INDEX

Sr.no	Practical Name	Date	Remark
1	Data collection,cleaning,modeling and compilation	02/08/2023	
2	Explore data visualization techniques	09/08/2023	
3	Implement statistics distribution: a) Normal distribution b) Binomial distribution c) Poisson distribution d) Chi-square distribution	18/08/2023	
4	Perform hypothesis testing: a) One sample t test for small samples b) Two sample t test for small samples c) Paired t-test d) One sample z test for large samples e) Two sample z test for large samples	08/09/2023	
5	Exploring categorical and binary data	13/09/2023	
6	Implementation of non-parametric test: a) Sign test b) Wilcoxon matched pair test (or sign rank test) c) Kruskal wallis test	20/09/2023	
7	Perform ANOVA testing: a) One way ANOVA b) Two way ANOVA	27/09/2023	
8	Implement regression analysis and fit a straight line	12/10/2023	
9	Perform time series analysis	18/10/2023	

Practical 1

Data collection,

cleaning, modeling and compilation

Input and output

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

data=pd.read_csv(r"C:\Users\ocmodels03dev\Desktop\SAMEER\risk_analytics_train - risk_analytics_train - risk_analytics_train - risk_analytics_train.csv",index_col=0,header=0)
data.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
Loan_ID												
LP001002	Male	No	0.0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
LP001003	Male	Yes	1.0	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
LP001005	Male	Yes	0.0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
LP001006	Male	Yes	0.0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
LP001008	Male	No	0.0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

```
data=pd.read_csv(r"C:\Users\ocmodels03dev\Desktop\SAMEER\risk_analytics_test - risk_analytics_test.csv",index_col=0,header=0)
data.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
Loan_ID											
LP001015	Male	Yes	0.0	Graduate	No	5720	0	110.0	360.0	1.0	Urban
LP001022	Male	Yes	1.0	Graduate	No	3076	1500	126.0	360.0	1.0	Urban
LP001031	Male	Yes	2.0	Graduate	No	5000	1800	208.0	360.0	1.0	Urban
LP001035	Male	Yes	2.0	Graduate	No	2340	2546	100.0	360.0	NaN	Urban
LP001051	Male	No	0.0	Not Graduate	No	3276	0	78.0	360.0	1.0	Urban

```
data.dtypes
```

```
Gender                object
Married               object
Dependents           float64
Education             object
Self_Employed         object
ApplicantIncome        int64
CoapplicantIncome      int64
LoanAmount           float64
Loan_Amount_Term     float64
Credit_History       float64
Property_Area         object
dtype: object
```

data.shape # for train

```
(614, 12)
```

data.shape # for test

```
(367, 11)
```

data.describe(include='all') # for train

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
count	601	611	599.000000	614	582	614.000000	614.000000	592.000000	600.00000	564.000000	614	614
unique	2	2	NaN	2	2	NaN	NaN	NaN	NaN	NaN	3	2
top	Male	Yes	NaN	Graduate	No	NaN	NaN	NaN	NaN	NaN	Semiurban	Y
freq	489	398	NaN	480	500	NaN	NaN	NaN	NaN	NaN	233	422
mean	NaN	NaN	0.762938	NaN	NaN	5403.459283	1621.245798	146.412162	342.00000	0.842199	NaN	NaN
std	NaN	NaN	1.015216	NaN	NaN	6109.041673	2926.248369	85.587325	65.12041	0.364878	NaN	NaN
min	NaN	NaN	0.000000	NaN	NaN	150.000000	0.000000	9.000000	12.00000	0.000000	NaN	NaN
25%	NaN	NaN	0.000000	NaN	NaN	2877.500000	0.000000	100.000000	360.00000	1.000000	NaN	NaN
50%	NaN	NaN	0.000000	NaN	NaN	3812.500000	1188.500000	128.000000	360.00000	1.000000	NaN	NaN
75%	NaN	NaN	2.000000	NaN	NaN	5795.000000	2297.250000	168.000000	360.00000	1.000000	NaN	NaN
max	NaN	NaN	3.000000	NaN	NaN	81000.000000	41667.000000	700.000000	480.00000	1.000000	NaN	NaN

data.describe(include='all') # for test

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
count	356	367	357.000000	367	344	367.000000	367.000000	362.000000	361.000000	338.000000	367
unique	2	2	NaN	2	2	NaN	NaN	NaN	NaN	NaN	3
top	Male	Yes	NaN	Graduate	No	NaN	NaN	NaN	NaN	NaN	Urban
freq	286	233	NaN	283	307	NaN	NaN	NaN	NaN	NaN	140
mean	NaN	NaN	0.829132	NaN	NaN	4805.599455	1569.577657	136.132597	342.537396	0.825444	NaN
std	NaN	NaN	1.071302	NaN	NaN	4910.685399	2334.232099	61.366652	65.156643	0.380150	NaN
min	NaN	NaN	0.000000	NaN	NaN	0.000000	0.000000	28.000000	6.000000	0.000000	NaN
25%	NaN	NaN	0.000000	NaN	NaN	2864.000000	0.000000	100.250000	360.000000	1.000000	NaN
50%	NaN	NaN	0.000000	NaN	NaN	3786.000000	1025.000000	125.000000	360.000000	1.000000	NaN
75%	NaN	NaN	2.000000	NaN	NaN	5060.000000	2430.500000	158.000000	360.000000	1.000000	NaN
max	NaN	NaN	3.000000	NaN	NaN	72529.000000	24000.000000	550.000000	480.000000	1.000000	NaN

data.isnull().sum() # for train

```
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

data.isnull().sum() # for test

```
Gender                 11
Married                 0
Dependents             10
Education               0
Self_Employed          23
ApplicantIncome         0
CoapplicantIncome       0
LoanAmount              5
Loan_Amount_Term        6
Credit_History         29
Property_Area           0
dtype: int64
```

```
for value in ['Gender','Married','Dependents','Self_Employed','Loan_Amount_Term']:
    data[value].fillna(data[value].mode()[0],inplace=True)
data.isnull().sum()                          # for train
```

```
Gender                  0
Married                 0
Dependents              0
Education               0
Self_Employed           0
ApplicantIncome         0
CoapplicantIncome       0
LoanAmount             22
Loan_Amount_Term        0
Credit_History         50
Property_Area           0
Loan_Status             0
dtype: int64
```

```
categorical_data=['Gender','Dependents','Self_Employed','Credit_History']
for i in categorical_data:
    data[i].fillna(data[i].mode()[0],inplace=True)
data.isnull().sum()                          # for test
```

```
Gender                  0
Married                 0
Dependents              0
Education               0
Self_Employed           0
ApplicantIncome         0
CoapplicantIncome       0
LoanAmount              5
Loan_Amount_Term        6
Credit_History          0
Property_Area           0
dtype: int64
```

```
data['LoanAmount'].fillna(round(data['LoanAmount'].mean(),0),inplace=True)
data.isnull().sum()                          # for train
```

```
Gender                0
Married               0
Dependents            0
Education             0
Self_Employed         0
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term      0
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

```
numerical_data=['LoanAmount','Loan_Amount_Term']
for i in numerical_data:
    data[i].fillna(data[i].mean(),inplace=True)
data.isnull().sum()                         # for test
```

```
Gender                0
Married               0
Dependents            0
Education             0
Self_Employed         0
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term      0
Credit_History        0
Property_Area         0
dtype: int64
```

```
data['Credit_History'].fillna(value=0,inplace=True)
print(data.isnull().sum())                  # for train
```

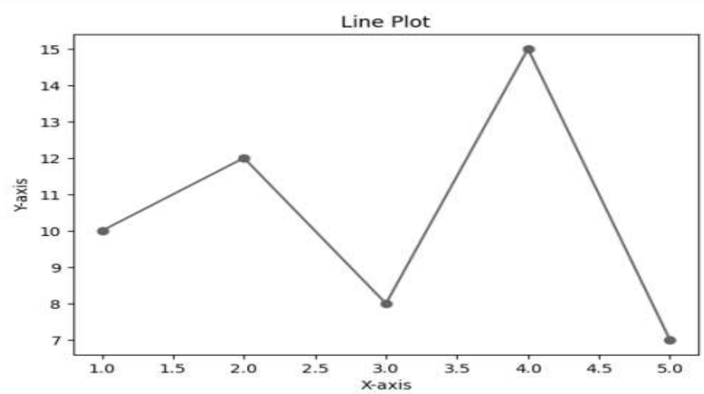
```
Gender                0
Married               0
Dependents            0
Education             0
Self_Employed         0
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term      0
Credit_History        0
Property_Area         0
Loan_Status           0
dtype: int64
```

Signature : ____________

Practical 2
Explore Data Visualization Techniques

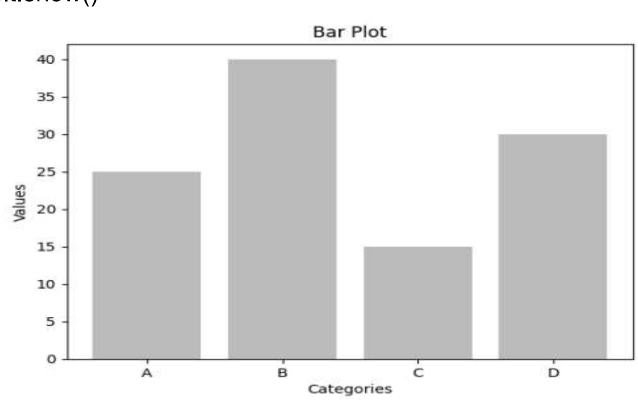
input and output:

1 Line Plot:
```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y = [10, 12, 8, 15, 7]
plt.plot(x, y, marker='o')
plt.title("Line Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```

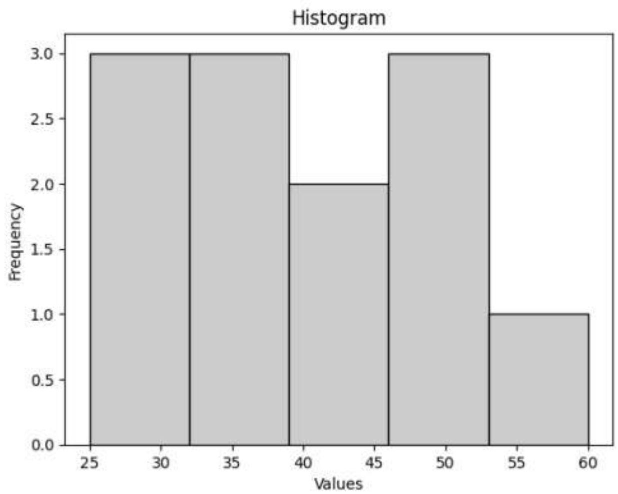


1

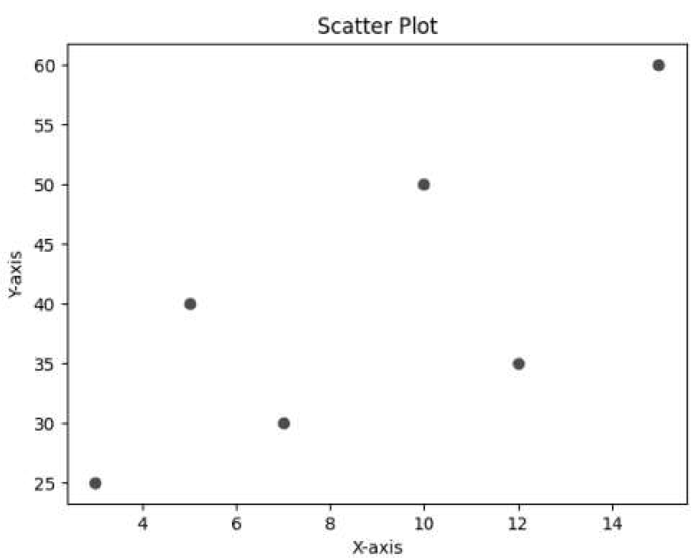
2 Bar Plot:
```
categories = ['A', 'B', 'C', 'D']
values = [25, 40, 15, 30]
plt.bar(categories, values, color='skyblue')
plt.title("Bar Plot")
plt.xlabel("Categories")
plt.ylabel("Values")
plt.show()
```



3 Histogram:
data = [32, 45, 50, 28, 36, 42, 38, 25, 29, 52, 48, 60] plt.hist(data, bins=5, color='lightblue', edgecolor='black')
plt.title("Histogram")
plt.xlabel("Values")
plt.ylabel("Frequency")

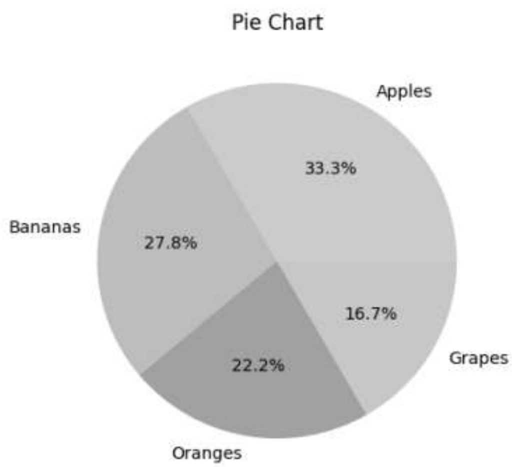


4 Scatter Plot:
x = [3, 5, 7, 10, 12, 15]
y = [25, 40, 30, 50, 35, 60]
plt.scatter(x, y, color='red', marker='o')
plt.title("Scatter Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()



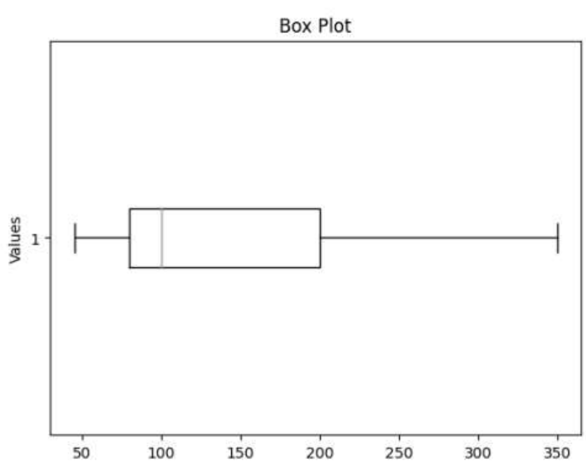
5 Pie Chart:

```
 labels = ['Apples', 'Bananas', 'Oranges', 'Grapes']
sizes = [30, 25, 20, 15]
plt.pie(sizes, labels=labels, autopct='%1.1f%%', colors=['gold', ␣
↪'lightskyblue', 'lightcoral', 'lightgreen'])
plt.title("Pie Chart")
plt.show()
```



6 Box Plot:

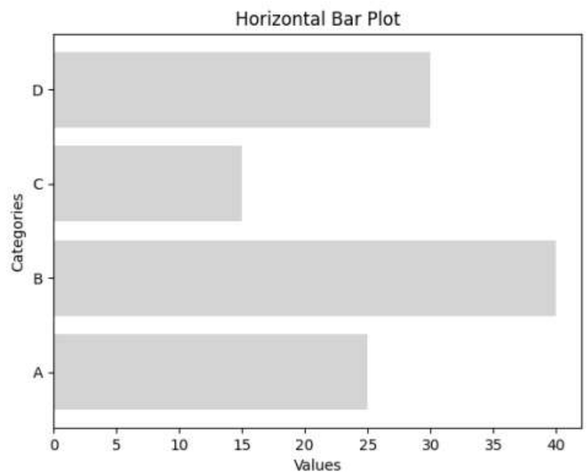
```
 data = [45, 60, 75,300, 80,350, 90, 95, 100, 110, 120,250,200]
plt.boxplot(data,vert=False,)
plt.title("Box Plot")
plt.ylabel("Values")
plt.show()
```



7 Barh Plot:

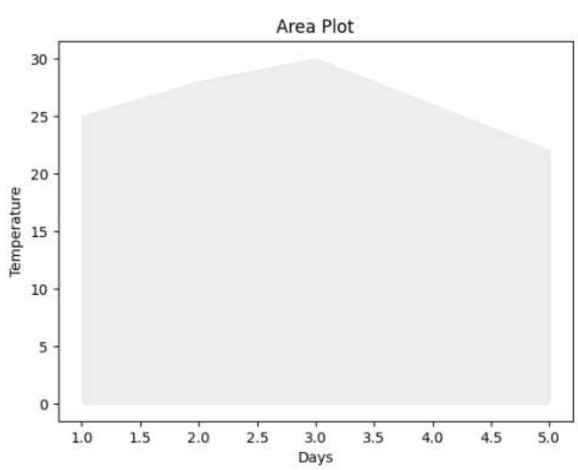
```
categories = ['A', 'B', 'C', 'D']
```

```
values = [25, 40, 15, 30]
plt.barh(categories, values, color='pink')
plt.title("Horizontal Bar Plot")
plt.xlabel("Values")
plt.ylabel("Categories")
plt.show(
```



8 Area Plot:

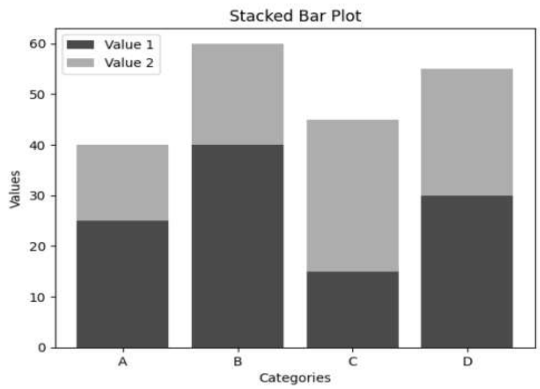
```
days = [1, 2, 3, 4, 5]
temperature = [25, 28, 30, 26, 22]
plt.fill_between(days, temperature, color='yellow', alpha=0.6)
plt.title("Area Plot")
plt.xlabel("Days")
plt.ylabel("Temperature")
plt.show()
```



9 Stacked Bar Plot:

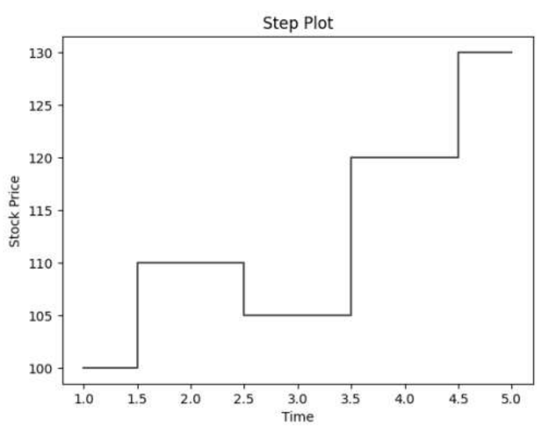
```
categories = ['A', 'B', 'C', 'D']
values1 = [25, 40, 15, 30]
values2 = [15, 20, 30, 25]
```

```
plt.bar(categories, values1, color='green', label='Value 1')
plt.bar(categories, values2, bottom=values1, color='orange', label='Value 2')
plt.title("Stacked Bar Plot")
plt.xlabel("Categories")
plt.ylabel("Values")
plt.legend()
plt.show()
```



10 Step Plot:

```
time = [1, 2, 3, 4, 5]
stock_price = [100, 110, 105, 120, 130]
plt.step(time, stock_price, color='red', where='mid')
plt.title("Step Plot")
plt.xlabel("Time")
plt.ylabel("Stock Price")
plt.show()
```



Signature ____________________

Practical 3 Implement statistics distribution

a) Normal distribution

Input and output

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
from scipy import stats

# Given information
mean = 78
std_dev = 25
total_students = 100
score = 60

# Calculate z-score for 60
z_score = (score - mean) / std_dev

# Calculate the probability of getting a score less than 60
prob = norm.cdf(z_score)

# Calculate the percentage of students who got less than 60 marks
percent = prob * 100

# Print the result
print("Percentage of students who got less than 60 marks:", round(percent, 2), "%")
```

```
Percentage of students who got less than 60 marks: 23.58 %
```

```
# Given information
mean = 78
std_dev = 25
total_students = 100
score = 70

# Calculate z-score for 70
z_score = (score - mean) / std_dev

# Calculate the probability of getting a more than 70
prob = norm.cdf(z_score)

# Calculate the percentage of students who got more than 70 marks
percent = (1-prob) * 100

# Print the result
print("Percentage of students who got more than 70 marks: ", round(percent, 2), " %")
```

```
Percentage of students who got more than 70 marks:  62.55  %
```

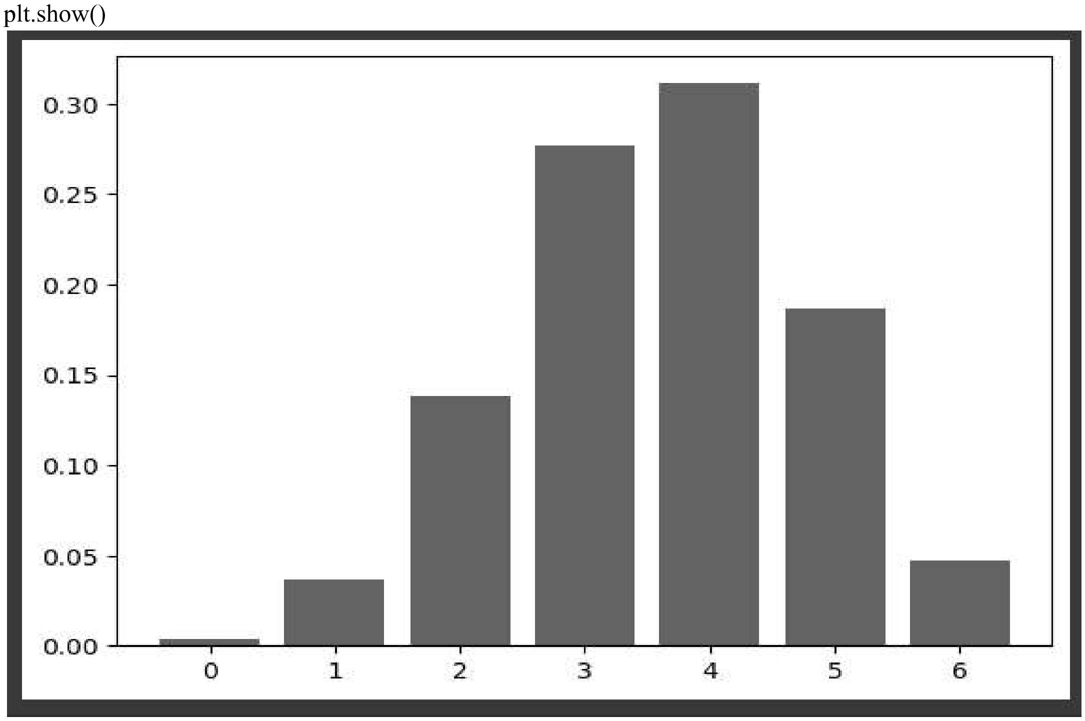
b) Binomial distribution

Input and output

```
from scipy.stats import binom

# setting the values
# of n and p
n = 6
p = 0.6

# defining list of r values
r_values = list(range(n + 1))
# list of pmf values
dist = [binom.pmf(r, n, p) for r in r_values ]
print(dist)
```

```
[0.004096000000000003, 0.03686400000000002, 0.1382400000000001, 0.2764800000000001, 0.3110400000000001, 0.1866240000000001, 0.04665599999999999]
```

```
# plotting the graph
plt.bar(r_values, dist)
plt.show()
```



When success and failure are equally likely, the binomial distribution is a normal distribution.

```
#Expected number of successful trials = 5
#total number of trails =30
#The probability of success = 0.15
stats.binom.pmf(5, 30, 0.15)
```

```
0.18610694845752918
```

the corresponding probability is 0.1861, that is the probability that exactly 5 customers will return items is approximately 18.61%

c) Poisson distribution

Input and output

```
#Question : The number of calls arriving at a call center follows a Poisson distribution at 20 calls per hour.
#Calculate the probability that the number of calls will be maximum 10.

stats.poisson.cdf(10,20)
```

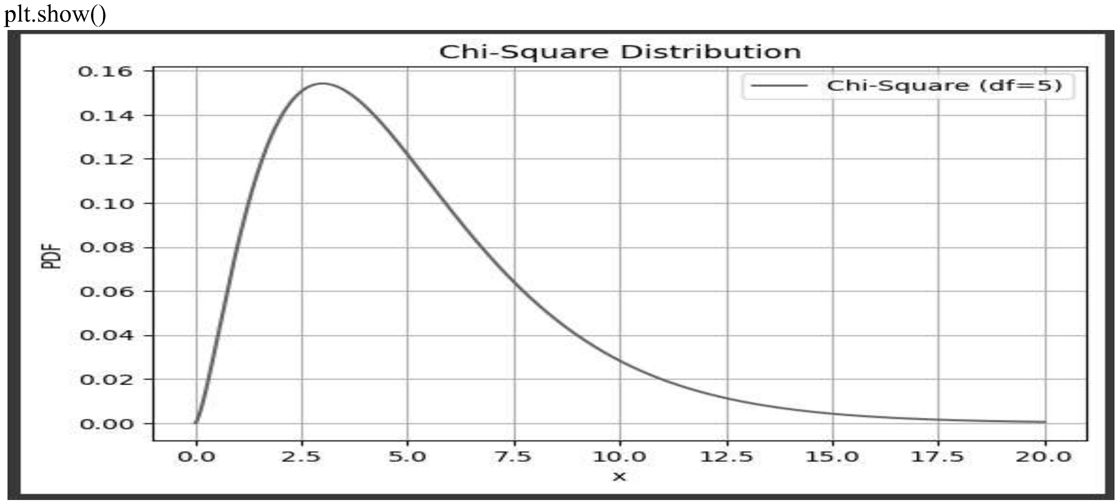
```
0.010811718826652723
```

the corresponding probability is 0.0108

d) Chi-square distribution

Input and output

```
from scipy.stats import chi2
# Parameters for the Chi-Square distribution
degrees_of_freedom = 5  # Degrees of freedom
# Generate a range of x values
x = np.linspace(0, 20, 1000)
# Calculate the probability density function (PDF) for the Chi-Square distribution
pdf = chi2.pdf(x, degrees_of_freedom)
# Plot the PDF
plt.plot(x, pdf, label=f'Chi-Square (df={degrees_of_freedom})')
plt.xlabel('x')
plt.ylabel('PDF')
plt.legend()
plt.title('Chi-Square Distribution')
plt.grid()
plt.show()
```



Signature : ____________

Practical 4 Perform hypothesis testing

a) One sample t test for small samples

i) Two-sided

Input and output

Q) The Violet industries produces furniture for home and office use. The production engineer claims that the average number of furniture items produced is 30. Test his claim.

```
import pandas as pd
df=pd.DataFrame({
    'Sr_no':[1,2,3,4,5,6,7,8,9,10],
    'Items_Produced':[34,34,23,45,23,45,34,23,37,35]
})
df
```

	Sr_no	Items_Produced
0	1	34
1	2	34
2	3	23
3	4	45
4	5	23
5	6	45
6	7	34
7	8	23
8	9	37
9	10	35

Let
X : The number of furniture items produced by machine A.
Thus μ : The average number of furniture items produced by machine A.
The Hypothesis is
H0 : The mean number of items produced by machine A is equal to 30 i.e $\mu = 30$ Against
H1 : The mean number of items produced by machine A is not equal toi 30 i.e $\mu \neq 30$

```
#import library
from scipy.stats import ttest_1samp
#conduct the test
ttest_1samp(df['Items_Produced'],30)
```

```
TtestResult(statistic=1.2674612253287467, pvalue=0.23680061493494728, df=9)
```

We see that the p-value(0.2368) > 0.05(l.o.s) , Hence we do not reject H0
We may conclude that the average number of items produced by machine A = 30.

ii) one-sided

Input and output

The blaze industries pvt.ltd claims that their manufactured electric bulbs have LIFE OF 45000 hrs.Test their claims at 5% level of significance.

```
data=pd.DataFrame({
    'Sr_no':[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15],
'Life':[35000,49000,44000,45900,43400,50000,47200,32700,33200,43000,46300,47800,32100,33200,33900]
})
data.head()
```

	Sr_no	Life
0	1	35000
1	2	49000
2	3	44000
3	4	45900
4	5	43400

Let
X : The life of bulb in hours
Thus µA : The average life of bulb in hours

The hypothesis is
H0: The mean life of bulb in hours is equal to 45000 , i.e. µA = 45000
AGAINST
H1 : The mean life of bulb in hours is less than equal to 45000, i.e. µA ≤ 45000\

```
#conduct the test
#this is the test is two tailed
t_test_less=ttest_1samp(data['Life'],45000)

#to get the p-value for one-tailed test
#divide the p-value by 2
p_value_less=t_test_less.pvalue/2

#print p-value
p_value_less
```

```
0.022628491566062506
```

we see that p-value(0.02) is less than 0.05(l.o.s), hence we reject H0.
We may conclude that the avg life of a bulb in an hour is less than or equal to 45000. And the manufacturer's claim is wrong.

Signature : ____________

b) Two sample t test for small samples

Input and output

In a pig farm, to increase the weight of the pig, two different diets were given. The data for gain in weight is as given. Test whether the two diets differ significantly regarding their effect on increase in weight.

```
f1=pd.DataFrame({
  'Sr_no':[1,2,3,4,5,6,7,8,9,10],
  'Diet A':[45,34,46,23,67,45,65,23,65,23],
  'Diet B':[45,35,32,47,37,51,42,38,32,46]
})
df1
```

	Sr_no	Diet A	Diet B
0	1	45	45
1	2	34	35
2	3	46	32
3	4	23	47
4	5	67	37
5	6	45	51
6	7	65	42
7	8	23	38
8	9	65	32
9	10	23	46

Let
X : The gain in weight due to diet A
Y : The gain in weight due to diet B
Thus
μa : The average gain in weight due to diet A
AGAINST
μb : The average gain in weight due to diet B
The Hypothesis is
H0 : The average gain in weight due to diet A is equal to The average gain in weight due to diet B i.e μa = μb
H1 : The average gain in weight due to diet A is not equal to The average gain in weight due to diet B i.e μa ≠μb

```
#t-test for testing equality of means
from scipy.stats import ttest_ind
#conduct the test
ttest_ind(df1['Diet A'],df1['Diet B'])
```

```
TtestResult(statistic=0.5169140149580205, pvalue=0.6115094596012607, df=18.0)
```

since the p-value (0.611) > 0.05(level of significance) hence we fail to reject H0
we may conclude that the average gain in weight due to diet A is not equal to the average gain in weight due to diet B

Signature : ____________

c) Paired t test

Input and output

A Pharma company claims to have produced a new drug which improves sleep of 18 year old teenagers . The hours of sleep for 10 teenagers before and after giving the new drug is recorded. Test whether there is a significant difference in the average hours of sleep.

Let
X : Hours of sleep before medication
Y : Hours of sleep after medication
Thus,
The Hypothesis is
H0 : The average hours of sleep before medication is equal to the average of sleep after medication.
i.e.$\mu A=\mu B$
H1 : The average hours of sleep before medication is not equal to the average of sleep after medication.
i.e.$\mu A\neq\mu B$

```
data=pd.DataFrame({
    'sr no.':[1,2,3,4,5,6,7,8,9,10],
    'Hours of sleep(before)':[5.0,6.0,6.5,5.5,6.5,7,8,7.5,6,7],
    'Hours of sleep(after)':[7,7.5,9,7,8,7.5,7,7,7,6.5]
})
data
```

	sr no.	Hours of sleep(before)	Hours of sleep(after)
0	1	5.0	7.0
1	2	6.0	7.5
2	3	6.5	9.0
3	4	5.5	7.0
4	5	6.5	8.0
5	6	7.0	7.5
6	7	8.0	7.0
7	8	7.5	7.0
8	9	6.0	7.0
9	10	7.0	6.5

```
from scipy.stats import ttest_rel
ttest_rel(data['Hours of sleep(before)'],data['Hours of sleep(after)'])
```

```
TtestResult(statistic=-2.2785119632047284, pvalue=0.048680802344322635, df=9)
```

Since the p-value(0.048) is less than 0.05 (level of significance),hence we reject H0.
We may conclude that the average hours of sleep before medication is not equal to the average of sleep after medication.
i.e $\mu a \neq \mu b$

Signature : ____________

d) One sample z test for large samples

Input and output

A random sample of 40 glass rod is taken from a lot manufactured under a new process. they are tested for their strength. Can we claim that the breaking strength is equal to 55 lbs?

The hypothesis is,
H0 : The mean of the breaking strength of glass rod is equal to 55 i.e. $\mu = 55$
H1 : The mean of the breaking strength of glass rod is not equal to 55 i.e. $\mu \neq 55$

```
data1=pd.DataFrame({
    'Strength(in lbs)':[48.6,38.8,52.6,48,60.2,54.6,42.7,42.1,49.8,57.8,57.9,46.4,52.5,55.5,53.5,54.9,58.8,
41.8,58.6,48.7,55,50.2,57,51.2,39.5,58.2,46.8,38,56.6,41,54.8,55.6,58.4,45,42.8,54.8,45.7,44.3,38.4,54.3]})
data1.head
```

	Strength(in lbs)
0	48.6
1	38.8
2	52.6
3	48.0
4	60.2

```
#import the library
from statsmodels.stats import weightstats as stests

# conduct the test
stests.ztest(data1['Strength(in lbs)'],value=55)
```

```
(-4.431740748679182, 9.347536680921983e-06)
```

Since, the p-value(0.00) is less than 0.05 (LO.S), hence we reject Ho.
We may conclude that the average breaking strength of the glass rod is not equal to 55 lbs.

Signature : ____________

e) Two sample z test for large samples

Input and output

Two types of fertilisers are tested on farm lands of the same size. The yields in quintals are given below. Can we conclude that the two fertilisers have the same yield?

Let,
x:yield due to fertiliser 1
y:yield due to fertiliser 2
Thus,
μA=The average yield due to fertiliser 1
μB=The average yield due to fertiliser 2
The Hypothesis is,
i.e. μA=μB
Ho:The average yield due to fertiliser 1 is equal to the average yield due to fertiliser 2
Ha:The average yield due to fertiliser 1 is not equal to the average yield due to fertiliser 2
i.e. μA≠μB

```
df_fertilizer=pd.DataFrame({
    'Fertilizer 1':[23,34,45,32,34,51,39,34,28,30,46,41,35,43,24,36,25,33,36,43,26,45,32,47,42,31,35,46,34,32,46,36,25,46,34],
    'Fertilizer 2':[45,34,21,34,45,32,46,53,24,23,54,53,34,45,43,45,32,47,36,37,37,29,30,31,38,42,48,29,37,35,47,43,31,43,43]
})
df_fertilizer.head()
```

	Fertilizer 1	Fertilizer 2
0	23	45
1	34	34
2	45	21
3	32	34
4	34	45

```
#import the library
from statsmodels.stats import weightstats as stests
#conduct the test
stests.ztest(df_fertilizer['Fertilizer 1'],df_fertilizer['Fertilizer 2'])
```

```
(-1.1390306751872912, 0.25469036008906554)
```

Since the p-value (0.255) is more than 0.05 (level of significance).
hence,we accept Ho.
We may conclude that the average yeild due to fertilizer 1 is equal to the average yeild due to fertilizer 2.

Signature : ____________

Practical 5 Exploring categorical and binary data

Input and output

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load your dataset
df = pd.read_csv(r"C:\Users\rahul\Downloads\sales_data.csv")
df.head()
```

	Date	Day	Month	Year	Customer_Age	Age_Group	Customer_Gender	Country	State	Product_Category	Sub_Category
0	26-11-2013	26	November	2013	19	Youth (<25)	M	Canada	British Columbia	Accessories	Bike Racks
1	26-11-2015	26	November	2015	19	Youth (<25)	M	Canada	British Columbia	Accessories	Bike Racks
2	23-03-2014	23	March	2014	49	Adults (35-64)	M	Australia	New South Wales	Accessories	Bike Racks
3	23-03-2016	23	March	2016	49	Adults (35-64)	M	Australia	New South Wales	Accessories	Bike Racks
4	15-05-2014	15	May	2014	47	Adults (35-64)	F	Australia	New South Wales	Accessories	Bike Racks

```
#df.info()
df.iloc[:,5:12].nunique()
#df[['Age_Group ','Customer_Gender','Country','State','Product_Category','Sub_Category','Product']]
```

```
Age_Group             4
Customer_Gender       2
Country               6
State                53
Product_Category      3
Sub_Category         17
Product             130
dtype: int64
```

```
df.describe()
```

	Day	Year	Customer_Age	Order_Quantity	Unit_Cost	Unit_Price	Profit	Cost	Revenue
count	113036.000000	113036.000000	113036.000000	113036.000000	113036.000000	113036.000000	113036.000000	113036.000000	113036.000000
mean	15.665753	2014.401739	35.919212	11.901660	267.296366	452.938427	285.051665	469.318695	754.370360
std	8.781567	1.272510	11.021936	9.561857	549.835483	922.071219	453.887443	884.866118	1309.094674
min	1.000000	2011.000000	17.000000	1.000000	1.000000	2.000000	-30.000000	1.000000	2.000000
25%	8.000000	2013.000000	28.000000	2.000000	2.000000	5.000000	29.000000	28.000000	63.000000
50%	16.000000	2014.000000	35.000000	10.000000	9.000000	24.000000	101.000000	108.000000	223.000000
75%	23.000000	2016.000000	43.000000	20.000000	42.000000	70.000000	358.000000	432.000000	800.000000
max	31.000000	2016.000000	87.000000	32.000000	2171.000000	3578.000000	15096.000000	42978.000000	58074.000000

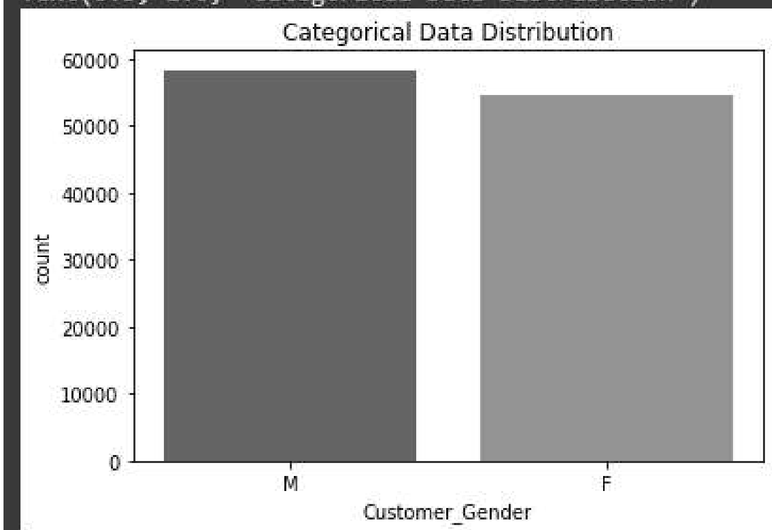
```
category_counts = df['Country'].value_counts()
```

print(category_counts)

```
United States     39206
Australia         23936
Canada            14178
United Kingdom    13620
Germany           11098
France            10998
Name: Country, dtype: int64
```

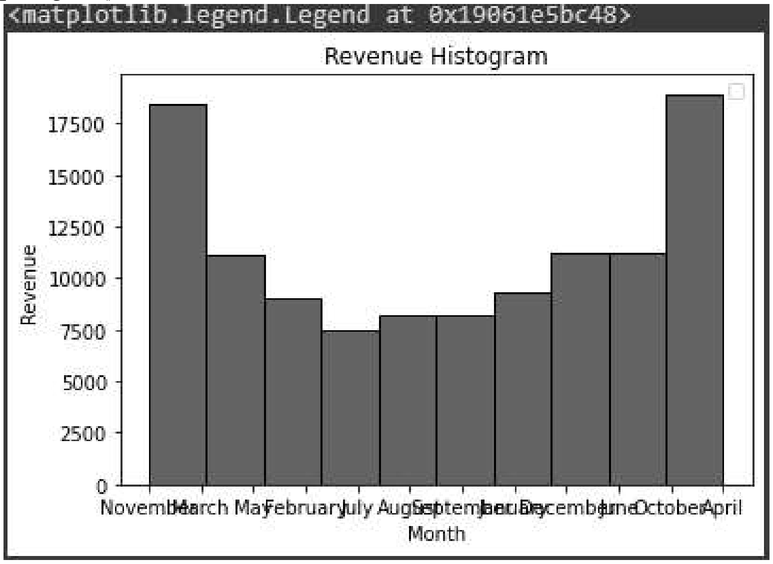
sns.countplot(data=df, x='Customer_Gender')
plt.title("Categorical Data Distribution")

```
Text(0.5, 1.0, 'Categorical Data Distribution')
```



plt.hist(df['Month'], bins=10, edgecolor='black')
plt.title('Revenue Histogram')
plt.xlabel('Month')
plt.ylabel('Revenue')
plt.legend()

```
<matplotlib.legend.Legend at 0x19061e5bc48>
```



```
one_hot_encoded_data = pd.get_dummies(df, columns = ['Product'])
one_hot_encoded_data
```

	Date	Day	Month	Year	Customer_Age	Age_Group	Customer_Gender	Country	State	Product_Category	...	Product_Touring-3000 Blue, 62
0	26-11-2013	26	November	2013	19	Youth (<25)	M	Canada	British Columbia	Accessories	...	0
1	26-11-2015	26	November	2015	19	Youth (<25)	M	Canada	British Columbia	Accessories	...	0
2	23-03-2014	23	March	2014	49	Adults (35-64)	M	Australia	New South Wales	Accessories	...	0
3	23-03-2016	23	March	2016	49	Adults (35-64)	M	Australia	New South Wales	Accessories	...	0
4	15-05-2014	15	May	2014	47	Adults (35-64)	F	Australia	New South Wales	Accessories	...	0

```
one_hot_encoded_data = pd.get_dummies(df, columns = ['State'])
one_hot_encoded_data
```

	Date	Day	Month	Year	Customer_Age	Age_Group	Customer_Gender	Country	Product_Category	Sub_Category	...	State_Tasmania
0	26-11-2013	26	November	2013	19	Youth (<25)	M	Canada	Accessories	Bike Racks	...	0
1	26-11-2015	26	November	2015	19	Youth (<25)	M	Canada	Accessories	Bike Racks	...	0
2	23-03-2014	23	March	2014	49	Adults (35-64)	M	Australia	Accessories	Bike Racks	...	0
3	23-03-2016	23	March	2016	49	Adults (35-64)	M	Australia	Accessories	Bike Racks	...	0
4	15-05-2014	15	May	2014	47	Adults (35-64)	F	Australia	Accessories	Bike Racks	...	0
...	...	...	...	...	...	...	...	...	...	...	...	...
113031	12-04-2016	12	April	2016	41	Adults (35-64)	M	United Kingdom	Clothing	Vests	...	0

Signature : ____________

Practical 6 Implementation of non parametric test

a) Sign test

Input and output

```
import numpy as np
from scipy.stats import binom

data = [
    (4, 3),
    (5, 6),
    (7, 6),
    (4, 5),
    (3, 2),
    (6, 6),

    (8, 7),
    (5, 4),
    (6, 6),
    (4, 4),
]

differences = [x - y for x, y in data]
positive_diff = sum(1 for diff in differences if diff > 0)
negative_diff = sum(1 for diff in differences if diff < 0)
n = len(data)
k = positive_diff
print(k)
```

```
5
```

```
p_value = binom.cdf(k, n, 0.5) + binom.sf(k - 1, n, 0.5)

print("p-value:", p_value)
```

```
p-value: 1.2460937499999998
```

b) Wilcoxon matched pair test (or sign rank test)

Input and output

```
import numpy as np
from scipy.stats import wilcoxon

before = [35, 42, 38, 46, 33, 29, 42, 37, 40, 32]
after = [30, 40, 36, 44, 28, 24, 41, 35, 38, 30]

differences = [after[i] - before[i] for i in range(len(before))]
```

```
statistic, p_value = wilcoxon(differences)

print("Test Statistic:", statistic)
print("p-value:", p_value)
```

```
Test Statistic: 0.0
p-value: 0.004016514660032747
```

c) Kruskal wallis test

Input and output

```
import scipy.stats as stats

group_A = [23, 27, 21, 19, 25]
group_B = [18, 16, 20, 24, 22]
group_C = [10, 13, 15, 11, 12]

H, p_value = stats.kruskal(group_A, group_B, group_C)

print("Kruskal-Wallis H:", H)
print("p-value:", p_value)
```

```
Kruskal-Wallis H: 10.220000000000006
p-value: 0.006036082923599548
```

Signature : ____________

Practical 7 Perform ANOVA testing

a) One way ANOVA

Input and output

Q. A firm wishes to compare 4 programmes for training workers to perform a certain task. 20 new employees are randomly assigned to these programmes with 5 in each. At the end of the training, a test is conducted to see how quickly the trainees perform that task. The number of times the task is performed per hour is recorded for each trainee. Perform ANOVA to check the effectiveness of the programme. (Assume the level of significance = 0.05) Programme_1= [9, 12, 14, 11, 13] programme_2=[10, 6, 9, 9, 10] Programme_3= [12, 14, 11, 13, 11] Programme_4= [9, 8, 11, 7,8]

Null Hypothesis: The average number of times the task is performed are equal
Alternative Hypothesis: At least one pair of averages is not equal

```
from scipy.stats import f_oneway

programme_1=[9,12,14,11,13]
programme_2=[10,6,9,9,10]
programme_3=[12,14,11,13,11]
programme_4=[9,8,11,7,8]

f_oneway(programme_1,programme_2,programme_3,programme_4)
```

```
F_onewayResult(statistic=7.044871794871795, pvalue=0.0031129438989961743)
```

Conclusion:- we see that p-value(0.003) is less than 0.05(level of significance)
hence,we reject null null hypothesis
we may conclude that at least one of the training programme has different effect

Signature : ___________

b) Two way ANOVA

Input and output

A botanist wants to know whether or not plant growth is influenced by sunlight exposure and watering frequency. She plants 30 seeds and lets them grow for two months under different conditions for sunlight exposure and watering frequency. After two months, she records the height of each plant, in inches. Use the following steps to perform a two-way ANOVA to determine if watering frequency and sunlight exposure ave a significant effect on plant growth, and to determine if there is any interaction effect between watering frequency and sunlight exposure.

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
from statsmodels.formula.api import ols

df=pd.DataFrame({
    'water':np.repeat(['daily','weekly'],15),
    'sun':np.tile(np.repeat(['low','med','high'],5),2),
    'height':[6,6,6,5,6,5,5,6,4,5,6,6,7,8,7,3,4,4,4,5,4,4,4,4,4,5,6,6,7,8,]
})
print(df)
```

```
      water   sun  height
0     daily   low       6
1     daily   low       6
2     daily   low       6
3     daily   low       5
4     daily   low       6
5     daily   med       5
6     daily   med       5
7     daily   med       6
8     daily   med       4
9     daily   med       5
10    daily  high       6
11    daily  high       6
12    daily  high       7
13    daily  high       8
14    daily  high       7
15   weekly   low       3
16   weekly   low       4
17   weekly   low       4
18   weekly   low       4
19   weekly   low       5
20   weekly   med       4
21   weekly   med       4
22   weekly   med       4
23   weekly   med       4
...
26   weekly  high       6
27   weekly  high       6
28   weekly  high       7
29   weekly  high       8
```

```
model=ols('height ~ C(water) + C(sun) + C(water):C(sun)', data=df).fit()
sm.stats.anova_lm(model,typ=2)
```

	sum_sq	df	F	PR(>F)
C(water)	8.533333	1.0	16.0000	0.000527
C(sun)	24.866667	2.0	23.3125	0.000002
C(water):C(sun)	2.466667	2.0	2.3125	0.120667
Residual	12.800000	24.0	NaN	NaN

```
# water: p-value = 0.000527
# sun: p-value = 0.000002
# water*sun: p-value = 0.120667
```

Since the values for water and sun lite both less than 05 this means that both factors have a statistically significant effect on plant height. And since the p-value for the interaction effect (120667) is not less than 05, this tells us that there is no significant interaction effect between sunlight exposure and watering frequency

Signature : ___________

Practical 8 Implement regression analysis and fit a straight line

Input and output

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

x=np.array([1,2,3,4,5])
y=np.array([7,14,15,18,19])
n=np.size(x)

x_mean=np.mean(x)
y_mean=np.mean(y)
x_mean,y_mean
```

```
(3.0, 14.6)
```

```
Sxx=np.sum(x*x)-n*x_mean*x_mean
Sxy=np.sum(x*y)-n*x_mean*y_mean
Sxx,Sxy
```

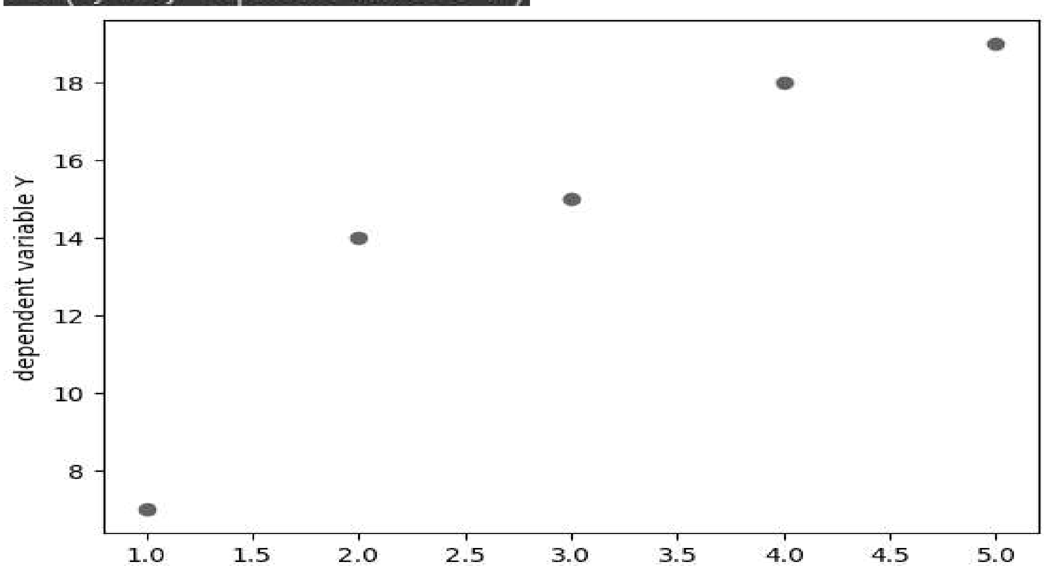
```
(10.0, 28.0)
```

```
b1=Sxy/Sxx
b0=y_mean - b1*x_mean
print('Slope b1 is' ,b1)
print('Intercept b0 is' ,b0)
```

```
Slope b1 is 2.8
Intercept b0 is 6.200000000000001
```

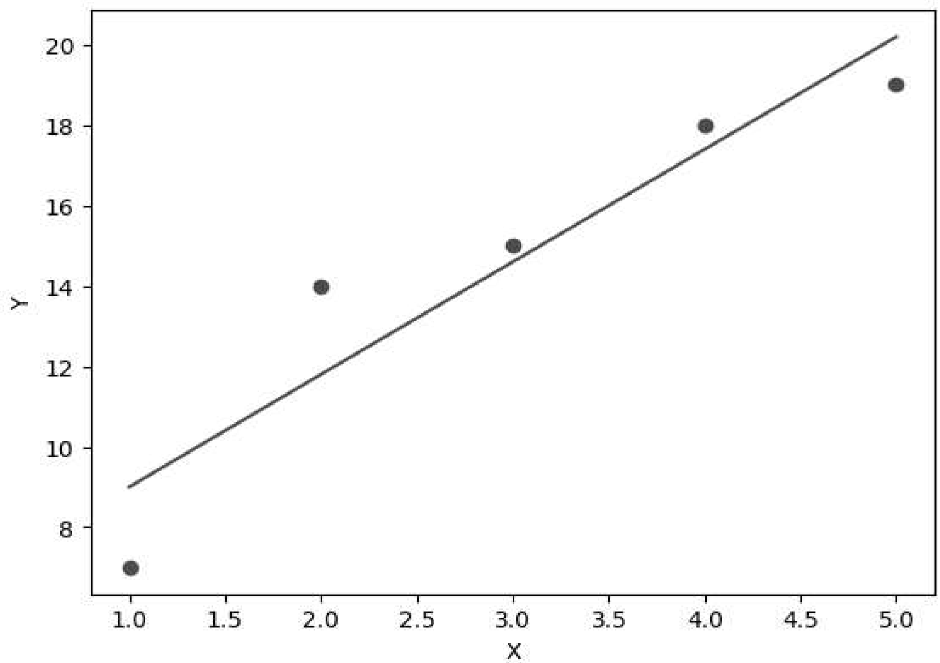
```
plt.scatter(x,y)
plt.xlabel('Independent variable X')
plt.ylabel('dependent variable Y')
```

```
Text(0, 0.5, 'dependent variable Y')
```



```
y_pred=b1*x+b0
plt.scatter(x,y,color='red')
plt.plot(x,y_pred,color='green')
plt.xlabel('X')
plt.ylabel('Y')
```

```
Text(0, 0.5, 'Y')
```



Signature : ___________

Practical 9 Perform time series analysis

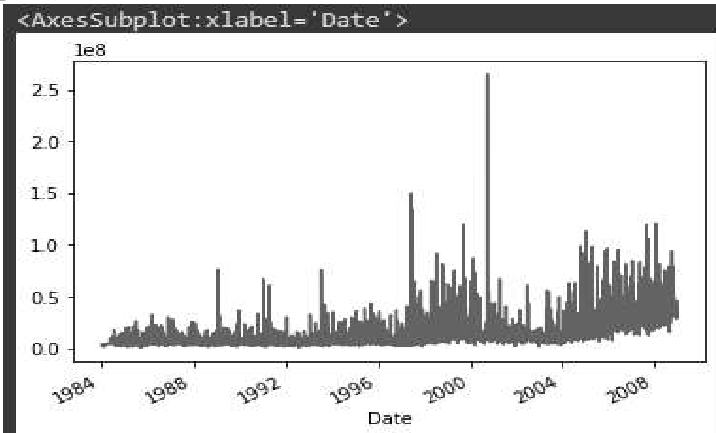
Input and output

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv(r"C:\Users\rahul\Desktop\RJ COLLEGE\aapl.csv", parse_dates=True,index_col="Date")
df.head()
```

	Open	High	Low	Close	Volume	Adj Close
Date						
2008-10-14	116.26	116.40	103.14	104.08	70749800	104.08
2008-10-13	104.55	110.53	101.02	110.26	54967000	110.26
2008-10-10	85.70	100.00	85.00	96.80	79260700	96.80
2008-09-10	93.35	95.80	86.60	88.74	57763700	88.74
2008-08-10	85.91	96.33	85.68	89.79	78847900	89.79

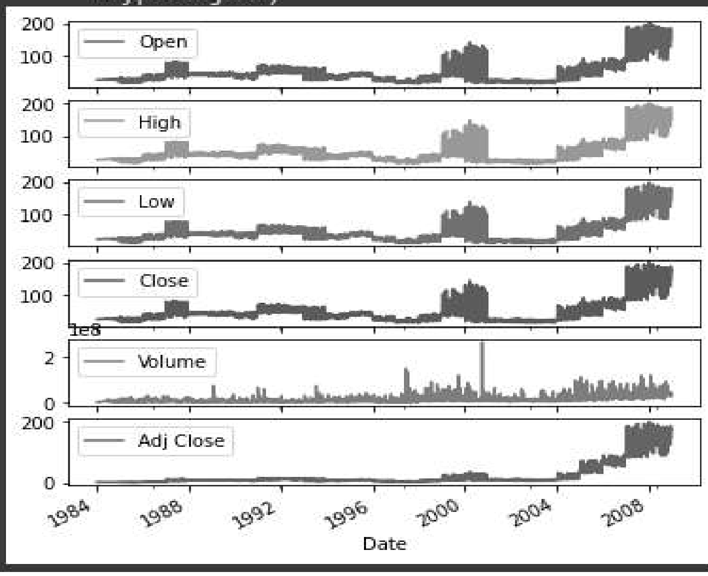
```
print(df)
```

<AxesSubplot:xlabel='Date'>




```
df.plot(subplots=True, figsize=(6, 6))
```

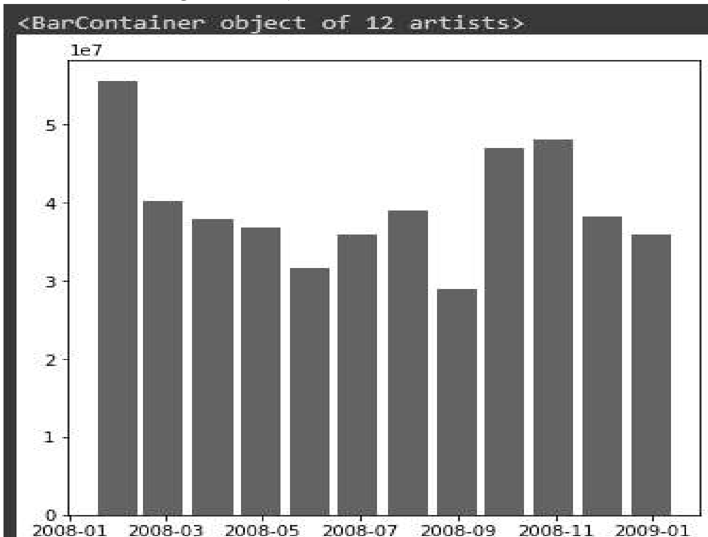
```
array([<AxesSubplot:xlabel='Date'>, <AxesSubplot:xlabel='Date'>,
       <AxesSubplot:xlabel='Date'>, <AxesSubplot:xlabel='Date'>,
       <AxesSubplot:xlabel='Date'>, <AxesSubplot:xlabel='Date'>],
      dtype=object)
```



Resampling: Resampling is a methodology of economically using a data sample to improve the accuracy and quantify the uncertainty of a population parameter. Resampling for months or weeks and making bar plots is another very simple and widely used method of finding seasonality.

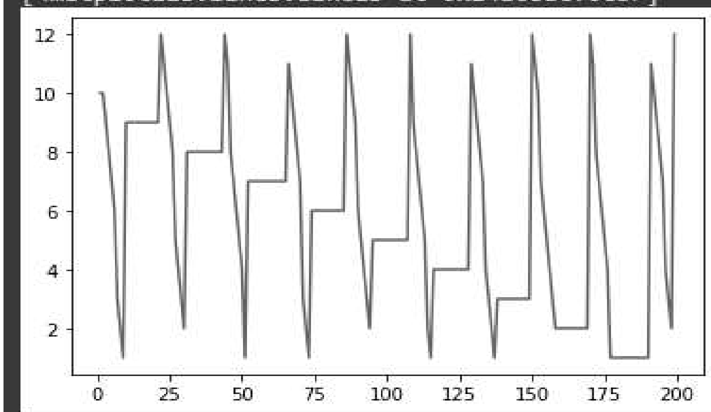
```
# Resampling the time series data based on monthly 'M' frequency
df_month = df.resample("M").mean()

# using subplot
fig, ax = plt.subplots(figsize=(6, 6))

# plotting bar graph
ax.bar(df_month['2008':].index,
      df_month.loc['2008':, "Volume"],
      width=25, align='center')
```

```
<BarContainer object of 12 artists>
```



```
df = pd.read_csv(r"C:\Users\rahul\Desktop\RJ COLLEGE\aapl.csv")
import datetime as dt
df['Date']= pd.to_datetime(df['Date'])
plt.plot(df['Date'][1:200].dt.month)
```

```
[<matplotlib.lines.Line2D at 0x14a68dcf0c8>]
```




```
df.iloc[1:200,:]
```

Date	Open	High	Low	Close	Volume	Adj Close
2008-10-13	104.55	110.53	101.02	110.26	54967000	110.26
2008-10-10	85.70	100.00	85.00	96.80	79260700	96.80
2008-09-10	93.35	95.80	86.60	88.74	57763700	88.74
2008-08-10	85.91	96.33	85.68	89.79	78847900	89.79
2008-07-10	100.48	101.50	88.95	89.16	67099000	89.16
...	...	...	...	...	...	...
2008-07-01	181.25	183.60	170.23	177.64	74006900	177.64
2008-04-01	191.45	193.00	178.89	180.05	51994000	180.05
2008-03-01	195.41	197.39	192.69	194.93	30073800	194.93
2008-02-01	199.27	200.26	192.55	194.84	38542100	194.84
2007-12-31	199.50	200.50	197.75	198.08	19261900	198.08

199 rows × 6 columns

Signature : ___________