.0	1814.0	1810.0	1800.0	1802.0	1799	1800.0	1798.0	1792	1767.0	1778.0	1786.0	1789.0	1787.0	1789.0	1790.0
Chile	2263.0	2242.0	2250.0	2235.0	2232.0	2157	2165.0	2128.0	2095	2074.0	2069.6	2047.4	2024.0	2015.3	1990.1	1987.5

result.transpose().plot(title = 'graph')

asia

```
In [19]: result.transpose().plot(title = 'graph')
Out[19]: <AxesSubplot:title={'center':'graph'}>
```

=



pd.read_csv('asia_2000_2015.csv',index_col=0)
asia

```
In [20]: asia = pd.read_csv('asia_2000_2015.csv',index_col=0)
         asia
```

Out[20]:

Country	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
Israel	2017	1979	1993	1974	1942	1931	1919	1931	1929	1927	1918	1920	1910	1867	1853	1858
Japan	1821	1809	1798	1799	1787	1775	1784	1785	1771	1714	1733	1728	1745	1734	1729	1719
Korea	2512	2499	2464	2424	2392	2351	2346	2306	2246	2232	2187	2090	2163	2079	2124	2113
Russia	1982	1980	1982	1993	1993	1989	1998	1999	1997	1974	1976	1979	1982	1980	1985	1978

europe = pd.read_csv('europe_2000_2015.csv',index_col=0)
europe

In [21]:
```
europe = pd.read_csv('europe_2000_2015.csv',index_col=0)
europe
```

Out[21]:

Country	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
Austria	1807.400000	1794.60000	1792.200000	1783.800000	1786.800000	1764.000000	1746.2	1736.00	1728.500000	1673.000000	1668.600000	1675.90	1652.90	1636.70	1629.40	1624.90
Belgium	1595.000000	1588.00000	1583.000000	1578.000000	1573.000000	1565.000000	1572.0	1577.00	1570.000000	1548.000000	1546.000000	1560.00	1560.00	1558.00	1560.00	1541.00
Switzerland	1673.600000	1635.00000	1614.000000	1626.800000	1656.500000	1651.700000	1643.2	1632.70	1623.100000	1614.900000	1612.400000	1605.40	1590.90	1572.90	1568.30	1589.70
Czech Republic	1896.000000	1818.00000	1816.000000	1806.000000	1817.000000	1817.000000	1799.0	1784.00	1790.000000	1779.000000	1800.000000	1806.00	1776.00	1763.00	1771.00	1779.00
Germany	1452.000000	1441.90000	1430.900000	1424.800000	1422.200000	1411.300000	1424.7	1424.40	1418.400000	1372.700000	1389.900000	1392.80	1375.30	1361.70	1366.40	1371.00
Denmark	1490.000000	1493.00000	1487.000000	1482.000000	1481.000000	1474.000000	1479.0	1456.00	1450.000000	1446.000000	1436.000000	1455.00	1437.00	1457.00	1458.00	1457.00
Spain	1752.800000	1762.50000	1764.600000	1755.900000	1741.500000	1725.600000	1715.7	1703.50	1712.600000	1719.700000	1710.400000	1715.50	1699.90	1695.60	1698.00	1691.30
Estonia	1978.000000	1970.00000	1973.000000	1978.000000	1986.000000	2008.000000	2001.0	1998.00	1968.000000	1831.000000	1875.000000	1919.00	1886.00	1866.00	1859.00	1852.00
Finland	1742.000000	1723.00000	1714.000000	1705.000000	1707.000000	1697.000000	1693.0	1691.00	1685.000000	1661.000000	1668.000000	1662.00	1650.00	1640.00	1643.00	1646.00
France	1534.800049	1525.98999	1487.319946	1484.380005	1513.209961	1507.439941	1484.0	1500.25	1507.170044	1489.069946	1493.959961	1496.33	1490.23	1474.30	1473.45	1482.00
United Kingdom	1700.000000	1705.00000	1684.000000	1674.000000	1674.000000	1673.000000	1669.0	1677.00	1659.000000	1651.000000	1650.000000	1634.00	1654.00	1666.00	1677.00	1674.00
Greece	2108.000000	2101.00000	2093.000000	2091.000000	2083.000000	2136.000000	2125.0	2111.00	2106.000000	2081.000000	2020.000000	2038.00	2055.00	2063.00	2026.00	2042.00
Hungary	2032.800000	1993.30000	2005.300000	1978.200000	1986.100000	1986.900000	1983.5	1978.60	1981.700000	1963.100000	1958.400000	1975.70	1889.20	1879.80	1857.90	1748.60
Ireland	1933.000000	1924.00000	1904.000000	1887.000000	1875.000000	1883.000000	1879.0	1865.00	1844.000000	1812.000000	1801.000000	1801.00	1806.00	1815.00	1821.26	1819.54
Iceland	2040.100000	2053.20000	2011.800000	1971.900000	1979.600000	1970.000000	1957.6	1931.60	1933.800000	1848.700000	1833.600000	1878.40	1853.40	1846.10	1864.10	1879.50
Italy	1850.800000	1837.80000	1826.600000	1815.800000	1815.300000	1812.100000	1812.6	1818.20	1807.000000	1775.700000	1777.300000	1773.30	1734.20	1719.51	1718.80	1724.83
Lithuania	1846.000000	1831.00000	1802.000000	1786.000000	1879.000000	1879.000000	1874.0	1904.00	1934.000000	1863.000000	1884.000000	1859.00	1857.00	1841.00	1834.00	1860.00
Luxembourg	1603.000000	1586.00000	1581.000000	1579.000000	1579.000000	1555.000000	1556.0	1570.00	1570.000000	1519.000000	1521.000000	1519.00	1513.00	1503.00	1509.00	1507.00
Latvia	1976.000000	1987.00000	1938.000000	1928.000000	1878.000000	1906.000000	1907.0	1878.00	2002.000000	1952.000000	1935.000000	1952.00	1934.00	1928.00	1938.00	1903.00
Netherlands	1462.000000	1452.00000	1435.000000	1427.000000	1448.000000	1434.000000	1430.0	1430.00	1430.000000	1422.000000	1421.000000	1422.00	1413.00	1415.00	1420.00	1419.00
Norway	1455.000000	1429.00000	1414.000000	1400.700000	1420.500000	1422.800000	1419.8	1426.00	1429.500000	1406.800000	1415.300000	1421.10	1419.70	1408.10	1426.90	1423.90
Poland	1988.000000	1974.00000	1979.000000	1984.000000	1983.000000	1994.000000	1985.0	1976.00	1969.000000	1948.000000	1940.000000	1938.00	1929.00	1918.00	1923.00	1963.00
Portugal	1917.000000	1900.00000	1894.000000	1887.000000	1893.000000	1895.000000	1883.0	1900.00	1887.000000	1887.000000	1890.000000	1867.00	1849.00	1859.00	1865.00	1868.00
Slovak Republic	1816.000000	1801.00000	1754.000000	1698.000000	1742.000000	1769.000000	1774.0	1791.00	1793.000000	1780.000000	1805.000000	1793.00	1789.00	1772.00	1760.00	1754.00
Slovenia	1710.000000	1696.00000	1721.000000	1724.000000	1737.000000	1697.000000	1667.0	1655.00	1674.000000	1679.000000	1680.000000	1663.00	1636.00	1655.00	1676.00	1676.00
Sweden	1642.000000	1618.00000	1595.000000	1582.000000	1605.000000	1605.000000	1599.0	1612.00	1617.000000	1609.000000	1635.000000	1632.00	1618.00	1609.00	1611.00	1612.00

south_pacific = pd.read_csv('south_pacific_2000_2015.csv',index_col=0)
south_pacific

In [22]:
```
south_pacific = pd.read_csv('south_pacific_2000_2015.csv',index_col=0)
south_pacific
```

Out[22]:

Country	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
Australia	1778.7	1736.7	1731.7	1735.8	1734.5	1729.2	1720.5	1712.5	1717.2	1690	1691.5	1699.5	1678.6	1662.7	1663.6	1665
New Zealand	1836.0	1825.0	1826.0	1823.0	1830.0	1815.0	1795.0	1774.0	1761.0	1740	1755.0	1746.0	1734.0	1752.0	1762.0	1757

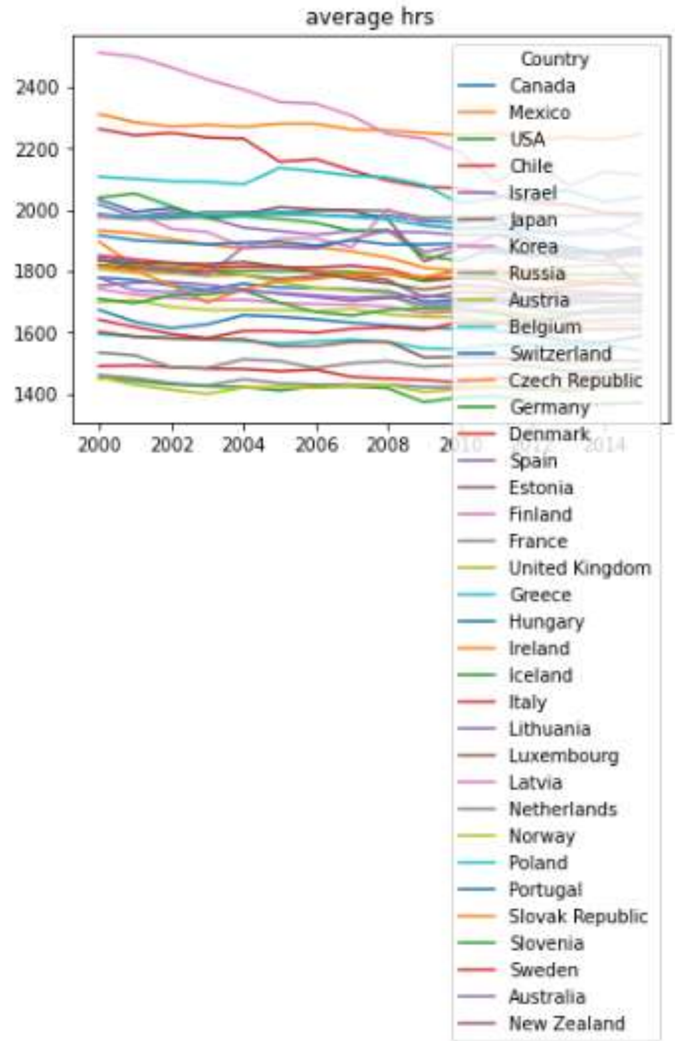
world = result.append([asia,europe,south_pacific])
world.index

```
In [23]: world = result.append([asia,europe,south_pacific])
         world.index
```

```
Out[23]: Index(['Canada', 'Mexico', 'USA', 'Chile', 'Israel', 'Japan', 'Korea',
               'Russia', 'Austria', 'Belgium', 'Switzerland', 'Czech Republic',
               'Germany', 'Denmark', 'Spain', 'Estonia', 'Finland', 'France',
               'United Kingdom', 'Greece', 'Hungary', 'Ireland', 'Iceland', 'Italy',
               'Lithuania', 'Luxembourg', 'Latvia', 'Netherlands', 'Norway', 'Poland',
               'Portugal', 'Slovak Republic', 'Slovenia', 'Sweden', 'Australia',
               'New Zealand'],
              dtype='object', name='Country')
```

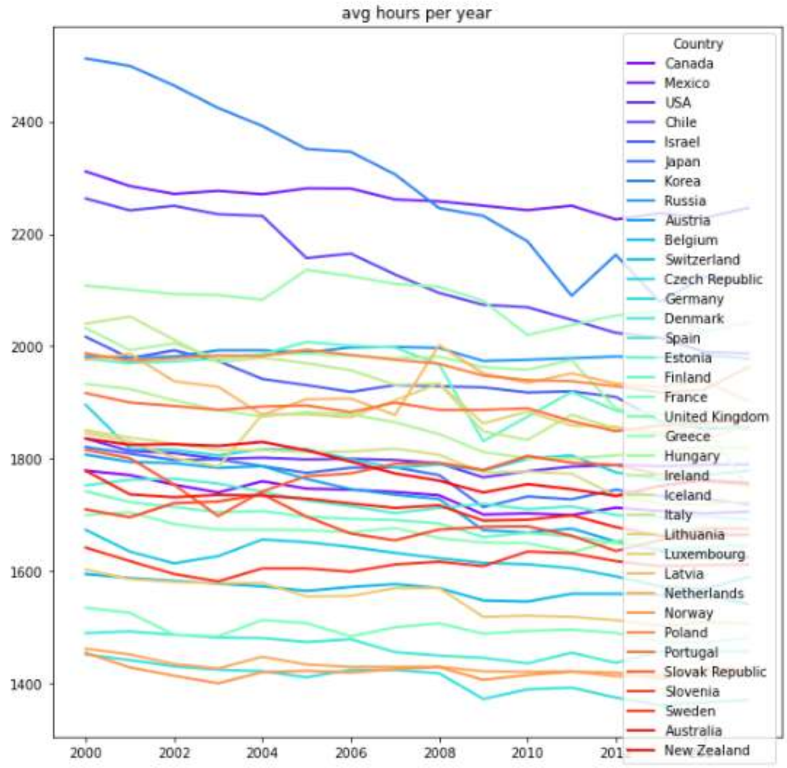
world.transpose().plot(title = 'average hrs')
plt.show()

```
In [24]: world.transpose().plot(title = 'average hrs')
         plt.show()
```



world.transpose().plot(figsize=(10,10),colormap = 'rainbow',linewidth = 2,title = 'avg hours per year')
plt.show

```
In [25]: world.transpose().plot(figsize=(10,10),colormap = 'rainbow',linewidth = 2,title = 'avg hours per year')
         plt.show

Out[25]: <function matplotlib.pyplot.show(close=None, block=None)>
```



historical = pd.read_csv('historical.csv',index_col = 0)
historical.head()

```
In [27]: historical = pd.read_csv('historical.csv',index_col = 0)
         historical.head()

Out[27]:
```

	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	...	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
Country																					
Australia	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	1779.5	1774.90	1773.70	1786.50	1797.60	1793.400	1782.700	1783.600	1768.40	1778.8
Austria	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	1619.200	1637.150	1648.500	1641.65	1654.0
Belgium	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	1662.9	1625.79	1602.72	1558.59	1558.59	1515.835	1500.295	1510.315	1513.33	1514.5
Canada	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	1789.5	1767.50	1766.00	1764.50	1773.00	1771.500	1786.500	1782.500	1778.50	1778.5
Switzerland	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	1673.10	1684.80	1685.80	1706.20	1685.500	1658.900	1648.600	1656.60	1678.4

5 rows × 50 columns

print('world rows,columns',world.shape)
print('historical rows and columns',historical.shape)

```
In [28]: print('world rows,columns',world.shape)
         print('historical rows and columns',historical.shape)

         world rows,columns (36, 16)
         historical rows and columns (39, 50)
```

world_historical = pd.merge(historical,world,left_index = True,right_index = True,how = 'right')
print(world_historical.shape)
world_historical.head()

```
In [31]: world_historical = pd.merge(historical,world,left_index = True,right_index = True,how = 'right')
         print(world_historical.shape)
         world_historical.head()

         (36, 66)
```

Out[31]:

	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	...	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
Country																					
Canada	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	1745.0	1741.0	1735.0	1701.0	1703.0	1700.0	1713.0	1707.0	1703.0	1706.0
Mexico	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	2280.6	2261.4	2258.0	2250.2	2242.4	2250.2	2225.8	2236.6	2228.4	2246.4
USA	1960.0	1975.5	1978.0	1980.0	1970.5	1992.5	1990.0	1962.0	1936.5	1947.0	...	1800.0	1798.0	1792.0	1767.0	1778.0	1786.0	1789.0	1787.0	1789.0	1790.0
Chile	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	2165.0	2128.0	2095.0	2074.0	2069.6	2047.4	2024.0	2015.3	1990.1	1987.5
Israel	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	1919.0	1931.0	1929.0	1927.0	1918.0	1920.0	1910.0	1867.0	1853.0	1858.0

5 rows × 66 columns

world_historical = historical.join(world,how = 'right')
world_historical.head()

```
In [32]: world_historical = historical.join(world,how = 'right')
         world_historical.head()
```

Out[32]:

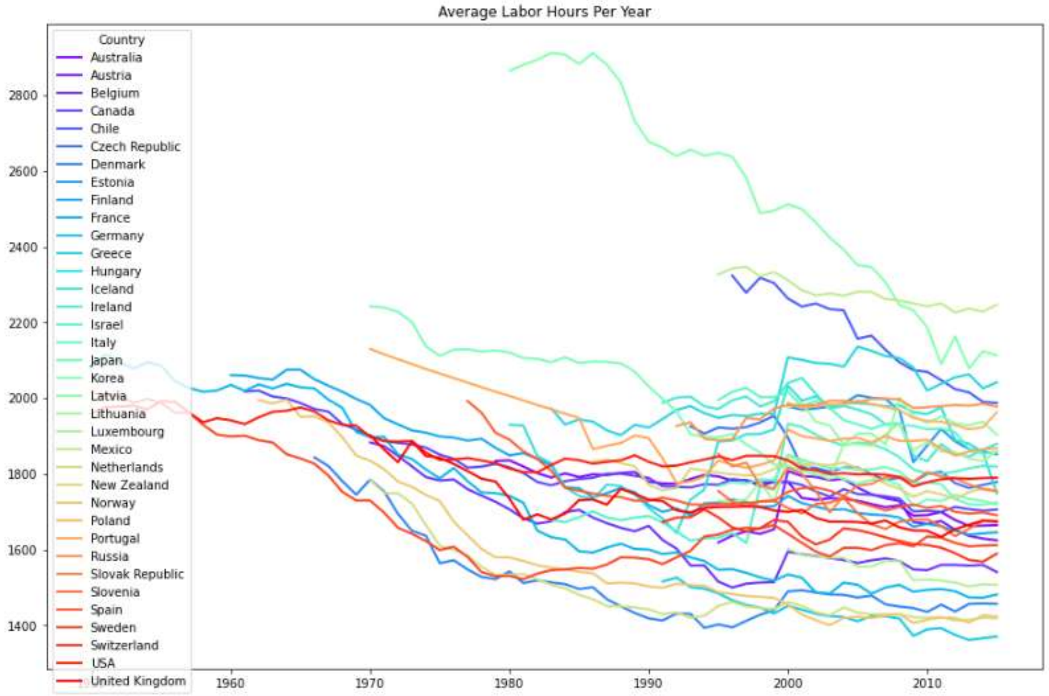
	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	...	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
Country																					
Canada	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	1745.0	1741.0	1735.0	1701.0	1703.0	1700.0	1713.0	1707.0	1703.0	1706.0
Mexico	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	2280.6	2261.4	2258.0	2250.2	2242.4	2250.2	2225.8	2236.6	2228.4	2246.4
USA	1960.0	1975.5	1978.0	1980.0	1970.5	1992.5	1990.0	1962.0	1936.5	1947.0	...	1800.0	1798.0	1792.0	1767.0	1778.0	1786.0	1789.0	1787.0	1789.0	1790.0
Chile	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	2165.0	2128.0	2095.0	2074.0	2069.6	2047.4	2024.0	2015.3	1990.1	1987.5
Israel	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	1919.0	1931.0	1929.0	1927.0	1918.0	1920.0	1910.0	1867.0	1853.0	1858.0

5 rows × 66 columns

world_historical.sort_index(inplace = True)

world_historical.transpose().plot(figsize = (15,10),colormap = 'rainbow',linewidth = 2,title = 'Average Labor Hours Per Year')

```
In [33]: orical.sort_index(inplace = True)
         orical.transpose().plot(figsize = (15,10),colormap = 'rainbow',linewidth = 2,title = 'Average Labor Hours Per Year')

Out[33]: <AxesSubplot:title={'center':'Average Labor Hours Per Year'}>
```



Conclusion: Combining data from multiple sources was implemented successfully.

Practical 05

Date:06/03/2023

Aim: Web scraping

Data set :
url = “https://www.ratemyprofessors.com/ShowRatings.jsp?tid=1986099

code:

```
import requests
from bs4 import BeautifulSoup
url='https://www.ratemyprofessors.com/professor?tid=1986099'
page=requests.get(url)
page
soup=BeautifulSoup(page.text,"html.parser")
proftag=soup.findAll("span",)

print(proftag)
```

```
In [15]: print(proftag)
```

```
[<span class="TeacherSchoolToggleButton__IconWrapper-sc-15dbb0q-4 kljOpt"> <svg fill="none" height="21" viewbox="0
0 18 21" width="18" xmlns="http://www.w3.org/2000/svg"><path clip-rule="evenodd" d="M9.67889 4.14964C10.078 2.0159
11.9509 0.400024 14.2004 0.400024C14.7527 0.400024 15.2004 0.84774 15.2004 1.40002C15.2004 3.03165 14.3507 4.46501
13.0696 5.28168C15.6839 5.77973 17.6004 8.4971 17.6004 11.8C17.6004 15.7456 14.7573 20.8 11.8004 20.8C10.5678 20.8
9.8655 20.6127 9.3257 20.1809C9.2611 20.1292 9.20071 20.076 9.13303 20.0151C9.12474 20.0078 9.12021 20.0028 9.1171
8 19.9994L9.11717 19.9994C9.11505 19.9971 9.11367 19.9955 9.11225 19.9946L9.00039 20C8.94042 20 8.91449 19.9963 8.
90181 19.9944C8.89392 19.9933 8.89116 19.9929 8.88853 19.9946L8.86775 20.0151C8.80008 20.076 8.73968 20.1292 8.675
09 20.1809C8.13528 20.6127 7.43296 20.8 6.20039 20.8C3.24351 20.8 0.400391 15.7456 0.400391 11.8C0.400391 8.14065
2.75289 5.20002 5.80039 5.20002C6.73116 5.20002 7.4215 5.30554 8.00039 5.5141V5.40002C8.00039 5.10508 7.77984 4.85
504 7.49683 4.80798L7.40039 4.80002H5.80039C5.24811 4.80002 4.80039 4.35231 4.80039 3.80002C4.80039 3.28719 5.1864
3 2.86452 5.68377 2.80675L5.80039 2.80002H7.40039C8.37994 2.80002 9.23574 3.3462 9.67889 4.14964ZM8.05318 7.76945C
8.35477 7.92025 8.65108 8.00002 9.00039 8.00002C9.3497 8.00002 9.64601 7.92025 9.9476 7.76945L10.4476 7.49445C10.8
269 7.30481 11.3159 7.20002 12.2004 7.20002C14.0329 7.20002 15.6004 9.1594 15.6004 11.8C15.6004 14.7878 13.3435 1
8.8 11.8004 18.8L11.4603 18.795C10.953 18.7785 10.731 18.7223 10.6089 18.6435L10.5253 18.5768L10.4535 18.513C10.05
57 18.1632 9.6465 18 9.00039 18C8.35428 18 7.94506 18.1632 7.54724 18.513L7.4257 18.6192C7.2905 18.7273 7.01782 1
8.8 6.20039 18.8C4.65727 18.8 2.40039 14.7878 2.40039 11.8C2.40039 9.1594 3.96789 7.20002 5.80039 7.20002C6.68489
7.20002 7.17389 7.30481 7.55318 7.49445L8.05318 7.76945Z" data-testid="APPLE_PATH_TESTID" fill="#ffffff" fill-rule
="evenodd"></path></svg></span>, <span>Jump To Ratings</span>, <span>Trevor</span>, <span class="NameTitle__LastNa
meWrapper-dowf0z-2 glXOHH">Tomesh<!-- --> <button class="TeacherBookmark__StyledTeacherBookmark-sc-17dr6wh-0 hVjzh
H" type="button"><img alt="Bookmark" data-for="GLOBAL_TOOLTIP" data-tip="Save Professor" data-tooltip="true" src
="/static/media/bookmark-default.b056b070.svg"/></button></span>, <span>Professor in the<!-- --> <a class="Teacher
Department__StyledDepartmentLink-fl79e8-0 fuypDB" href="/search/teachers?query=*&amp;sid=1468&amp;did=11"><b>Compu
ter Science<!-- --> department</b></a> <!-- -->at<!-- --> </span>, <span class="Tag-bs9vf4-0 hHOVKF">Hilarious</sp
an>, <span class="Tag-bs9vf4-0 hHOVKF">Accessible outside class</span>, <span class="Tag-bs9vf4-0 hHOVKF">Amazing
lectures </span>, <span class="Tag-bs9vf4-0 hHOVKF">Caring</span>, <span class="Tag-bs9vf4-0 hHOVKF">Clear grading
criteria</span>, <span class="SimilarProfessorListItem__TeacherScoreSpan-x7cr0c-0 eLbzKR">4.75</span>, <span class
="SimilarProfessorListItem__TeacherNameSpan-x7cr0c-1 gMAFfH">Howard Hamilton</span>, <span class="SimilarProfessor
ListItem__TeacherScoreSpan-x7cr0c-0 eLbzKR">4.75</span>, <span class="SimilarProfessorListItem__TeacherNameSpan-x7
cr0c-1 gMAFfH">Richard Dosselmann</span>, <span class="SimilarProfessorListItem__TeacherScoreSpan-x7cr0c-0 eLbzK
R">4.70</span>, <span class="SimilarProfessorListItem__TeacherNameSpan-x7cr0c-1 gMAFfH">Catherine Song</span>, <sp
an aria-label="Sunglasses" role="img">😎</span>, <span aria-label="Sunglasses" role="img">😎</span>, <span>Yes</sp
```

for mytag in proftag:
    print(mytag.get_text())

```
In [19]: for mytag in proftag:
             print(mytag.get_text())

         Jump To Ratings
         Trevor
         Tomesh
         Professor in the Computer Science department at
         Hilarious
         Accessible outside class
         Amazing lectures
         Caring
         Clear grading criteria
         4.75
         Howard Hamilton
         4.75
         Richard Dosselmann
         4.70
         Catherine Song
         😎
         😎
         Yes
```

```
In [20]: soup.title
Out[20]: <title data-react-helmet="true">Trevor Tomesh at University of Regina - RateMyProfessors.com</title>
```

soup.title


```
In [20]: soup.title
Out[20]: <title data-react-helmet="true">Trevor Tomesh at University of Regina - RateMyProfessors.com</title>
```


page.text

```
In [21]: page.text
Out[21]: '\n    <!DOCTYPE html>\n    <!-- SSR -->\n    <html >\n      <head>\n        <meta name="viewport" content="widt
h=device-width, initial-scale=1" />\n        <meta name="theme-color" content="#000000" />\n        \n\n
<link rel="manifest" href="/build/manifest.json">\n        <link rel="stylesheet" type="text/css" href="/static/
css/main.1773c5b7.css">\n\n        <!-- Google Optimize Anti-flicker snippet -->\n        <style>.async-hide { o
pacity: 0 !important} </style>\n        <script>(function(a,s,y,n,c,h,i,d,e){s.className+=\' \'+y;h.start=1*new
Date;\n        h.end=i=function(){s.className=s.className.replace(RegExp(\' ?\'+y),\'\')};\n        (a[n]=a[n]||
[]).hide=h;setTimeout(function(){i();h.end=null},c);h.timeout=c;\n        })(window,document.documentElement,\'a
sync-hide\',\'dataLayer\',4000,\n        {\'OPT-MLW3VTZ\':true});</script>\n\n        <!-- Google Optimize -->\n
<script async src="https://www.googleoptimize.com/optimize.js?id=OPT-MLW3VTZ"></script>\n\n        <script src
="https://cdn.browsiprod.com/bootstrap/bootstrap.js" id="browsi-tag" data-pubKey="alticermp" data-siteKey="rmp1"
async></script>\n        <script src="//ads.ratemyprofessors.com/adBundle.js?v=5"></script>\n        \n        <
link data-react-helmet="true" rel="icon" href="/favicons/favicon-16.png" sizes/><link data-react-helmet="true" r
el="icon" href="/favicons/favicon-32.png" sizes="32x32"/><link data-react-helmet="true" rel="apple-touch-icon" h
ref="/favicons/favicon-57.png" sizes/><link data-react-helmet="true" rel="apple-touch-icon" href="/favicons/favi
con-48.png" sizes="48x48"/><link data-react-helmet="true" rel="apple-touch-icon" href="/favicons/favicon-72.png"
sizes="72x72"/><link data-react-helmet="true" rel="apple-touch-icon" href="/favicons/favicon-114.png" sizes="114
x114"/><link data-react-helmet="true" rel="apple-touch-icon" href="/favicons/favicon-152.png" sizes="152x152"/><
link data-react-helmet="true" rel="apple-touch-icon-precomposed" href="/favicons/favicon-196.png" sizes/><link d
ata-react-helmet="true" rel="canonical" href="https://www.ratemyprofessors.com/professor?tid=1986099"/>\n<title

In [ ]:
```

Conclusion : Web scraping was implemented successfully.

Practical 06

Date:20/03/2023

Aim: implementation of K-means algorithm

code:

from pandas import DataFrame

Data = {'x':

[25,34,22,27,33,33,31,22,35,34,67,54,57,43,50,57,59,52,65,47,49,48,35,33,44,45,38,43,51,46],
'y':

[79,51,53,78,59,74,73,57,69,75,51,32,40,47,53,36,35,58,59,50,25,20,14,12,20,5,29,27,8,7]
}

df = DataFrame(Data,columns=['x','y'])
print(df)

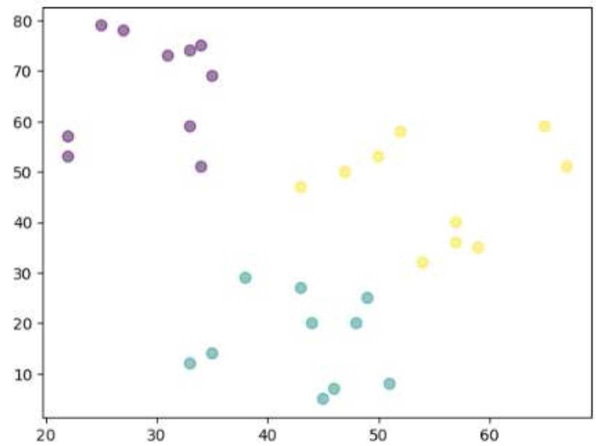
```
In [3]: df = DataFrame(Data,columns=['x','y'])
        print(df)

     x   y
0   25  79
1   34  51
2   22  53
3   27  78
4   33  59
5   33  74
6   31  73
7   22  57
8   35  69
9   34  75
10  67  51
11  54  32
12  57  40
13  43  47
14  50  53
15  57  36
16  59  35
17  52  58
18  65  59
19  47  50
20  49  25
21  48  20
22  35  14
23  33  12
24  44  20
25  45   5
26  38  29
27  43  27
28  51   8
29  46   7
```

```
import matplotlib.pyplot as plt
from sklearn.cluster import Kmeans

kmeans = Kmeans(n_clusters=3).fit(df)

centroids = Kmeans.cluster_centers_

print(centroids)
```

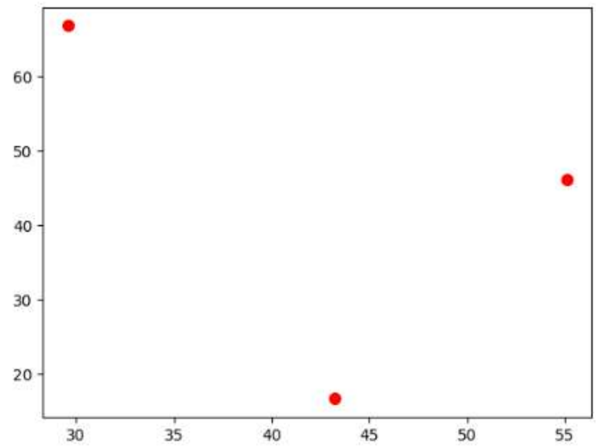
```
In [16]: print(centroids)

[[29.6 66.8]
 [43.2 16.7]
 [55.1 46.1]]
```

```
plt.scatter(df['x'],df['y'],c=kmeans.labels_.astype(float),s=50,alpha=0.5)
plt.scatter(centroids[:,0],centroids[:,1],c='red',s=50)
```

```
In [17]: plt.scatter(df['x'],df['y'],c=kmeans.labels_.astype(float),s=50,alpha=0.5)
Out[17]: <matplotlib.collections.PathCollection at 0x7f97403e88e0>
```

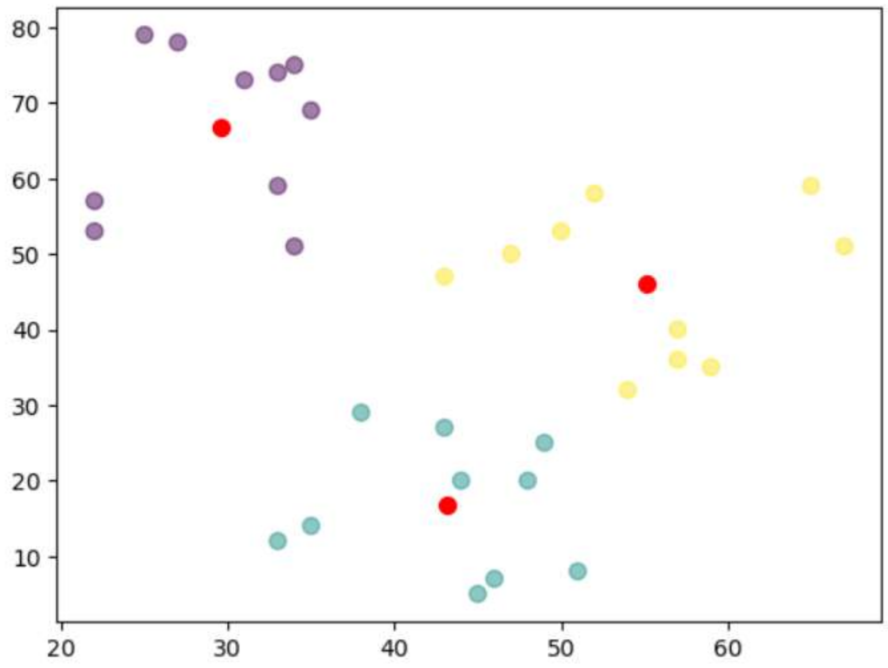



```
In [12]: plt.scatter(centroids[:,0],centroids[:,1],c='red',s=50)
Out[12]: <matplotlib.collections.PathCollection at 0x7f97405559a0>
```



plt.scatter(df['x'],df['y'],c=kmeans.labels_.astype(float),s=50,alpha=0.5)
plt.scatter(centroids[:,0],centroids[:,1],c='red',s=50)

```
In [18]: plt.scatter(df['x'],df['y'],c=kmeans.labels_.astype(float),s=50,alpha=0.5)
         plt.scatter(centroids[:,0],centroids[:,1],c='red',s=50)

Out[18]: <matplotlib.collections.PathCollection at 0x7f974036a3d0>
```



Conclusion : K – Means Algorithm was implemented successfully.

Practical 07

Date:03/04/2023

Aim: implementation of classification algorithm – decision tree

code:

```
import pandas as pd

golf_df = pd.DataFrame()

golf_df['Outlook'] =
['sunny','sunny','overcast','rainy','rainy','rainy','overcast','sunny','sunny','rain
y','sunny','overcast','overcast','rainy']

golf_df['Temperature'] =
['hot','hot','hot','cool','cool','cool','mild','cool','mild','mild','mild','mild','hot','
mild']

golf_df['Humidity'] =
['high','high','high','high','normal','normal','normal','high','normal','normal','
normal','high','normal','high']

golf_df['windy'] =
['false','true','false','false','false','true','true','false','false','false','true','true','fal
se','true']

golf_df['Play'] =
['no','no','yes','yes','yes','no','yes','no','yes','yes','yes','yes','yes','no']
```

print(golf_df)

```
In [57]: print(golf_df)
      Outlook Temperature Humidity  windy Play
0       sunny         hot     high  false   no
1       sunny         hot     high   true   no
2    overcast         hot     high  false  yes
3       rainy        cool     high  false  yes
4       rainy        cool   normal  false  yes
5       rainy        cool   normal   true   no
6    overcast        mild   normal   true  yes
7       sunny        cool     high  false   no
8       sunny        mild   normal  false  yes
9       rainy        mild   normal  false  yes
10      sunny        mild   normal   true  yes
11   overcast        mild     high   true  yes
12   overcast         hot   normal  false  yes
13      rainy        mild     high   true   no
```

one_hot_data

```
In [59]: one_hot_data
Out[59]:
```

	Outlook_overcast	Outlook_rainy	Outlook_sunny	Temperature_cool	Temperature_hot	Temperature_mild	Humidity_high	Humidity_normal	Play_no	Play_
0	0	0	1	0	1	0	1	0	1	
1	0	0	1	0	1	0	1	0	1	
2	1	0	0	0	1	0	1	0	0	
3	0	1	0	1	0	0	1	0	0	
4	0	1	0	1	0	0	0	1	0	
5	0	1	0	1	0	0	0	1	1	
6	1	0	0	0	0	1	0	1	0	
7	0	0	1	1	0	0	1	0	1	
8	0	0	1	0	0	1	0	1	0	
9	0	1	0	0	0	1	0	1	0	
10	0	0	1	0	0	1	0	1	0	
11	1	0	0	0	0	1	1	0	0	
12	1	0	0	0	1	0	0	1	0	
13	0	1	0	0	0	1	1	0	1	

one_hot_data=pd.get_dummies(golf_df[['Outlook','Temperature','Humidity','windy']])

from sklearn import tree

clf=tree.DecisionTreeClassifier()

clf_train=clf.fit(one_hot_data,golf_df['Play'])

print(tree.export_graphviz(clf_train,None))

```
In [75]:
print(tree.export_graphviz(clf_train,None))

digraph Tree {
node [shape=box, fontname="helvetica"] ;
edge [fontname="helvetica"] ;
0 [label="X[0] <= 0.5\ngini = 0.459\nsamples = 14\nvalue = [5, 9]"] ;
1 [label="X[7] <= 0.5\ngini = 0.5\nsamples = 10\nvalue = [5, 5]"] ;
0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"] ;
2 [label="X[3] <= 0.5\ngini = 0.32\nsamples = 5\nvalue = [4, 1]"] ;
1 -> 2 ;
3 [label="gini = 0.0\nsamples = 3\nvalue = [3, 0]"] ;
2 -> 3 ;
4 [label="X[1] <= 0.5\ngini = 0.5\nsamples = 2\nvalue = [1, 1]"] ;
2 -> 4 ;
5 [label="gini = 0.0\nsamples = 1\nvalue = [1, 0]"] ;
4 -> 5 ;
6 [label="gini = 0.0\nsamples = 1\nvalue = [0, 1]"] ;
4 -> 6 ;
7 [label="X[5] <= 0.5\ngini = 0.32\nsamples = 5\nvalue = [1, 4]"] ;
1 -> 7 ;
8 [label="X[9] <= 0.5\ngini = 0.5\nsamples = 2\nvalue = [1, 1]"] ;
7 -> 8 ;
9 [label="gini = 0.0\nsamples = 1\nvalue = [0, 1]"] ;
8 -> 9 ;
10 [label="gini = 0.0\nsamples = 1\nvalue = [1, 0]"] ;
8 -> 10 ;
11 [label="gini = 0.0\nsamples = 3\nvalue = [0, 3]"] ;
7 -> 11 ;
12 [label="gini = 0.0\nsamples = 4\nvalue = [0, 4]"] ;
0 -> 12 [labeldistance=2.5, labelangle=-45, headlabel="False"] ;
}
```

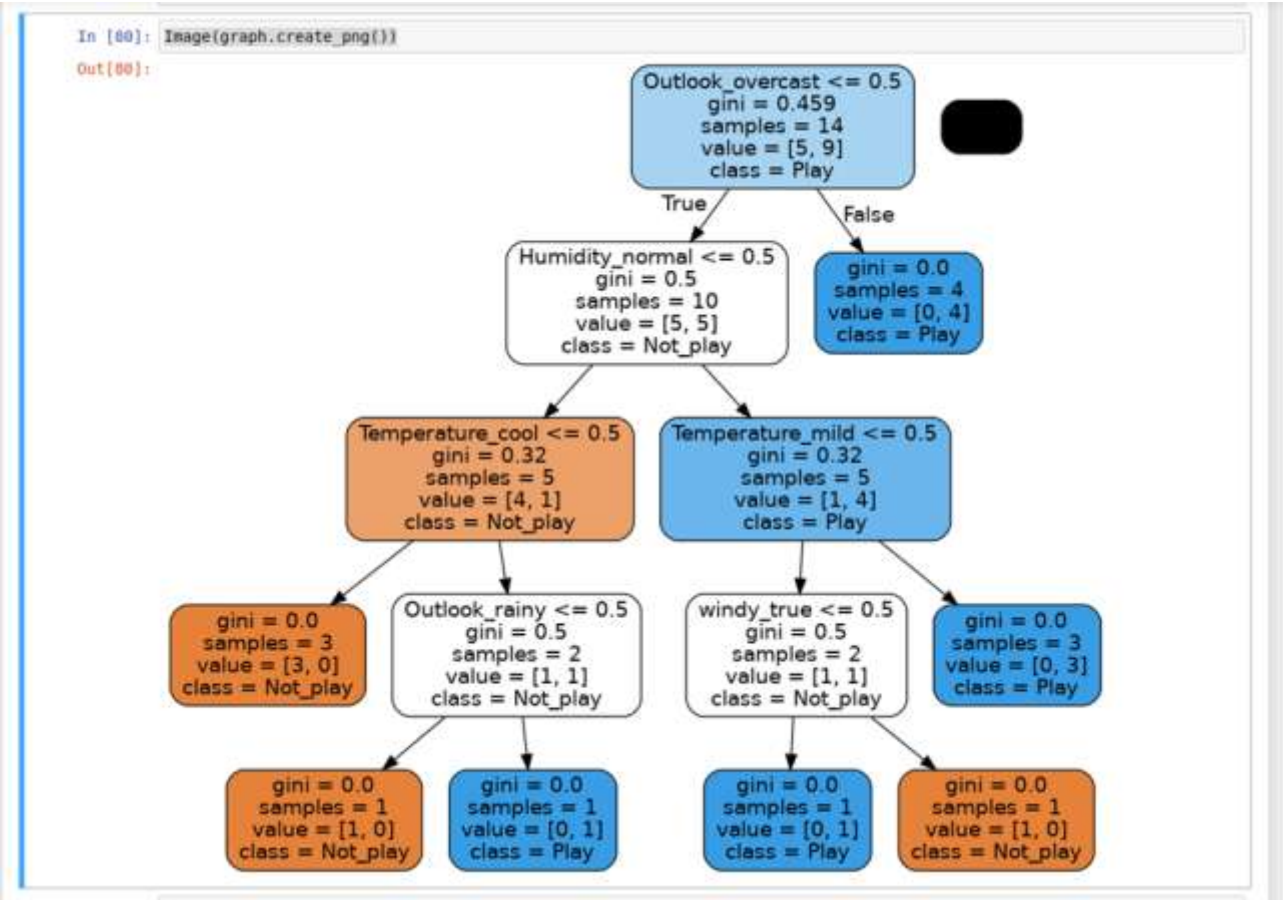
---

import pydotplus

dot_data=tree.export_graphviz(clf_train,out_file=None,feature_names=list(one_hot_data.columns.values),class_names=['Not_play','Play'],rounded=True,filled=True) graph=pydotplus.graph_from_dot_data(dot_data)

from IPython.display import Image

Image(graph.create_png())



Conclusion : Decision tree algorithm was implemented successfully.

Practical 08

Date:10/04/2023

Aim: Data visualization using box plot, scatter plot, heat maps, histogram

code:

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inlineplt.title("Number fo cars by make")

sns.set(color_codes=True)

df=pd.read_csv("data.csv")

#to display the top 5 row
df.head(5)

```
In [9]: #to display the top 5 row
        df.head(5)
Out[9]:
```

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size	Vehicle Style	highway MPG	city mpg	Popularity	MSRP
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Factory Tuner,Luxury,High-Performance	Compact	Coupe	26	19	3916	46135
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Convertible	28	19	3916	40650
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,High-Performance	Compact	Coupe	28	20	3916	36350
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Coupe	28	18	3916	29450
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury	Compact	Convertible	28	18	3916	34500

df=df.rename(columns={"Engine HP":"HP","Engine Cylinders":"Cyliinders","Transmission Type":"Transmission","Driven_Wheels":"Drive MOde","highway MPG":"MPG-H","city mpg":"MPG-C","MSRP":"Price"})

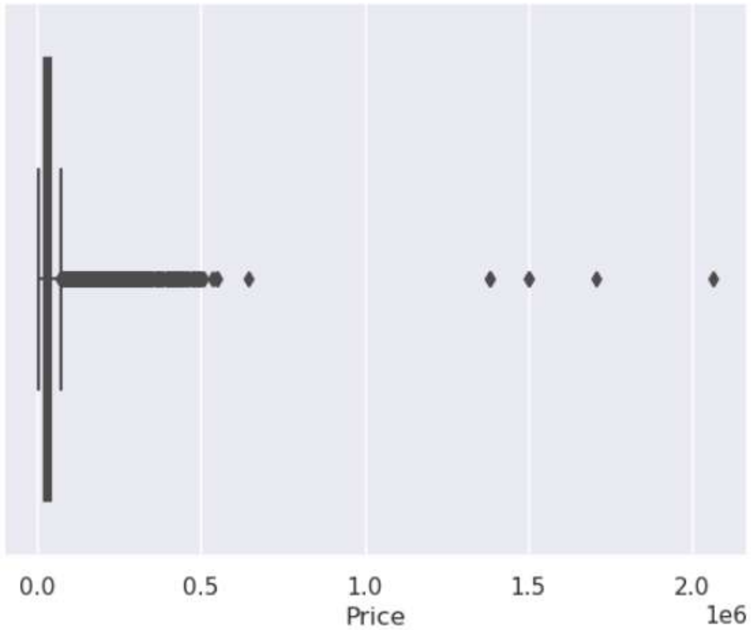
df.head(5)

```
In [10]: df=df.rename(columns={"Engine HP":"HP","Engine Cylinders":"Cyliinders","Transmission Type":"Transmission","Driven_W

In [11]: df.head(5)
Out[11]:
```

	Make	Model	Year	Engine Fuel Type	HP	Cylinders	Transmission	Drive MOde	Number of Doors	Market Category	Vehicle Size	Vehicle Style	MPG-H	MPG-C	Popularity	Price
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Factory Tuner,Luxury,High-Performance	Compact	Coupe	26	19	3916	46135
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Convertible	28	19	3916	40650
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,High-Performance	Compact	Coupe	28	20	3916	36350
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Coupe	28	18	3916	29450
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury	Compact	Convertible	28	18	3916	34500

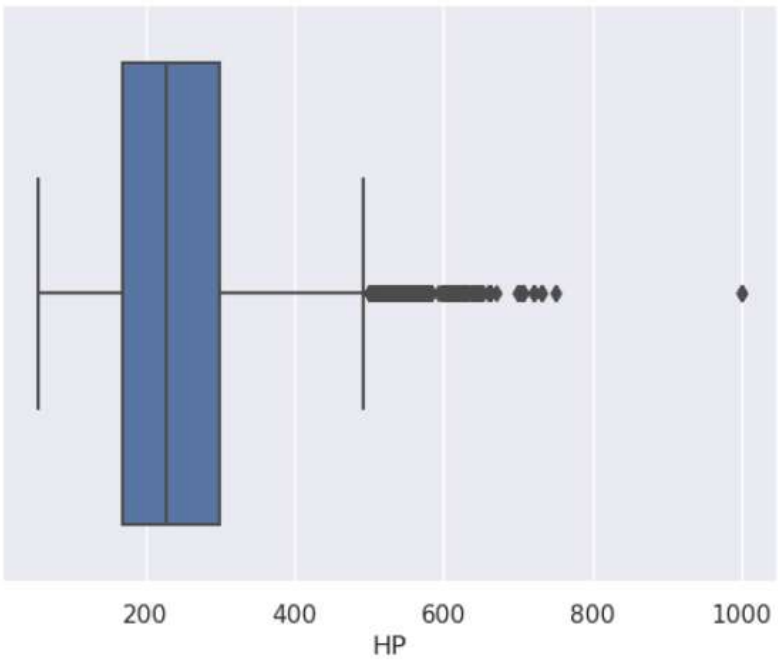
sns.boxplot(x=df['Price'])

```
In [13]: sns.boxplot(x=df['Price'])
Out[13]: <AxesSubplot:xlabel='Price'>
```



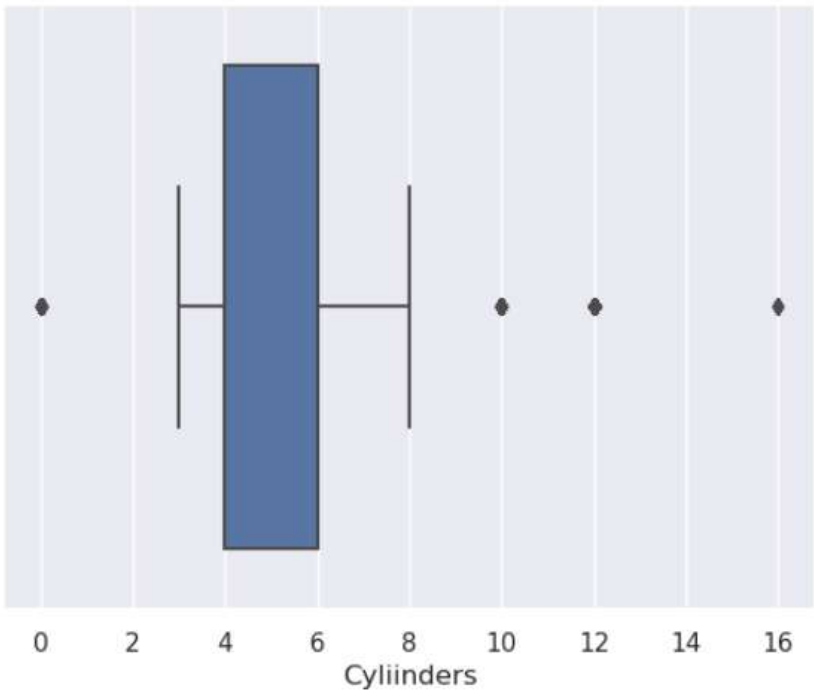
sns.boxplot(x=df['HP'])

```
In [14]: sns.boxplot(x=df['HP'])
Out[14]: <AxesSubplot:xlabel='HP'>
```



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

```
In [16]: sns.boxplot(x=df['Cyliinders'])
Out[16]: <AxesSubplot:xlabel='Cyliinders'>
```



```
Q1 = df.quantile(0.25)

Q3 = df.quantile(0.75)

IQR = Q3 – Q1

print(IQR)
```

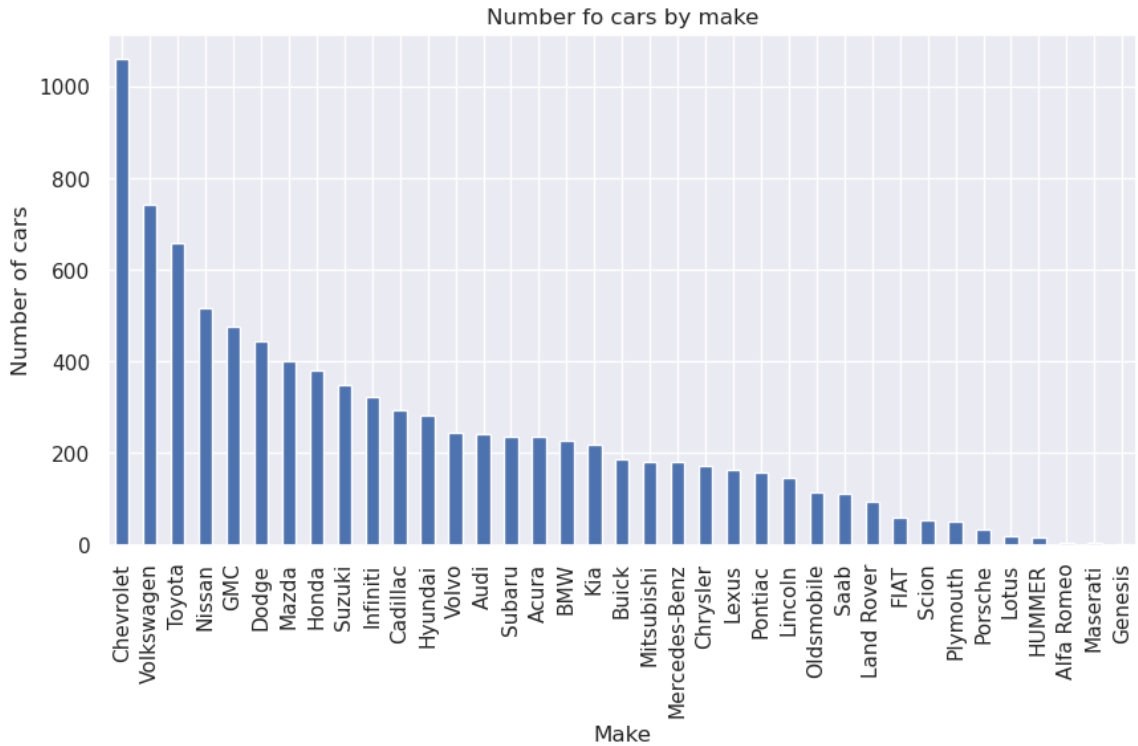
```
In [20]: print(IQR)

Year                    9.00
HP                    130.00
Cyliinders              2.00
Number of Doors         2.00
MPG-H                   8.00
MPG-C                   6.00
Popularity           1460.00
Price               21231.25
dtype: float64
```

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

df.shape
```

```
In [22]: df.shape
Out[22]: (9084, 16)
```

df.Make.value_counts().nlargest(40).plot(kind='bar',figsize=(10,5))
plt.title("Number fo cars by make")
plt.ylabel('Number of cars')
plt.xlabel('Make')

```
In [26]: df.Make.value_counts().nlargest(40).plot(kind='bar',figsize=(10,5))
         plt.title("Number fo cars by make")
         plt.ylabel('Number of cars')
         plt.xlabel('Make')

Out[26]: Text(0.5, 0, 'Make')
```

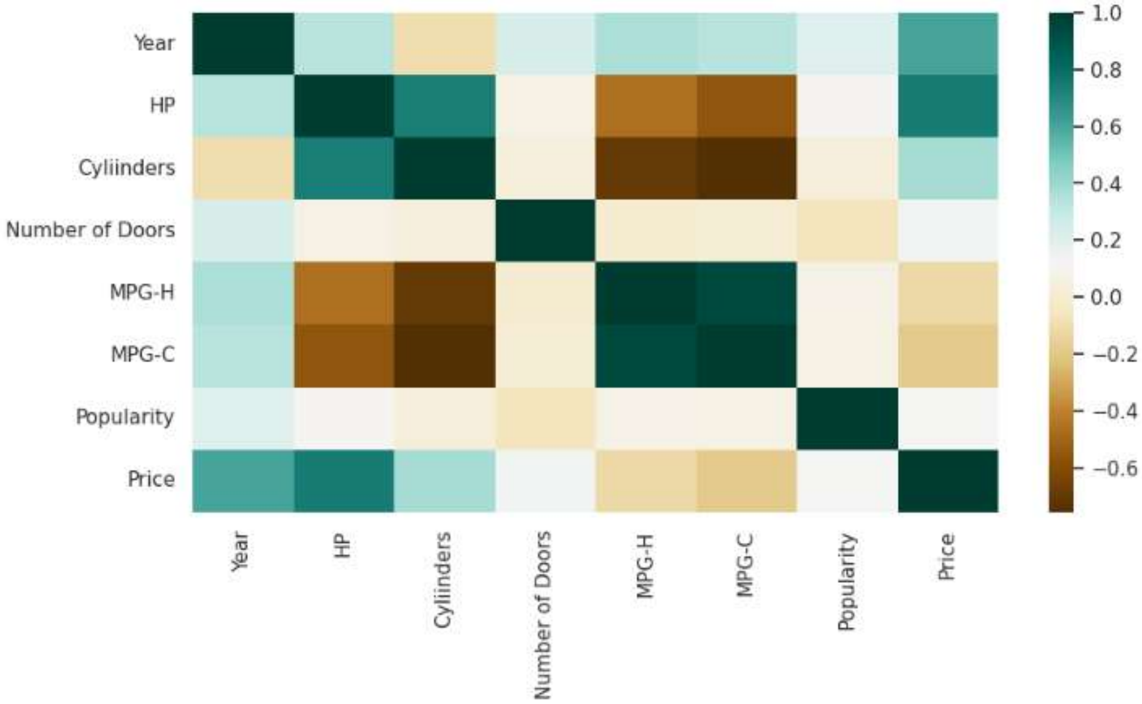


plt.figure(figsize=(10,5))
c=df.corr()
sns.heatmap(c,cmap="BrBG",annot=False)
c

In [39]:
```
plt.figure(figsize=(10,5))
c=df.corr()
sns.heatmap(c,cmap="BrBG",annot=False)
c
```

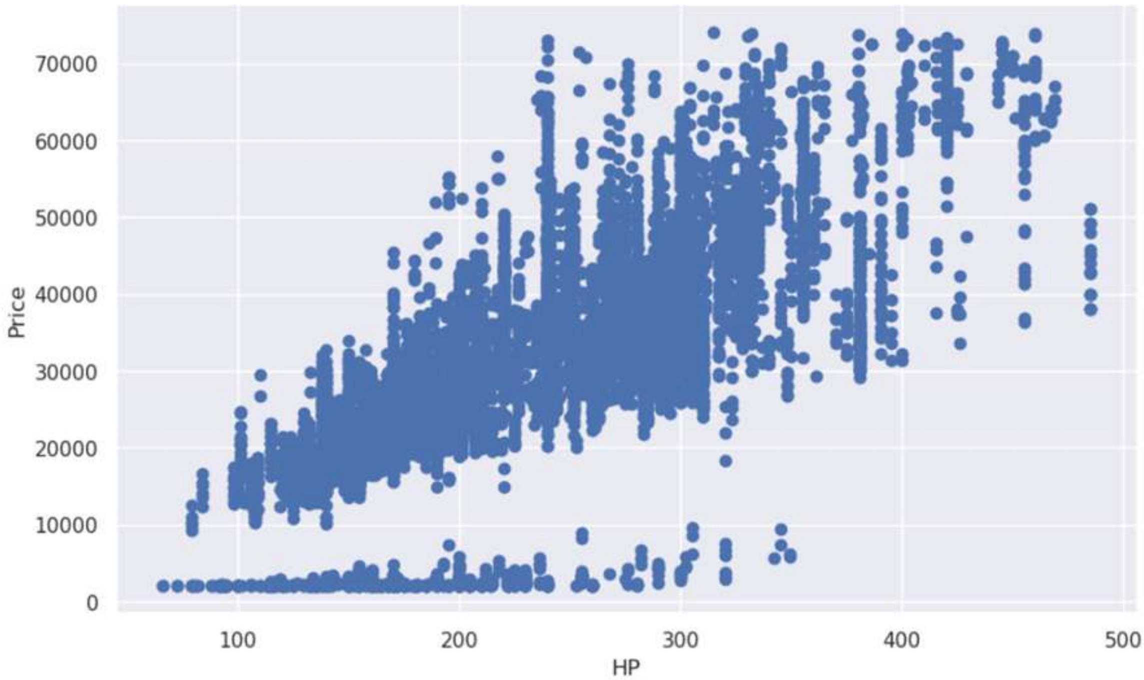
Out[39]:

	Year	HP	Cyliinders	Number of Doors	MPG-H	MPG-C	Popularity	Price
Year	1.000000	0.330647	-0.103458	0.233774	0.362741	0.328945	0.202637	0.606315
HP	0.330647	1.000000	0.737088	0.061315	-0.467352	-0.561886	0.109569	0.740117
Cyliinders	-0.103458	0.737088	1.000000	0.037569	-0.705089	-0.755840	0.029992	0.379463
Number of Doors	0.233774	0.061315	0.037569	1.000000	-0.005238	0.007052	-0.066323	0.142302
MPG-H	0.362741	-0.467352	-0.705089	-0.005238	1.000000	0.938849	0.073455	-0.120786
MPG-C	0.328945	-0.561886	-0.755840	0.007052	0.938849	1.000000	0.064806	-0.192441
Popularity	0.202637	0.109569	0.029992	-0.066323	0.073455	0.064806	1.000000	0.118664
Price	0.606315	0.740117	0.379463	0.142302	-0.120786	-0.192441	0.118664	1.000000



```
fig,ax = plt.subplots(figsize=(10,6))
ax.scatter(df['HP'],df['Price'])
ax.set_xlabel('HP')
ax.set_ylabel('Price')
plt.show()
```

```
In [40]: fig,ax = plt.subplots(figsize=(10,6))
         ax.scatter(df['HP'],df['Price'])
         ax.set_xlabel('HP')
         ax.set_ylabel('Price')
         plt.show()
```



Conclusion : Data visualization using box plot, scatter plot, heat maps, histogram was implemented successfully.