INDEX

Sr.No	Aim	Date	Sign
1	Write a python program to plot word cloud for a Wikipedia page of any topic	18/12/23	
2	Write a python program to perform Web Scrapping 01.Html scrapping- use Beautiful Soup 02.json scrapping	 18/12/23 1/1/24	
3	Perform Exploratory Data Analysis(EDA) of mtcars.csv in R	1/1/24	
4	Exploratory data analysis in Python using Titanic Dataset	15/1/24	

5	1)Write a python program to build a regression model that could predict the salary of an employee from the given experience and visualize univariate linear regression on it. 2) Write a python program to simulate linear model Y=10+7*x+e for random 100 samples and visualize univariate linear regression on it.	5/2/24 5/2/24	
6	Write a python program to implement multiple linear regression on the Dataset Boston.csv	19/2/24	
7	Write a python program to implement KNN algorithm to predict breast cancer using breast cancer wisconsin dataset .	19/2/24	
8	Introduction to NOSQL using MongoDB	27/2/24	

Practical No. 1

Aim :- Write a python program to plot word cloud for a wikipedia page of any topic.

```
# Install module wikipedia
!pip install wikipedia
```

```
Collecting wikipedia
  Downloading wikipedia-1.4.0.tar.gz (27 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Requirement already satisfied: beautifulsoup4 in c:\users\admin\anaconda3\lib\site-packages (from wikipedia) (4.11.1)
Requirement already satisfied: requests<3.0.0,>=2.0.0 in c:\users\admin\anaconda3\lib\site-packages (from wikipedia) (2.28.1)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\admin\anaconda3\lib\site-packages (from requests<3.0.0,>=2.
0.0->wikipedia) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\admin\anaconda3\lib\site-packages (from requests<3.0.0,>=2.0.0->w
ikipedia) (2022.9.14)
Requirement already satisfied: idna<4,>=2.5 in c:\users\admin\anaconda3\lib\site-packages (from requests<3.0.0,>=2.0.0->wikiped
ia) (3.3)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\admin\anaconda3\lib\site-packages (from requests<3.0.0,>=2.0.0
->wikipedia) (1.26.11)
Requirement already satisfied: soupsieve>1.2 in c:\users\admin\anaconda3\lib\site-packages (from beautifulsoup4->wikipedia) (2.
3.1)
Building wheels for collected packages: wikipedia
  Building wheel for wikipedia (setup.py): started
  Building wheel for wikipedia (setup.py): finished with status 'done'
  Created wheel for wikipedia: filename=wikipedia-1.4.0-py3-none-any.whl size=11680 sha256=29e6faaf9decd2c2c12e4104bbf1e9605829
b07f295a2fbdb32daf251b94ade2
  Stored in directory: c:\users\admin\appdata\local\pip\cache\wheels\c2\46\f4\caa1bee71096d7b0cdca2f2a2af45cacf35c5760bee8f0094
8
Successfully built wikipedia
Installing collected packages: wikipedia
Successfully installed wikipedia-1.4.0
```

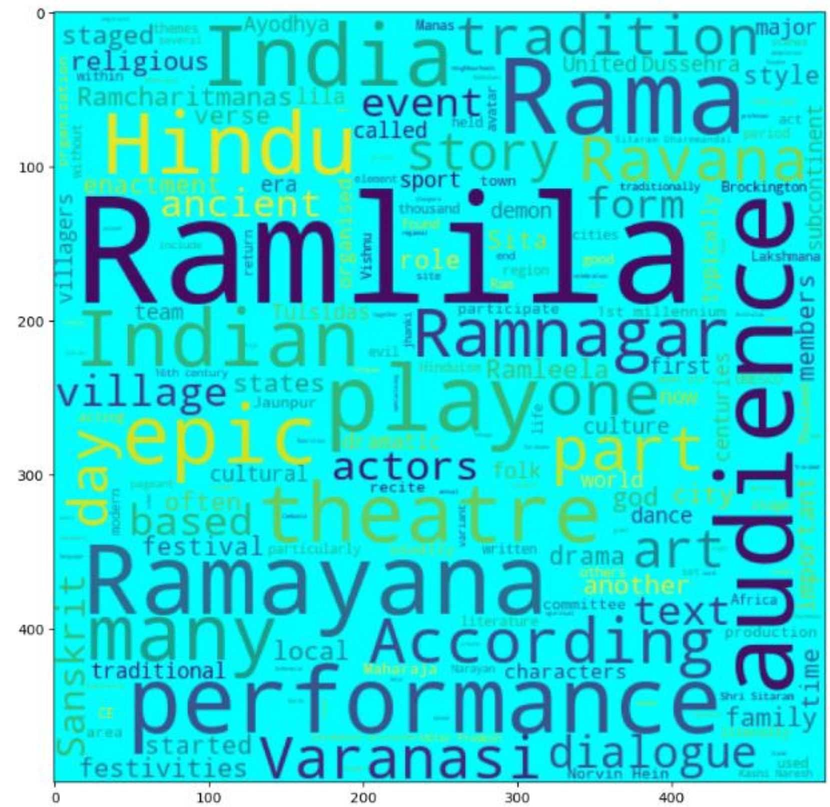
```
#Install module wordcloud
!pip install wordcloud
```

```
Collecting wordcloud
  Downloading wordcloud-1.9.3-cp39-cp39-win_amd64.whl (300 kB)
     -------------------------------------- 300.6/300.6 kB 6.2 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.6.1 in c:\users\admin\anaconda3\lib\site-packages (from wordcloud) (1.21.5)
Requirement already satisfied: pillow in c:\users\admin\anaconda3\lib\site-packages (from wordcloud) (9.2.0)
Requirement already satisfied: matplotlib in c:\users\admin\anaconda3\lib\site-packages (from wordcloud) (3.5.2)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\admin\anaconda3\lib\site-packages (from matplotlib->wordcloud) (3.
0.9)
Requirement already satisfied: cycler>=0.10 in c:\users\admin\anaconda3\lib\site-packages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\admin\anaconda3\lib\site-packages (from matplotlib->wordcloud)
(2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\admin\anaconda3\lib\site-packages (from matplotlib->wordcloud) (1.
4.2)
Requirement already satisfied: packaging>=20.0 in c:\users\admin\anaconda3\lib\site-packages (from matplotlib->wordcloud) (21.
3)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\admin\anaconda3\lib\site-packages (from matplotlib->wordcloud) (4.
25.0)
Requirement already satisfied: six>=1.5 in c:\users\admin\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib->w
ordcloud) (1.16.0)
Installing collected packages: wordcloud
Successfully installed wordcloud-1.9.3
```

Code :-

```
from wordcloud import STOPWORDS,WordCloud
import matplotlib.pyplot as plt
import wikipedia as wp

result = wp.page("Ramtilak")
final_result =result.content

#Define function for plotting WordCloud
def plot_wordcloud(wc):
  plt.axis("off")
  plt.figure(figsize=(10,10))
  plt.imshow(wc)
  plt.show()

wc = WordCloud(width=500,height=500,background_color="cyan",random_state=10,stopwords=STOPWORDS).generate(final_result)
plot_wordcloud(wc)

print(STOPWORDS)
```

Output :-




```
{'for', 'then', 'could', "didn't", 'off', 'because', 'here', 'been', 'those', 'same', "you've", 'has', 'until', 'did', 'had',
'it', 'your', "i'm", 'by', 'else', 'them', "he'd", 'therefore', 'under', 'being', 'that', 'why', 'my', 'his', "weren't", 'yours
elf', 'further', 'shall', 'otherwise', 'however', "he's", 'own', 'were', "why's", "won't", 'but', 'or', 'since', 'have', 'itsel
f', "you'll", 'into', "doesn't", "he'll", 'if', 'is', 'most', 'before', 'ours', 'against', 'no', 'of', 'her', 'out', 'each', "i
t's", "wasn't", "we'll", 'like', 'are', 'was', 'its', 'we', "hadn't", "we've", 'do', 'an', 'and', "haven't", 'ought', 'me', "sh
an't", "she'll", "couldn't", 'as', 'few', "there's", "don't", 'yourselves', 'from', 'over', 'when', 'on', 'be', 'hers', 'thes
e', 'how', 'should', "you're", 'just', 'he', 'r', 'too', 'after', "i've", "isn't", 'more', 'theirs', 'between', 'myself', 'up',
'at', "you'd", "what's", 'hence', "she'd", 'all', 'doing', "i'll", 'again', 'above', 'ever', 'www', 'http', 'where', 'com', 'wh
at', 'this', "we'd", 'having', "mustn't", 'ourselves', "that's", 'such', "shouldn't", 'any', "who's", 'does', 'to', 'k', 'onc
e', 'i', "aren't", "i'd", "they'd", 'herself', 'also', 'they', 'can', "when's", 'so', 'some', 'only', 'the', "hasn't", 'than',
'through', 'which', 'would', 'with', 'not', "can't", 'about', "she's", 'both', 'while', "how's", 'himself', 'down', "they'll",
'our', "they're", 'there', "we're", 'him', 'below', 'whom', "let's", 'get', "they've", 'other', 'she', 'very', 'yours', 'canno
t', "here's", 'their', 'themselves', 'in', "wouldn't", 'you', 'nor', 'am', "where's", 'during', 'a', 'who'}
```

Practical No. 2

Aim :- Write a python program to perform Web Scrapping

Description :- Web Scrapping :- Web scraping is the process of collecting and parsing raw data from the Web.

2a. HTML Web Scrapping [Use BeautifulSoup]

Code :-

```python
import pandas as pd
from bs4 import BeautifulSoup
from urllib.request import urlopen

url = "https://en.wikipedia.org/wiki/List_of_Asian_countries_by_area"
page = urlopen(url)
html_page = page.read().decode("utf-8")
soup = BeautifulSoup(html_page,"html.parser")
table = soup.find("table")

SrNo = []
Country = []
Area = []
rows = table.find("tbody").find_all("tr")
for row in rows:
    cells = row.find_all("td")
    if(cells):
        SrNo.append(cells[0].get_text().strip("\n"))
        Country.append(cells[1].get_text().strip("\xa0").strip("\n").strip("\[2]*"))
        Area.append(cells[3].get_text().strip("\n").replace(",",""))

countries_df= pd.DataFrame()
countries_df["ID"] = SrNo
countries_df["Country"] = Country
countries_df["Area"] = Area

print(countries_df.head(10))
print(countries_df.tail(10))
```

Output :-

```
   ID       Country              Area
0   1        Russia  13083100 (5051400)
1   2         China   9596961 (3705407)
2   3         India   3287263 (1269219)
3   4    Kazakhstan   2600000 (1000000)
4   5  Saudi Arabia    2149690 (830000)
5   6          Iran    1648195 (636372)
6   7      Mongolia    1564110 (603910)
7   8     Indonesia    1488509 (574717)
8   9      Pakistan     881913 (340509)
9  10        Turkey     759805 (293362)
    ID            Country          Area
43  44              Qatar  11586 (4473)
44  45            Lebanon  10452 (4036)
45  46             Cyprus   9251 (3572)
46  47          Palestine   6220 (2400)
47  48             Brunei   5765 (2226)
48     Hong Kong (China)   2755 (1064)
49  49            Bahrain     786 (303)
50  50          Singapore     728 (281)
51  51           Maldives     300 (120)
52         Macao (China)      115 (44)
```

2b. JSON Web Scrapping

Code :-

```
import pandas as pd
import urllib,json
url = "https://jsonplaceholder.typicode.com/users"
response = urllib.request.urlopen(url)
data = json.loads(response.read())
id = []
username =[]
email = []

for item in data:
    if "id" in item.keys():
        id.append(item["id"])
    else:
        id.append("NA")

    if "username" in item.keys():
        username.append(item["username"])
    else:
        username.append("NA")

    if "email" in item.keys():
        email.append(item["email"])
    else:
        email.append("NA")

users = pd.DataFrame()
users["id"] = id
users["username"] = username
users["email"] = email
users
```

Output :-

	id	username	email
0	1	Bret	Sincere@april.biz
1	2	Antonette	Shanna@melissa.tv
2	3	Samantha	Nathan@yesenia.net
3	4	Karianne	Julianne.OConner@kory.org
4	5	Kamren	Lucio_Hettinger@annie.ca
5	6	Leopoldo_Corkery	Karley_Dach@jasper.info
6	7	Elwyn.Skiles	Telly.Hoeger@billy.biz
7	8	Maxime_Nienow	Sherwood@rosamond.me
8	9	Delphine	Chaim_McDermott@dana.io
9	10	Moriah.Stanton	Rey.Padberg@karina.biz

Practical No. 3

Aim :- Perform Exploratory Data Analysis(EDA) of mtcars.csv in R

Code & Output :-

```
cars_df = read.csv("mtcars.csv")
View(cars_df)
```

	model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
2	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
3	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
4	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
5	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
6	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
7	Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
8	Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
9	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
10	Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
11	Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
12	Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
13	Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
14	Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
15	Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
16	Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
17	Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
18	Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
19	Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
20	Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1

```
> str(cars_df)
'data.frame':	32 obs. of  12 variables:
 $ model: chr  "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
 $ mpg  : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl  : int  6 6 4 6 8 6 8 4 4 6 ...
 $ disp : num  160 160 108 258 360 ...
 $ hp   : int  110 110 93 110 175 105 245 62 95 123 ...
 $ drat : num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt   : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec : num  16.5 17 18.6 19.4 17 ...
 $ vs   : int  0 0 1 1 0 1 0 1 1 1 ...
 $ am   : int  1 1 1 0 0 0 0 0 0 0 ...
 $ gear : int  4 4 4 3 3 3 3 4 4 4 ...
 $ carb : int  4 4 1 1 2 1 4 2 2 4 ...
```

```
> dim(cars_df)
[1] 32 12
```

```
> names(cars_df)
 [1] "model" "mpg"   "cyl"   "disp"  "hp"    "drat"  "wt"    "qsec"  "vs"    "am"
[11] "gear"  "carb"
```

```
> row.names(cars_df)
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14" "15" "16"
[17] "17" "18" "19" "20" "21" "22" "23" "24" "25" "26" "27" "28" "29" "30" "31" "32"
```

```
row.names(cars_df) = cars_df$model
View(cars_df)
```

	model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear
Mazda RX4	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4
Mazda RX4 Wag	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4
Datsun 710	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4
Hornet 4 Drive	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3
Hornet Sportabout	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3
Valiant	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3
Duster 360	Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3
Merc 240D	Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4
Merc 230	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4
Merc 280	Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4
Merc 280C	Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4
Merc 450SE	Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3
Merc 450SL	Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3
Merc 450SLC	Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3
Cadillac Fleetwood	Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3
Lincoln Continental	Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3
Chrysler Imperial	Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3
Fiat 128	Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4
Honda Civic	Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4

```
new_cars_df = cars_df[,-1] #row,column, -1 ignores first column
View(new_cars_df)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1

```
library(dplyr)
df1 = select(new_cars_df,c(1:4))
View(df1)
```

	mpg	cyl	disp	hp
Mazda RX4	21.0	6	160.0	110
Mazda RX4 Wag	21.0	6	160.0	110
Datsun 710	22.8	4	108.0	93
Hornet 4 Drive	21.4	6	258.0	110
Hornet Sportabout	18.7	8	360.0	175
Valiant	18.1	6	225.0	105
Duster 360	14.3	8	360.0	245
Merc 240D	24.4	4	146.7	62
Merc 230	22.8	4	140.8	95
Merc 280	19.2	6	167.6	123
Merc 280C	17.8	6	167.6	123
Merc 450SE	16.4	8	275.8	180
Merc 450SL	17.3	8	275.8	180
Merc 450SLC	15.2	8	275.8	180
Cadillac Fleetwood	10.4	8	472.0	205
Lincoln Continental	10.4	8	460.0	215
Chrysler Imperial	14.7	8	440.0	230
Fiat 128	32.4	4	78.7	66
Honda Civic	30.4	4	75.7	52
Toyota Corolla	33.9	4	71.1	65

```
df2 = new_cars_df%>%select(c(1:4)) #Using Pipe Operator
View(df2)
```

	mpg	cyl	disp	hp
Mazda RX4	21.0	6	160.0	110
Mazda RX4 Wag	21.0	6	160.0	110
Datsun 710	22.8	4	108.0	93
Hornet 4 Drive	21.4	6	258.0	110
Hornet Sportabout	18.7	8	360.0	175
Valiant	18.1	6	225.0	105
Duster 360	14.3	8	360.0	245
Merc 240D	24.4	4	146.7	62
Merc 230	22.8	4	140.8	95
Merc 280	19.2	6	167.6	123
Merc 280C	17.8	6	167.6	123
Merc 450SE	16.4	8	275.8	180
Merc 450SL	17.3	8	275.8	180
Merc 450SLC	15.2	8	275.8	180
Cadillac Fleetwood	10.4	8	472.0	205
Lincoln Continental	10.4	8	460.0	215
Chrysler Imperial	14.7	8	440.0	230
Fiat 128	32.4	4	78.7	66
Honda Civic	30.4	4	75.7	52
Toyota Corolla	33.9	4	71.1	65

```
df3 = cars_df%>%select(c(mpg,disp,wt,gear)) #To randomly select columns
View(df3)
```

	mpg	disp	wt	gear
Mazda RX4	21.0	160.0	2.620	4
Mazda RX4 Wag	21.0	160.0	2.875	4
Datsun 710	22.8	108.0	2.320	4
Hornet 4 Drive	21.4	258.0	3.215	3
Hornet Sportabout	18.7	360.0	3.440	3
Valiant	18.1	225.0	3.460	3
Duster 360	14.3	360.0	3.570	3
Merc 240D	24.4	146.7	3.190	4
Merc 230	22.8	140.8	3.150	4
Merc 280	19.2	167.6	3.440	4
Merc 280C	17.8	167.6	3.440	4
Merc 450SE	16.4	275.8	4.070	3
Merc 450SL	17.3	275.8	3.730	3
Merc 450SLC	15.2	275.8	3.780	3
Cadillac Fleetwood	10.4	472.0	5.250	3
Lincoln Continental	10.4	460.0	5.424	3
Chrysler Imperial	14.7	440.0	5.345	3
Fiat 128	32.4	78.7	2.200	4
Honda Civic	30.4	75.7	1.615	4
Toyota Corolla	33.9	71.1	1.835	4

```
df4 = filter(df3,gear==4,)
View(df4)
```

	mpg	disp	wt	gear
Mazda RX4	21.0	160.0	2.620	4
Mazda RX4 Wag	21.0	160.0	2.875	4
Datsun 710	22.8	108.0	2.320	4
Merc 240D	24.4	146.7	3.190	4
Merc 230	22.8	140.8	3.150	4
Merc 280	19.2	167.6	3.440	4
Merc 280C	17.8	167.6	3.440	4
Fiat 128	32.4	78.7	2.200	4
Honda Civic	30.4	75.7	1.615	4
Toyota Corolla	33.9	71.1	1.835	4
Fiat X1-9	27.3	79.0	1.935	4
Volvo 142E	21.4	121.0	2.780	4

```
df5 = cars_df%>%filter(gear==4)%>%select(c(mpg,wt,disp,gear))
View(df5)
```

	mpg	wt	disp	gear
Mazda RX4	21.0	2.620	160.0	4
Mazda RX4 Wag	21.0	2.875	160.0	4
Datsun 710	22.8	2.320	108.0	4
Merc 240D	24.4	3.190	146.7	4
Merc 230	22.8	3.150	140.8	4
Merc 280	19.2	3.440	167.6	4
Merc 280C	17.8	3.440	167.6	4
Fiat 128	32.4	2.200	78.7	4
Honda Civic	30.4	1.615	75.7	4
Toyota Corolla	33.9	1.835	71.1	4
Fiat X1-9	27.3	1.935	79.0	4
Volvo 142E	21.4	2.780	121.0	4

```
df6 = cars_df%>%filter(cyl == 4 | mpg>20)%>%select(c(mpg,cyl))
View(df6)
```

	mpg	cyl
Mazda RX4	21.0	6
Mazda RX4 Wag	21.0	6
Datsun 710	22.8	4
Hornet 4 Drive	21.4	6
Merc 240D	24.4	4
Merc 230	22.8	4
Fiat 128	32.4	4
Honda Civic	30.4	4
Toyota Corolla	33.9	4
Toyota Corona	21.5	4
Fiat X1-9	27.3	4
Porsche 914-2	26.0	4
Lotus Europa	30.4	4
Volvo 142E	21.4	4

```
df7 = new_cars_df%>%filter(mpg<20&carb==3)%>%select(c(mpg,carb))
View(df7)
```

	mpg	carb
Merc 450SE	16.4	3
Merc 450SL	17.3	3
Merc 450SLC	15.2	3

```
df8 = new_cars_df%>%arrange(desc(mpg))
View(df8)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4

```
df9 = new_cars_df%>%arrange(cyl)%>%arrange(desc(mpg))
View(df9)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4

```
df10 = cars_df%>%rename(cylinders=cyl,milespergallon=mpg)
View(df10)
```

	model	milespergallon	cylinders	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1

```
df11 = df10%>%mutate(power=hp*wt)
View(df11)
```

	model	milespergallon	cylinders	disp	hp	drat	wt	qsec	vs	am	gear	carb	power
Mazda RX4	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4	288.200
Mazda RX4 Wag	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4	316.250
Datsun 710	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1	215.760
Hornet 4 Drive	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1	353.650
Hornet Sportabout	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2	602.000
Valiant	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1	363.300
Duster 360	Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4	874.650
Merc 240D	Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2	197.780
Merc 230	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2	299.250
Merc 280	Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4	423.120
Merc 280C	Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4	423.120
Merc 450SE	Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3	732.600
Merc 450SL	Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3	671.400
Merc 450SLC	Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3	680.400
Cadillac Fleetwood	Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4	1076.250
Lincoln Continental	Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4	1166.160
Chrysler Imperial	Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4	1229.350
Fiat 128	Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1	145.200
Honda Civic	Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2	83.980
Toyota Corolla	Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1	119.275

```
> str(df11)
'data.frame':	32 obs. of  13 variables:
 $ model         : chr  "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
 $ milespergallon: num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cylinders     : int  6 6 4 6 8 6 8 4 4 6 ...
 $ disp          : num  160 160 108 258 360 ...
 $ hp            : int  110 110 93 110 175 105 245 62 95 123 ...
 $ drat          : num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt            : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec          : num  16.5 17 18.6 19.4 17 ...
 $ vs            : int  0 0 1 1 0 1 0 1 1 1 ...
 $ am            : int  1 1 1 0 0 0 0 0 0 0 ...
 $ gear          : int  4 4 4 3 3 3 3 4 4 4 ...
 $ carb          : int  4 4 1 1 2 1 4 2 2 4 ...
 $ power         : num  288 316 216 354 602 ...
```

```
#To convert int to factor
cars_df$gear = as.factor(cars_df$gear)
View(cars_df)
```

	model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1

```
> str(cars_df)
'data.frame':	32 obs. of  12 variables:
 $ model: chr  "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
 $ mpg  : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl  : int  6 6 4 6 8 6 8 4 4 6 ...
 $ disp : num  160 160 108 258 360 ...
 $ hp   : int  110 110 93 110 175 105 245 62 95 123 ...
 $ drat : num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt   : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec : num  16.5 17 18.6 19.4 17 ...
 $ vs   : int  0 0 1 1 0 1 0 1 1 1 ...
 $ am   : int  1 1 1 0 0 0 0 0 0 0 ...
 $ gear : Factor w/ 3 levels "3","4","5": 2 2 2 1 1 1 1 2 2 2 ...
 $ carb : int  4 4 1 1 2 1 4 2 2 4 ...
```

```
df12 = cars_df%>%group_by(gear)%>%summarise(no=n(),mean_mpg=mean(mpg),mean_wt=mean(wt))
View(df12)
```

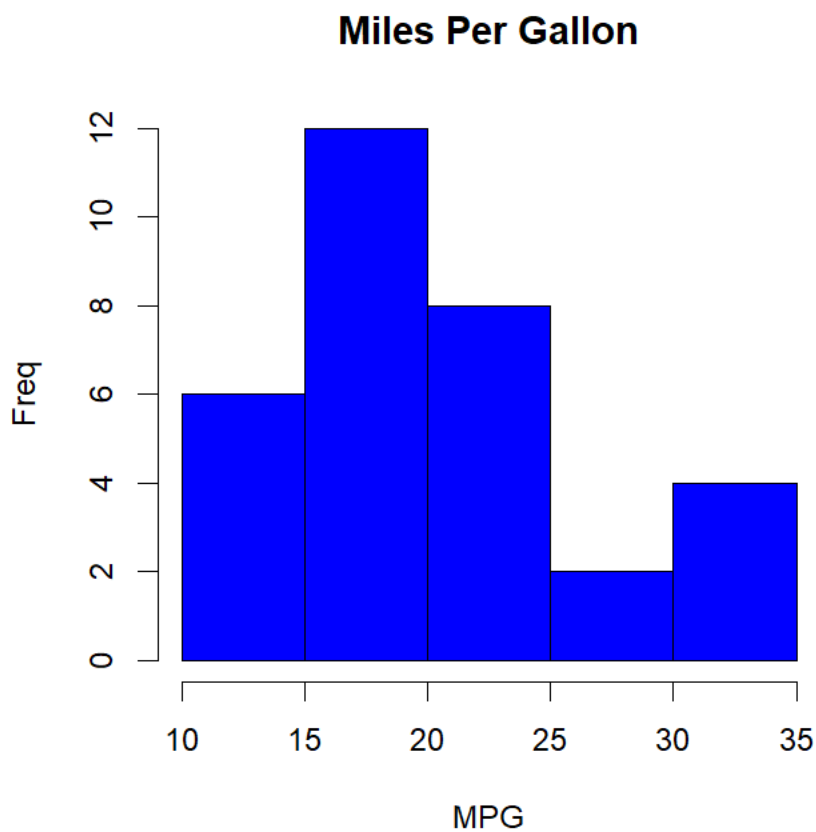
	gear	no	mean_mpg	mean_wt
1	3	15	16.10667	3.892600
2	4	12	24.53333	2.616667
3	5	5	21.38000	2.632600

```
> cars_df$cyl = as.factor(cars_df$cyl)
> str(cars_df)
'data.frame':	32 obs. of  12 variables:
 $ model: chr  "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
 $ mpg  : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl  : Factor w/ 3 levels "4","6","8": 2 2 1 2 3 2 3 1 1 2 ...
 $ disp : num  160 160 108 258 360 ...
 $ hp   : int  110 110 93 110 175 105 245 62 95 123 ...
 $ drat : num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt   : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec : num  16.5 17 18.6 19.4 17 ...
 $ vs   : int  0 0 1 1 0 1 0 1 1 1 ...
 $ am   : int  1 1 1 0 0 0 0 0 0 0 ...
 $ gear : Factor w/ 3 levels "3","4","5": 2 2 2 1 1 1 1 2 2 2 ...
 $ carb : int  4 4 1 1 2 1 4 2 2 4 ...
```

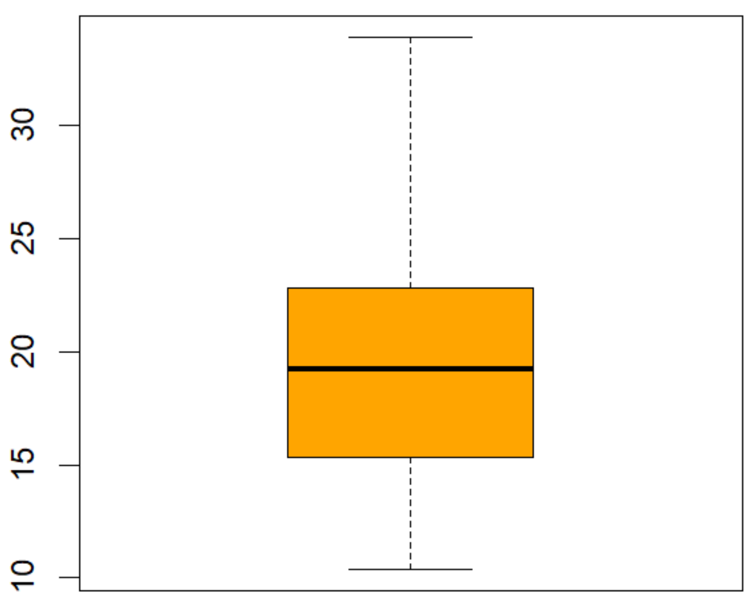
```
df13 = cars_df%>%group_by(cyl)%>%summarise(no=n(),mean_mpg=mean(mpg),mean_wt = mean(wt),pro=mean_mpg*mean_wt)
View(df13)
```

	cyl	no	mean_mpg	mean_wt	pro
1	4	11	26.66364	2.285727	60.94580
2	6	7	19.74286	3.117143	61.54131
3	8	14	15.10000	3.999214	60.38814

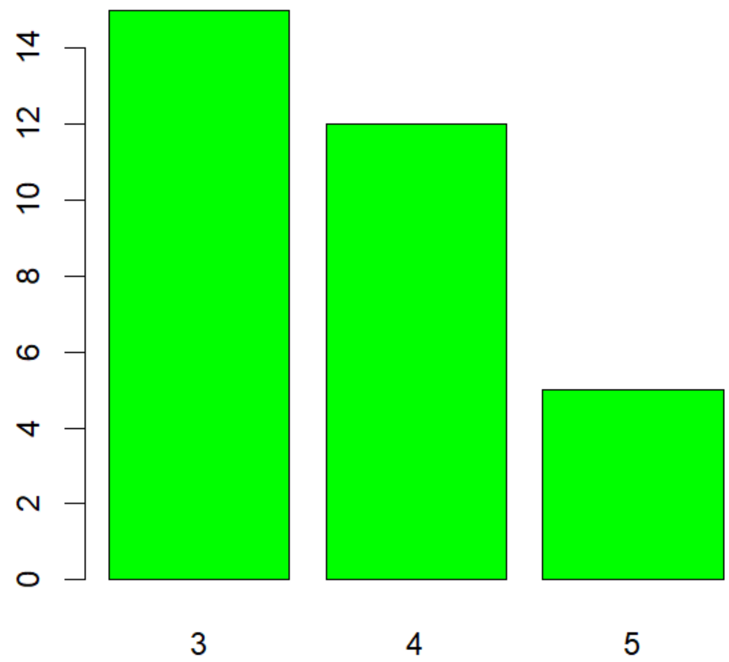
```
#Data Visualizations
hist(cars_df$mpg,xlab = "MPG",ylab = "Freq",main = "Miles Per Gallon",col="blue") #Single column frequency
```




```
boxplot(cars_df$mpg,col="orange") #Diagrametic representation of summary
```

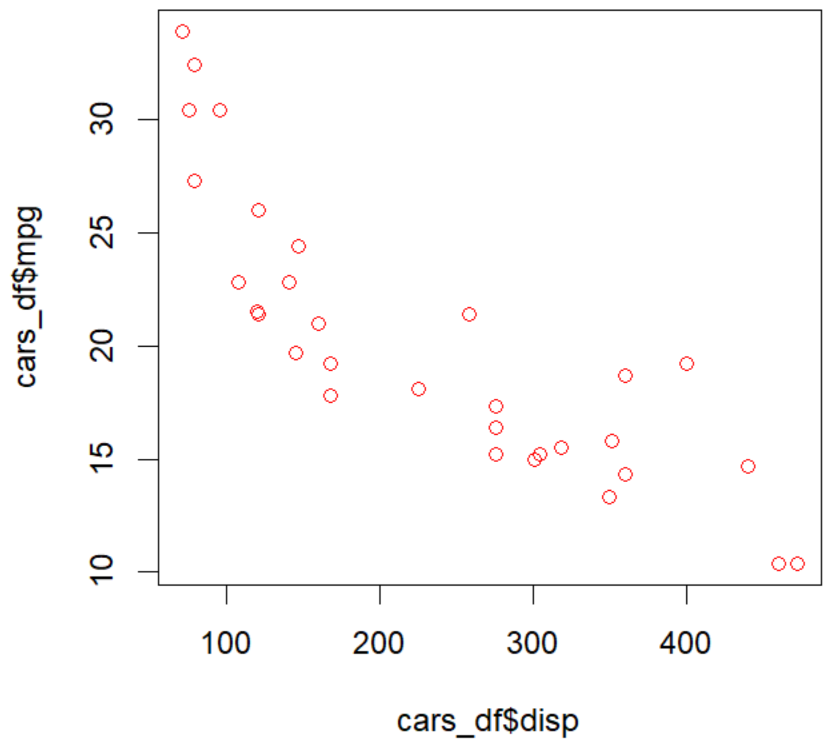


```r
barplot(table(cars_df$gear),col="green") #Used for categorical variable
```

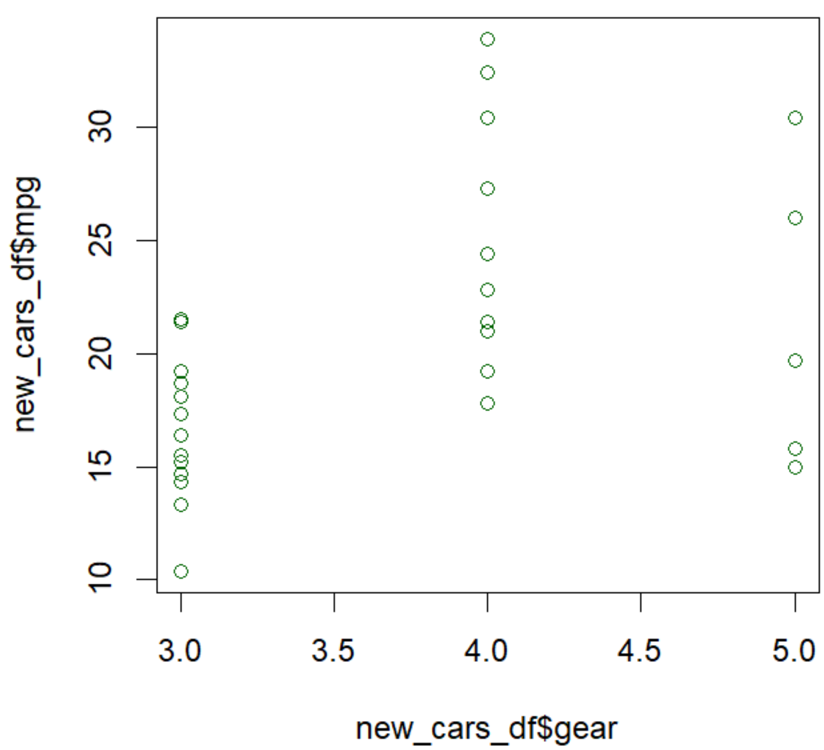



```
> table(cars_df$gear)

 3  4  5 
15 12  5 
```

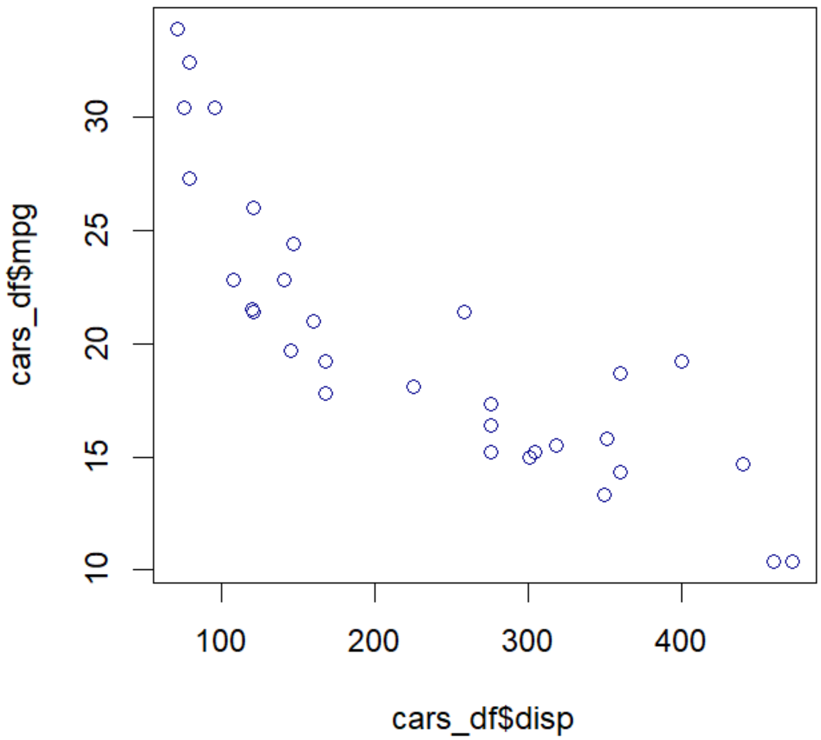
```r
plot(cars_df$mpg~cars_df$disp,col="red") #Plots relationship between two variables
```




```
plot(new_cars_df$mpg~new_cars_df$gear,col="darkgreen")
```



```
plot(cars_df$mpg~cars_df$disp,col="darkblue")
```



Practical No. 4

Aim :- Exploratory data analysis in Python using Titanic Dataset

Description :- It is one of the most popular datasets used for understanding machine learning basics. It contains information of all the passengers aboard the RMS Titanic, which unfortunately was shipwrecked. This dataset can be used to predict whether a given passenger survived or not.

Seaborn: It is a python library used to statistically visualize data. Seaborn, built over Matplotlib, provides a better interface and ease of usage. It can be installed using the following command, pip3 install seaborn

Features: The titanic dataset has roughly the following types of features:

Categorical/Nominal: Variables that can be divided into multiple categories but having no order or priority. Eg. Embarked (C = Cherbourg; Q = Queenstown; S = Southampton) ¶

Binary: A subtype of categorical features, where the variable has only two categories. Eg: Sex (Male/Female)

Ordinal: They are similar to categorical features but they have an order(i.e can be sorted). Eg. Pclass (1, 2, 3)

Continuous: They can take up any value between the minimum and maximum values in a column. Eg. Age, Fare

Count: They represent the count of a variable. Eg. SibSp, Parch

Useless: They don’t contribute to the final outcome of an ML model. Here, PassengerId, Name, Cabin and Ticket might fall into this category.

Code :-

```
from re import A
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

%matplotlib inline

titanic = pd.read_csv("train.csv")
titanic.head()
titanic.info() # str() of R
titanic.describe() #similar to summary() of R
titanic.isnull().sum() #is.na() of R
titanic_clean = titanic.drop(['PassengerId','Name','Ticket','Cabin','Fare'],axis=1)
print(titanic_clean)

sns.catplot(x='Sex',hue='Survived',kind='count',data=titanic_clean)
titanic_clean.groupby(['Sex','Survived']).count()

group1 =titanic_clean.groupby(['Sex','Survived'])
gender_survived = group1.size().unstack()
sns.heatmap(gender_survived,annot=True,fmt='d')

group2 =titanic_clean.groupby(['Pclass','Survived'])
Pclass_survived = group2.size().unstack()
sns.heatmap(Pclass_survived,annot=True,fmt='d')

sns.violinplot(x='Sex',y='Age',hue='Survived',data=titanic_clean,split=True)
```

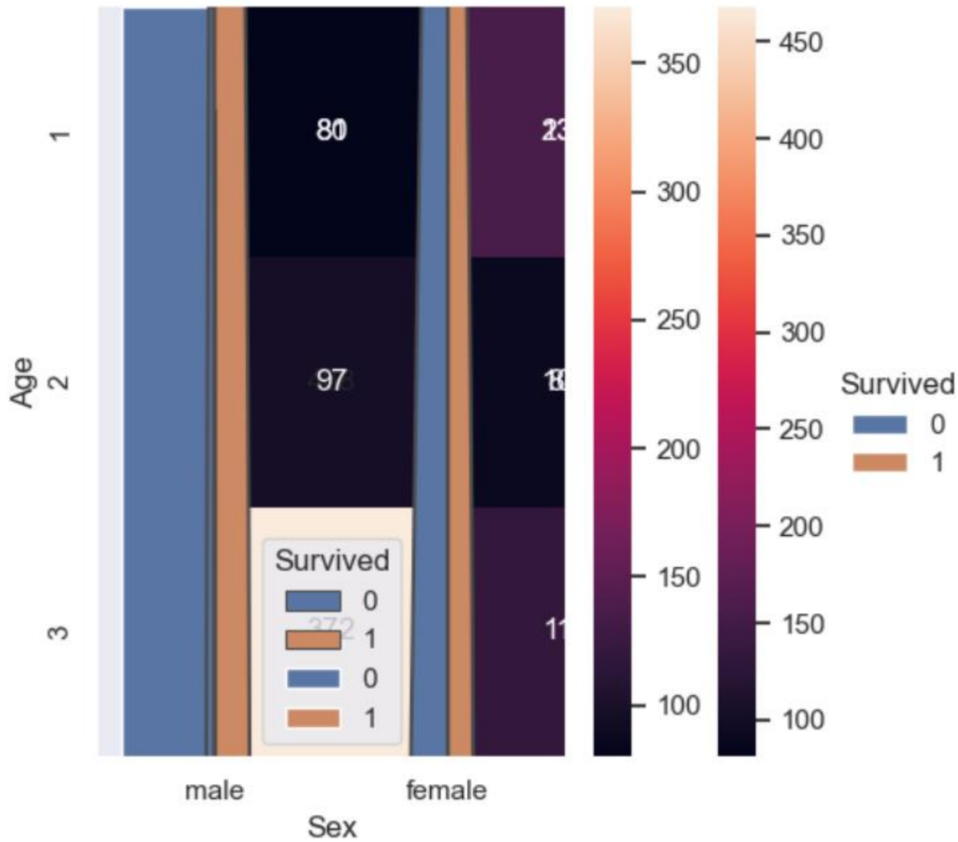
```python
print("Oldest person on board : ",titanic_clean['Age'].max())
print("Youngest person on board : ",titanic_clean['Age'].min())
print("Average age of people on board : ",titanic_clean['Age'].mean())

def impute(cols):
    Age = cols[0]
    Pclass = cols[1]

    if pd.isnull(Age):
        if Pclass == 1:
            return 32
        elif Pclass==2:
            return 29
        else:
            return 24
    else:
        return Age

titanic_clean['Age'] = titanic_clean[['Age','Pclass']].apply(impute,axis=1)

titanic_clean.isnull().sum()

titanic_clean.corr('pearson')

sns.heatmap(titanic_clean.corr('pearson'),annot=True,vmax=1)
```

Output :-

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
     Survived  Pclass     Sex   Age  SibSp  Parch Embarked
0           0       3    male  22.0      1      0        S
1           1       1  female  38.0      1      0        C
2           1       3  female  26.0      0      0        S
3           1       1  female  35.0      1      0        S
4           0       3    male  35.0      0      0        S
..        ...     ...     ...   ...    ...    ...      ...
886         0       2    male  27.0      0      0        S
887         1       1  female  19.0      0      0        S
888         0       3  female   NaN      1      2        S
889         1       1    male  26.0      0      0        C
890         0       3    male  32.0      0      0        Q
```

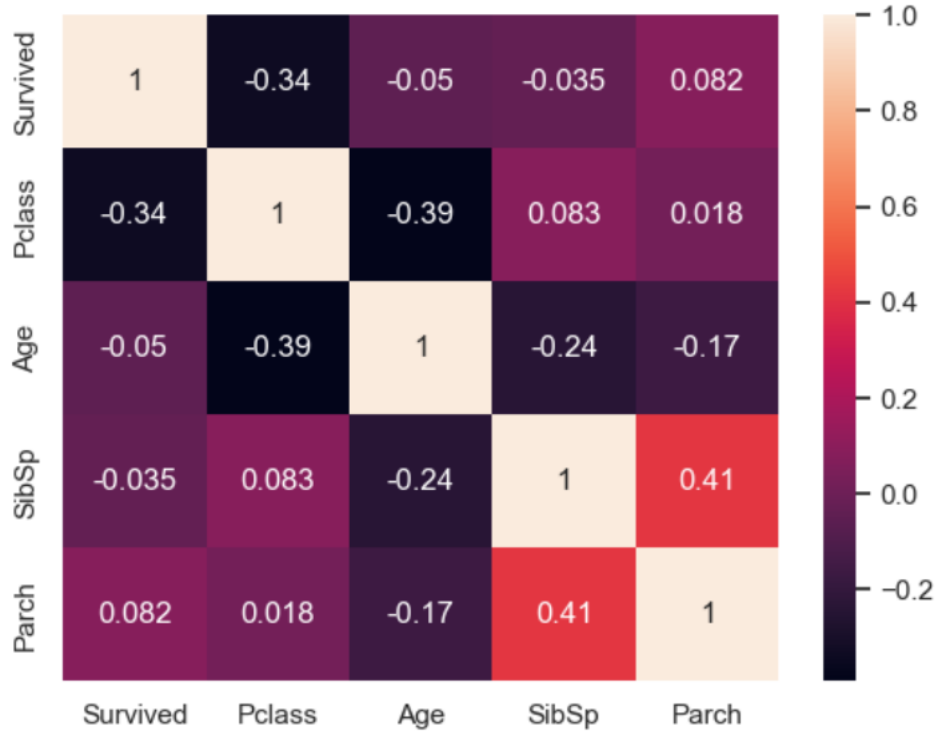
```
[891 rows x 7 columns]
```

Out[23]: `<AxesSubplot:xlabel='Sex', ylabel='Age'>`



```
Oldest person on board :  80.0
Youngest person on board :  0.42
Average age of people on board :  29.69911764705882
```

Out[24]: <AxesSubplot:>



Practical No. 5

Aim :- 5a. Write a python program to build a regression model that could predict the salary of an employee from the given experience and visualize univariate linear regression on it.

Description :- The package scikit-learn is a widely used Python library for machine learning, built on top of NumPy and some other packages, It provides the means for preprocessing data, reducing dimensionality, implementing regression, classification, clustering, and more. Like NumPy, scikit-learn is also open source.

It is used as sklearn in python

Code :-

```
import numpy as np
from sklearn import datasets
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import pandas as pd
x,y,coef = datasets.make_regression(n_samples=100,n_features=1, n_informative=1,noise=10,coef=True,random_state=10)
x = np.interp(x,(x.min(),x.max()),(0,20))
y = np.interp(y,(y.min(),y.max()),(20000,160000))
plt.plot(x,y,'*',label="Training Data")
plt.xlabel("Years Of Experience")
plt.ylabel("Salary")
plt.title("Experience vs Salary")
reg_mode = LinearRegression()
reg_mode.fit(x,y)
y_pred = reg_mode.predict(x)
plt.plot(x,y_pred,color='red')
data = {'Experience':np.round(x.flatten()),'Salary':np.round(y)}
df = pd.DataFrame(data)
df.head()
x1 = [[31.0]]
y1 = reg_mode.predict(x1)
print(np.round(y1))
```

Output :-

[220097.]

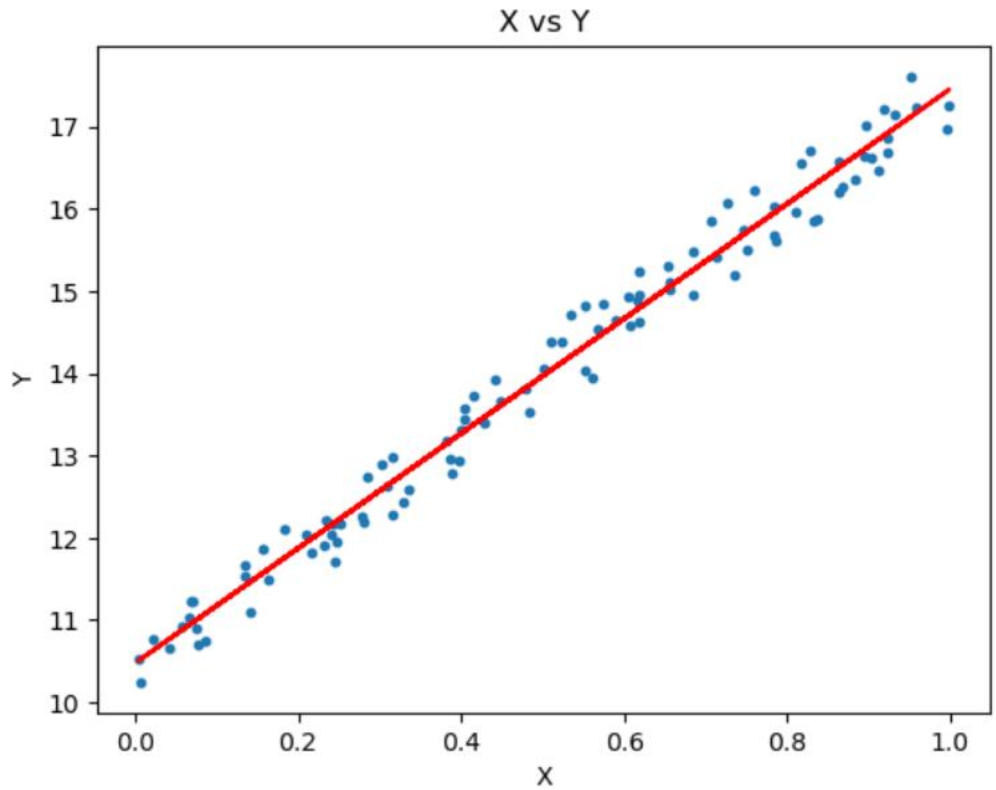
Aim :- 5b. Write a python program to simulate linear model

Y=10+7*x+e for random 100 samples and visualize univariate linear regression on it.

```
reg_model1 = LinearRegression()
x = np.random.rand(100,1)
yintercept = 10
slope = 7
error = np.random.rand(100,1)
y = yintercept + slope * x + error
reg_model1.fit(x,y)
y_predicted = reg_model1.predict(x)
plt.scatter(x,y,s=10)
plt.xlabel("X")
plt.ylabel("Y")
plt.title("X vs Y")
plt.plot(x,y_predicted,color='red')
```

Output :-

Out[16]: [<matplotlib.lines.Line2D at 0x14992af03d0>]



Practical No. 6

Aim :- Write a python program to implement multiple linear regression on the Dataset Boston.csv

Description :- The dataset provides Housing Values in Suburbs of Boston

The medv(Price) variable is the target /dependent variable.

Data description

The Boston data frame has 506 rows and 14 columns.

This data frame contains the following columns:

crim per capita crime rate by town.

zn proportion of residential land zoned for lots over 25,000 sq.ft.

indus proportion of non-retail business acres per town.

chas Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).

nox nitrogen oxides concentration (parts per 10 million).

rm average number of rooms per dwelling.

age proportion of owner-occupied units built prior to 1940.

dis weighted mean of distances to five Boston employment centres.

rad index of accessibility to radial highways.

tax full-value property-tax rate per dollor 10,000.

ptratio pupil-teacher ratio by town.

black 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town.

lstat lower status of the population (percent).

Medv(Price) median value of owner-occupied homes in $1000s.

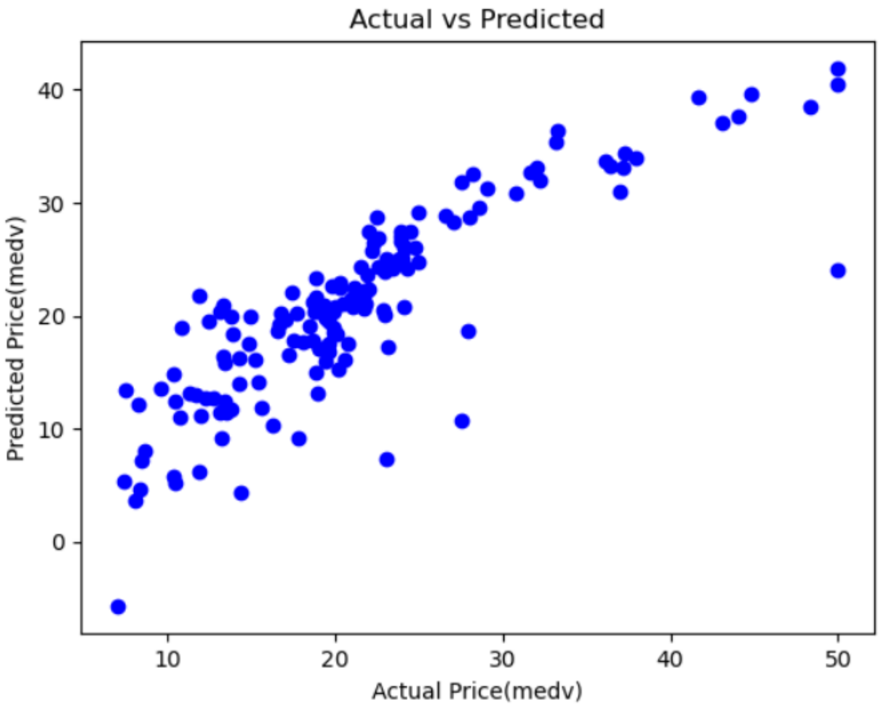
Code :-

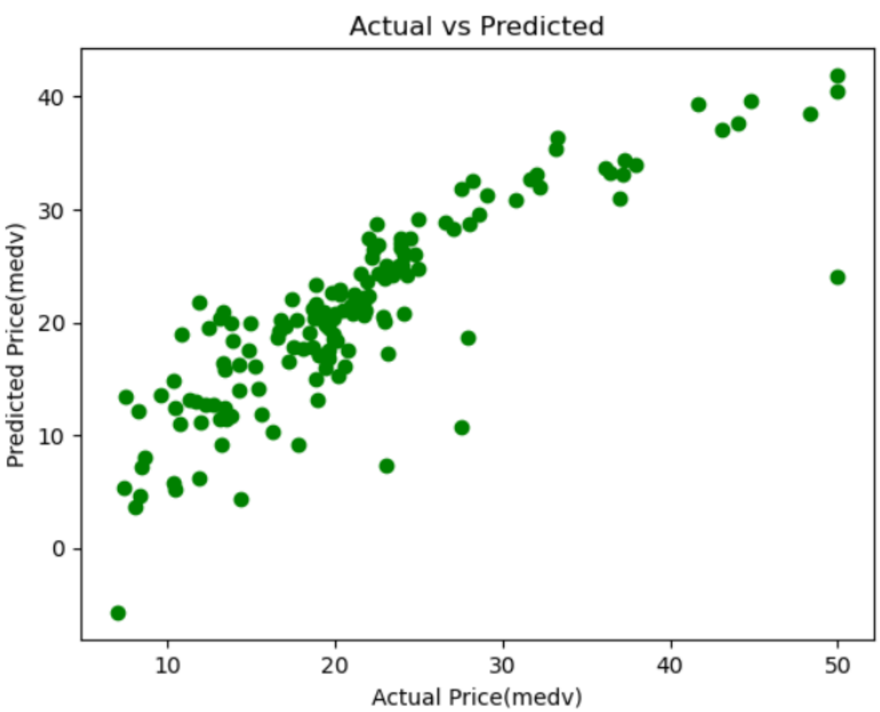
```python
import pandas as pd
import matplotlib.pyplot as plt
import sklearn

boston = pd.read_csv("Boston.csv")
boston.head()
boston.info()
boston = boston.drop(columns="Unnamed: 0") #Removing particular column
boston.info()

boston_x = pd.DataFrame(boston.iloc[:,:13]) # Ceating a DataFrame with independent variables
boston_y = pd.DataFrame(boston.iloc[:,-1]) # Creating a DataFrame with dependent variable
boston_x.head()
boston_y.head()

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(boston_x,boston_y,test_size=0.3)
print(f'XTrain Shape : {x_train.shape}\nYTrain Shape : {y_train.shape}\nXTest Shape : {x_test.shape}\nYTest Shape : {y_test.shape

from sklearn.linear_model import LinearRegression
regression = LinearRegression()
regression.fit(x_train,y_train)
predicted_y = regression.predict(x_test)
predicted_y_df = pd.DataFrame(predicted_y,columns=["Predicted"])
predicted_y_df.head()

plt.scatter(y_test,predicted_y_df,c="blue")
plt.xlabel("Actual Price(medv)")
plt.ylabel("Predicted Price(medv)")
plt.title("Actual vs Predicted")
plt.show()

plt.scatter(y_test,predicted_y,c="green")
plt.xlabel("Actual Price(medv)")
plt.ylabel("Predicted Price(medv)")
plt.title("Actual vs Predicted")
plt.show()
```

Output :-

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  506 non-null    int64
 1   crim        506 non-null    float64
 2   zn          506 non-null    float64
 3   indus       506 non-null    float64
 4   chas        506 non-null    int64
 5   nox         506 non-null    float64
 6   rm          506 non-null    float64
 7   age         506 non-null    float64
 8   dis         506 non-null    float64
 9   rad         506 non-null    int64
 10  tax         506 non-null    int64
 11  ptratio     506 non-null    float64
 12  black       506 non-null    float64
 13  lstat       506 non-null    float64
 14  medv        506 non-null    float64
dtypes: float64(11), int64(4)
memory usage: 59.4 KB
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   crim     506 non-null    float64
 1   zn       506 non-null    float64
 2   indus    506 non-null    float64
 3   chas     506 non-null    int64
 4   nox      506 non-null    float64
 5   rm       506 non-null    float64
 6   age      506 non-null    float64
 7   dis      506 non-null    float64
 8   rad      506 non-null    int64
 9   tax      506 non-null    int64
 10  ptratio  506 non-null    float64
 11  black    506 non-null    float64
 12  lstat    506 non-null    float64
 13  medv     506 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 55.5 KB
XTrain Shape : (354, 13)
YTrain Shape : (354, 1)
XTest Shape : (152, 13)
YTest Shape : (152, 1)
```


Actual vs Predicted
Predicted Price(medv)
Actual Price(medv)
40
30
20
10
0
10
20
30
40
50


Actual vs Predicted
Predicted Price(medv)
Actual Price(medv)
40
30
20
10
0
10
20
30
40
50

Practical No. 7

Aim :- K Nearest Neighbor classification Algorithm

Write a python program to implement KNN algorithm to predict breast cancer using breast cancer wisconsin dataset .

Description :- Data Set Information:

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

Attribute Information:

1) ID number 2) Diagnosis (M = malignant, B = benign) (3-32) Ten real-valued features are computed for each cell nucleus: a) radius (mean of distances from center to points on the perimeter) b) texture (standard deviation of gray-scale values) c) perimeter d) area e) smoothness (local variation in radius lengths) f) compactness (perimeter^2 / area - 1.0) g) concavity (severity of concave portions of the contour) h) concave points (number of concave portions of the contour) i) symmetry j) fractal dimension ("coastline approximation" - 1)

Code :-

```
from sklearn.datasets import load_breast_cancer
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
import seaborn as sns

breast_cancer_df = load_breast_cancer()
x = pd.DataFrame(breast_cancer_df.data,columns=breast_cancer_df.feature_names)
x.head()

x = x[["mean area","mean compactness"]]
x.head()

y = pd.Categorical.from_codes(breast_cancer_df.target,breast_cancer_df.target_names)
print(y)

y = pd.get_dummies(y,drop_first=True)
print(y)

x_train, x_test, y_train ,y_test = train_test_split(x,y,random_state=1)
print(f'XTrain Shape : {x_train.shape}\nYTrain Shape : {y_train.shape}\nXTest Shape : {x_test.shape}\nYTest Shape :
      {y_test.shape}')

Knn = KNeighborsClassifier(n_neighbors=5,metric="euclidean")
Knn.fit(x_train,y_train)

sns.set()
sns.scatterplot(x="mean area",y="mean compactness", hue="benign",data=x_test.join(y_test,how="outer"))
```

```
predicted_y = Knn.predict(x_test)
plt.scatter(x_test["mean area"],x_test["mean compactness"],c=predicted_y,cmap="coolwarm",alpha=0.7)
```

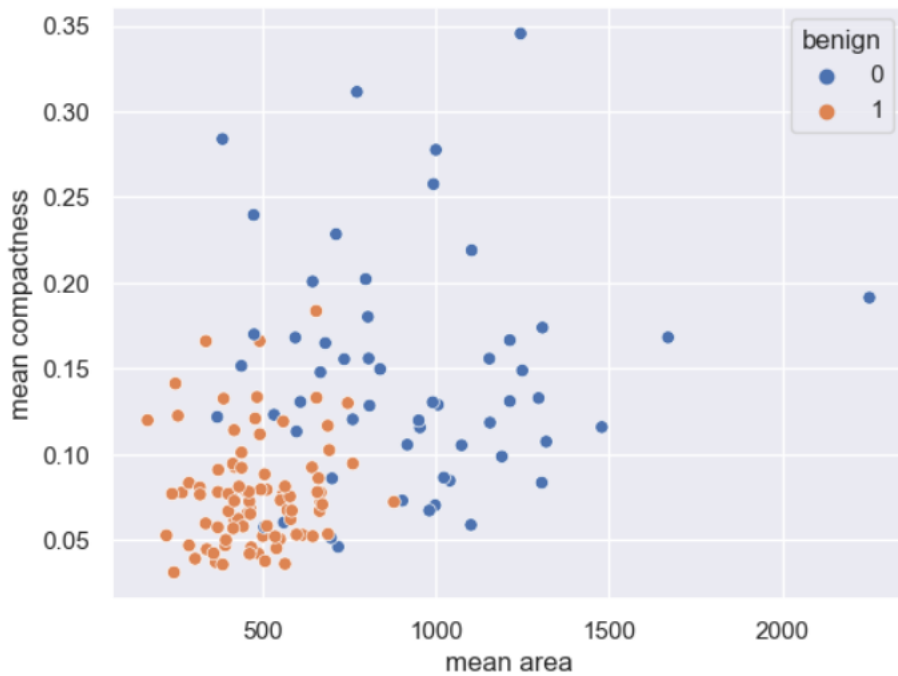
```python
cf = confusion_matrix(y_test,predicted_y)
print(cf)

labels = ["True Positive","True Negative","False Positive","False Negative"]
labels = np.asarray(labels).reshape(2,2)
categories = ["Zero","One"]
ax = plt.subplot()
sns.heatmap(cf,annot=True,ax=ax)
ax.set_xlabel("Predicted Values")
ax.set_ylabel("Actual Values")
ax.set_title("Confusion Matrix")
ax.xaxis.set_ticklabels(["Malignant","Benign"])
ax.yaxis.set_ticklabels(["Malignant","Benign"])
```

Output :-

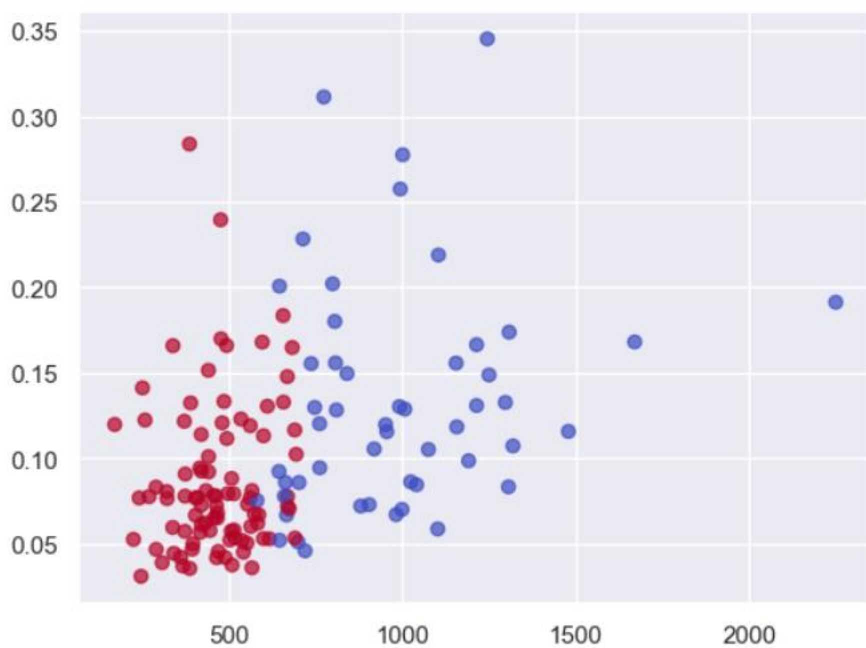
```
['malignant', 'malignant', 'malignant', 'malignant', 'malignant', ..., 'malignant', 'malignant', 'malignant', 'malignant', 'ben
ign']
Length: 569
Categories (2, object): ['malignant', 'benign']
     benign
0         0
1         0
2         0
3         0
4         0
..      ...
564       0
565       0
566       0
567       0
568       1

[569 rows x 1 columns]
XTrain Shape : (426, 2)
YTrain Shape : (426, 1)
XTest Shape : (143, 2)
YTest Shape : (143, 1)
```

```
C:\Users\admin\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:198: DataConversionWarning: A column-vector y w
as passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  return self._fit(X, y)
```

<AxesSubplot:xlabel='mean area', ylabel='mean compactness'>

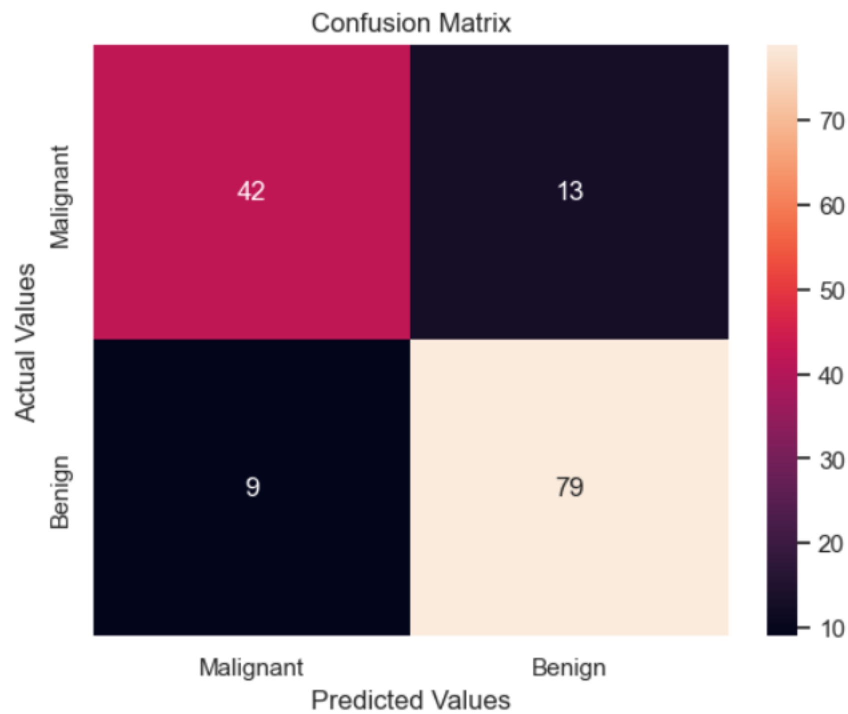


<matplotlib.collections.PathCollection at 0x149906fe850>



```
[[42 13]
 [ 9 79]]
```

Out[21]:
```
[Text(0, 0.5, 'Malignant'), Text(0, 1.5, 'Benign')]
```




```
True Postive : 79
True Negative : 42
False Positive : 13
False Negative : 9
Accuracy :  0.8461538461538461
Precision :  0.8586956521739131
Recall :  0.8977272727272727
F1Score :  0.8777777777777778
0.8777777777777778
```

Out[22]:
```
0.8306818181818182
```

Practical No. 8

Aim :- Introduction to NOSQL using MongoDB

Perform the following:

1.Create a database Company ,Create a Collection Staff and Insert ten documents in it with fields: empid, empname, salary and designation.

```
> use Company;
switched to db Company
```

```
> db.Staff.insertMany( [ {empid:'E001',empname:'Employee1',salary:122000,designation:'Manager'}, {empid:'E002',empname:'Employee2',salary:112000,designation:'Accountant'}, {empid:'E003',empname:'Employee3',salary:102000,designation:'Python Developer'}, {empid:'E004',empname:'Employee4',salary:92000,designation:'Manager'}, {empid:'E005',empname:'Employee5',salary:82000,designation:'Data Analyst'}, {empid:'E006',empname:'Employee6',salary:72000,designation:'Java Developer'}, {empid:'E007',empname:'Employee7',salary:62000,designation:'dotNET Developer'}, {empid:'E008',empname:'Employee8',salary:52000,designation:'Andriod Developer'}, {empid:'E009',empname:'Employee9',salary:45000,designation:'Accountant'}, {empid:'E010',empname:'Employee10',salary:32000,designation:'Manager'} ] );
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("65e9f979257325a977e114cd"),
                ObjectId("65e9f979257325a977e114ce"),
                ObjectId("65e9f979257325a977e114cf"),
                ObjectId("65e9f979257325a977e114d0"),
                ObjectId("65e9f979257325a977e114d1"),
                ObjectId("65e9f979257325a977e114d2"),
                ObjectId("65e9f979257325a977e114d3"),
                ObjectId("65e9f979257325a977e114d4"),
                ObjectId("65e9f979257325a977e114d5"),
                ObjectId("65e9f979257325a977e114d6")
        ]
}
```

Display all documents in Staff and display only empid and designation

```
> db.Staff.find({},{empid:1,designation:1})
{ "_id" : ObjectId("65e9f979257325a977e114cd"), "empid" : "E001", "designation" : "Manager" }
{ "_id" : ObjectId("65e9f979257325a977e114ce"), "empid" : "E002", "designation" : "Accountant" }
{ "_id" : ObjectId("65e9f979257325a977e114cf"), "empid" : "E003", "designation" : "Python Developer" }
{ "_id" : ObjectId("65e9f979257325a977e114d0"), "empid" : "E004", "designation" : "Manager" }
{ "_id" : ObjectId("65e9f979257325a977e114d1"), "empid" : "E005", "designation" : "Data Analyst" }
{ "_id" : ObjectId("65e9f979257325a977e114d2"), "empid" : "E006", "designation" : "Java Developer" }
{ "_id" : ObjectId("65e9f979257325a977e114d3"), "empid" : "E007", "designation" : "dotNET Developer" }
{ "_id" : ObjectId("65e9f979257325a977e114d4"), "empid" : "E008", "designation" : "Andriod Developer" }
{ "_id" : ObjectId("65e9f979257325a977e114d5"), "empid" : "E009", "designation" : "Accountant" }
{ "_id" : ObjectId("65e9f979257325a977e114d6"), "empid" : "E010", "designation" : "Manager" }
```

Sort the documents in descending order of Salary

```
> db.Staff.find().sort({salary:-1});
{ "_id" : ObjectId("65e9f979257325a977e114cd"), "empid" : "E001", "empname" : "Employee1", "salary" : 122000, "designati
on" : "Manager" }
{ "_id" : ObjectId("65e9f979257325a977e114ce"), "empid" : "E002", "empname" : "Employee2", "salary" : 112000, "designati
on" : "Accountant" }
{ "_id" : ObjectId("65e9f979257325a977e114cf"), "empid" : "E003", "empname" : "Employee3", "salary" : 102000, "designati
on" : "Python Developer" }
{ "_id" : ObjectId("65e9f979257325a977e114d0"), "empid" : "E004", "empname" : "Employee4", "salary" : 92000, "designatio
n" : "Manager" }
{ "_id" : ObjectId("65e9f979257325a977e114d1"), "empid" : "E005", "empname" : "Employee5", "salary" : 82000, "designatio
n" : "Data Analyst" }
{ "_id" : ObjectId("65e9f979257325a977e114d2"), "empid" : "E006", "empname" : "Employee6", "salary" : 72000, "designatio
n" : "Java Developer" }
{ "_id" : ObjectId("65e9f979257325a977e114d3"), "empid" : "E007", "empname" : "Employee7", "salary" : 62000, "designatio
n" : "dotNET Developer" }
{ "_id" : ObjectId("65e9f979257325a977e114d4"), "empid" : "E008", "empname" : "Employee8", "salary" : 52000, "designatio
n" : "Andriod Developer" }
{ "_id" : ObjectId("65e9f979257325a977e114d5"), "empid" : "E009", "empname" : "Employee9", "salary" : 45000, "designatio
n" : "Accountant" }
{ "_id" : ObjectId("65e9f979257325a977e114d6"), "empid" : "E010", "empname" : "Employee10", "salary" : 32000, "designati
on" : "Manager" }
```

Display employee with designation with “Manager” or salary greater than Rs. 50,000/-.

```
> db.Staff.find({$or : [{designation:"Manager"},{salary : {$gt : 50000}}]});
{ "_id" : ObjectId("65e9f979257325a977e114d0"), "empid" : "E004", "empname" : "Employee4", "salary" : 92000, "designation" : "Manager
" }
{ "_id" : ObjectId("65e9f979257325a977e114d1"), "empid" : "E005", "empname" : "Employee5", "salary" : 82000, "designation" : "Data An
alyst" }
{ "_id" : ObjectId("65e9f979257325a977e114d2"), "empid" : "E006", "empname" : "Employee6", "salary" : 72000, "designation" : "Java De
veloper" }
{ "_id" : ObjectId("65e9f979257325a977e114d3"), "empid" : "E007", "empname" : "Employee7", "salary" : 62000, "designation" : "dotNET
Developer" }
{ "_id" : ObjectId("65e9f979257325a977e114d4"), "empid" : "E008", "empname" : "Employee8", "salary" : 52000, "designation" : "Andriod
 Developer" }
{ "_id" : ObjectId("65e9f979257325a977e114d6"), "empid" : "E010", "empname" : "Employee10", "salary" : 32000, "designation" : "Manage
r" }
```

Update the salary of all employees with designation as “Accountant” to Rs.45000

```
> db.Staff.updateOne({designation:"Accountant"},{$set :{salary : 45000}});
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.Staff.find().pretty();
{
        "_id" : ObjectId("65e9f979257325a977e114cd"),
        "empid" : "E001",
        "empname" : "Employee1",
        "salary" : 122000,
        "designation" : "Manager"
}
{
        "_id" : ObjectId("65e9f979257325a977e114ce"),
        "empid" : "E002",
        "empname" : "Employee2",
        "salary" : 45000,
        "designation" : "Accountant"
}
{
        "_id" : ObjectId("65e9f979257325a977e114cf"),
        "empid" : "E003",
        "empname" : "Employee3",
        "salary" : 102000,
        "designation" : "Python Developer"
}
{
        "_id" : ObjectId("65e9f979257325a977e114d0"),
        "empid" : "E004",
        "empname" : "Employee4",
        "salary" : 92000,
        "designation" : "Manager"
}
```

```
{
        "_id" : ObjectId("65e9f979257325a977e114d1"),
        "empid" : "E005",
        "empname" : "Employee5",
        "salary" : 82000,
        "designation" : "Data Analyst"
}
{
        "_id" : ObjectId("65e9f979257325a977e114d2"),
        "empid" : "E006",
        "empname" : "Employee6",
        "salary" : 72000,
        "designation" : "Java Developer"
}
{
        "_id" : ObjectId("65e9f979257325a977e114d3"),
        "empid" : "E007",
        "empname" : "Employee7",
        "salary" : 62000,
        "designation" : "dotNET Developer"
}
{
        "_id" : ObjectId("65e9f979257325a977e114d4"),
        "empid" : "E008",
        "empname" : "Employee8",
        "salary" : 52000,
        "designation" : "Andriod Developer"
}
```

```
{
        "_id" : ObjectId("65e9f979257325a977e114d5"),
        "empid" : "E009",
        "empname" : "Employee9",
        "salary" : 45000,
        "designation" : "Accountant"
}
{
        "_id" : ObjectId("65e9f979257325a977e114d6"),
        "empid" : "E010",
        "empname" : "Employee10",
        "salary" : 32000,
        "designation" : "Manager"
}
```

Remove the documents of employees whose salary is greater than Rs100000.

```
> db.Staff.remove({salary : {$gt : 100000}});
WriteResult({ "nRemoved" : 2 })
> db.Staff.find().pretty();
{
        "_id" : ObjectId("65e9f979257325a977e114ce"),
        "empid" : "E002",
        "empname" : "Employee2",
        "salary" : 45000,
        "designation" : "Accountant"
}
{
        "_id" : ObjectId("65e9f979257325a977e114d0"),
        "empid" : "E004",
        "empname" : "Employee4",
        "salary" : 92000,
        "designation" : "Manager"
}
{
        "_id" : ObjectId("65e9f979257325a977e114d1"),
        "empid" : "E005",
        "empname" : "Employee5",
        "salary" : 82000,
        "designation" : "Data Analyst"
}
{
        "_id" : ObjectId("65e9f979257325a977e114d2"),
        "empid" : "E006",
        "empname" : "Employee6",
        "salary" : 72000,
        "designation" : "Java Developer"
}
```

```
{
        "_id" : ObjectId("65e9f979257325a977e114d3"),
        "empid" : "E007",
        "empname" : "Employee7",
        "salary" : 62000,
        "designation" : "dotNET Developer"
}
{
        "_id" : ObjectId("65e9f979257325a977e114d4"),
        "empid" : "E008",
        "empname" : "Employee8",
        "salary" : 52000,
        "designation" : "Andriod Developer"
}
{
        "_id" : ObjectId("65e9f979257325a977e114d5"),
        "empid" : "E009",
        "empname" : "Employee9",
        "salary" : 45000,
        "designation" : "Accountant"
}
{
        "_id" : ObjectId("65e9f979257325a977e114d6"),
        "empid" : "E010",
        "empname" : "Employee10",
        "salary" : 32000,
        "designation" : "Manager"
}
```

2. Create a database Institution .Create a Collection Student and Insert ten documents in it with fields: RollNo, Name, Class and TotalMarks(out of 500).

```
> use Institution;
switched to db Institution
> db.Student.insertMany([
... {RollNo : "S001", Name : "Ramtilak", Class : "MSC", TotalMarks : 500},
... {RollNo : "S002", Name : "Ram", Class : "MSC", TotalMarks : 499},
... {RollNo : "S003", Name : "Tilak", Class : "MSC", TotalMarks : 498},
... {RollNo : "S004", Name : "RAMTILAK", Class : "TYBSc CS", TotalMarks : 497},
... {RollNo : "S005", Name : "RAM", Class : "TYBSc CS", TotalMarks : 496},
... {RollNo : "S006", Name : "TILAK", Class : "TYBSc CS", TotalMarks : 495},
... {RollNo : "S007", Name : "Ayaan", Class : "MSC", TotalMarks : 402},
... {RollNo : "S008", Name : "Aryan", Class : "TYBSc CS", TotalMarks : 201},
... {RollNo : "S009", Name : "Ananya", Class : "MSC", TotalMarks : 196},
... {RollNo : "S010", Name : "Arya", Class : "TYBSc CS", TotalMarks : 193}
... ]);
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("65eabe8d257325a977e114d7"),
                ObjectId("65eabe8d257325a977e114d8"),
                ObjectId("65eabe8d257325a977e114d9"),
                ObjectId("65eabe8d257325a977e114da"),
                ObjectId("65eabe8d257325a977e114db"),
                ObjectId("65eabe8d257325a977e114dc"),
                ObjectId("65eabe8d257325a977e114dd"),
                ObjectId("65eabe8d257325a977e114de"),
                ObjectId("65eabe8d257325a977e114df"),
                ObjectId("65eabe8d257325a977e114e0")
        ]
}
```

Display all documents in Student

```
> db.Student.find();
{ "_id" : ObjectId("65eabe8d257325a977e114d7"), "RollNo" : "S001", "Name" : "Ramtilak", "Class" : "MSC", "TotalMarks" : 500 }
{ "_id" : ObjectId("65eabe8d257325a977e114d8"), "RollNo" : "S002", "Name" : "Ram", "Class" : "MSC", "TotalMarks" : 499 }
{ "_id" : ObjectId("65eabe8d257325a977e114d9"), "RollNo" : "S003", "Name" : "Tilak", "Class" : "MSC", "TotalMarks" : 498 }
{ "_id" : ObjectId("65eabe8d257325a977e114da"), "RollNo" : "S004", "Name" : "RAMTILAK", "Class" : "TYBSc CS", "TotalMarks" : 497 }
{ "_id" : ObjectId("65eabe8d257325a977e114db"), "RollNo" : "S005", "Name" : "RAM", "Class" : "TYBSc CS", "TotalMarks" : 496 }
{ "_id" : ObjectId("65eabe8d257325a977e114dc"), "RollNo" : "S006", "Name" : "TILAK", "Class" : "TYBSc CS", "TotalMarks" : 495 }
{ "_id" : ObjectId("65eabe8d257325a977e114dd"), "RollNo" : "S007", "Name" : "Ayaan", "Class" : "MSC", "TotalMarks" : 402 }
{ "_id" : ObjectId("65eabe8d257325a977e114de"), "RollNo" : "S008", "Name" : "Aryan", "Class" : "TYBSc CS", "TotalMarks" : 201 }
{ "_id" : ObjectId("65eabe8d257325a977e114df"), "RollNo" : "S009", "Name" : "Ananya", "Class" : "MSC", "TotalMarks" : 196 }
{ "_id" : ObjectId("65eabe8d257325a977e114e0"), "RollNo" : "S010", "Name" : "Arya", "Class" : "TYBSc CS", "TotalMarks" : 193 }
>
```

Sort the documents in descending order of TotalMarks.

```
> db.Student.find().sort({TotalMarks:-1});
{ "_id" : ObjectId("65eabe8d257325a977e114d7"), "RollNo" : "S001", "Name" : "Ramtilak", "Class" : "MSC", "TotalMarks" : 500 }
{ "_id" : ObjectId("65eabe8d257325a977e114d8"), "RollNo" : "S002", "Name" : "Ram", "Class" : "MSC", "TotalMarks" : 499 }
{ "_id" : ObjectId("65eabe8d257325a977e114d9"), "RollNo" : "S003", "Name" : "Tilak", "Class" : "MSC", "TotalMarks" : 498 }
{ "_id" : ObjectId("65eabe8d257325a977e114da"), "RollNo" : "S004", "Name" : "RAMTILAK", "Class" : "TYBSc CS", "TotalMarks" : 497 }
{ "_id" : ObjectId("65eabe8d257325a977e114db"), "RollNo" : "S005", "Name" : "RAM", "Class" : "TYBSc CS", "TotalMarks" : 496 }
{ "_id" : ObjectId("65eabe8d257325a977e114dc"), "RollNo" : "S006", "Name" : "TILAK", "Class" : "TYBSc CS", "TotalMarks" : 495 }
{ "_id" : ObjectId("65eabe8d257325a977e114dd"), "RollNo" : "S007", "Name" : "Ayaan", "Class" : "MSC", "TotalMarks" : 402 }
{ "_id" : ObjectId("65eabe8d257325a977e114de"), "RollNo" : "S008", "Name" : "Aryan", "Class" : "TYBSc CS", "TotalMarks" : 201 }
{ "_id" : ObjectId("65eabe8d257325a977e114df"), "RollNo" : "S009", "Name" : "Ananya", "Class" : "MSC", "TotalMarks" : 196 }
{ "_id" : ObjectId("65eabe8d257325a977e114e0"), "RollNo" : "S010", "Name" : "Arya", "Class" : "TYBSc CS", "TotalMarks" : 193 }
>
```

Display students of class “MSc” or marks greater than 400.

```
> db.Student.find({$or : [{Class:"MSC"},{TotalMarks : {$gt : 400}}]});
{ "_id" : ObjectId("65eabe8d257325a977e114d7"), "RollNo" : "S001", "Name" : "Ramtilak", "Class" : "MSC", "TotalMarks" : 500 }
{ "_id" : ObjectId("65eabe8d257325a977e114d8"), "RollNo" : "S002", "Name" : "Ram", "Class" : "MSC", "TotalMarks" : 499 }
{ "_id" : ObjectId("65eabe8d257325a977e114d9"), "RollNo" : "S003", "Name" : "Tilak", "Class" : "MSC", "TotalMarks" : 498 }
{ "_id" : ObjectId("65eabe8d257325a977e114da"), "RollNo" : "S004", "Name" : "RAMTILAK", "Class" : "TYBSc CS", "TotalMarks" : 497 }
{ "_id" : ObjectId("65eabe8d257325a977e114db"), "RollNo" : "S005", "Name" : "RAM", "Class" : "TYBSc CS", "TotalMarks" : 496 }
{ "_id" : ObjectId("65eabe8d257325a977e114dc"), "RollNo" : "S006", "Name" : "TILAK", "Class" : "TYBSc CS", "TotalMarks" : 495 }
{ "_id" : ObjectId("65eabe8d257325a977e114dd"), "RollNo" : "S007", "Name" : "Ayaan", "Class" : "MSC", "TotalMarks" : 402 }
{ "_id" : ObjectId("65eabe8d257325a977e114df"), "RollNo" : "S009", "Name" : "Ananya", "Class" : "MSC", "TotalMarks" : 196 }
```

Remove all the documents with TotalMarks<200

```
> db.Student.remove({TotalMarks : {$lt : 200}})
WriteResult({ "nRemoved" : 2 })
> db.Student.find();
{ "_id" : ObjectId("65eabe8d257325a977e114d7"), "RollNo" : "S001", "Name" : "Ramtilak", "Class" : "MSC", "TotalMarks" : 500 }
{ "_id" : ObjectId("65eabe8d257325a977e114d8"), "RollNo" : "S002", "Name" : "Ram", "Class" : "MSC", "TotalMarks" : 499 }
{ "_id" : ObjectId("65eabe8d257325a977e114d9"), "RollNo" : "S003", "Name" : "Tilak", "Class" : "MSC", "TotalMarks" : 498 }
{ "_id" : ObjectId("65eabe8d257325a977e114da"), "RollNo" : "S004", "Name" : "RAMTILAK", "Class" : "TYBSc CS", "TotalMarks" : 497 }
{ "_id" : ObjectId("65eabe8d257325a977e114db"), "RollNo" : "S005", "Name" : "RAM", "Class" : "TYBSc CS", "TotalMarks" : 496 }
{ "_id" : ObjectId("65eabe8d257325a977e114dc"), "RollNo" : "S006", "Name" : "TILAK", "Class" : "TYBSc CS", "TotalMarks" : 495 }
{ "_id" : ObjectId("65eabe8d257325a977e114dd"), "RollNo" : "S007", "Name" : "Ayaan", "Class" : "MSC", "TotalMarks" : 402 }
{ "_id" : ObjectId("65eabe8d257325a977e114de"), "RollNo" : "S008", "Name" : "Aryan", "Class" : "TYBSc CS", "TotalMarks" : 201 }
> 
```

SIES College of Arts, Science & Commerce

RISE WITH EDUCATION

NAAC REACCREDITED - 'A' GRADE

ISO 9001 : 2008

S.I.E.S College of Arts, Science and Commerce
Sion(W), Mumbai – 400 022.

CERTIFICATE

This is to certify that Mr. / Miss. NADAR RAMTILAK SAIT SANKARALINGAM.
Roll No. **TCS2324047**Has successfully completed the necessary course of experiments in the subject of during the academic year **2023 – 2024** complying with the requirements of **University of Mumbai**, for the course of **T.Y.BSc. Computer Science [Semester-VI]**

Prof. In-Charge
MAYA NAIR

Examination Date:
Examiner's Signature & Date:

Head of the Department
Prof. Manoj Singh

College Seal
And
Date