

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

**CERTIFICATE**

This is to certify that Mr. **NADAR KIBSON SAMUEL P JOSEPH**  
Roll No. **TCS2324045** has successfully completed the necessary course of  
experiments in the subject of **Data Science** during the academic year **2023 –**  
**2024** complying with the requirements of **University of Mumbai**, for the  
course of **TYBSc Computer Science [Semester-VI]**.

Prof. In-Charge  
**Maya Nair**

Examination date:

Examiner's Signature & Date:

Head of the Department  
**Prof. Manoj Singh**

College Seal

### Index Page

Sr. No	Description	Page No	Date	Faculty Signature
1	Write a python program to plot word cloud for a wikipedia page of any topic.	3	12/12/23	
2	Write a python program to perform Web Scrapping	5	19/12/23	
3	Exploratory Data Analysis of mtcars.csv Dataset in R ( Use functions of dplyr like select, filter, mutate , rename, arrange, group by, summarize and data visualizations)	8	09/1/24	
4	Exploratory data analysis in Python using Titanic Dataset	19	23/1/24	
5	Exploratory data analysis in Python using Titanic Dataset	27	23/1/24	
6	Write a python program to implement multiple linear regression on the Dataset Boston.csv	31	30/2/24	
7	Write a python program to implement KNN algorithm to predict breast cancer using breast cancer wisconsin dataset	35	13/2/24	
8	Introduction to NOSQL using MongoDB	39	05/2/24	

## Practical 1

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

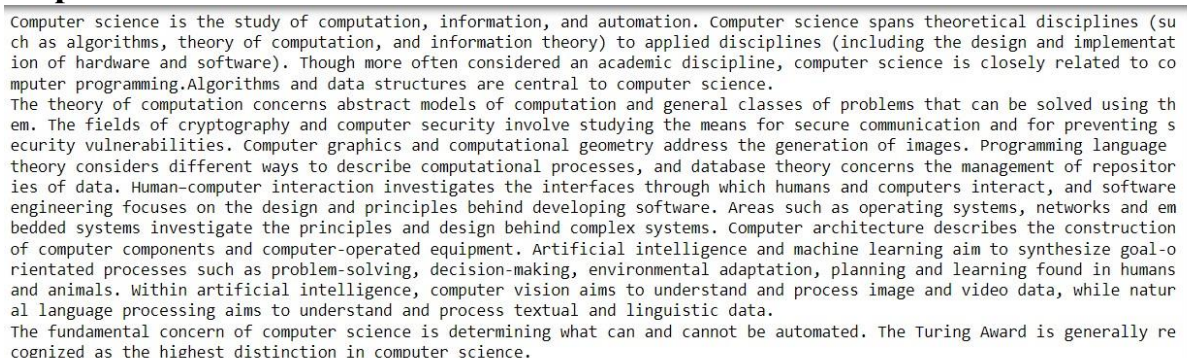
```
Code: from wordcloud import WordCloud,
STOPWORDS
import matplotlib.pyplot as plt
import wikipedia as wp

result = wp.page('Computer Science')
final_result = result.content print(final_result)

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="blue",
random_state=10,stopwords=STOPWORDS).generate(final_result)
wc.to_file("cs.png")
plot_wordcloud(wc)
```

## Output:



Computer science is the study of computation, information, and automation. Computer science spans theoretical disciplines (such as algorithms, theory of computation, and information theory) to applied disciplines (including the design and implementation of hardware and software). Though more often considered an academic discipline, computer science is closely related to computer programming. Algorithms and data structures are central to computer science.

The theory of computation concerns abstract models of computation and general classes of problems that can be solved using them. The fields of cryptography and computer security involve studying the means for secure communication and for preventing security vulnerabilities. Computer graphics and computational geometry address the generation of images. Programming language theory considers different ways to describe computational processes, and database theory concerns the management of repositories of data. Human-computer interaction investigates the interfaces through which humans and computers interact, and software engineering focuses on the design and principles behind developing software. Areas such as operating systems, networks and embedded systems investigate the principles and design behind complex systems. Computer architecture describes the construction of computer components and computer-operated equipment. Artificial intelligence and machine learning aim to synthesize goal-oriented processes such as problem-solving, decision-making, environmental adaptation, planning and learning found in humans and animals. Within artificial intelligence, computer vision aims to understand and process image and video data, while natural language processing aims to understand and process textual and linguistic data.

The fundamental concern of computer science is determining what can and cannot be automated. The Turing Award is generally recognized as the highest distinction in computer science.



## **Practical 2:**

**Web scraping :** Web scraping is the process of collecting and parsing raw data from the Web.

**Aim:** Write a python program to perform Web Scrapping

### **01.Html scrapping- use BeautifulSoup**

#### **Code:**

```
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")
print(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
Name"]=Country
countries_df["Area"] = Area print(countries_df.head(10))
```

#### **Output:**

```

<table class="sorttable wikitables sticky-header col2left" style="text-align: center">
<tbody><tr>
<th></th>
<th>Country / dependency</th>
<th>%<br>total</th>
<th>Asia area<br>in km<sup>2</sup> (mi<sup>2</sup>)</th>
<th class="unsortable">
</th></tr>
<tr>
<td>1</td>
<td><span class="flagicon"><span class="mw-image-border" typeof="mw:File"><span></span></span> </span><a href="/wiki/Russia" title="Russia">Russia</a></td>
<td>29.3%</td>
<td><span data-sort-value="7013130831000000000"></span>13,083,100 (5,051,400)</td>
<td><sup class="reference" id="cite_ref-3"><a href="#cite_note-3">[a]</a></sup>
</td></tr>

```

	ID	Country Name	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)

## 02.json scrapping

### Code:

```

import pandas as pd
import urllib.request
import 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")

user_df = pd.DataFrame()
user_df["User ID"] = id
user_df["User Name"] = username
user_df["Email Address"] = email
print(user_df.head(10))

```

### Output:

	User ID	User Name	Email Address
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 3:

**Aim:** Exploratory Data Analysis of mtcars.csv Dataset in R ( Use functions of dplyr like select, filter, mutate , rename, arrange, group by, summarize and data visualizations) mtcars.csv:

Motor Trend Car Road Tests-The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973--74 models).

Format

A data frame with 32 observations on 12 (numeric) variables.

[, 1]	mpg	Miles/(US) gallon
[, 2]	cyl	Number of cylinders
[, 3]	disp	Displacement (cu.in.)
[, 4]	hp	Gross horsepower
[, 5]	drat	Rear axle ratio
[, 6]	wt	Weight (1000 lbs)
[, 7]	qsec	1/4 mile time
[, 8]	vs	Engine (0 = V-shaped, 1 = straight)
[, 9]	am	Transmission (0 = automatic, 1 = manual)
[,10]	gear	Number of forward gears
[,11]	Carb	Number of carburetors

Motor Trend is a magazine about the automobile industry. Looking at a data set of a collection of cars, they are interested in exploring the relationship between a set of variables and miles per gallon (MPG) (outcome).



**Code: #Do In**

Rstudio

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

```
View(cars_df) str(cars_df)
```

```
dim(cars_df) names(cars_df)
```

```
row.names(cars_df)
```

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

```
cars_df=cars_df[,-1] View(cars_df)
```

```
library(dplyr)
```

```
#Select fuction - for extracting specific columns
```

```
#df1=select(cars_df,mpg:hp)
```

```
df1=cars_df %>% select(mpg:hp) #pipe of dplyr it will take out content of one  
column to the output of other column View(df1)
```

```
df1 = cars_df %>% select(c(mpg,disp,wt,gear))
```

```
View(df1)
```

```
#Filter function - for extracting specific rows or observation
```

```
#extract record where gears=4 and columns to be displayed are mpg, disp, wt  
and gears. df1 = cars_df %>% filter(gear==4) %>%
```

```
select(c(mpg,disp,wt,gear))
```

```
View(df1)
```

```
# extract record where cyl=4 or mpg>20 and columns are required are mpg,cyl  
df1 = cars_df %>% filter(cyl==4 | mpg > 20) %>% select(c(mpg,cyl))
```

```
View(df1)
```

```
#extract records where mpg<20 and carb = 3 and coumns needed are mpg and  
carb
```

```

df1 = cars_df %>% filter(mpg < 20 & carb == 3) %>% select(c(mpg,carb))
view(df1)

# Arrange function -Sort as per specific columns df1
=cars_df %>% arrange(cyl,desc(mpg))
View(df1)

#Rename function - change names of one or more column df1
= cars_df %>%
rename(MilesPerGallon=mpg,Cylinders=cyl,Displacement=disp)
View(df1)

#Mutate function - creating new columns on the basis of existing column df1
= cars_df %>% mutate(Power=hp*wt)
View(df1)

#Group_by and summaries - segregating data as per categorical variable and
summarizing
df1$gear = as.factor(df1$gear) str(df1)
summary_df = df1 %>% group_by(df1$gear) %>% summarise(no=n(),
mean_mpg=mean(mpg), mean_wt=mean(wt)) summary_df
summary_df = df1 %>% group_by(df1$Cylinders) %>% summarise(no=n(),
mean_mpg=mean(mpg), mean_wt=mean(wt)) summary_df
#Data Visualization
#histogram - for single column frequency hist(df1$mpg,
main="Histogeam of
MilePergallon(mtcars)",col="lightgreen",xlab="Miles Per Gallon")
#box plot - diagrammatic representation of summary
summary(df1$mpg) boxplot(df1$mpg)

```

#bar plot - categorical variable representation'

```
table(df1$gear) barplot(table(df1$gear))
```

#scatter plot - plot() - plots relationship between two variable

```
plot(df1$mpg~df1$disp) plot(df1$mpg~df1$cyl)
```

```
plot(df1$mpg~df1$wt)
```

## Output:

	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

```
'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 ...
```

```
[1] 32 12
```

```
[1] "model" "mpg" "cyl" "disp" "hp" "drat" "wt" "qsec" "vs" "am" "gear"
```

```
[12] "carb"
```

```
[1] "1" "2" "3" "4" "5" "6" "7" "8" "9" "10" "11" "12" "13" "14" "15" "16" "17" "18"
```

```
[19] "19" "20" "21" "22" "23" "24" "25" "26" "27" "28" "29" "30" "31" "32"
```

	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

	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

	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

	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

	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

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

	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

Name of Instructor: Maya Nair



	MilesPerGallon	Cylinders	Displacement	hp	drat	wt	qsec	vs	am	gear	ci
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	Power
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4	288.200
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4	316.250
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1	215.760
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1	353.650
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2	602.000
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1	363.300
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4	874.650
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2	197.780
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2	299.250
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4	423.120
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4	423.120
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3	732.600
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3	671.400
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3	680.400

'data.frame': 32 obs. of 12 variables:

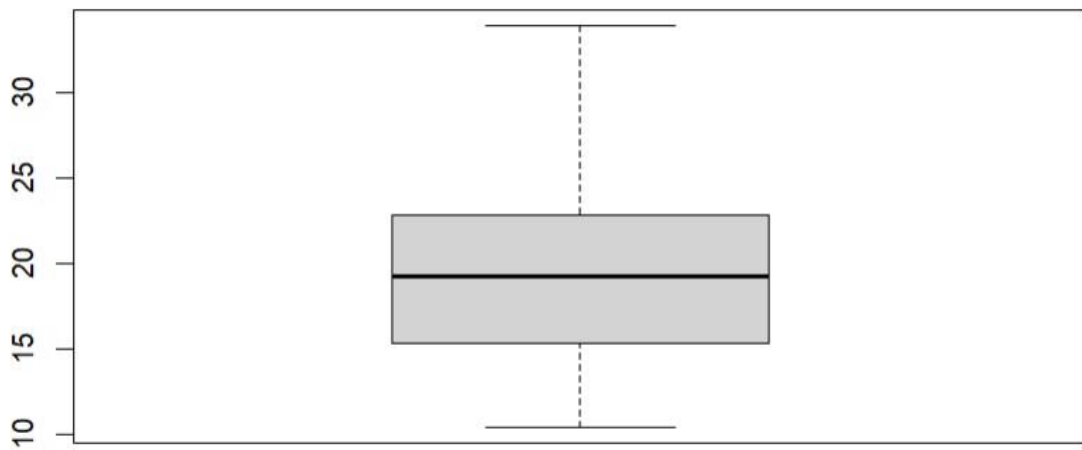
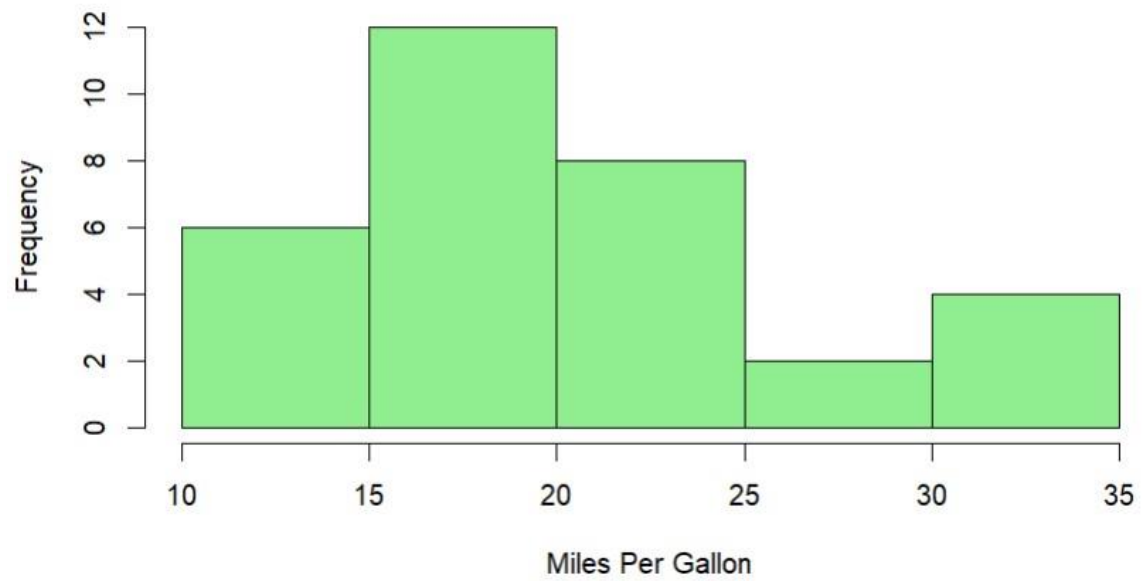
```
$ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
$ cyl : int 6 6 4 6 8 6 8 4 4 6 ...
$ disp : num 160 160 108 258 360 ...
$ hp : int 110 110 93 110 175 105 245 62 95 123 ...
$ drat : num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
$ wt : num 2.62 2.88 2.32 3.21 3.44 ...
$ qsec : 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 ...
$ Power: num 288 316 216 354 602 ...
```

# Summary

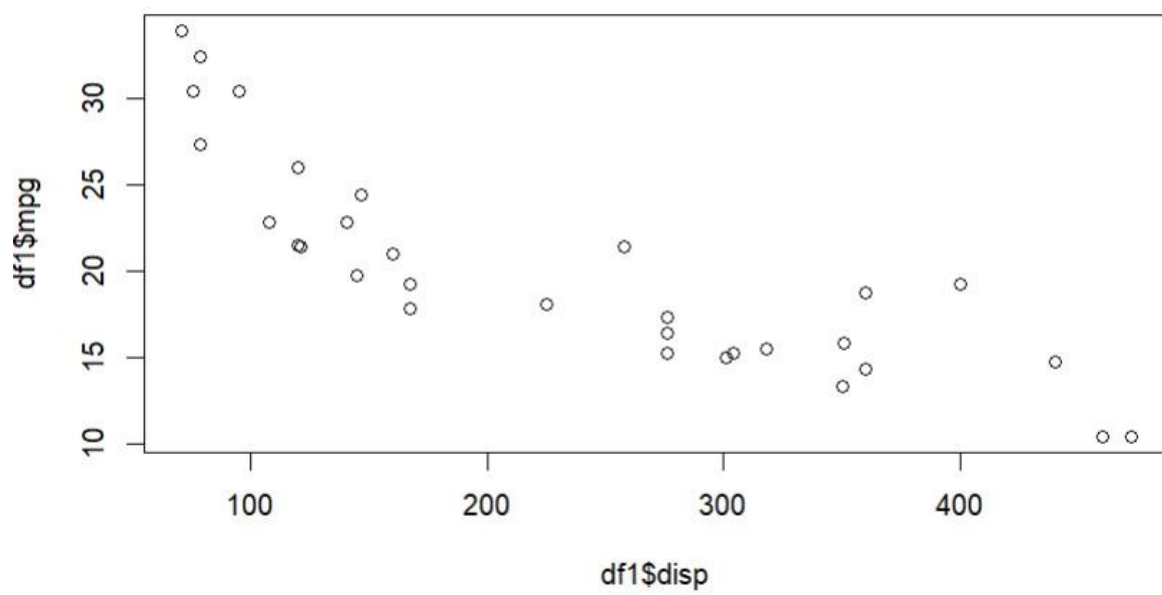
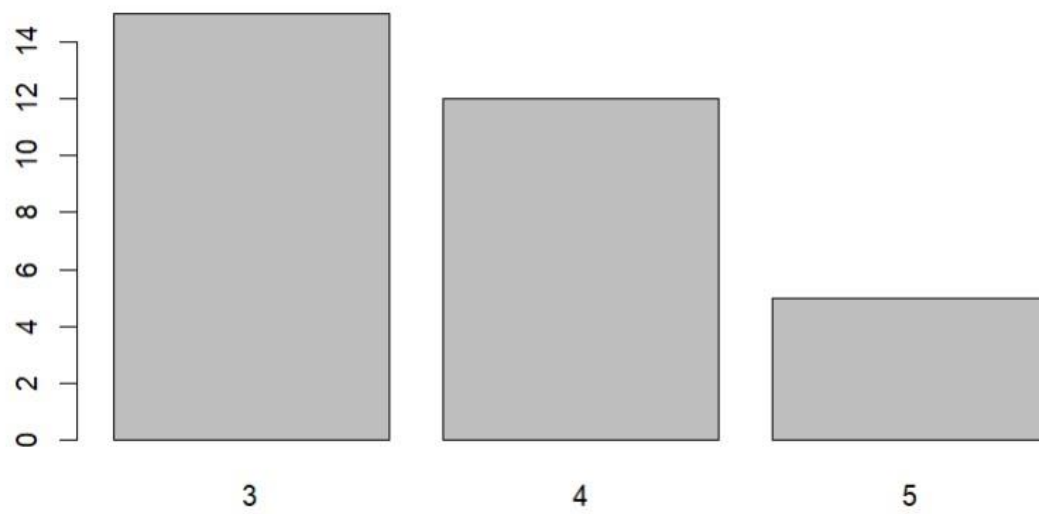
# A tibble: 3 x 4

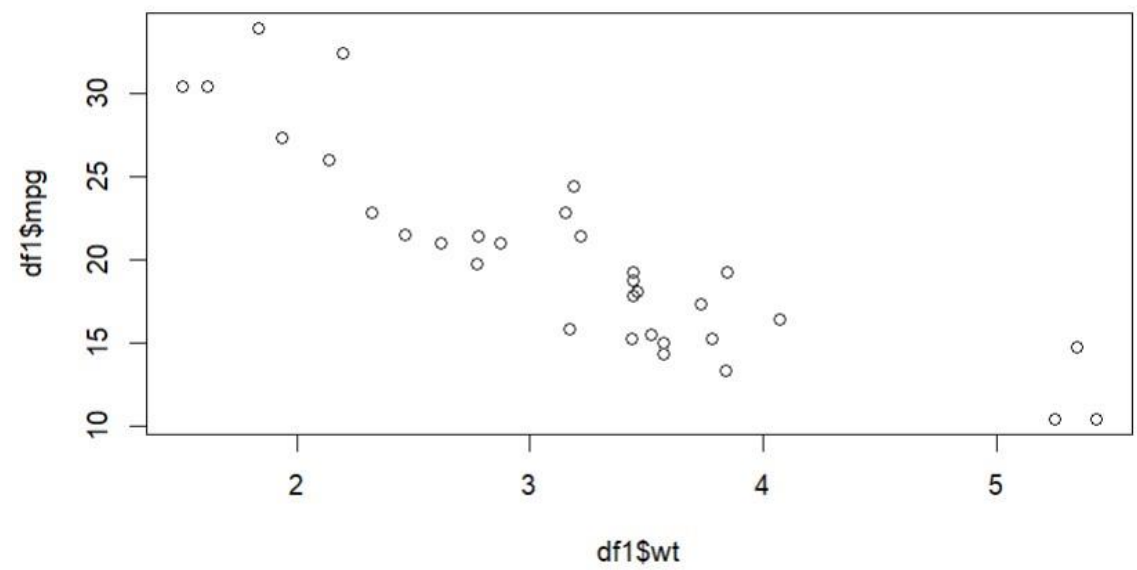
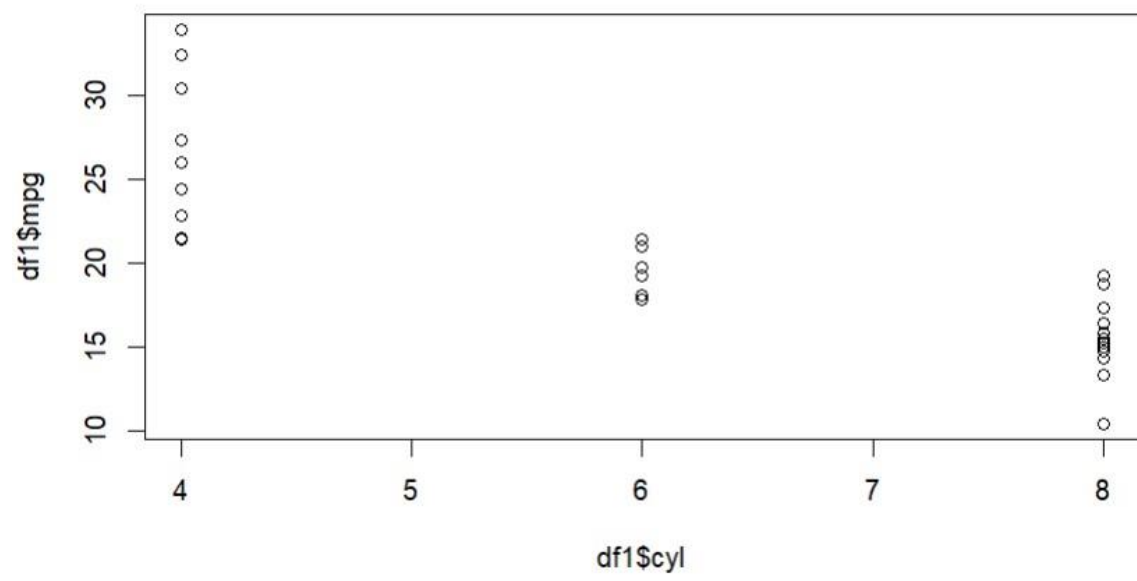
```
`df1$gear` no mean_mpg mean_wt
<fct> <int> <dbl> <dbl>
1 3 15 16.1 3.89
2 4 12 24.5 2.62
3 5 5 21.4 2.63
```

**Histogram of MilePerGallon(mtcars)**









## Practical No:4

**Aim:** Exploratory data analysis in Python using Titanic Dataset **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.**

Data Dictionary

Variable	Definition	Key
Survival	Survival	0 = No, 1 = Yes
Pclass	Ticket class	1 = 1st, 2 = 2nd, 3 = 3rd
Sex	Sex	
Age	Age in years	
Sibsp	# of siblings / spouses aboard the Titanic	
Parch	# of parents / children aboard the Titanic	
Ticket	Ticket number	
Fare	Passenger fare	
Cabin	Cabin number	
Embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

## 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:

```
import pandas as pd
titanic = pd.read_csv("train.csv")
titanic.head()
titanic.info()
titanic.describe()
titanic.isnull().sum()
titanic_cleaned = titanic.drop(['PassengerId', 'Name', 'Ticket', 'Fare', 'Cabin'], axis=1)
titanic_cleaned.info()
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.catplot(x="Sex", hue="Survived", kind="count", data=titanic_cleaned)
titanic_cleaned.groupby(['Sex', 'Survived'])['Survived'].count()
group1 = titanic_cleaned.groupby(['Sex', 'Survived'])
sns.heatmap(titanic_cleaned.groupby(['Sex', 'Survived']).get_group('male', 'Survived'), annot=True, fmt="d")
sns.heatmap(titanic_cleaned.groupby(['Sex', 'Survived']).get_group('female', 'Survived'), annot=True, fmt="d")
sns.violinplot(x="Sex", y="Age", hue="Survived", data=titanic_cleaned, split=True)
print("Oldest Person on Board:", titanic_cleaned['Age'].max())
print("Youngest Person on Board:", titanic_cleaned['Age'].min())
print("Average age of Person on Board:", titanic_cleaned['Age'].mean())
titanic_cleaned.isnull().sum()
def impute(cols):
    Age = cols[0]
    Pclass = cols[1]
    if pd.isnull(Age):
        if Pclass == 1:
            return 38
        elif Pclass == 2:
            return 29
        else:
            return 24
    else:
        return Age
titanic_cleaned['Age'] = titanic_cleaned[['Age', 'Pclass']].apply(impute, axis=1)
titanic_cleaned.isnull().sum()
titanic_cleaned.corr(method='pearson')
sns.heatmap(titanic_cleaned.corr(method="pearson"), annot=True, vmax=1)
import numpy as np
from sklearn import datasets
x, y, coef = datasets.make_regression(n_samples=100, n_features=1, n_informative=1, noise=10, coef=True, random_state=0)
```

```

x=np.interp(x,(x.min(),x.max()),(0,20))
print(len(x)) print(x)
y=np.interp(y,(y.min(),y.max()),(20000,150000)) print(len(y))
print(y)

```

## Output:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```

<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

```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

```

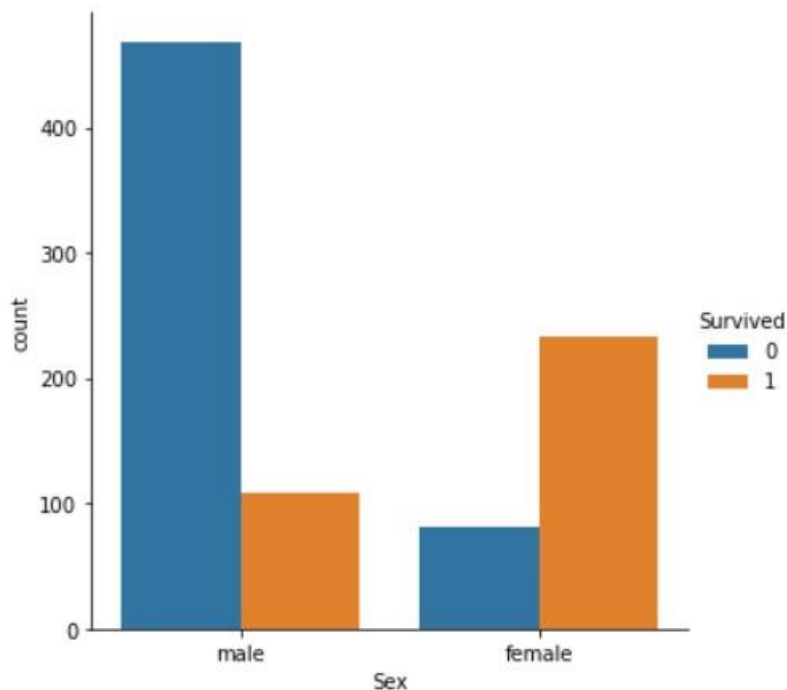
PassengerId    0
Survived        0
Pclass         0
Name           0
Sex            0
Age           177
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin         687
Embarked       2
dtype: int64

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 7 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Survived    891 non-null    int64
1   Pclass      891 non-null    int64
2   Sex         891 non-null    object
3   Age         714 non-null    float64
4   SibSp       891 non-null    int64
5   Parch       891 non-null    int64
6   Embarked    889 non-null    object
dtypes: float64(1), int64(4), object(2)
memory usage: 48.9+ KB

```



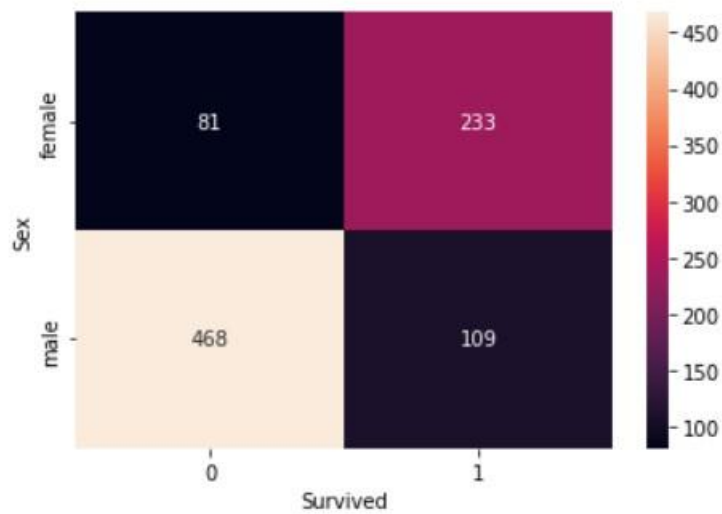
```

: Sex      Survived
  female  0          81
         1         233
  male    0         468
         1         109
Name: Survived, dtype: int64

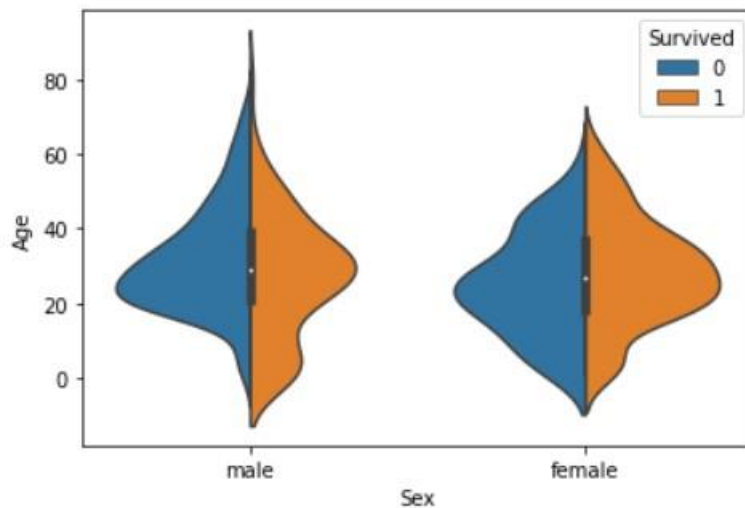
```

Survived		0	1
Sex			
female		81	233
male		468	109

```
<AxesSubplot:xlabel='Survived', ylabel='Sex'>
```



```
<AxesSubplot:xlabel='Sex', ylabel='Age'>
```



Oldest Person on Board: 80.0

Youngest Person on Board: 0.42

---

Average age of Person on Board: 29.69911764705882

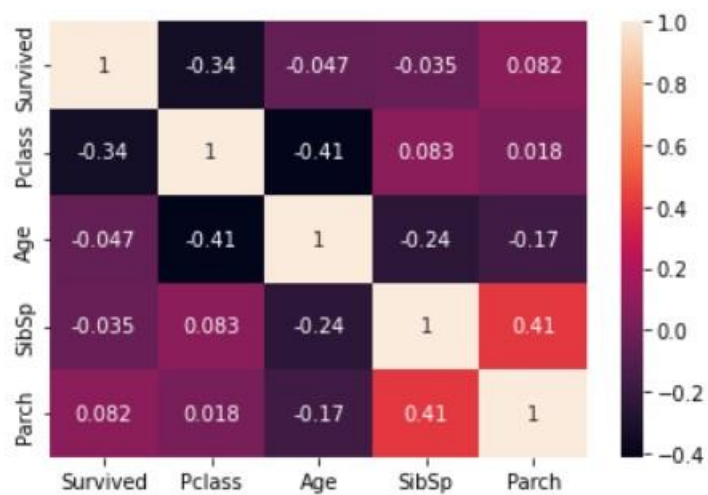
```
Survived    0
Pclass      0
Sex         0
Age        177
SibSp       0
Parch       0
Embarked    2
dtype: int64
```



```
Survived    0
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Embarked    2
dtype: int64
```

	Survived	Pclass	Age	SibSp	Parch
Survived	1.000000	-0.338481	-0.046746	-0.035322	0.081629
Pclass	-0.338481	1.000000	-0.411805	0.083081	0.018443
Age	-0.046746	-0.411805	1.000000	-0.243877	-0.171917
SibSp	-0.035322	0.083081	-0.243877	1.000000	0.414838
Parch	0.081629	0.018443	-0.171917	0.414838	1.000000

<AxesSubplot:>



100

```
[ 9.09621765]
[14.63742853]
[12.25580785]
[ 7.21515957]
[ 6.90562848]
[12.42799856]
[ 6.53450315]
[12.36358975]
[11.45101022]
[ 9.29527704]
[ 8.46897323]
[11.11359701]
[ 4.21646281]
[ 8.92109838]
[13.29785748]
[15.47570863]
[ 9.84113925]
[17.99332461]
[16.61818648]
[ 7.74737185]
```

100

```
[ 78311.16075377 103897.6645258 97836.26101499 80550.25638039
 68555.820963 108021.44227128 55778.0199934 101586.97979347
103966.61856971 76826.00913959 73657.03907056 96439.33831133
 43282.85644907 73119.73495559 109692.0380975 128125.74670244
 87499.26503386 136438.82955292 140414.06203468 75920.22641562
122765.94046351 138676.79599883 90840.21480164 99453.36502726
118663.17132396 125247.52951645 144470.99004202 98454.6493064
 92321.3919241 133162.35931048 61723.07434352 77095.35501897
 59042.68149761 109559.00643186 77206.62874325 109743.44545302
103902.53136675 82585.66146856 81088.97054957 62200.35300958
111971.74647069 101515.0451792 47090.60230288 141613.36480828
 99370.060872 72953.14343772 131312.34257614 68957.25418311
135509.14233685 90658.86260334 75147.59074288 46071.12989863
 01120 01042552 105126 42540022 110217 0745170 102720 70107702
```

## Practical 5

**Aim:** Exploratory data analysis in Python using Titanic Dataset The following figure illustrates simple linear regression:

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

**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.**

### Code:

```
import numpy as np from sklearn import datasets
x,y,coef=datasets.make_regression(n_samples=100, n_features=1,
n_informative=1, noise=10,coef=True, random_state = 0)
x=np.interp(x,(x.min(),x.max()),(0,20))
print(len(x)) print(x)
y=np.interp(y,(y.min(),y.max()),(20000,150000))
print(len(y)) print(y) import matplotlib.pyplot as
plt plt.plot(x,t,'.',label="training data")
plt.xlabel("Years of Experience")
plt.ylabel("Salary") plt.title("Experience vs
Salary") from sklearn.linear_model import
LinearRegression reg_model = LinearRegression()
reg_model.fit(x,y) y_pred=reg_model.predict(x)
plt.plot(x,y_pred,color="black")
plt.plot(x,y,'.',label="training data")
plt.xlabel("Years of Experience")
plt.ylabel("Salary") plt.title("Experience vs
Salary")
import pandas as pd
data = {'Experience':np.round(x.flatten()),'Salary':np.round(y)}
df=pd.DataFrame(data)
df.head(10)
```

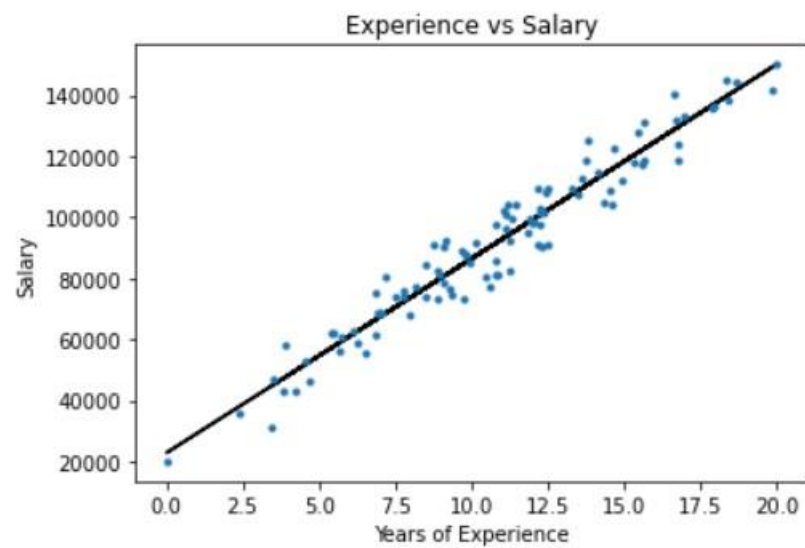
### Output:

---

```
Text(0.5, 1.0, 'Experience vs Salary')
```



```
Text(0.5, 1.0, 'Experience vs Salary')
```



	Experience	Salary
0	9.0	78311.0
1	15.0	103898.0
2	12.0	97836.0
3	7.0	80550.0
4	7.0	68556.0
5	12.0	108021.0
6	7.0	55778.0
7	12.0	101587.0
8	11.0	103967.0
9	9.0	76826.0

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.

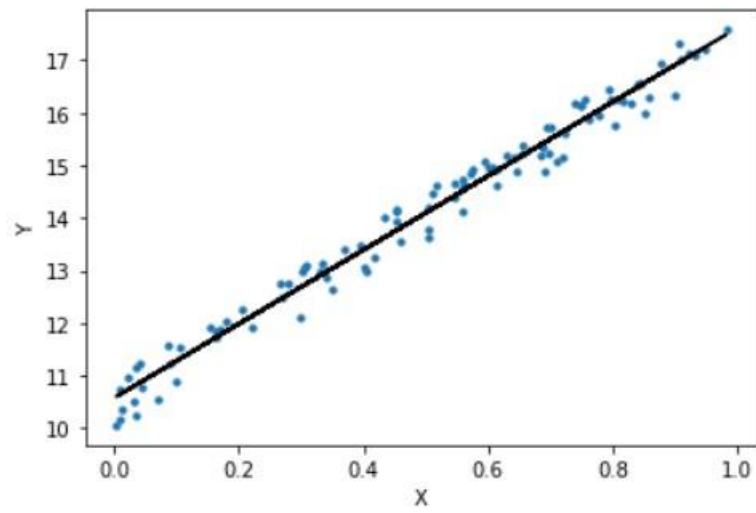
**Code:**

```
x1=[[13.0]] y1=reg_model.predict(x1)
print(np.round(y1))
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_pred=reg_model1.predict(x)
plt.scatter(x,y,s=10)
plt.xlabel("X") plt.ylabel("Y")
plt.plot(x,y_pred,color="black")
```

**Output:**

```
[105534.]
```

[<matplotlib.lines.Line2D at 0x26fbd74c8b0>]



## Practical 6

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

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  
\\$10,000.

*ptratio* pupil-teacher ratio  
by town.

*black*

$1000(B_k - 0.63)^2$  where  $B_k$  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:

```
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")
boston.info()
boston_x = pd.DataFrame(boston.iloc[:,13])
boston_y = pd.DataFrame(boston.iloc[:,-1])
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("xtrain shape", X_train.shape) print("ytrain shape",
Y_train.shape) print("xtest shape", X_test.shape) print("ytest shape",
Y_test.shape)
from sklearn.linear_model import LinearRegression
regression=LinearRegression() regression.fit(X_train,Y_train)
Y_pred_linear = regression.predict(X_test)
Y_pred_df = pd.DataFrame(Y_pred_linear,columns=["Predicted"])
Y_pred_df.head()
plt.scatter(Y_test, Y_pred_linear, c="green")
plt.xlabel("Actual Price(medv)")
plt.ylabel("Predicted Pric(medv)")
plt.title("Actual vs Prediction") plt.show()

```

## Output:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2



```

<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

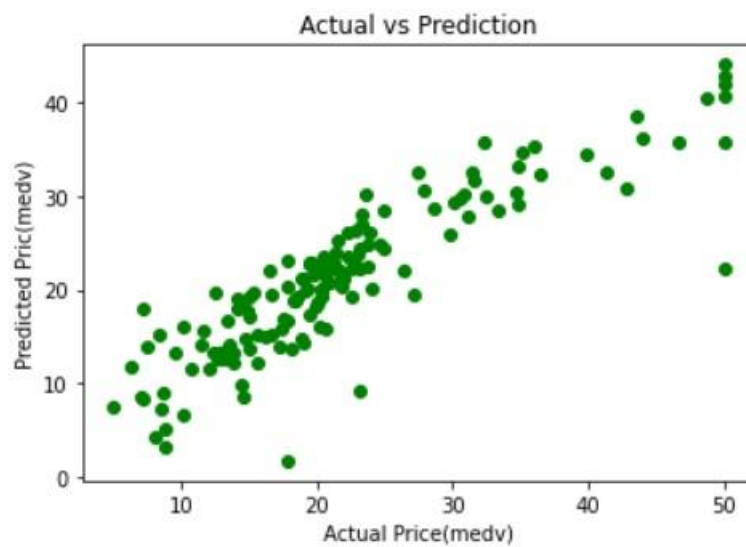
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33

	medv
0	24.0
1	21.6
2	34.7
3	33.4
4	36.2

```
xtrain shape (354, 13)
ytrain shape (354, 1)
xtest shape (152, 13)
ytest shape (152, 1)
```

	Predicted
0	22.984672
1	23.549732
2	19.280036
3	25.237437
4	36.235906



## Practical 7:

### K Nearest Neighbor classification Algorithm

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

### 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 ( $\text{perimeter}^2 / \text{area} - 1.0$ )  
g) concavity (severity of concave portions of the contour)

h) concave points (number of concave contour)  
portions of the i) symmetry

j) fractal dimension ("coastline approximation" - 1)

### Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
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()

columns_to_select = ["mean area", "mean compactness"]
x = x[columns_to_select]
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)
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"))

y_pred = knn.predict(X_test)
plt.scatter(X_test["mean area"], X_test["mean compactness"], c=y_pred, cmap="coolwarm", alpha=0.7)

cf = confusion_matrix(Y_test, y_pred)
print(cf)

labels = ["True Negative", "False Positive", "False Negative", "True Positive"]
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:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter	worst area	worst smoothness
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	25.38	17.33	184.60	2019.0	0.1622
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	24.99	23.41	158.80	1956.0	0.1238
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	23.57	25.53	152.50	1709.0	0.1444
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	14.91	26.50	98.87	567.7	0.2098
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	22.54	16.67	152.20	1575.0	0.1374

5 rows x 30 columns

	mean area	mean compactness
0	1001.0	0.27760
1	1326.0	0.07864
2	1203.0	0.15990
3	386.1	0.28390
4	1297.0	0.13280

```
['malignant', 'malignant', 'malignant', 'malignant', 'malignant', ..., 'malignant', 'malignant', 'malignant', 'malignant', 'benign']
```

Length: 569

Categories (2, object): ['malignant', 'benign']

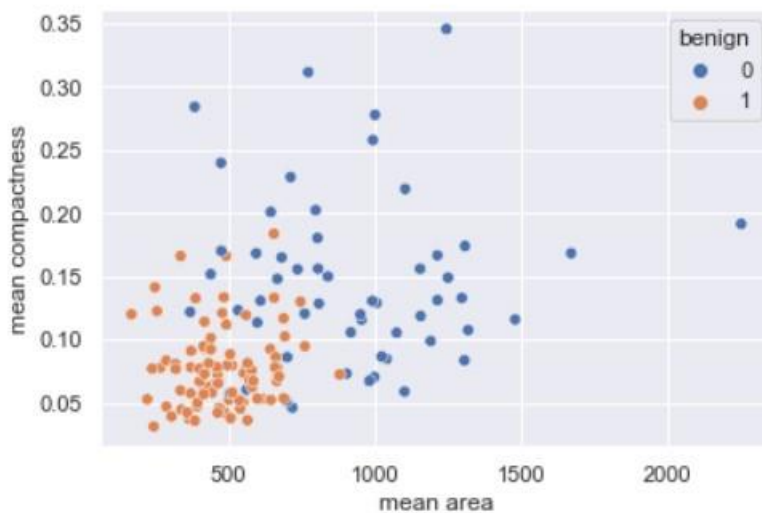
```

benign
0      0
1      0
2      0
3      0
4      0
..      ...
564    0
565    0
566    0
567    0
568    1

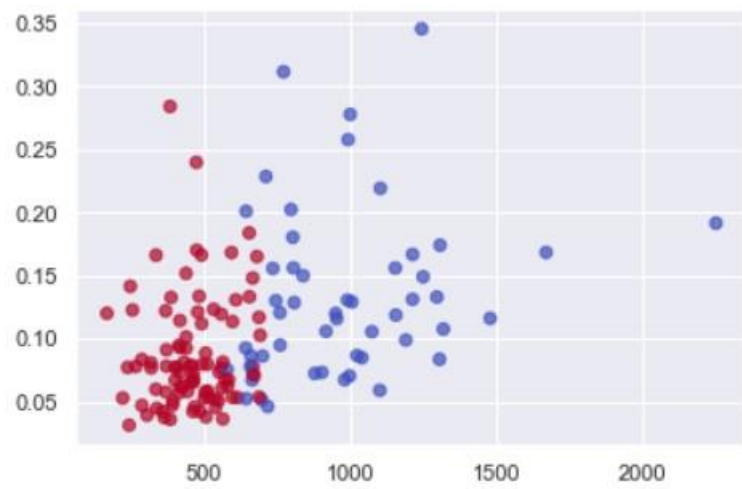
```

[569 rows x 1 columns]

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

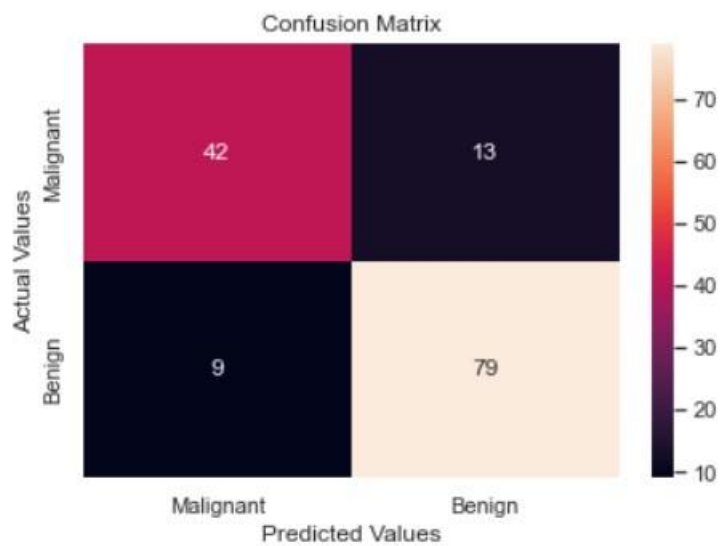


```
<matplotlib.collections.PathCollection at 0x260054d1fa0>
```



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

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



## Practical 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.

- Display all documents in Staff and display only empid and designation.db
- Sort the documents in descending order of Salary
- Display employee with designation with “Manager” or salary greater than Rs. 50,000/-.
- Update the salary of all employees with designation as “Accountant” to Rs.45000.
- Remove the documents of employees whose salary is greater than Rs100000.

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).

- Display all documents in Student.
- Sort the documents in descending order of TotalMarks.
- Display students of class “MSc” or marks greater than 400.
- Remove all the documents with TotalMarks<200

### Code and Output:

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

use Institution db.createCollection(“Staff”)

```
> use Institution
switched to db Institution
> db.createCollection("Staff")
```

db

```
db.Staff.insertMany([ { "empid": 1, "empname": "John Doe", "salary": 60000,
"designation": "Manager" }, { "empid": 2, "empname": "Jane Smith", "salary":
55000, "designation": "Accountant" }, { "empid": 3, "empname": "Michael
Johnson", "salary": 70000, "designation": "Manager" }, { "empid": 4,
```



```
"empname": "Emily Brown", "salary": 45000, "designation": "Accountant" }, {
"empid": 5, "empname": "David Wilson", "salary": 80000, "designation":
"Developer" }, { "empid": 6, "empname": "Sarah Lee", "salary": 95000,
"designation": "Manager" }, { "empid": 7, "empname": "Christopher Martinez",
"salary": 50000, "designation": "Accountant" }, { "empid": 8, "empname":
"Amanda Davis", "salary": 120000, "designation": "Manager" }, { "empid": 9,
"empname": "Jason Rodriguez", "salary": 40000, "designation": "Intern" }, {
"empid": 10, "empname": "Jessica Taylor", "salary": 110000, "designation":
"Manager" } ])
```

```
> db
Institution
> db.Staff.insertMany([
...   { "empid": 1, "empname": "John Doe", "salary": 60000, "designation": "Manager" },
...   { "empid": 2, "empname": "Jane Smith", "salary": 55000, "designation": "Accountant" },
...   { "empid": 3, "empname": "Michael Johnson", "salary": 70000, "designation": "Manager" },
...   { "empid": 4, "empname": "Emily Brown", "salary": 45000, "designation": "Accountant" },
...   { "empid": 5, "empname": "David Wilson", "salary": 80000, "designation": "Developer" },
...   { "empid": 6, "empname": "Sarah Lee", "salary": 95000, "designation": "Manager" },
...   { "empid": 7, "empname": "Christopher Martinez", "salary": 50000, "designation": "Accountant" },
...   { "empid": 8, "empname": "Amanda Davis", "salary": 120000, "designation": "Manager" },
...   { "empid": 9, "empname": "Jason Rodriguez", "salary": 40000, "designation": "Intern" },
...   { "empid": 10, "empname": "Jessica Taylor", "salary": 110000, "designation": "Manager" }
... ])
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("65deecde8cbdbaebf7136c4f"),
    ObjectId("65deecde8cbdbaebf7136c50"),
    ObjectId("65deecde8cbdbaebf7136c51"),
    ObjectId("65deecde8cbdbaebf7136c52"),
    ObjectId("65deecde8cbdbaebf7136c53"),
    ObjectId("65deecde8cbdbaebf7136c54"),
    ObjectId("65deecde8cbdbaebf7136c55"),
    ObjectId("65deecde8cbdbaebf7136c56"),
    ObjectId("65deecde8cbdbaebf7136c57"),
    ObjectId("65deecde8cbdbaebf7136c58")
  ]
}
```

```
Db.Staff.find().pretty()
```



```

> db.Staff.find().pretty()
{
  "_id" : ObjectId("65deecde8cbdbaebf7136c4f"),
  "empid" : 1,
  "empname" : "John Doe",
  "salary" : 60000,
  "designation" : "Manager"
}
{
  "_id" : ObjectId("65deecde8cbdbaebf7136c50"),
  "empid" : 2,
  "empname" : "Jane Smith",
  "salary" : 55000,
  "designation" : "Accountant"
}
{
  "_id" : ObjectId("65deecde8cbdbaebf7136c51"),
  "empid" : 3,
  "empname" : "Michael Johnson",
  "salary" : 70000,
  "designation" : "Manager"
}
{
  "_id" : ObjectId("65deecde8cbdbaebf7136c52"),
  "empid" : 4,
  "empname" : "Emily Brown",
  "salary" : 45000,
  "designation" : "Accountant"
}

```

- Display all documents in Staff and display only empid and designation. `db.Staff.find().pretty()`

```

> db.Staff.find().pretty()
{
  "_id" : ObjectId("65deecde8cbdbaebf7136c4f"),
  "empid" : 1,
  "empname" : "John Doe",
  "salary" : 60000,
  "designation" : "Manager"
}
{
  "_id" : ObjectId("65deecde8cbdbaebf7136c50"),
  "empid" : 2,
  "empname" : "Jane Smith",
  "salary" : 55000,
  "designation" : "Accountant"
}
{
  "_id" : ObjectId("65deecde8cbdbaebf7136c51"),
  "empid" : 3,
  "empname" : "Michael Johnson",
  "salary" : 70000,
  "designation" : "Manager"
}
{
  "_id" : ObjectId("65deecde8cbdbaebf7136c52"),
  "empid" : 4,
  "empname" : "Emily Brown",
  "salary" : 45000,
  "designation" : "Accountant"
}

```

```
{
  "_id" : ObjectId("65deecde8cbdbaebf7136c54"),
  "empid" : 6,
  "empname" : "Sarah Lee",
  "salary" : 95000,
  "designation" : "Manager"
}
{
  "_id" : ObjectId("65deecde8cbdbaebf7136c55"),
  "empid" : 7,
  "empname" : "Christopher Martinez",
  "salary" : 50000,
  "designation" : "Accountant"
}
{
  "_id" : ObjectId("65deecde8cbdbaebf7136c56"),
  "empid" : 8,
  "empname" : "Amanda Davis",
  "salary" : 120000,
  "designation" : "Manager"
}
{
  "_id" : ObjectId("65deecde8cbdbaebf7136c57"),
  "empid" : 9,
  "empname" : "Jason Rodriguez",
  "salary" : 40000,
  "designation" : "Intern"
}
```

Db.staff.find({}, {"\_id":0, "empid":1, "designation":1}).pretty()

```
> db.Staff.find({}, { "_id": 0, "empid": 1, "designation": 1 }).pretty()
{ "empid" : 1, "designation" : "Manager" }
{ "empid" : 2, "designation" : "Accountant" }
{ "empid" : 3, "designation" : "Manager" }
{ "empid" : 4, "designation" : "Accountant" }
{ "empid" : 5, "designation" : "Developer" }
{ "empid" : 6, "designation" : "Manager" }
{ "empid" : 7, "designation" : "Accountant" }
{ "empid" : 8, "designation" : "Manager" }
{ "empid" : 9, "designation" : "Intern" }
{ "empid" : 10, "designation" : "Manager" }
```

- Sort the documents in descending order of Salary

db.Staff.find().sort({ "salary": -1 })

```
> db.Staff.find().sort({ "salary": -1 })
{ "_id" : ObjectId("65deecde8cbdbaebf7136c56"), "empid" : 8, "empname" : "Amanda Davis", "salary" : 120000, "designation" : "Manager" }
{ "_id" : ObjectId("65deecde8cbdbaebf7136c58"), "empid" : 10, "empname" : "Jessica Taylor", "salary" : 110000, "designation" : "Manager" }
{ "_id" : ObjectId("65deecde8cbdbaebf7136c54"), "empid" : 6, "empname" : "Sarah Lee", "salary" : 95000, "designation" : "Manager" }
{ "_id" : ObjectId("65deecde8cbdbaebf7136c53"), "empid" : 5, "empname" : "David Wilson", "salary" : 80000, "designation" : "Developer" }
{ "_id" : ObjectId("65deecde8cbdbaebf7136c51"), "empid" : 3, "empname" : "Michael Johnson", "salary" : 70000, "designation" : "Manager" }
{ "_id" : ObjectId("65deecde8cbdbaebf7136c4f"), "empid" : 1, "empname" : "John Doe", "salary" : 60000, "designation" : "Manager" }
{ "_id" : ObjectId("65deecde8cbdbaebf7136c50"), "empid" : 2, "empname" : "Jane Smith", "salary" : 55000, "designation" : "Accountant" }
{ "_id" : ObjectId("65deecde8cbdbaebf7136c55"), "empid" : 7, "empname" : "Christopher Martinez", "salary" : 50000, "designation" : "Accountant" }
{ "_id" : ObjectId("65deecde8cbdbaebf7136c52"), "empid" : 4, "empname" : "Emily Brown", "salary" : 45000, "designation" : "Accountant" }
{ "_id" : ObjectId("65deecde8cbdbaebf7136c57"), "empid" : 9, "empname" : "Jason Rodriguez", "salary" : 40000, "designation" : "Intern" }
```

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

```
db.Staff.find({ $or: [{ "designation": "Manager" }, { "salary": { $gt: 50000 } } ] })
```

```
> db.Staff.find({ $or: [{ "designation": "Manager" }, { "salary": { $gt: 50000 } } ] })
{ "_id" : ObjectId("65deecde8cbdbae7136c4f"), "empid" : 1, "empname" : "John Doe", "salary" : 60000, "designation" : "Manager" }
{ "_id" : ObjectId("65deecde8cbdbae7136c51"), "empid" : 3, "empname" : "Michael Johnson", "salary" : 70000, "designation" : "Manager" }
{ "_id" : ObjectId("65deecde8cbdbae7136c53"), "empid" : 5, "empname" : "David Wilson", "salary" : 80000, "designation" : "Developer" }
{ "_id" : ObjectId("65deecde8cbdbae7136c54"), "empid" : 6, "empname" : "Sarah Lee", "salary" : 95000, "designation" : "Manager" }
```

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

```
db.Staff.updateMany({ "designation": "Accountant" }, { $set: { "salary": 45000 } })
```

```
> db.Staff.updateMany({ "designation": "Accountant" }, { $set: { "salary": 45000 } })
{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 2 }
```

```
db.Staff.find({ "designation": "Accountant" })
```

```
> db.Staff.find({ "designation": "Accountant" })
{ "_id" : ObjectId("65deecde8cbdbae7136c50"), "empid" : 2, "empname" : "Jane Smith", "salary" : 45000, "designation" : "Accountant" }
{ "_id" : ObjectId("65deecde8cbdbae7136c52"), "empid" : 4, "empname" : "Emily Brown", "salary" : 45000, "designation" : "Accountant" }
{ "_id" : ObjectId("65deecde8cbdbae7136c55"), "empid" : 7, "empname" : "Christopher Martinez", "salary" : 45000, "designation" : "Accountant" }
```

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

```
db.Staff.deleteMany({ "salary": { $gt: 100000 } }) db.Staff.find()
```

```
> db.Staff.deleteMany({ "salary": { $gt: 100000 } })
{ "acknowledged" : true, "deletedCount" : 2 }
> db.Staff.find()
{ "_id" : ObjectId("65deecde8cbdbae7136c4f"), "empid" : 1, "empname" : "John Doe", "salary" : 60000, "designation" : "Manager" }
{ "_id" : ObjectId("65deecde8cbdbae7136c50"), "empid" : 2, "empname" : "Jane Smith", "salary" : 45000, "designation" : "Accountant" }
{ "_id" : ObjectId("65deecde8cbdbae7136c51"), "empid" : 3, "empname" : "Michael Johnson", "salary" : 70000, "designation" : "Manager" }
{ "_id" : ObjectId("65deecde8cbdbae7136c52"), "empid" : 4, "empname" : "Emily Brown", "salary" : 45000, "designation" : "Accountant" }
{ "_id" : ObjectId("65deecde8cbdbae7136c53"), "empid" : 5, "empname" : "David Wilson", "salary" : 80000, "designation" : "Developer" }
{ "_id" : ObjectId("65deecde8cbdbae7136c54"), "empid" : 6, "empname" : "Sarah Lee", "salary" : 95000, "designation" : "Manager" }
{ "_id" : ObjectId("65deecde8cbdbae7136c55"), "empid" : 7, "empname" : "Christopher Martinez", "salary" : 45000, "designation" : "Accountant" }
{ "_id" : ObjectId("65deecde8cbdbae7136c57"), "empid" : 9, "empname" : "Jason Rodriguez", "salary" : 40000, "designation" : "Intern" }
```

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).  
db.createCollection(“Student”) db

```
> db.createCollection("Student")
{ "ok" : 1 }
> db
Institution
```

```
db.Student.insertMany([ { "RollNo": 101, "Name": "Alice Johnson", "Class":
"BSc", "TotalMarks": 480 }, { "RollNo": 102, "Name": "Bob Smith", "Class":
"MSc", "TotalMarks": 450 }, { "RollNo": 103, "Name": "Charlie Brown",
"Class": "MSc", "TotalMarks": 420 }, { "RollNo": 104, "Name": "David
Davis", "Class": "BSc", "TotalMarks": 400 }, { "RollNo": 105, "Name": "Eva
Wilson", "Class": "MSc", "TotalMarks": 490 }, { "RollNo": 106, "Name":
"Frank Martinez", "Class": "BSc", "TotalMarks": 360 }, { "RollNo": 107,
"Name": "Grace Lee", "Class": "MSc", "TotalMarks": 510 }, { "RollNo": 108,
"Name": "Henry Taylor", "Class": "BSc", "TotalMarks": 320 }, { "RollNo":
109, "Name": "Isabel Rodriguez", "Class": "MSc", "TotalMarks": 380 }, {
"RollNo": 110, "Name": "Jack Harris", "Class": "BSc", "TotalMarks": 250 } ])
```

```
> db.Student.insertMany([
...   { "RollNo": 101, "Name": "Alice Johnson", "Class": "BSc", "TotalMarks": 480 },
...   { "RollNo": 102, "Name": "Bob Smith", "Class": "MSc", "TotalMarks": 450 },
...   { "RollNo": 103, "Name": "Charlie Brown", "Class": "MSc", "TotalMarks": 420 },
...   { "RollNo": 104, "Name": "David Davis", "Class": "BSc", "TotalMarks": 400 },
...   { "RollNo": 105, "Name": "Eva Wilson", "Class": "MSc", "TotalMarks": 490 },
...   { "RollNo": 106, "Name": "Frank Martinez", "Class": "BSc", "TotalMarks": 360 },
...   { "RollNo": 107, "Name": "Grace Lee", "Class": "MSc", "TotalMarks": 510 },
...   { "RollNo": 108, "Name": "Henry Taylor", "Class": "BSc", "TotalMarks": 320 },
...   { "RollNo": 109, "Name": "Isabel Rodriguez", "Class": "MSc", "TotalMarks": 380 },
...   { "RollNo": 110, "Name": "Jack Harris", "Class": "BSc", "TotalMarks": 250 }
... ])
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("65def1918cbdbaebf7136c59"),
    ObjectId("65def1918cbdbaebf7136c5a"),
    ObjectId("65def1918cbdbaebf7136c5b"),
    ObjectId("65def1918cbdbaebf7136c5c"),
    ObjectId("65def1918cbdbaebf7136c5d"),
    ObjectId("65def1918cbdbaebf7136c5e"),
    ObjectId("65def1918cbdbaebf7136c5f"),
    ObjectId("65def1918cbdbaebf7136c60"),
    ObjectId("65def1918cbdbaebf7136c61"),
    ObjectId("65def1918cbdbaebf7136c62")
  ]
}
```

db.Student.find({})

```
> db.Student.find({})
{ "_id" : ObjectId("65def1918cbdbaebf7136c59"), "RollNo" : 101, "Name" : "Alice Johnson", "Class" : "BSc", "TotalMarks" : 480 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5a"), "RollNo" : 102, "Name" : "Bob Smith", "Class" : "MSc", "TotalMarks" : 450 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5b"), "RollNo" : 103, "Name" : "Charlie Brown", "Class" : "MSc", "TotalMarks" : 420 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5c"), "RollNo" : 104, "Name" : "David Davis", "Class" : "BSc", "TotalMarks" : 400 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5d"), "RollNo" : 105, "Name" : "Eva Wilson", "Class" : "MSc", "TotalMarks" : 490 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5e"), "RollNo" : 106, "Name" : "Frank Martinez", "Class" : "BSc", "TotalMarks" : 360 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5f"), "RollNo" : 107, "Name" : "Grace Lee", "Class" : "MSc", "TotalMarks" : 510 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c60"), "RollNo" : 108, "Name" : "Henry Taylor", "Class" : "BSc", "TotalMarks" : 320 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c61"), "RollNo" : 109, "Name" : "Isabel Rodriguez", "Class" : "MSc", "TotalMarks" : 380 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c62"), "RollNo" : 110, "Name" : "Jack Harris", "Class" : "BSc", "TotalMarks" : 250 }
>
```

- Display all documents in Student.



```
db.Student.find({})
```

- Sort the documents in descending order of TotalMarks.

```
db.Student.find().sort({ "TotalMarks": -1 })
```

```
> db.Student.find({})
{ "_id" : ObjectId("65def1918cbdbaebf7136c59"), "RollNo" : 101, "Name" : "Alice Johnson", "Class" : "BSc", "TotalMarks" : 480 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5a"), "RollNo" : 102, "Name" : "Bob Smith", "Class" : "MSc", "TotalMarks" : 450 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5b"), "RollNo" : 103, "Name" : "Charlie Brown", "Class" : "MSc", "TotalMarks" : 420 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5c"), "RollNo" : 104, "Name" : "David Davis", "Class" : "BSc", "TotalMarks" : 400 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5d"), "RollNo" : 105, "Name" : "Eva Wilson", "Class" : "MSc", "TotalMarks" : 490 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5e"), "RollNo" : 106, "Name" : "Frank Martinez", "Class" : "BSc", "TotalMarks" : 360 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5f"), "RollNo" : 107, "Name" : "Grace Lee", "Class" : "MSc", "TotalMarks" : 510 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c60"), "RollNo" : 108, "Name" : "Henry Taylor", "Class" : "BSc", "TotalMarks" : 320 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c61"), "RollNo" : 109, "Name" : "Isabel Rodriguez", "Class" : "MSc", "TotalMarks" : 380 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c62"), "RollNo" : 110, "Name" : "Jack Harris", "Class" : "BSc", "TotalMarks" : 250 }
>
> db.Student.find().sort({ "TotalMarks": -1 })
{ "_id" : ObjectId("65def1918cbdbaebf7136c5f"), "RollNo" : 107, "Name" : "Grace Lee", "Class" : "MSc", "TotalMarks" : 510 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5d"), "RollNo" : 105, "Name" : "Eva Wilson", "Class" : "MSc", "TotalMarks" : 490 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c59"), "RollNo" : 101, "Name" : "Alice Johnson", "Class" : "BSc", "TotalMarks" : 480 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5a"), "RollNo" : 102, "Name" : "Bob Smith", "Class" : "MSc", "TotalMarks" : 450 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5b"), "RollNo" : 103, "Name" : "Charlie Brown", "Class" : "MSc", "TotalMarks" : 420 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5c"), "RollNo" : 104, "Name" : "David Davis", "Class" : "BSc", "TotalMarks" : 400 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c61"), "RollNo" : 109, "Name" : "Isabel Rodriguez", "Class" : "MSc", "TotalMarks" : 380 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5e"), "RollNo" : 106, "Name" : "Frank Martinez", "Class" : "BSc", "TotalMarks" : 360 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c60"), "RollNo" : 108, "Name" : "Henry Taylor", "Class" : "BSc", "TotalMarks" : 320 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c62"), "RollNo" : 110, "Name" : "Jack Harris", "Class" : "BSc", "TotalMarks" : 250 }
```

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

```
db.Student.find({ $or: [{ "Class": "MSc" }, { "TotalMarks": { $gt: 400 } }] })
```

```
> db.Student.find({ $or: [{ "Class": "MSc" }, { "TotalMarks": { $gt: 400 } }] })
{ "_id" : ObjectId("65def1918cbdbaebf7136c59"), "RollNo" : 101, "Name" : "Alice Johnson", "Class" : "BSc", "TotalMarks" : 480 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5a"), "RollNo" : 102, "Name" : "Bob Smith", "Class" : "MSc", "TotalMarks" : 450 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5b"), "RollNo" : 103, "Name" : "Charlie Brown", "Class" : "MSc", "TotalMarks" : 420 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5d"), "RollNo" : 105, "Name" : "Eva Wilson", "Class" : "MSc", "TotalMarks" : 490 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5f"), "RollNo" : 107, "Name" : "Grace Lee", "Class" : "MSc", "TotalMarks" : 510 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c61"), "RollNo" : 109, "Name" : "Isabel Rodriguez", "Class" : "MSc", "TotalMarks" : 380 }
>
```

- Remove all the documents with TotalMarks<200

```
db.Student.deleteMany({ "TotalMarks": { $lt: 200 } })
```

```
db.Student.find({})
```

```
> db.Student.deleteMany({ "TotalMarks": { $lt: 200 } })
{ "acknowledged" : true, "deletedCount" : 0 }
> db.Student.find({})
{ "_id" : ObjectId("65def1918cbdbaebf7136c59"), "RollNo" : 101, "Name" : "Alice Johnson", "Class" : "BSc", "TotalMarks" : 480 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5a"), "RollNo" : 102, "Name" : "Bob Smith", "Class" : "MSc", "TotalMarks" : 450 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5b"), "RollNo" : 103, "Name" : "Charlie Brown", "Class" : "MSc", "TotalMarks" : 420 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5c"), "RollNo" : 104, "Name" : "David Davis", "Class" : "BSc", "TotalMarks" : 400 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5d"), "RollNo" : 105, "Name" : "Eva Wilson", "Class" : "MSc", "TotalMarks" : 490 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5e"), "RollNo" : 106, "Name" : "Frank Martinez", "Class" : "BSc", "TotalMarks" : 360 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c5f"), "RollNo" : 107, "Name" : "Grace Lee", "Class" : "MSc", "TotalMarks" : 510 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c60"), "RollNo" : 108, "Name" : "Henry Taylor", "Class" : "BSc", "TotalMarks" : 320 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c61"), "RollNo" : 109, "Name" : "Isabel Rodriguez", "Class" : "MSc", "TotalMarks" : 380 }
{ "_id" : ObjectId("65def1918cbdbaebf7136c62"), "RollNo" : 110, "Name" : "Jack Harris", "Class" : "BSc", "TotalMarks" : 250 }
>
```