Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Academic Year: 2023-2024

Name: Bhavvya Jain	**Sap Id:** 60018220108
Course: Data Mining and Analytics	**Roll No:** S019
Branch: Artificial Intelligence and Data Science	**Div:** S
Course Code: DJS22ADL403	**Batch:** A1

Experiment 01

Title	**Measures of Central tendency and dispersion**
Aim	To measure central tendency and dispersion of data using Python
Software	Google Colab

Implementation

```
import numpy as np
import statistics as st
from scipy import stats
```

1. **Find arithmetic mean of 20, 2, 7, 1, 34**

Code:

```
A=[20, 2, 7, 1, 34]
print("Arithmetic Mean of A = ",str(np.mean(A)))
```

Output:

```
Arithmetic Mean of A =  12.8
```

2. **Create following matrix using multidimensional array and calculate arithmetic mean for each column, each row and considering entire data.**
   14 17 12 33 44
   15 6 27 8 19
   23 2 54 1 4

Code:

```
B=np.array([[14, 17, 12, 33, 44],[15, 6, 27, 8, 19],[23, 2, 54, 1,
4]])
print("Mean of Matrix=",np.mean(B))
print("Row-wise Mean=",np.mean(B,axis=1))
print("Column-wise Mean=",np.mean(B,axis=0))
```

Output:

```
Mean of Matrix= 18.6
Row-wise Mean= [24.  15.  16.8]
Column-wise Mean= [17.33333333  8.33333333 31.          14.
22.33333333]
```

3. **Find minimum value, maximum value and range for entire data, column wise and row wise.**

Code:

```
c=np.array([[3,7,5],[8,4,3],[2,4,9]])
print("Minimum:",np.min(c))
print("Columnwise minimum:",np.min(c,axis=0))
print("Rowwise minimum:",np.min(c,axis=1))
print("Maximum:",np.max(c))
print("Columnwise maximum:",np.max(c,axis=0))
print("Rowwise maximum:",np.max(c,axis=1))
print("Range:",np.ptp(c))
print("Columnwise range:",np.ptp(c,axis=0))
print("Rowwise range:",np.ptp(c,axis=1))
```

Output:

```
Minimum: 2
Columnwise minimum: [2 4 3]
Rowwise minimum: [3 3 2]
Maximum: 9
Columnwise maximum: [8 7 9]
Rowwise maximum: [7 8 9]
Range: 7
Columnwise range: [6 3 6]
Rowwise range: [4 5 7]
```

4. **Find weighted average for the data given below.**

Outcomes	**Frequency**
1	**4**
2	**3**
3	**2**
4	**1**

Code:

```
x=np.array([1,2,3,4])
n=np.array([4,3,2,1])
print("Weighted Mean:",np.average(x,weights=n))
```

Output:

```
Weighted Mean: 2.0
```

5. **The speed of 13 vehicles is 99,86,87,88,111,86,103,87,94,78,77,85,86 Find mean, median and mode**

Code:

```
from scipy import stats
s=[99,86,87,88,111,86,103,87,94,78,77,85,86]
print("Mean:",np.mean(s))
print("Median:",np.median(s))
print("Mode:",stats.mode(s))
```

Output:

```
Mean: 89.76923076923077
Median: 87.0
Mode: ModeResult(mode=86, count=3)
```

6. **Calculate geometric mean for each column, each row and considering entire data.**
   1 3 27
   3 4 6
   7 6 3
   3 6 8

Code:

```
from scipy.stats.mstats import gmean
d=([1,3,27],[3,4,6],[7,6,3],[3,6,8])
print("Geometric mean:",gmean(gmean(d)))
print("Col GM:",gmean(d,axis=0))
print("Row GM:",gmean(d,axis=1))
```

Output:

```
Geometric mean: 4.663506066439651
Col GM: [2.81731325 4.55901411 7.89644408]
Row GM: [4.32674871 4.16016765 5.01329793 5.24148279]
```

7. **Calculate harmonic mean for 1, 3, 5, 7, 9**

Code:

```
import statistics as st
H=[1,3,5,7,9]
print("Harmonic Mean:",st.harmonic_mean(H))
```

Output:

```
Harmonic Mean: 2.797513321492007
```

8. **Calculate median for each column, each row and considering entire data.**
   30 65 70
   80 95 10
   50 90 60

Code:

```
M=np.array([[30,65,70],[80,95,10],[50,90,60]])
print("Median:",np.median(M))
print("Col Median:",np.median(M,axis=0))
print("Row Median:",np.median(M,axis=1))
```

Output:

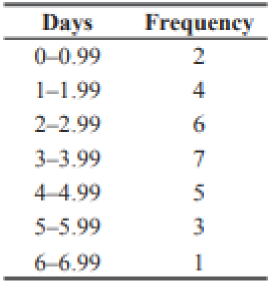
```
Median: 65.0
Col Median: [50. 90. 60.]
Row Median: [65. 80. 60.]
```

9. The number of solar heating systems available to the public is quite large, and their heat-storage capacities are quite varied. Here is a distribution of heat-storage capacity (in days) of 28 systems that were tested recently by University Laboratories, Inc.:

University Laboratories, Inc., knows that its report on the tests will be widely circulated and used as the basis for tax legislation on solar-heat allowances. Ittherefore wants the measures it uses to be as reflective of the data as possible.

a. Compute the mean for these data.
b. Compute the mode for these data.
c. Compute the median for these data.

Select the answer among parts (a), (b), and (c) that best reflects the central tendency of the test data and justify your choice.



Days	Frequency
0–0.99	2
1–1.99	4
2–2.99	6
3–3.99	7
4–4.99	5
5–5.99	3
6–6.99	1

Code:

```
A = np.array([[0,0.99,2,0],[1,1.99,4,0],[2,2.99,6,0],[3,3.99,7,0]
,[4,4.99,5,0],[5,5.99,3,0],[6,6.99,1,0]])
def
  grouped_mean(A):n
  = len(A)
  sum
  =0f=0
  for i in range(n):
    sum = sum +
    (((A[i][0]+A[i][1])/2)*A[i][2])f = f+
    A[i][2]
  mean =
  sum/freturn
  mean

def
  grouped_median(A):n
  = len(A)
  f = 0
  for i in range
    (n):if i ==0:
      A[i][3] = A[i][2]
    else:
      A[i][3] = A[i-1][3] +
    A[i][2]f = f + A[i][2]
  mid = f/2
  for i in range (1,n):
    if (A[i-1][3]< mid <= A[i][3]):
      med_class = i
  median = A[med_class][0] + (A[med_class][1] - A [med_class][0])
*(mid - A[med_class -
  1][3])/A[med_class][2]return (median)
```

```
 def grouped_mode(A):
n = len(A)
mod_class = 0
   for i in range (n):
     if (A[i][2]>
       A[mod_class][2]):mod_class
       = i
   mode = A[mod_class][0] + (A[mod_class][1] - A
 [mod_class][0])*((A[mod_class][2]-A[mod_class -
 1][2])/(2*A[mod_class][2]- A[mod_class-1][2]-A[mod_class
 +1][2]))
   return mode
 print("Mean of data =",grouped_mean(A))
 print("Median of data
 =",grouped_median(A))print("Mode of data
 =",grouped_mode(A))
 print("The mean of the data is a calculated value whereas the
 median and the mode of the data are observed values only. In
 this case there are no extreme outliers, so in this case, the
 mean best reflects the central tendency of the data.")
```

Output:

```
 Mean of data = 3.2807142857142857
 Median of data =
 3.282857142857143Mode of data =
 3.33
 The mean of the data is a calculated value whereas the median
 andthe mode of the data are observed values only. In this case
 thereare no extreme outliers, so in this case, the mean best
 reflects the central tendency of the data.
```

10. Write a function for calculating percentile and determine 30, 50, 75 and 90 percentiles for the following data. 30,40,72,83,25,10,50,90,60,15,5,9,34,23,67,80,67,45

Code:

```
A=[30,40,72,83,25,10,50,90,60,15,5,9,34,23,67,80,67,45]
def percentile(A,p):
  A.sort()
  n=len(A)
  pr=p*(n+1)/100
  i=int(pr)-1
  fr=pr-int(pr)
  ans=A[i]+fr*(A[i+1]-A[i])
  return ans
print("30th percentile =",percentile(A,30))
print("50th percentile =",percentile(A,50))
print("75th percentile =",percentile(A,75))
print("90th percentile =",percentile(A,90))
```

Output:

```
30th percentile = 24.4
50th percentile = 42.5
75th percentile = 68.25
90th percentile = 83.70000000000002
```

11. There are 39 plants in the garden. A few plants were selected randomly and their heights in cm were recorded as follows: 51, 38, 79, 46, 57. Calculate the standard deviation of their heights.

Code:

```
B=np.array([51,38,79,46,57])
print("Standard deviation of sample= ",st.stdev(B))
print("Standard deviation of population= ",np.std(B))
```

Output:

```
Standard deviation of sample=  15.491933384829668
Standard deviation of population=  13.876599006961325
```

12. The head chef of The Flying Taco has just received two d ozen tomatoes from her supplier, but she isn′ t ready to accept them. She knows from the invoice that the average weight of a tomato is 7.5 ounces, but she insists that all be of uniform weight. She will accept them only if the average weight is 7.5 ounces and the standard deviation is less than 0.5 ounce. Here are the weights of the tomatoes 6.3 7.2 7.3 8.1 7.8 6.8 7.5 7.8 7.2 7.5 8.1 8.2 8.0 7.4 7.6 7.7 7.6 7.4 7.5 8.4 7.4 7.6 6.2 7.4

What is the chef′ s decision and why?

Code:

```
W = [6.3, 7.2, 7.3, 8.1, 7.8, 6.8, 7.5, 7.8, 7.2, 7.5, 8.1, 8.2 ,
8.0, 7.4, 7.6, 7.7, 7.6, 7.4, 7.5, 8.4, 7.4, 7.6,
6.2, 7.4]
mean = np.mean(W)
stddev = np.std(W)
print("Mean =",mean," Standard deviation =",stddev)
if(mean == 7.5 and stddev<0.5):
 print("The batch will be accepted.")
else:
 print("The chef will not accept the batch.")
```

Output:

```
Mean = 7.5  Standard deviation = 0.5163977794943222
The chef will not accept the batch.
```

13. A company is considering employing one of two training programs. Two groups were trained for the same task. Group 1 was trained by program A; group 2, by program B. For the first group, the times required to train the employees had an average of 32.11 hours and a variance of 68.09. In the second group, the average was 19.75 hours and the variance was 71.14. Which training program has less relative variability in its performance?

Code:

```
W = [6.3, 7.2, 7.3, 8.1, 7.8, 6.8, 7.5, 7.8, 7.2, 7.5, 8.1, 8.2
,8.0, 7.4, 7.6, 7.7, 7.6, 7.4, 7.5, 8.4, 7.4, 7.6,
6.2, 7.4]
mean = np.mean(W)
stddev = np.std(W)
print("Mean =",mean," Standard deviation =",stddev)
if(mean == 7.5 and stddev<0.5):
  print("The batch will be accepted.")
else:
  print("The chef will not accept the batch.")
```

Output:

```
Mean = 7.5  Standard deviation = 0.5163977794943222
The chef will not accept the batch.
```

14. Test scores for a college statistics class held during the day are:99 56 78 55.5 32 90 80 81 56 59 45 77 84.5 84 70 72 68 32 79 90

Test scores for a college statistics class held during the evening are:98 78 68 83 81 89 88 76 65 45 98 90 80 84.5 85 79 78 98 90 79 81 25.5

- **a. Find the smallest and largest values, the median, and the first and thirdquartile for the day class.**
- **b. Find the smallest and largest values, the median, and the first and thirdquartile for the night class.**
- **c. For each data set, what percentage of the data is between the smallest value and the first quartile? the first quartile and the median? the medianand the third quartile? the third quartile and the largest value? What percentage of the data is between the first quartile and the largest value?**
- **d. Create a box plot for each set of data. Use one number line for both boxplots.**
- **e. Which box plot has the widest spread for the middle 50% of the data (thedata between the first and third quartiles)? What does this mean for thatset of data in comparison to the other set of data**

Code:

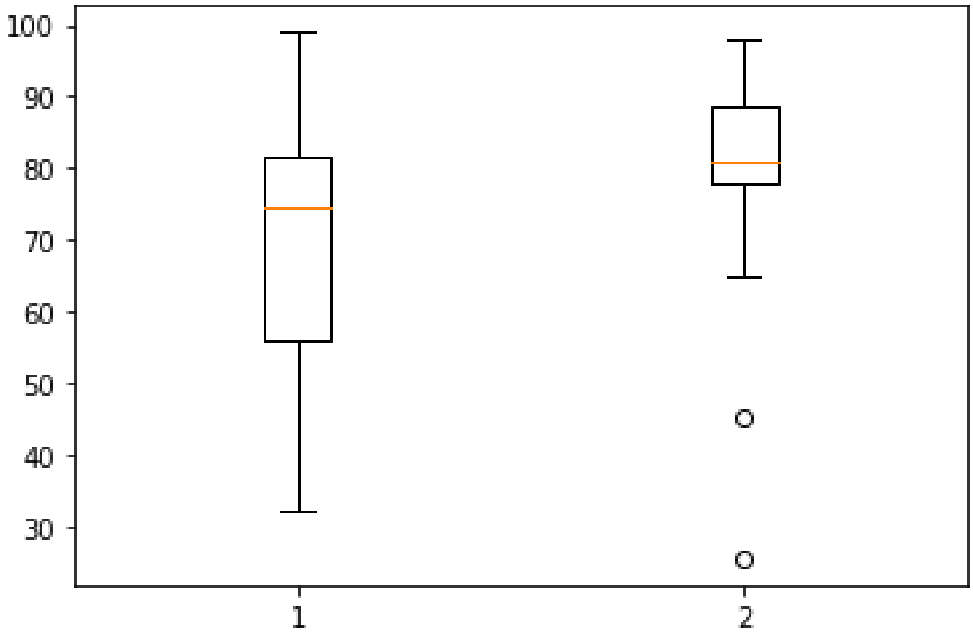
```
import matplotlib.pyplot as plt
# DAY
day = [99, 56, 78, 55.5, 32, 90, 80, 81, 56, 59, 45, 77, 84.5, 84, 70, 72, 68, 32, 79, 90]
day.sort()
c1 = 0
c2 = 0
c3 = 0
c4 = 0
c5 = 0
print("Sorted Day: ", day)
median_day = np.median(day)
print("Smallest Value is:", *day[:1])
print("Largest Value is:", *day[-1:])
print("Median is:", median_day)
q1 = np.quantile(day, 0.25)
q3 = np.quantile(day, 0.75)
print("1st Quartile of Day class is:", q1)
print("3rd Quartile of Day class is:", q3)
for i in day:
 if(i<=56):
  c1 = c1 + 1
  i = i + 1
print(c1)
p1 = (c1/len(day))*100
print(p1, "% of the data is between the smallest value and the first quartile.")
for i in day:
 if(i>=q1 and i<=median_day):
  c2 = c2 + 1
  i = i + 1
print(c2)
p2 = (c2/len(day))*100
print(p2, "% of the data is between the first quartile and the median.")
for i in day:
 if(i>=median_day and i<=q3):
  c3 = c3 + 1
  i = i + 1
print(c3)
p3 = (c3/len(day))*100
print(p3, "% of the data is between the median and the third quartile.")
for i in day:
 if(i>=q3 and i<=99):
  c4 = c4 + 1
  i = i + 1
print(c4)
p4 = (c4/len(day))*100
print(p4, "% of the data is between the third quartile and the largest value.")
for i in day:
 if(i>=q1 and i<=99):
  c5 = c5 + 1
  i = i + 1
print(c5)
p5 = (c5/len(day))*100
print(p5, "% of the data is between the first quartile and the largest value.")
# NIGHT
night = [98, 78, 68, 83, 81, 89, 88, 76, 65, 45, 98, 90, 80, 84.5, 85, 79, 78, 98, 90, 79, 81, 25.5]
night.sort()
n1 = 0
n2 = 0
n3 = 0
n4 = 0
n5 = 0
print("Sorted night: ", night)
median_night = np.median(night)
print("Smallest Value is:", *night[:1])
print("Largest Value is:", *night[-1:])
print("Median is:", np.median(night))
qn_1 = np.quantile(night, 0.25)
qn_3 = np.quantile(night, 0.75)
print("1st Quartile of night class is:", qn_1)
print("3rd Quartile of night class is:", qn_3)
for i in night:
 if(i<=qn_1):
  n1 = n1 + 1
  i = i + 1
p_n1 = (n1/len(night))*100
```

```
print(p_n1, "% of the data is between the smallest value and the
first quartile.")
for i in night:
 if(i>=qn_1 and i<=median_night):
  n2 = n2 + 1
  i = i + 1
p_n2 = (n2/len(night))*100
print(p_n2, "% of the data is between the first quartile and the
median.")
for i in night:
 if(i>=median_night and i<=qn_3):
  n3 = n3 + 1
  i = i + 1
p_n3 = (n3/len(night))*100
print(p_n3, "% of the data is between the median and the third
quartile.")
for i in night:
 if(i>=qn_3 and i<=98):
  n4 = n4 + 1
  i = i + 1
p_n4 = (n4/len(night))*100
print(p_n4, "% of the data is between the third quartile and the
largest value.")
for i in night:
 if(i>=qn_1 and i<=98):
  n5 = n5 + 1
  i = i + 1
p_n5 = (n5/len(night))*100
print(p_n5, "% of the data is between the first quartile and the
largest value.")
# BOX PLOT
print("Box plot for day and night test scores:")
plt.boxplot((day,night))
print("The day test scores has widest spread for the middle 50% of
the data, which means the day test scores of students are not
consistent as compared to night scores.")
```

Output:

```
 Sorted Day:  [32, 32, 45, 55.5, 56, 56, 59, 68, 70, 72, 77, 78, 79,
 80, 81, 84, 84.5, 90, 90, 99]
 Smallest Value is:
 32Largest Value is:
 99 Median is: 74.5
 1st Quartile of Day class is: 56.0
 3rd Quartile of Day class is:
 81.756
 30.0 % of the data is between the smallest value and the first
 quartile.
 6
 30.0 % of the data is between the first quartile and the median.
 5
25.0 % of the data is between the median and the third quartile.
 5
 25.0 % of the data is between the third quartile and the largest
 value.
 16
```

```
80.0 % of the data is between the first quartile and the largest
value.
Sorted night:  [25.5, 45, 65, 68, 76, 78, 78, 79, 79, 80, 81, 81,
83, 84.5, 85, 88, 89, 90, 90, 98, 98, 98]
Smallest Value is:
25.5Largest Value is:
98 Median is: 81.0
1st Quartile of night class is: 78.0
3rd Quartile of night class is:
88.75
31.818181818181817 % of the data is between the smallest value and
the first quartile.
31.818181818181817 % of the data is between the first quartile and
the median.
27.27272727272727 % of the data is between the median and the third
quartile.
27.27272727272727 % of the data is between the third quartile
andthe largest value.
77.27272727272727 % of the data is between the first quartile and
the largest value.
Box plot for day and night test scores:
The day test scores has widest spread for the middle 50% of the
data, which means the day test scores of students are not
consistentas compared to night scores.
return array(a, dtype, copy=False, order=order)
```



Conclusion

In conclusion, understanding central tendency (mean, median, mode) and dispersion (variance, standard deviation) in Python is crucial for analyzing and interpreting data effectively, enabling informed decisionmaking and insightful data exploration.

Signature of Faculty

Shri Vile Parle Kelavani Mandal's

DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING

(Autonomous College Affiliated to the University of Mumbai)

NAAC Accredited with "A" Grade (CGPA : 3.18)

Academic Year: 2023-2024

Name: Bhavvya Jain	**Sap Id:** 60018220108
Course: Data Mining and Analytics	**Roll No:** S019
Branch: Artificial Intelligence and Data Science	**Div:** S
Course Code: DJS22ADL403	**Batch:** A1

Experiment 02

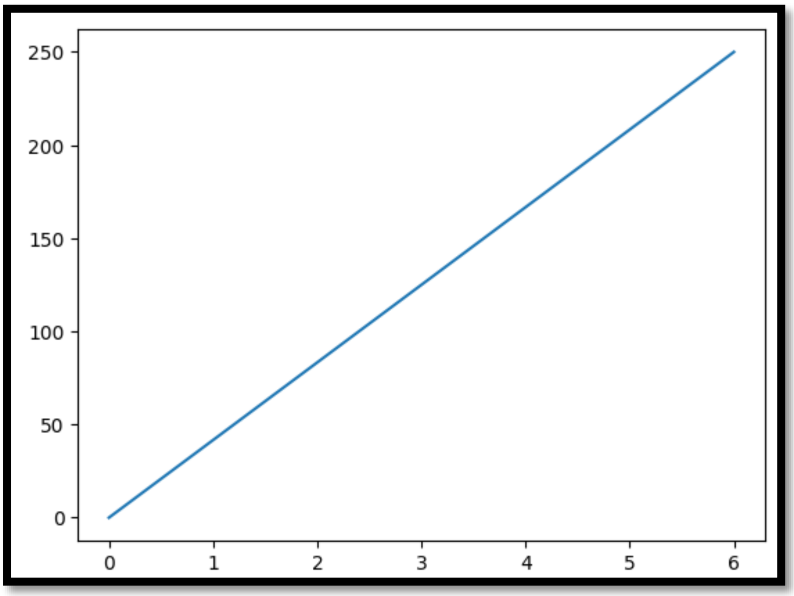
Title	Data Visualization using Matplotlib
Aim	To visualize data using Matplot library
Software	Google Colab

Implementation

Code:

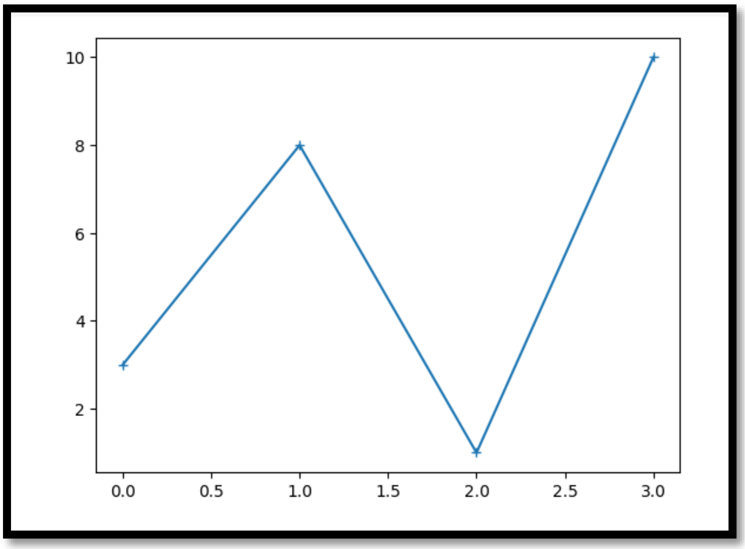
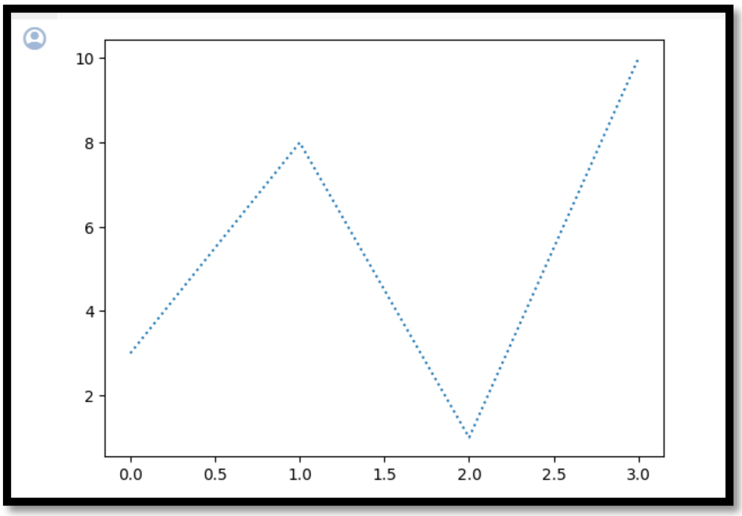
```
import matplotlib.pyplot as plt
import numpy as np
xpoints=np.array([0,6])
ypoints=np.array([0,250])
plt.plot(xpoints, ypoints)
plt.show()
```

Output:

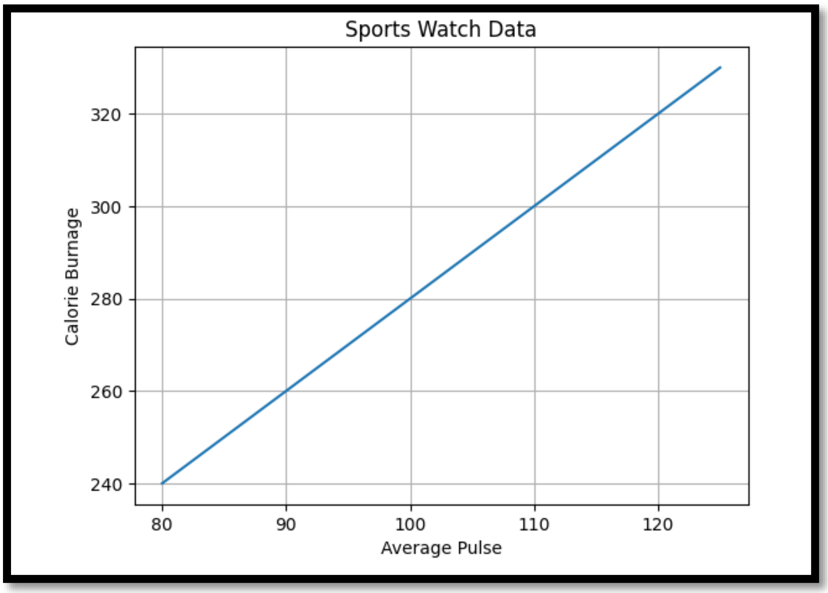


Code:

```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker='*')
plt.show()
```

	Output:
	
	
	Code:
	`ypoints=np.array([3, 8, 1, 10])`
	`plt.plot(ypoints, linestyle='dotted')`
	`plt.show()`
	Output:
	
	
	Code:
	`x=np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])`
	`y=np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])`
	`plt.plot(x,y)`
	`plt.title("Sports Watch Data")`
	`plt.xlabel("Äverage Pulse")`
	`plt.ylabel("Calorie Burnage")`
	`plt.grid()`
	`plt.show()`

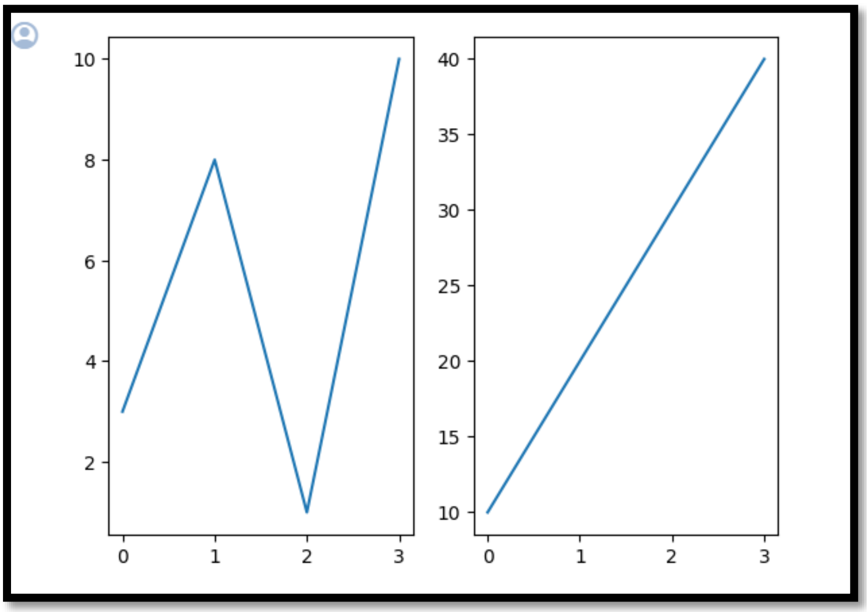
Output:



Code:

```
#plot 1:
x= np.array([0, 1, 2, 3])
y=np.array([3, 8, 1, 10])
plt.subplot(1, 2, 1)  #1 row, 2 column and 1st figures
plt.plot(x,y)

#plot 2:
x= np.array([0, 1, 2, 3])
y=np.array([10, 20, 30, 40])
plt.subplot(1, 2, 2)  #1 row, 2 column and 2nd figures
plt.plot(x,y)

plt.show()
```

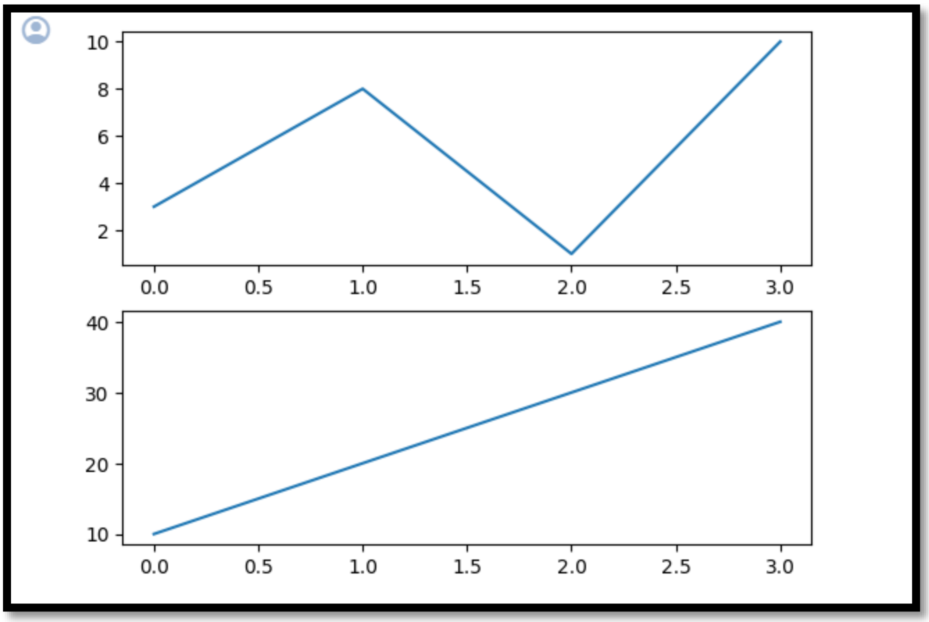
Output:



Code:

```
#plot 1:
x= np.array([0, 1, 2, 3])
y=np.array([3, 8, 1, 10])
plt.subplot(2, 1, 1)
plt.plot(x,y)

#plot 2:
x= np.array([0, 1, 2, 3])
y=np.array([10, 20, 30, 40])
plt.subplot(2, 1, 2)
plt.plot(x,y)

plt.show()
```

Output:

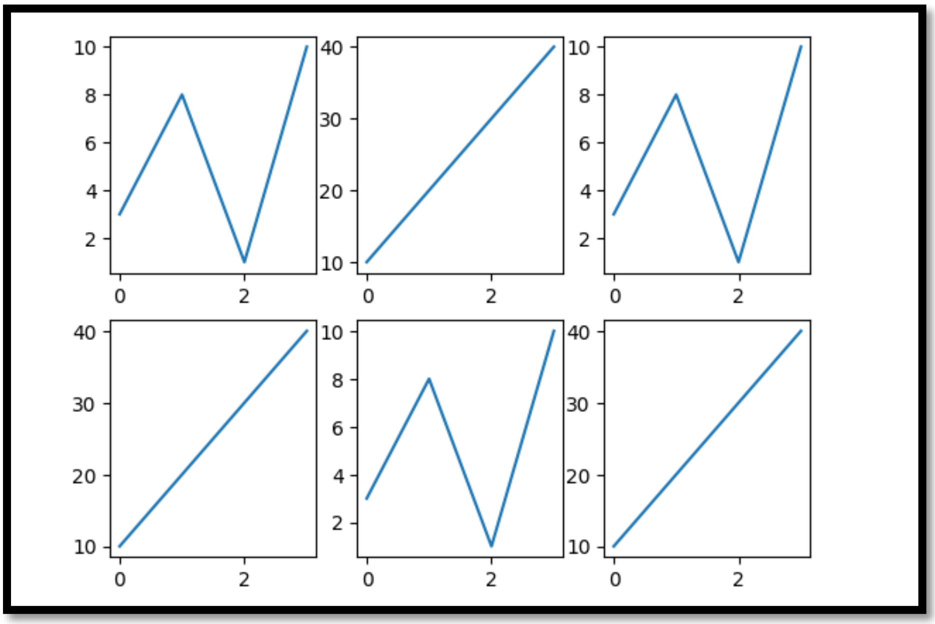


Code:

```
x= np.array([0, 1, 2, 3])
y=np.array([3, 8, 1, 10])
plt.subplot(2, 3, 1)
plt.plot(x,y)
x= np.array([0, 1, 2, 3])
y=np.array([10, 20, 30, 40])
plt.subplot(2, 3, 2)
plt.plot(x,y)
x= np.array([0, 1, 2, 3])
y=np.array([3, 8, 1, 10])
plt.subplot(2, 3, 3)
plt.plot(x,y)
x= np.array([0, 1, 2, 3])
y=np.array([10, 20, 30, 40])
plt.subplot(2, 3, 4)
plt.plot(x,y)
```

```
x= np.array([0, 1, 2, 3])
y=np.array([3, 8, 1, 10])
plt.subplot(2, 3, 5)
plt.plot(x,y)
x= np.array([0, 1, 2, 3])
y=np.array([10, 20, 30, 40])
plt.subplot(2, 3, 6)
plt.plot(x,y)
plt.show()
```

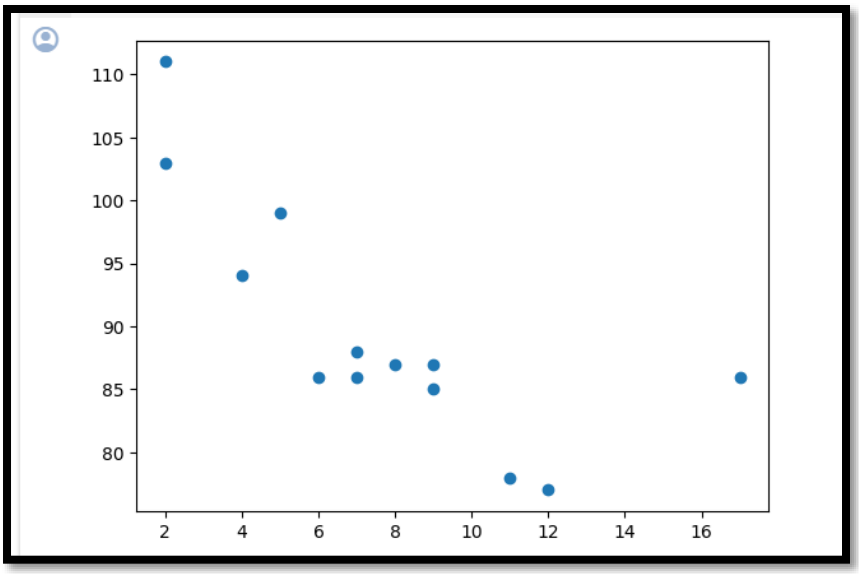
Output:



Code:

```
x=np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y=np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
plt.scatter(x,y)
plt.show()
```

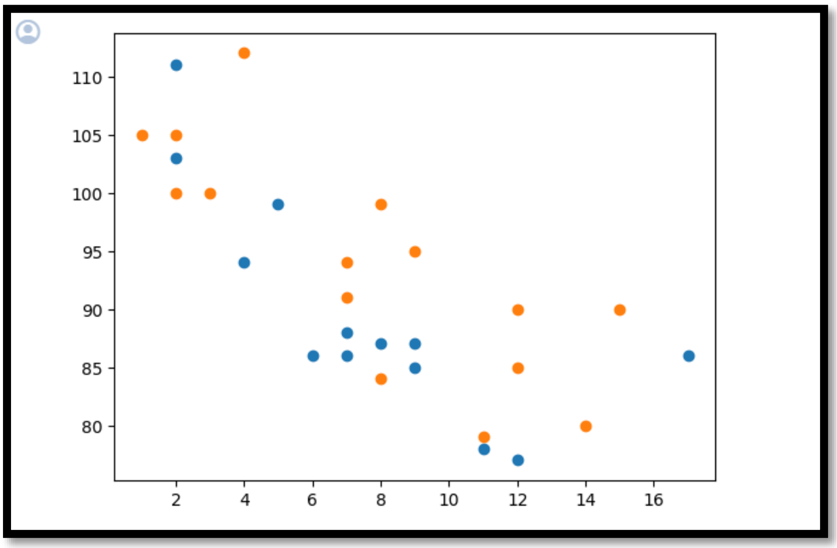
Output:



Code:

```
#Day 1, age and speed of 13 cars:
x=np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y=np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
plt.scatter(x,y)
#Day 2, age and speed of 13 cars:
x=np.array([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])
y=np.array([100,105,84,105,90,99,90,95,94,100,79,112,91,80,85])
plt.scatter(x,y)
plt.show()
```

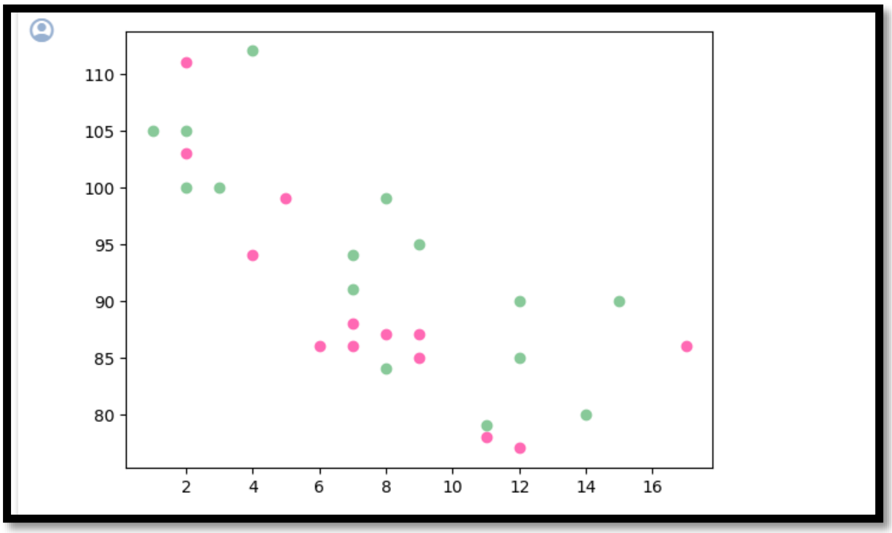
Output:



Code:

```
#Day 1, age and speed of 13 cars:
x=np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y=np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
plt.scatter(x,y,color='hotpink')
#Day 2, age and speed of 13 cars:
x=np.array([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])
y=np.array([100,105,84,105,90,99,90,95,94,100,79,112,91,80,85])
plt.scatter(x,y,color='#88c999')
plt.show()
```

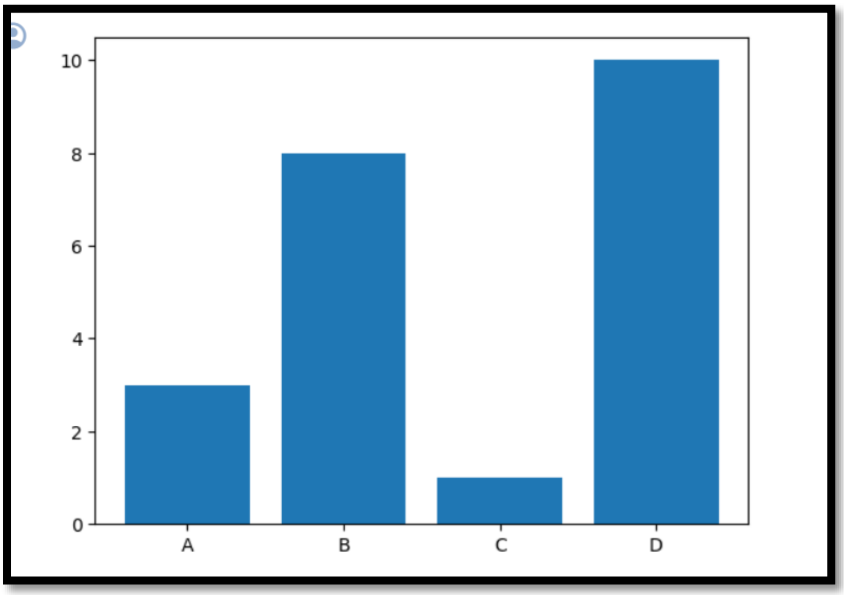
Output:



Code:

```
x=np.array(['A','B','C','D'])
y=np.array([3, 8, 1, 10])
plt.bar(x,y)
plt.show()
```

Output:

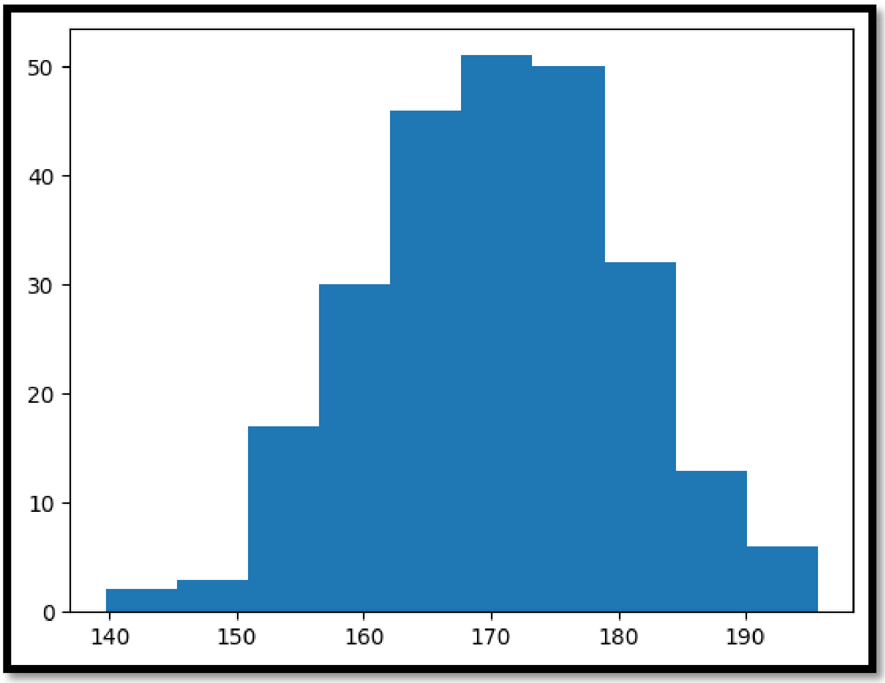


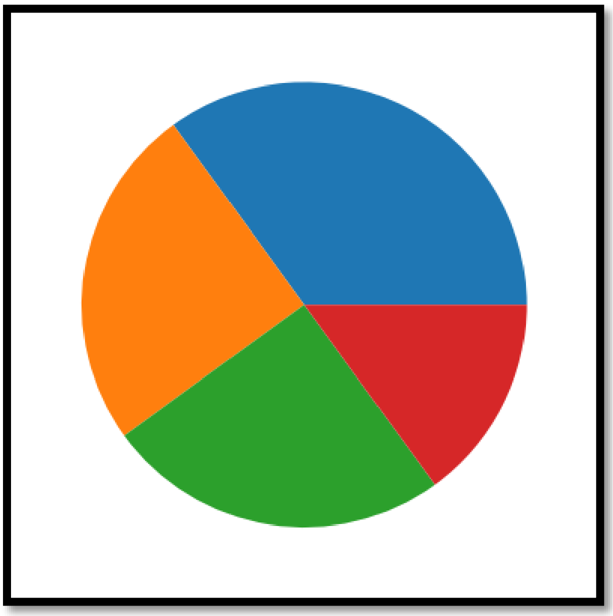
Code:

```
import numpy as np
x=np.random.normal(170,10,250) #250 values with sd of 10 and
concentrate on 170
print(x)
plt.hist(x)
plt.show()
```

Output:

```
[172.77007384 183.88565486 167.89516738 175.96070574 181.42416883
 174.94498816 152.38221018 162.68696803 172.99701697 167.88088208
 179.31673291 145.56937486 162.14054669 171.27526954 172.40080531
 158.08672151 185.15565383 168.51692316 174.93242886 173.33991834
 176.19514487 176.58130792 168.19212507 160.6071668  180.01540378
 167.25812747 177.81703923 174.56078742 161.66969512 181.81699059
 183.66306182 168.14349615 170.2387026  154.99092235 177.99103686
 171.89996673 163.33184974 185.80669482 164.6543335  188.86545366
 167.56273165 173.08123802 174.78418411 172.16633852 185.70563275
 184.82361526 172.32911443 151.72062936 178.7955621  170.06771168
 164.65993566 164.22091531 185.24065248 175.0767841  178.75716859
 171.04259641 183.37947323 175.64047065 172.99529507 163.51666831
 175.60845283 154.94552915 175.54798082 167.74504856 181.05733007
 170.50578746 179.60444614 159.55728031 164.84250315 183.39400517
 164.8707682  173.19800288 190.1351139  166.65196257 163.0781996
 163.17345035 183.84784531 150.73956882 173.56920595 164.47581533
 153.90690435 147.39089527 179.32645773 181.16000606 167.18136583
 169.28518161 181.92796571 151.4896221  161.3861592  184.22205085
 173.88156957 157.11241178 166.54240425 171.26676371 180.17704924
 165.17988974 191.20417926 188.16107245 173.13786888 168.97282036
 177.63287046 172.97928245 174.34019172 176.60624027 171.95889228
 177.89283184 178.1414327  175.99604175 174.05599683 163.88865246
 173.64883833 168.06124846 154.77885301 181.99724537 175.38978933
 164.96385408 176.40211552 156.26678298 160.56877509 168.8922997
 153.73630971 168.44098392 169.85345268 178.18589692 164.08227709
163.69523756 170.48316935 164.28324124 185.97869087 170.43746401
169.03667299 161.96935493 161.717051   169.21688244 164.05069942
164.87486853 160.76520413 173.61258003 169.34727551 155.47561662
165.49580922 172.52060073 139.71699331 174.37825579 181.01363151
165.30203748 155.97717599 165.68985528 160.01592595 159.97515425
174.41250489 169.53743333 184.37861109 173.75263529 158.65048456
174.19666359 158.23457265 139.77069703 157.15179436 188.10689198
182.48553394 170.68947205 165.89566883 179.54540688 153.869799
176.62670641 165.82951961 157.67923225 162.79323035 154.18939239
175.20635137 158.11149528 182.38362059 166.32629915 156.97181781
160.27923528 166.68155443 162.18270956 178.76195781 172.45162107
163.3773811  173.66367149 187.74249005 181.08431644 191.33140164
164.54202822 174.56061261 153.4205246  178.11123063 167.73838969
166.72652245 161.64024221 177.28706635 195.6602311  161.20304838
163.85486871 181.52204922 168.04674883 175.12302257 170.92453491
178.12170013 165.87884894 172.964871   169.74136423 170.43295555
166.62835171 152.48708009 179.56990683 165.42998274 178.03720616
156.58612318 157.31409511 158.11337432 163.90122515 180.58671449
169.75378438 157.70152038 160.33088428 163.51573213 169.59804725
190.78433054 178.58893518 159.26600903 174.69911745 172.03366094
190.3077134  180.87055586 185.36989368 171.96752379 169.08337524
162.80774803 166.87255713 168.81285003 179.16852606 184.29901325
175.50553334 184.82003699 179.33007139 181.58947475 161.15959339
177.55459407 166.99618341 157.29626538 185.67754251 175.61972321
151.53813803 155.91807799 180.06064187 175.13737981 158.03137204]
```



	Code:
	`y=np.array([35, 25, 25, 15])`
	`plt.pie(y)`
	`plt.show()`
	Output:
	

Conclusion	Data visualization tools like Matplotlib, Seaborn, Plotly, and ggplot2 in Python are essential for creating insightful graphs, plots, and visual representations of data, enabling effective communication and analysis.

Signature of Faculty

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Academic Year: 2023-2024

Name: Bhavvya Jain	**Sap Id:** 60018220108
Course: Data Mining and Analytics	**Roll No:** S019
Branch: Artificial Intelligence and Data Science	**Div:** S
Course Code: DJS22ADL403	**Batch:** A1

Experiment 03

Title	Data Preparation using NumPy and Pandas
Aim	1. Collect data from a specific source (eg. CSV file, API, database) and inspect its structure. 2. Generate summary statistics for a given dataset, including mean, median, standard deviation and quartiles for numerical columns.
Software	Google Colab

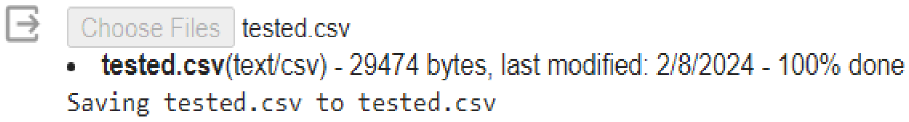
Implementation

Collect data from a specific source (eg. CSV file, API, database) and inspect its structure

Code:

```
# Mount Google Drive if your data is stored there
# If data is stored locally, you can skip this step
# from google.colab import drive
# drive.mount('/content/drive')
# Import necessary libraries
import numpy as np
import pandas as pd
from google.colab import files
# Specify the path to your CSV file
#csv_path = '/content/drive/MyDrive/PK Sir TA-exps/tested.csv'
#csv_path = '/content/drive/MyDrive/BSE/Tableu/Neel/tested.csv'
uploaded = files.upload()
file_name = list(uploaded.keys())[0]
```

Output:



tested.csv

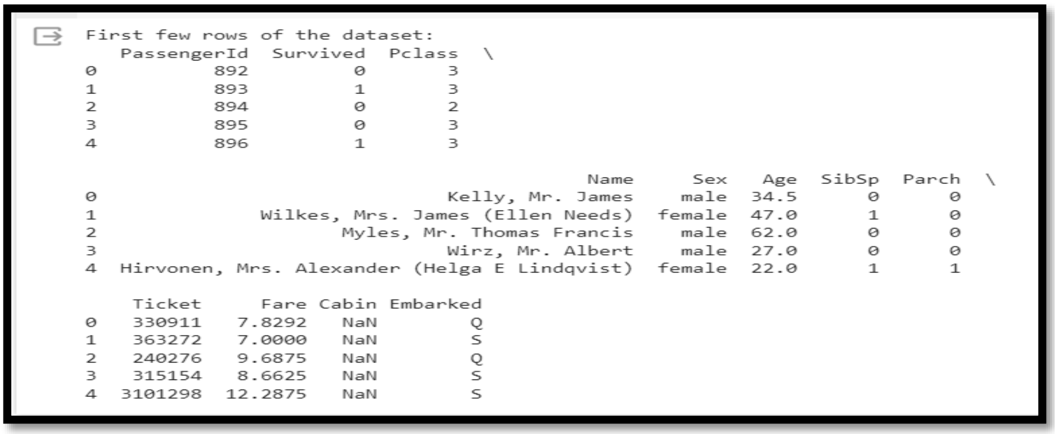
- **tested.csv**(text/csv) - 29474 bytes, last modified: 2/8/2024 - 100% done

```
Saving tested.csv to tested.csv
```

Code:

```
# Load the data into a Pandas DataFrame
#data = pd.read_csv(csv_path)
data = pd.read_csv(file_name)
# Display the first few rows of the dataset
print("First few rows of the dataset:")
print(data.head())
```

Output:

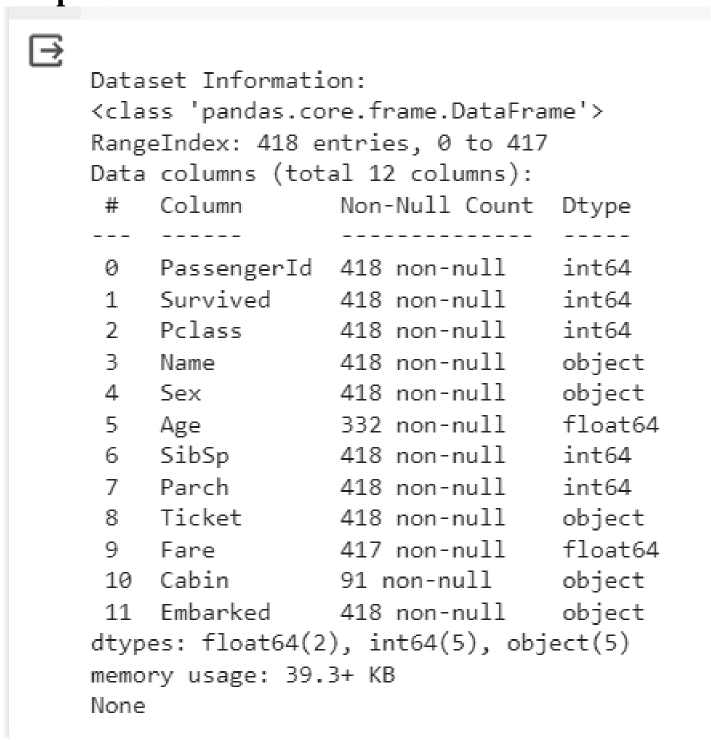


```
First few rows of the dataset:
   PassengerId  Survived  Pclass  \
0          892         0       3
1          893         1       3
2          894         0       2
3          895         0       3
4          896         1       3

                                           Name     Sex   Age  SibSp  Parch  \
0                              Kelly, Mr. James    male  34.5      0      0
1              Wilkes, Mrs. James (Ellen Needs)  female  47.0      1      0
2                     Myles, Mr. Thomas Francis    male  62.0      0      0
3                              Wirz, Mr. Albert    male  27.0      0      0
4  Hirvonen, Mrs. Alexander (Helga E Lindqvist)  female  22.0      1      1

    Ticket     Fare Cabin Embarked
0   330911   7.8292   NaN        Q
1   363272   7.0000   NaN        S
2   240276   9.6875   NaN        Q
3   315154   8.6625   NaN        S
4  3101298  12.2875   NaN        S
```

Code:

```
# Display basic information about the dataset
print("\nDataset Information:")
print(data.info())
```

Output:

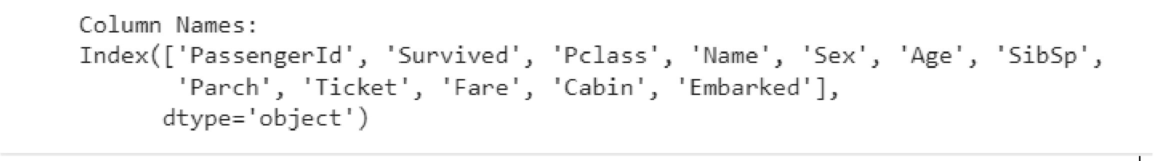


```
Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  418 non-null    int64
 1   Survived     418 non-null    int64
 2   Pclass       418 non-null    int64
 3   Name         418 non-null    object
 4   Sex          418 non-null    object
 5   Age          332 non-null    float64
 6   SibSp        418 non-null    int64
 7   Parch        418 non-null    int64
 8   Ticket       418 non-null    object
 9   Fare         417 non-null    float64
 10  Cabin        91 non-null     object
 11  Embarked     418 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 39.3+ KB
None
```

Code:

```
# Display the column names
print("\nColumn Names:")
print(data.columns)
```

Output:



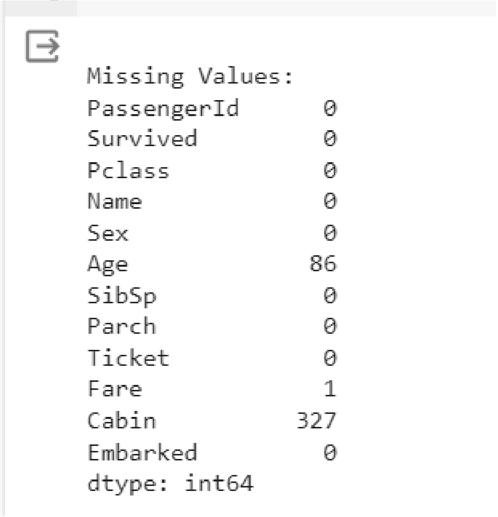
```
Column Names:
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

Code:

```
# Check for missing values in each column
```

```
print("\nMissing Values:")
print(data.isnull().sum())
```

Output:



```
Missing Values:
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

Code:

```
# Display unique values in categorical columns
categorical_columns =
data.select_dtypes(include=['object']).columns
for column in categorical_columns:
    print(f"\nUnique values in {column}:")
    print(data[column].unique())
```

Output:

```
Unique values in Name: ['Kelly, Mr. James' 'Wilkes, Mrs. James (Ellen Needs)'  'Myles, Mr. Thomas Francis' 'Wirz, Mr. Albert'
 'Hirvonen, Mrs. Alexander (Helga E Lindqvist)'  'Svensson, Mr. Johan Cervin' 'Connolly, Miss. Kate'  'Caldwell, Mr. Albert Francis'
 'Abrahim, Mrs. Joseph (Sophie Halaut Easu)' 'Davies, Mr. John Samuel' 'Ilieff, Mr. Ylio' 'Jones, Mr. Charles Cresson'  'Snyder, Mrs.
John Pillsbury (Nelle Stevenson)' 'Howard, Mr. Benjamin'  'Chaffee, Mrs. Herbert Fuller (Carrie Constance Toogood)'  'del Carlo,
Mrs. Sebastiano (Argenia Genovesi)' 'Keane, Mr. Daniel'  'Assaf, Mr. Gerios' 'Ilmakangas, Miss. Ida Livija'  'Assaf Khalil, Mrs.
Mariana (Miriam")"' 'Rothschild, Mr. Martin'  'Olsen, Master. Artur Karl' 'Flegenheim, Mrs. Alfred (Antoinette)'  'Williams, Mr.
Richard Norris II'  'Ryerson, Mrs. Arthur Larned (Emily Maria Borie)'  'Robins, Mr. Alexander A' 'Ostby, Miss. Helene Ragnhild'
 'Daher, Mr. Shedid' 'Brady, Mr. John Bertram' 'Samaan, Mr. Elias'  'Louch, Mr. Charles Alexander' 'Jefferys, Mr. Clifford Thomas'
 'Dean, Mrs. Bertram (Eva Georgetta Light)'  'Johnston, Mrs. Andrew G (Elizabeth Lily" Watson)"'  'Mock, Mr. Philipp Edmund'
 'Katavelas, Mr. Vassilios (Catavelas Vassilios")"' 'Roth, Miss. Sarah A'  'Cacic, Miss. Manda' 'Sap, Mr. Julius' 'Hee, Mr. Ling' 'Karun,
Mr. Franz'  'Franklin, Mr. Thomas Parham' 'Goldsmith, Mr. Nathan'  'Corbett, Mrs. Walter H (Irene Colvin)'  'Kimball, Mrs. Edwin
Nelson Jr (Gertrude Parsons)'  'Peltomaki, Mr. Nikolai Johannes' 'Chevre, Mr. Paul Romaine'  'Shaughnessy, Mr. Patrick'
 'Bucknell, Mrs. William Robert (Emma Eliza Ward)'  'Coutts, Mrs. William (Winnie Minnie" Treanor)"'  'Smith, Mr. Lucien Philip'
'Pulbaum, Mr. Franz'  'Hocking, Miss. Ellen Nellie""' 'Fortune, Miss. Ethel Flora'  'Mangiavacchi, Mr. Serafino Emilio' 'Rice, Master.
Albert' 'Cor, Mr. Bartol' 'Abelseth, Mr. Olaus Jorgensen'  'Davison, Mr. Thomas Henry' 'Chaudanson, Miss. Victorine'
 'Dika, Mr. Mirko' 'McCrae, Mr. Arthur Gordon'  'Bjorklund, Mr. Ernst Herbert' 'Bradley, Miss. Bridget Delia'
 'Ryerson, Master. John Borie'  'Corey, Mrs. Percy C (Mary Phyllis Elizabeth Miller)'  'Burns, Miss. Mary Delia' 'Moore, Mr. Clarence
Bloomfield'  'Tucker, Mr. Gilbert Milligan Jr' 'Fortune, Mrs. Mark (Mary McDougald)'  'Mulvihill, Miss. Bertha E' 'Minkoff, Mr.
Lazar'  'Nieminen, Miss. Manta Josefina' 'Ovies y Rodriguez, Mr. Servando'  'Geiger, Miss. Amalie' 'Keeping, Mr. Edwin' 'Miles,
Mr. Frank'  'Cornell, Mrs. Robert Clifford (Malvina Helen Lamson)'  'Aldworth, Mr. Charles Augustus' 'Doyle, Miss. Elizabeth'
 'Boulos, Master. Akar' 'Straus, Mr. Isidor' 'Case, Mr. Howard Brown' 'Demetri, Mr. Marinko' 'Lamb, Mr. John Joseph' 'Khalil, Mr.
Betros'  'Barry, Miss. Julia' 'Badman, Miss. Emily Louisa'  "O'Donoghue, Ms. Bridget" 'Wells, Master. Ralph Lester' 'Dyker, Mrs.
Adolf Fredrik (Anna Elisabeth Judith Andersson)'  'Pedersen, Mr. Olaf' 'Davidson, Mrs. Thornton (Orian Hays)' 'Guest, Mr. Robert'
'Birnbaum, Mr. Jakob' 'Tenglin, Mr. Gunnar Isidor'  'Cavendish, Mrs. Tyrell William (Julia Florence Siegel)' 'Makinen, Mr. Kalle
Edvard' 'Braf, Miss. Elin Ester Maria' 'Nancarrow, Mr. William Henry'  'Stengel, Mrs. Charles Emil Henry (Annie May Morris)'
 'Weisz, Mr. Leopold' 'Foley, Mr. William'  'Johansson Palmquist, Mr. Oskar Leander' 'Thomas, Mrs. Alexander (Thamine Thelma")"'
'Holthen, Mr. Johan Martin' 'Buckley, Mr. Daniel' 'Ryan, Mr. Edward'  'Willer, Mr. Aaron (Abi Weller")"' 'Swane, Mr. George'
 'Stanton, Mr. Samuel Ward' 'Shine, Miss. Ellen Natalia'  'Evans, Miss. Edith Corse' 'Buckley, Miss. Katherine'  'Straus, Mrs. Isidor
(Rosalie Ida Blun)' 'Chronopoulos, Mr. Demetrios'  'Thomas, Mr. John' 'Sandstrom, Miss. Beatrice Irene'  'Beattie, Mr. Thomson'
'Chapman, Mrs. John Henry (Sara Elizabeth Lawry)'  'Watt, Miss. Bertha J' 'Kiernan, Mr. John'  'Schabert, Mrs. Paul (Emma Mock)'
Carver, Mr. Alfred John'  'Kennedy, Mr. John' 'Cribb, Miss. Laura Alice' 'Brobeck, Mr. Karl Rudolf' 'McCoy, Miss. Alicia' 'Bowenur,
Mr. Solomon' 'Petersen, Mr. Marius'  'Spinner, Mr. Henry John' 'Gracie, Col. Archibald IV'  'Lefebre, Mrs. Frank (Frances)' 'Thomas,
Mr. Charles P'  'Dintcheff, Mr. Valtcho' 'Carlsson, Mr. Carl Robert'  'Zakarian, Mr. Mapriededer' 'Schmidt, Mr. August' 'Drapkin,
Miss. Jennie'  'Goodwin, Mr. Charles Frederick' 'Goodwin, Miss. Jessie Allis' 'Daniels, Miss. Sarah' 'Ryerson, Mr. Arthur Larned'
 'Beauchamp, Mr. Henry James'  'Lindeberg-Lind, Mr. Erik Gustaf (Mr Edward Lingrey")"' 'Vander Planke, Mr. Julius' 'Hilliard, Mr.
Herbert Henry' 'Davies, Mr. Evan' 'Crafton, Mr. John Bertram' 'Lahtinen, Rev. William' 'Earnshaw, Mrs. Boulton (Olive Potter)'
'Matinoff, Mr. Nicola' 'Storey, Mr. Thomas' 'Klasen, Mrs. (Hulda Kristina Eugenia Lofqvist)' 'Asplund, Master. Filip Oscar'
'Duquemin, Mr. Joseph' 'Bird, Miss. Ellen' 'Lundin, Miss. Olga Elida' 'Borebank, Mr. John James' 'Peacock, Mrs. Benjamin (Edith
Nile)' 'Smyth, Miss. Julia' 'Touma, Master. Georges Youssef' 'Wright, Miss. Marion' 'Pearce, Mr. Ernest' 'Peruschitz, Rev. Joseph
Maria' 'Kink-Heilmann, Mrs. Anton (Luise Heilmann)' 'Brandeis, Mr. Emil' 'Ford, Mr. Edward Watson' 'Cassebeer, Mrs. Henry
Arthur Jr (Eleanor Genevieve Fosdick)' 'Hellstrom, Miss. Hilda Maria' 'Lithman, Mr. Simon' 'Zakarian, Mr. Ortin' 'Dyker, Mr. Adolf
```

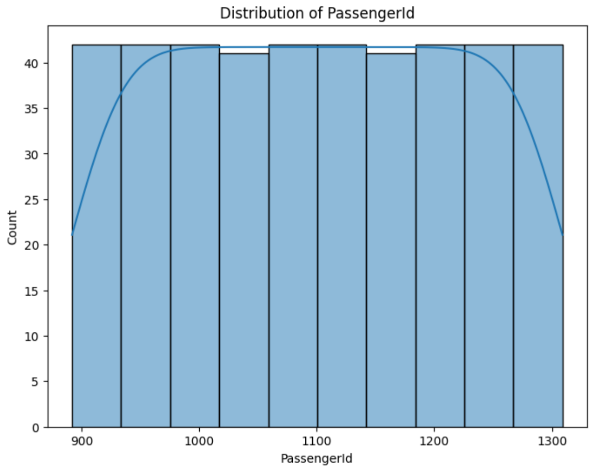
```
Fredrik' 'Torfa, Mr. Assad' 'Asplund, Mr. Carl Oscar Vilhelm Gustafsson' 'Brown, Miss. Edith Eileen' 'Sincock, Miss. Maude'
'Stengel, Mr. Charles Emil Henry' 'Becker, Mrs. Allen Oliver (Nellie E Baumgardner)' 'Compton, Mrs. Alexander Taylor (Mary
Eliza Ingersoll)' 'McCrie, Mr. James Matthew' 'Compton, Mr. Alexander Taylor Jr' 'Marvin, Mrs. Daniel Warner (Mary Graham
Carmichael Farquarson)' 'Lane, Mr. Patrick' 'Douglas, Mrs. Frederick Charles (Mary Helene Baxter)' 'Maybery, Mr. Frank Hubert'
'Phillips, Miss. Alice Frances Louisa' 'Davies, Mr. Joseph' 'Sage, Miss. Ada' 'Veal, Mr. James' 'Angle, Mr. William A' 'Salomon, Mr.
Abraham L' 'van Billiard, Master. Walter John' 'Lingane, Mr. John' 'Drew, Master. Marshall Brines' 'Karlsson, Mr. Julius Konrad
Eugen' 'Spedden, Master. Robert Douglas' 'Nilsson, Miss. Berta Olivia' 'Baimbrigge, Mr. Charles Robert' 'Rasmussen, Mrs. (Lena
Jacobsen Solvang)' 'Murphy, Miss. Nora' 'Danbom, Master. Gilbert Sigvard Emanuel' 'Astor, Col. John Jacob' 'Quick, Miss. Winifred
Vera' 'Andrew, Mr. Frank Thomas''Omont, Mr. Alfred Fernand' 'McGowan, Miss. Katherine' 'Collett, Mr. Sidney C Stuart'
'Rosenbaum, Miss. Edith Louise' 'Delalic, Mr. Redjo' 'Andersen, Mr. Albert Karvin' 'Finoli, Mr. Luigi' 'Deacon, Mr. Percy William'
 'Howard, Mrs. Benjamin (Ellen Truelove Arman)' 'Andersson, Miss. Ida Augusta Margareta' 'Head, Mr. Christopher' 'Mahon, Miss.
Bridget Delia' 'Wick, Mr. George Dennick' 'Widener, Mrs. George Dunton (Eleanor Elkins)' 'Thomson, Mr. Alexander Morrison'
'Duran y More, Miss. Florentina' 'Reynolds, Mr. Harold J' 'Cook, Mrs. (Selena Rogers)' 'Karlsson, Mr. Einar Gervasius'
 'Candee, Mrs. Edward (Helen Churchill Hungerford)' 'Moubarek, Mrs. George (Omine Amenia" Alexander)"' 'Asplund, Mr. Johan
Charles' 'McNeill, Miss. Bridget' 'Everett, Mr. Thomas James' 'Hocking, Mr. Samuel James Metcalfe' 'Sweet, Mr. George Frederick'
'Willard, Miss. Constance' 'Wiklund, Mr. Karl Johan' 'Linehan, Mr. Michael' 'Cumings, Mr. John Bradley' 'Vendel, Mr. Olof Edvin'
 'Warren, Mr. Frank Manley' 'Baccos, Mr. Raffull' 'Hiltunen, Miss. Marta' 'Douglas, Mrs. Walter Donald (Mahala Dutton)'
'Lindstrom, Mrs. Carl Johan (Sigrid Posse)' 'Christy, Mrs. (Alice Frances)' 'Spedden, Mr. Frederic Oakley' 'Hyman, Mr. Abraham'
'Johnston, Master. William Arthur Willie"""' 'Kenyon, Mr. Frederick R' 'Karnes, Mrs. J Frank (Claire Bennett)' 'Drew, Mr. James
Vivian' 'Hold, Mrs. Stephen (Annie Margaret Hill)' 'Khalil, Mrs. Betros (Zahie Maria" Elias)"' 'West, Miss. Barbara J' 'Abrahamsson,
Mr. Abraham August Johannes' 'Clark, Mr. Walter Miller' 'Salander, Mr. Karl Johan' 'Wenzel, Mr. Linhart' 'MacKay, Mr. George
William' 'Mahon, Mr. John' 'Niklasson, Mr. Samuel' 'Bentham, Miss. Lilian W' 'Midtsjo, Mr. Karl Albert' 'de Messemaeker, Mr.
Guillaume Joseph' 'Nilsson, Mr. August Ferdinand' 'Wells, Mrs. Arthur Henry (Addie" Dart Trevaskis)"' 'Klasen, Miss. Gertrud
Emilia' 'Portaluppi, Mr. Emilio Ilario Giuseppe' 'Lyntakoff, Mr. Stanko' 'Chisholm, Mr. Roderick Robert Crispin' 'Warren, Mr.
Charles William' 'Howard, Miss. May Elizabeth' 'Pokrnic, Mr. Mate' 'McCaffry, Mr. Thomas Francis' 'Fox, Mr. Patrick'
 'Clark, Mrs. Walter Miller (Virginia McDowell)' 'Lennon, Miss. Mary' 'Saade, Mr. Jean Nassr' 'Bryhl, Miss. Dagmar Jenny Ingeborg
' 'Parker, Mr. Clifford Richard' 'Faunthorpe, Mr. Harry' 'Ware, Mr. John James' 'Oxenham, Mr. Percy Thomas' 'Oreskovic, Miss.
Jelka' 'Peacock, Master. Alfred Edward' 'Fleming, Miss. Honora' 'Touma, Miss. Maria Youssef' 'Rosblom, Miss. Salli Helena'
'Dennis, Mr. William' 'Franklin, Mr. Charles (Charles Fardon)' 'Snyder, Mr. John Pillsbury' 'Mardirosian, Mr. Sarkis' 'Ford, Mr.
Arthur' 'Rheims, Mr. George Alexander Lucien' 'Daly, Miss. Margaret Marcella Maggie"""' 'Nasr, Mr. Mustafa' 'Dodge, Dr.
Washington' 'Wittevrongel, Mr. Camille' 'Angheloff, Mr. Minko' 'Laroche, Miss. Louise' 'Samaan, Mr. Hanna' 'Loring, Mr. Joseph
Holland' 'Johansson, Mr. Nils' 'Olsson, Mr. Oscar Wilhelm' 'Malachard, Mr. Noel' 'Phillips, Mr. Escott Robert' 'Pokrnic, Mr. Tome'
 'McCarthy, Miss. Catherine Katie"""' 'Crosby, Mrs. Edward Gifford (Catherine Elizabeth Halstead)' 'Allison, Mr. Hudson Joshua
Creighton' 'Aks, Master. Philip Frank' 'Hays, Mr. Charles Melville' 'Hansen, Mrs. Claus Peter (Jennie L Howard)' 'Cacic, Mr. Jego
Grga' 'Vartanian, Mr. David' 'Sadowitz, Mr. Harry' 'Carr, Miss. Jeannie' 'White, Mrs. John Stuart (Ella Holmes)' 'Hagardon, Miss.
Kate' 'Spencer, Mr. William Augustus' 'Rogers, Mr. Reginald Harry' 'Jonsson, Mr. Nils Hilding' 'Jefferys, Mr. Ernest Wilfred'
'Andersson, Mr. Johan Samuel' 'Krekorian, Mr. Neshan' 'Nesson, Mr. Israel' 'Rowe, Mr. Alfred G' 'Kreuchen, Miss. Emilie' 'Assam,
Mr. Ali' 'Becker, Miss. Ruth Elizabeth' 'Rosenshine, Mr. George (Mr George Thorne")"' 'Clarke, Mr. Charles Valentine' 'Enander,
Mr. Ingvar' 'Davies, Mrs. John Morgan (Elizabeth Agnes Mary White) ' 'Dulles, Mr. William Crothers' 'Thomas, Mr. Tannous'
 'Nakid, Mrs. Said (Waika Mary" Mowad)"' 'Cor, Mr. Ivan' 'Maguire, Mr. John Edward' 'de Brito, Mr. Jose Joaquim' 'Elias, Mr.
Joseph' 'Denbury, Mr. Herbert' 'Betros, Master. Seman' 'Fillbrook, Mr. Joseph Charles' 'Lundstrom, Mr. Thure Edvin' 'Sage, Mr. John
George' 'Cardeza, Mrs. James Warburton Martinez (Charlotte Wardle Drake)' 'van Billiard, Master. James William' 'Abelseth, Miss.
Karen Marie' 'Botsford, Mr. William Hull' 'Whabee, Mrs. George Joseph (Shawneene Abi-Saab)' 'Giles, Mr. Ralph' 'Walcroft, Miss.
Nellie' 'Greenfield, Mrs. Leo David (Blanche Strouse)' 'Stokes, Mr. Philip Joseph' 'Dibden, Mr. William' 'Herman, Mr. Samuel' 'Dean,
Miss. Elizabeth Gladys Millvina"""' 'Julian, Mr. Henry Forbes' 'Brown, Mrs. John Murray (Caroline Lane Lamson)' 'Lockyer, Mr.
Edward' "O'Keefe, Mr. Patrick" 'Lindell, Mrs. Edvard Bengtsson (Elin Gerda Persson)' 'Sage, Master. William Henry' 'Mallet, Mrs.
Albert (Antoinette Magnin)' 'Ware, Mrs. John James (Florence Louise Long)' 'Strilic, Mr. Ivan' 'Harder, Mrs. George Achilles
(Dorothy Annan)' 'Sage, Mrs. John (Annie Bullen)' 'Caram, Mr. Joseph' 'Riihivouri, Miss. Susanna Juhantytar Sanni"""'
 'Gibson, Mrs. Leonard (Pauline C Boeson)' 'Pallas y Castello, Mr. Emilio' 'Giles, Mr. Edgar' 'Wilson, Miss. Helen Alice' 'Ismay, Mr.
Joseph Bruce' 'Harbeck, Mr. William H' 'Dodge, Mrs. Washington (Ruth Vidaver)' 'Bowen, Miss. Grace Scott' 'Kink, Miss. Maria'
 'Cotterill, Mr. Henry Harry"""' 'Hipkins, Mr. William Edward' 'Asplund, Master. Carl Edgar' "O'Connor, Mr. Patrick" 'Foley, Mr.
Joseph' 'Risien, Mrs. Samuel (Emma)' "McNamee, Mrs. Neal (Eileen O'Leary)" 'Wheeler, Mr. Edwin Frederick"""' 'Herman, Miss.
Kate' 'Aronsson, Mr. Ernst Axel Algot' 'Ashby, Mr. John' 'Canavan, Mr. Patrick' 'Palsson, Master. Paul Folke' 'Payne, Mr. Vivian
Ponsonby' 'Lines, Mrs. Ernest H (Elizabeth Lindsey James)' 'Abbott, Master. Eugene Joseph' 'Gilbert, Mr. William'
 'Kink-Heilmann, Mr. Anton' 'Smith, Mrs. Lucien Philip (Mary Eloise Hughes)' 'Colbert, Mr. Patrick' 'Frolicher-Stehli, Mrs.
Maxmillian (Margaretha Emerentia Stehli)' 'Larsson-Rondberg, Mr. Edvard A' 'Conlon, Mr. Thomas Henry' 'Bonnell, Miss. Caroline'
'Gale, Mr. Harry' 'Gibson, Miss. Dorothy Winifred' 'Carrau, Mr. Jose Pedro' 'Frauenthal, Mr. Isaac Gerald' 'Nourney, Mr. Alfred
(Baron von Drachstedt")"' 'Ware, Mr. William Jeffery' 'Widener, Mr. George Dunton' 'Riordan, Miss. Johanna Hannah"""' 'Peacock,
Miss. Treasteall' 'Naughton, Miss. Hannah' 'Minahan, Mrs. William Edward (Lillian E Thorpe)' 'Henriksson, Miss. Jenny Lovisa'
'Spector, Mr. Woolf' 'Oliva y Ocana, Dona. Fermina' 'Saether, Mr. Simon Sivertsen' 'Ware, Mr. Frederick' 'Peter, Master. Michael J']
Unique values in Sex:
['male' 'female']

Unique values in Ticket:
['330911' '363272' '240276' '315154' '3101298' '7538' '330972' '248738' '2657' 'A/4 48871' '349220' '694' '21228' '24065' 'W.E.P.
5734' 'SC/PARIS 2167' '233734' '2692' 'STON/O2. 3101270' '2696' 'PC 17603' 'C 17368' 'PC 17598' 'PC 17597' 'PC 17608' 'A/5.
3337' '113509' '2698' '113054' '2662' 'SC/AH 3085' 'C.A. 31029' 'C.A. 2315' 'W./C. 6607' '13236' '2682' '342712' '315087' '345768'
'1601' '349256' '113778' 'SOTON/O.Q. 3101263' '237249' '11753' 'STON/O 2. 3101291' 'PC 17594' '370374' '11813' 'C.A. 37671'
'13695' 'SC/PARIS 2168' '29105' '19950' 'SC/A.3 2861' '382652' '349230' '348122' '386525' '349232' '237216' '347090' '334914'
'F.C.C. 13534' '330963' '113796' '2543' '382653' '349211' '3101297' 'PC 17562' '113503' '359306' '11770' '248744' '368702''2678' 'PC
17483' '19924' '349238' '240261' '2660' '330844' 'A/4 31416' '364856' '29103' '347072' '345498' 'F.C. 12750' '376563' '13905' '350033'
 '19877' 'STON/O 2. 3101268' '347471' 'A./5. 3338' '11778' '228414' '365235' '347070' '2625' 'C 4001' '330920' '383162' '3410'
'248734' '237734' '330968' 'PC 17531' '329944' '2680' '2681' 'PP 9549' '13050' 'SC/AH 29037' 'C.A. 33595' '367227' '392095' '368783'
'371362' '350045' '367226' '211535' '342441' 'STON/OQ. 369943' '113780' '4133' '2621' '349226' '350409' '2656' '248659'
'SOTON/OQ 392083' 'CA 2144' '113781' '244358' '17475' '345763' '17463' 'SC/A4 23568' '113791' '250651' '11767' '349255' '3701'
'350405' '347077' 'S.O./P.P. 752' '347469' '110489' 'SOTON/O.Q. 3101315' '335432' '2650' '220844' '343271' '237393' '315153' 'PC
17591' 'W./C. 6608' '17770' '7548' 'S.O./P.P. 251' '2670' '2673' '29750' 'C.A. 33112' '230136' 'PC 17756' '233478' '113773' '7935'
 'PC 17558' '239059' 'S.O./P.P. 2' 'A/4 48873' 'CA. 2343' '28221' '226875' '111163' 'A/5. 851' '235509' '28220' '347465' '16966'
'347066' 'C.A. 31030' '65305' '36568' '347080' 'PC 17757' '26360' 'C.A. 34050' 'F.C. 12998' '9232' '28034' 'PC 17613' '349250'
'SOTON/O.Q. 3101308' 'S.O.C. 14879' '347091' '113038' '330924' '36928' '32302' 'SC/PARIS 2148' '342684' 'W./C. 14266' '350053'
'PC 17606' '2661' '350054' '370368' 'C.A. 6212' '242963' '220845' '113795' '3101266' '330971' 'PC 17599' '350416' '110813' '2679'
'250650' 'PC 17761' '112377' '237789' '3470' '17464' '26707' 'C.A. 34651' 'SOTON/O2 3101284' '13508' '7266' '345775' 'C.A. 42795'
```

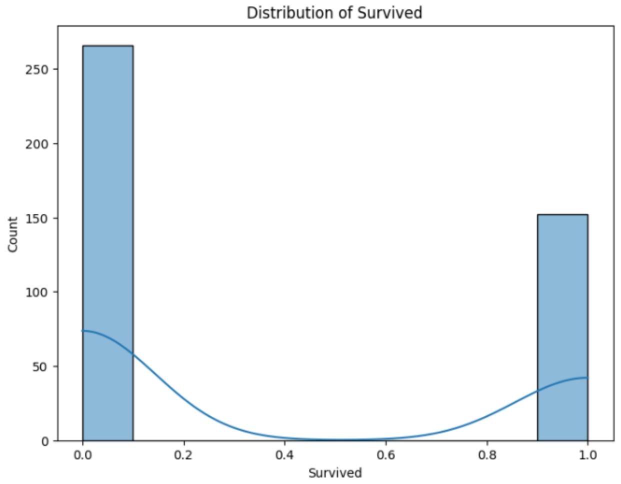
```
'AQ/4 3130' '363611' '28404' '345501' '345572' '350410' 'C.A. 34644' '349235' '112051' 'C.A. 49867' 'A. 2. 39186' '315095' '368573'
'370371' '2676' '236853' 'SC 14888' '2926' 'CA 31352' 'W./C. 14260' '315085' '364859' '370129' 'A/5 21175' 'SOTON/O.Q. 3101314'
 '2655' 'A/5 1478' 'PC 17607' '382650' '2652' '33638' '345771' '349202' 'SC/Paris 2123' '113801' '347467' '347079' '237735' '315092'
'383123' '112901' '392091' '12749' '350026' '315091' '2658' 'LP 1588' '368364' 'PC 17760' 'AQ/3. 30631' 'PC 17569' '28004' '350408'
'347075' '2654' '244368' '113790' '24160' 'SOTON/O.Q. 3101309' 'PC 17585' '2003' '236854' 'PC 17580' '2684' '2653' '349229'
'110469' '244360' '2675' '2622' 'C.A. 15185' '350403' 'PC 17755' '348125' '237670' '2688' '248726' 'F.C.C. 13528' 'PC 17759' 'F.C.C.
13540' '113044' '11769' '1222' '368402' '349910' 'S.C./PARIS 2079' '315083' '11765' '2689' '3101295' '112378' 'SC/PARIS 2147'
'28133' '112058' '248746' '315152' '29107' '680' '366713' '330910' '364498' '376566' 'SC/PARIS 2159' '349911' '244346' '364858'
'349909' 'PC 17592' 'C.A. 2673' 'C.A. 30769' '371109' '13567' '347065' '21332' '28664' '113059' '17765' 'SC/PARIS 2166' '28666'
'334915' '365237' '19928' '347086' 'A.5. 3236' 'PC 17758' 'SOTON/O.Q. 3101262' '359309' '2668']
Unique values in Cabin:
[nan 'B45' 'E31' 'B57 B59 B63 B66' 'B36' 'A21' 'C78' 'D34' 'D19' 'A9' 'D15' 'C31' 'C23 C25 C27' 'F G63' 'B61' 'C53' 'D43' 'C130'
'C132' 'C101' 'C55 C57' 'B71' 'C46' 'C116' 'F' 'A29' 'G6' 'C6' 'C28' 'C51' 'E46' 'C54' 'C97' 'D22' 'B10' 'F4' 'E45' 'E52' 'D30' 'B58 B60'
'E34' 'C62 C64' 'A11' 'B11' 'C80' 'F33' 'C85' 'D37' 'C86' 'D21' 'C89' 'F E46' 'A34' 'D' 'B26' 'C22 C26' 'B69' 'C32' 'B78' 'F E57' 'F2'
'A18' 'C106' 'B51 B53 B55' 'D10 D12' 'E60' 'E50' 'E39 E41' 'B52 B54 B56' 'C39' 'B24' 'D28' 'B41' 'C7' 'D40' 'D38' 'C105']
Unique values in Embarked:
['Q' 'S' 'C']
```

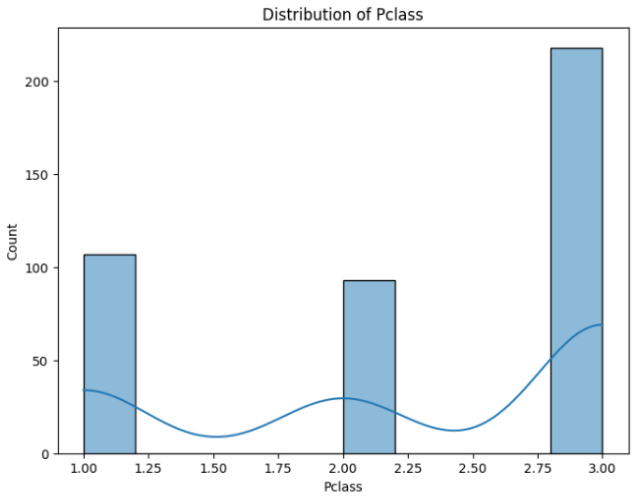
Code:

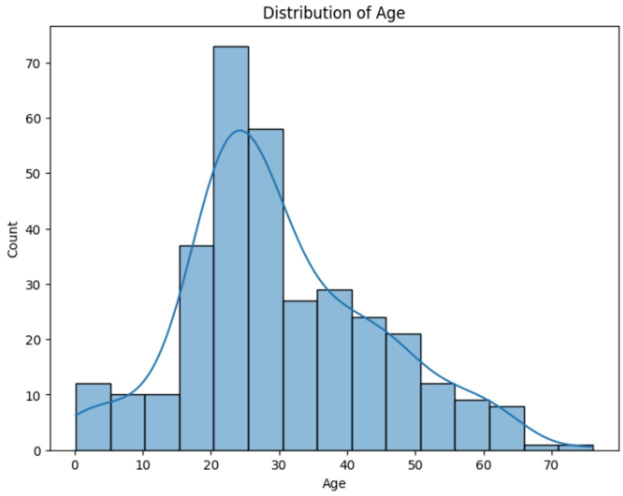
```python
# Visualize the distribution of numerical columns
import matplotlib.pyplot as plt
import seaborn as sns
numerical_columns = data.select_dtypes(include=['float64',
'int64']).columns
for column in numerical_columns:
    plt.figure(figsize=(8, 6))
    sns.histplot(data[column], kde=True)
    plt.title(f'Distribution of {column}')
    plt.show()
```

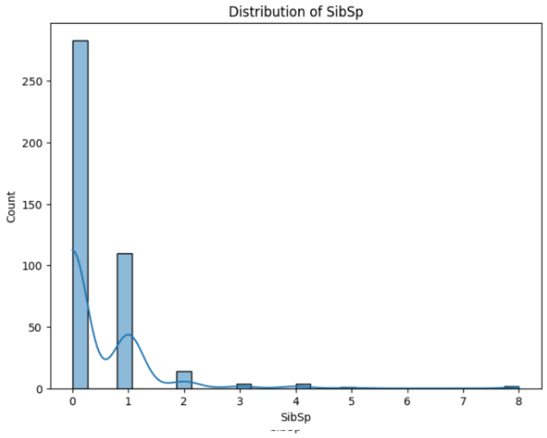
Output:

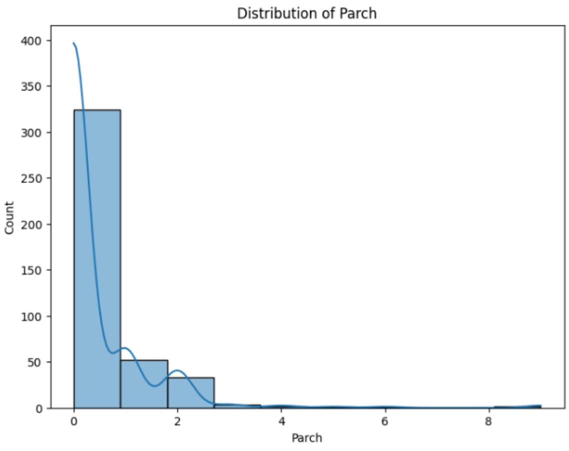


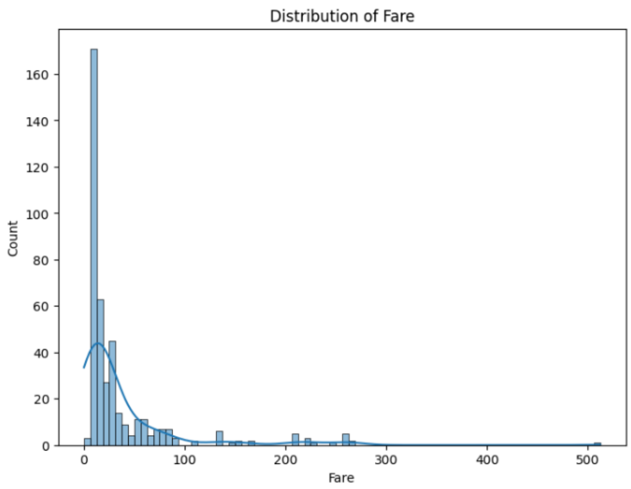



Distribution of Pclass
Count
0
50
100
150
200
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
Pclass


Distribution of Age
Count
0
10
20
30
40
50
60
70
0
10
20
30
40
50
60
70
Age


Distribution of SibSp
Count
0
50
100
150
200
250
0
1
2
3
4
5
6
7
8
SibSp


Distribution of Parch
Count
0
50
100
150
200
250
300
350
400
0
2
4
6
8
Parch

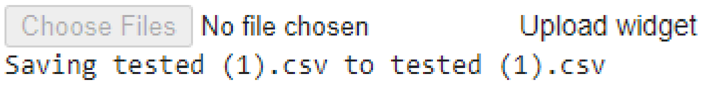


Generate summary statistics for a given dataset, including mean, median, standard deviation and quartiles for numerical columns.

Code:

```
# Import necessary libraries
import numpy as np
import pandas as pd
from google.colab import files
# Mount Google Drive if your data is stored there
# If data is stored locally, you can skip this step
# from google.colab import drive tested.csv
# drive.mount('/content/drive')
uploaded = files.upload()
file_name = list(uploaded.keys())[0]
```

Output:



```
Saving tested (1).csv to tested (1).csv
```

Code:

```
data = pd.read_csv(file_name)
# Display summary statistics for numerical columns
summary_statistics = data.describe()
# Display mean, median, standard deviation, and quartiles for each
numerical column
for column in data.select_dtypes(include=['float64',
'int64']).columns:
    print(f"\nSummary Statistics for {column}:")
    print(f"Mean: {summary_statistics[column]['mean']}")
    print(f"Median: {data[column].median()}")
    print(f"Standard Deviation:
{summary_statistics[column]['std']}")
    print(f"25th Percentile (Q1): {data[column].quantile(0.25)}")
    print(f"50th Percentile (Q2): {data[column].quantile(0.50)}")
    print(f"75th Percentile (Q3): {data[column].quantile(0.75)}")
```

Output:

```
Summary Statistics for PassengerId:
Mean: 1100.5
Median: 1100.5
Standard Deviation: 120.81045760473994
25th Percentile (Q1): 996.25
50th Percentile (Q2): 1100.5
75th Percentile (Q3): 1204.75

Summary Statistics for Survived:
Mean: 0.36363636363636365
Median: 0.0
Standard Deviation: 0.4816221409322309
25th Percentile (Q1): 0.0
50th Percentile (Q2): 0.0
75th Percentile (Q3): 1.0

Summary Statistics for Pclass:
Mean: 2.2655502392344498
Median: 3.0
Standard Deviation: 0.8418375519640503
25th Percentile (Q1): 1.0
50th Percentile (Q2): 3.0
75th Percentile (Q3): 3.0

Summary Statistics for Age:
Mean: 30.272590361445783
Median: 27.0
Standard Deviation: 14.181209235624422
25th Percentile (Q1): 21.0
50th Percentile (Q2): 27.0
75th Percentile (Q3): 39.0

Summary Statistics for SibSp:
Mean: 0.4473684210526316
Median: 0.0
Standard Deviation: 0.8967595611217135
25th Percentile (Q1): 0.0
50th Percentile (Q2): 0.0
75th Percentile (Q3): 1.0

Summary Statistics for Parch:
Mean: 0.3923444976076555
Median: 0.0
Standard Deviation: 0.9814288785371691
25th Percentile (Q1): 0.0
50th Percentile (Q2): 0.0
75th Percentile (Q3): 0.0

Summary Statistics for Fare:
Mean: 35.627188489208635
Median: 14.4542
Standard Deviation: 55.907576179973844
25th Percentile (Q1): 7.8958
50th Percentile (Q2): 14.4542
75th Percentile (Q3): 31.5
```

Code:

```
# Display summary statistics of the dataset
print("\nSummary Statistics:")
print(data.describe())
```

Output:

```
Summary Statistics:
       PassengerId    Survived      Pclass         Age       SibSp
count   418.000000  418.000000  418.000000  332.000000  418.000000
mean   1100.500000    0.363636    2.265550   30.272590    0.447368
std     120.810458    0.481622    0.841838   14.181209    0.896760
min     892.000000    0.000000    1.000000    0.170000    0.000000
25%     996.250000    0.000000    1.000000   21.000000    0.000000
50%    1100.500000    0.000000    3.000000   27.000000    0.000000
75%    1204.750000    1.000000    3.000000   39.000000    1.000000
max    1309.000000    1.000000    3.000000   76.000000    8.000000

            Parch        Fare
count  418.000000  417.000000
mean     0.392344   35.627188
std      0.981429   55.907576
min      0.000000    0.000000
25%      0.000000    7.895800
50%      0.000000   14.454200
75%      0.000000   31.500000
max      9.000000  512.329200
```

Conclusion	Summary statistics, such as mean, median, standard deviation, and quartiles, provide valuable insights into the distribution and characteristics of numerical data, aiding in comprehensive data analysis and decision-making processes.

Signature of Faculty

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Academic Year: 2023-2024

Name: Bhavvya Jain	**Sap Id:** 60018220108
Course: Data Mining and Analytics	**Roll No:** S019
Branch: Artificial Intelligence and Data Science	**Div:** S
Course Code: DJS22ADL403	**Batch:** A1

Experiment 04

Title	Data Pre-processing With Pandas
Aim	1. Identify the presence of missing values in a dataset and choose an appropriate method for handling them (eg. removal, imputation) 2. Remove duplicate records from a dataset and assess the impact on data quality.
Software	Google Colab

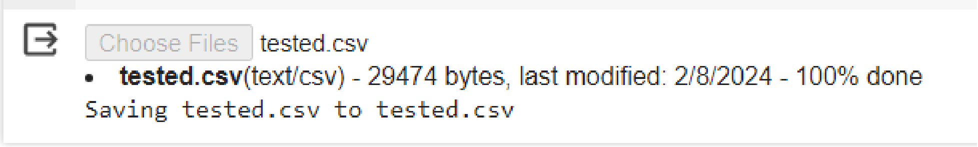
Implementation

Identify the presence of missing values in a dataset and choose an appropriate method for handling them (eg. removal, imputation)

Code:

```
# Import necessary libraries
import pandas as pd
from google.colab import files
uploaded = files.upload()
file_name = list(uploaded.keys())[0]
```

Output:

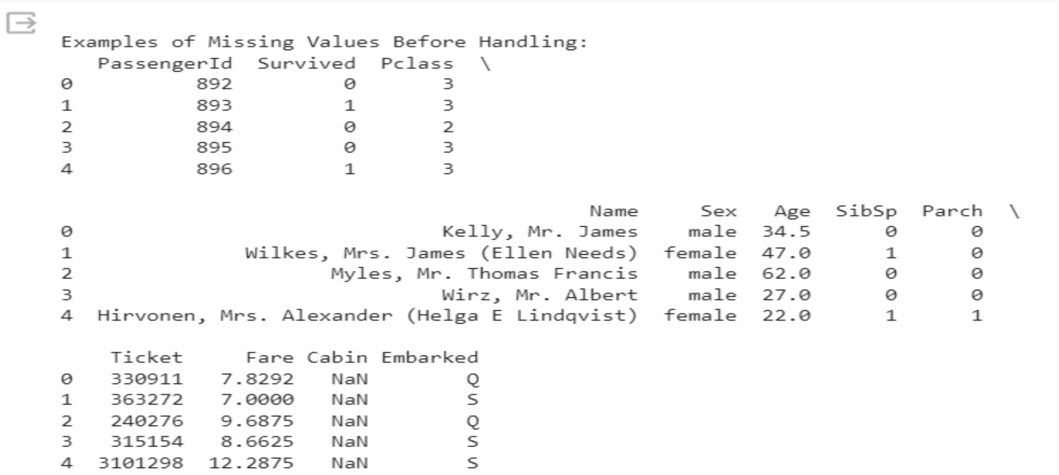


```
tested.csv
• tested.csv(text/csv) - 29474 bytes, last modified: 2/8/2024 - 100% done
Saving tested.csv to tested.csv
```

Code:

```
data = pd.read_csv(file_name)
```

Output:



```
Examples of Missing Values Before Handling:
   PassengerId  Survived  Pclass  \
0          892         0       3
1          893         1       3
2          894         0       2
3          895         0       3
4          896         1       3

                                           Name     Sex   Age  SibSp  Parch  \
0                              Kelly, Mr. James    male  34.5      0      0
1              Wilkes, Mrs. James (Ellen Needs)  female  47.0      1      0
2                     Myles, Mr. Thomas Francis    male  62.0      0      0
3                              Wirz, Mr. Albert    male  27.0      0      0
4  Hirvonen, Mrs. Alexander (Helga E Lindqvist)  female  22.0      1      1

    Ticket     Fare Cabin Embarked
0   330911   7.8292   NaN        Q
1   363272   7.0000   NaN        S
2   240276   9.6875   NaN        Q
3   315154   8.6625   NaN        S
4  3101298  12.2875   NaN        S
```

Code:

```
# Print a few examples of missing values before handling
print("\nExamples of Missing Values Before Handling:")
print(data[data.isnull().any(axis=1)].head())
```

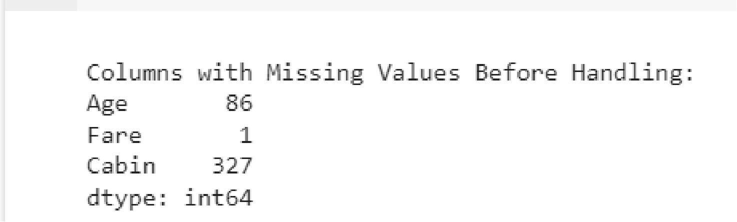
Output:

```
Examples of Missing Values Before Handling:
   PassengerId  Survived  Pclass  \
0          892         0       3
1          893         1       3
2          894         0       2
3          895         0       3
4          896         1       3

                                           Name     Sex   Age
SibSp  Parch  \
0                              Kelly, Mr. James    male  34.5
0      0
1              Wilkes, Mrs. James (Ellen Needs)  female  47.0
1      0
2                     Myles, Mr. Thomas Francis    male  62.0
0      0
3                              Wirz, Mr. Albert    male  27.0
0      0
4  Hirvonen, Mrs. Alexander (Helga E Lindqvist)  female  22.0
1      1

    Ticket      Fare Cabin Embarked
0   330911   7.8292   NaN        Q
1   363272   7.0000   NaN        S
2   240276   9.6875   NaN        Q
3   315154   8.6625   NaN        S
4  3101298  12.2875   NaN        S
```

Code:

```
# Check for missing values in each column
missing_values_before = data.isnull().sum()

# Display columns with missing values and their count before
handling
print("\nColumns with Missing Values Before Handling:")
print(missing_values_before[missing_values_before > 0])
```

Output:

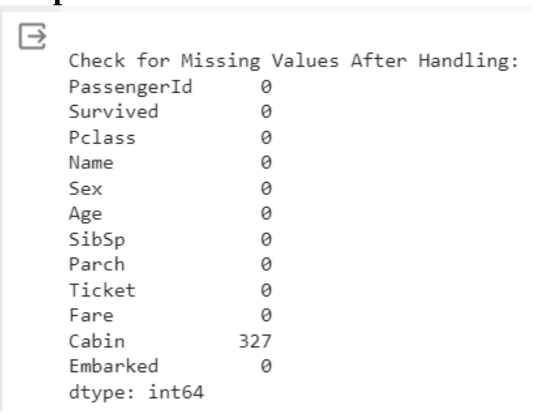


```
Columns with Missing Values Before Handling:
Age       86
Fare       1
Cabin    327
dtype: int64
```

Code:

```
# Choose an appropriate method for handling missing values
# Method 1: Removing rows with missing values
# data.dropna(inplace=True)
# Method 2: Imputing missing values with the mean for numerical columns
numerical_columns = data.select_dtypes(include=['float64', 'int64']).columns
for column in numerical_columns:
    data[column].fillna(data[column].mean(), inplace=True)
# Method 3: Imputing missing values with the median for numerical columns
# for column in numerical_columns:
#     data[column].fillna(data[column].median(), inplace=True)
# Method 4: Forward-fill missing values in a DataFrame
# data.ffill(inplace=True)
# After handling missing values, you can recheck for missing values
print("\nCheck for Missing Values After Handling:")
print(data.isnull().sum())
```

Output:



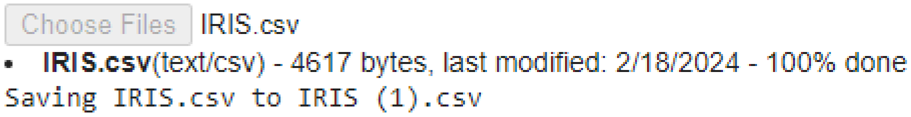
```
Check for Missing Values After Handling:
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          327
Embarked         0
dtype: int64
```

Remove duplicate records from a dataset and assess the impact on data quality

Code:

```
# Import necessary libraries
import pandas as pd
from google.colab import files
# Specify the path to your CSV file
#csv_path = '/content/drive/MyDrive/PK Sir TA-exps/IRIS.csv'
uploaded = files.upload()
file_name = list(uploaded.keys())[0]
```

Output:



IRIS.csv

- **IRIS.csv**(text/csv) - 4617 bytes, last modified: 2/18/2024 - 100% done

```
Saving IRIS.csv to IRIS (1).csv
```

Code:

```
# Load the data into a Pandas DataFrame
#data = pd.read_csv(csv_path)
data = pd.read_csv(file_name)
```

```
# Display the number of duplicate records before removal
print("Number of Duplicate Records Before Removal:",
data.duplicated().sum())
```

Output:

```
Number of Duplicate Records Before Removal: 3
```

Code:

```
# Remove duplicate records
data.drop_duplicates(inplace=True)
# Display the number of duplicate records after removal
print("Number of Duplicate Records After Removal:",
data.duplicated().sum())
```

Output:

```
Number of Duplicate Records After Removal: 0
```

Code:

```
# Assess impact on data quality
print("\nImpact on Data Quality:")
print("Original Number of Records:", len(data) +
data.duplicated().sum())
print("Number of Unique Records After Removal:", len(data))
```

Output:

```
Impact on Data Quality:
Original Number of Records: 147
Number of Unique Records After Removal: 147
```

Conclusion

Addressing missing values and removing duplicates not only enhances the reliability and integrity of the dataset but also ensures that analytical results are more robust and trustworthy, leading to more informed and effective decision-making processes.

Signature of Faculty

Academic Year: 2023-2024

Name: Bhavvya Jain	**Sap Id:** 60018220108
Course: Data Mining and Analytics	**Roll No:** S019
Branch: Artificial Intelligence and Data Science	**Div:** S
Course Code: DJS22ADL403	**Batch:** A1

Experiment 05

Title	Handling Categorical Data: One Hot Encoding
Aim	Implement one-hot encoding using Pandas to convert categorical variables into a format suitable for modelling.
Software	Google Colab

Implementation

Code:

```
import numpy as np
import pandas as pd
data = pd.read_csv('/content/temporary.csv')
print(data.head())
```

Output:

```
   Unnamed: 0  Employee ID  Gender Remarks
0           1          125    Male    Nice
1           2          164  Female    Good
2           3          321    Male   Great
3           4          415  Female   Great
```

Code:

```
data['Gender'].value_counts()
data['Remarks'].value_counts()
```

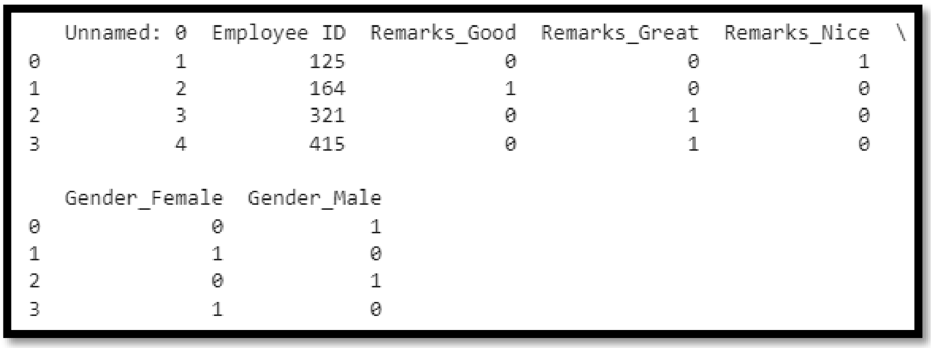
Output:

```
Great    2
Nice     1
Good     1
Name: Remarks, dtype: int64
```

Code:

```
one_hot_encoded_data = pd.get_dummies(data, columns = ['Remarks',
'Gender'])
print(one_hot_encoded_data)
```

Output:



```
   Unnamed: 0  Employee ID  Remarks_Good  Remarks_Great  Remarks_Nice  \
0           1          125             0              0             1
1           2          164             1              0             0
2           3          321             0              1             0
3           4          415             0              1             0

   Gender_Female  Gender_Male
0              0            1
1              1            0
2              0            1
3              1            0
```

Code:

```
df = pd.read_csv('/content/Apple.csv')
print(df.head())
```

Output:

```
Unnamed: 0  Categorical   Value
0       Apple               1      5
1      Orange               2     10
2      Grapes               3     15
```

Code:

```
df['Categorical '].value_counts()
df['Value'].value_counts()
```

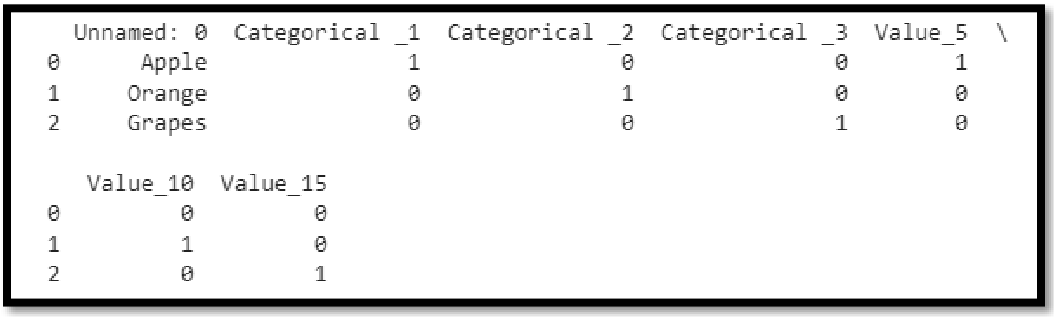
Output:

```
5   1
10   1
15   1
Name: Value, dtype: int64
```

Code:

```
one_hot_encoded_data = pd.get_dummies(df, columns = ['Categorical
', 'Value'])
print(one_hot_encoded_data)
```

Output:



```
  Unnamed: 0  Categorical _1  Categorical _2  Categorical _3  Value_5  \
0      Apple               1               0               0        1
1     Orange               0               1               0        0
2     Grapes               0               0               1        0

   Value_10  Value_15
0         0         0
1         1         0
2         0         1
```

Conclusion

Handling categorical data through one-hot encoding is crucial for preparing data for machine learning models, ensuring accurate interpretation and utilization of categorical variables in the algorithms.

Signature of Faculty

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Academic Year: 2023-2024

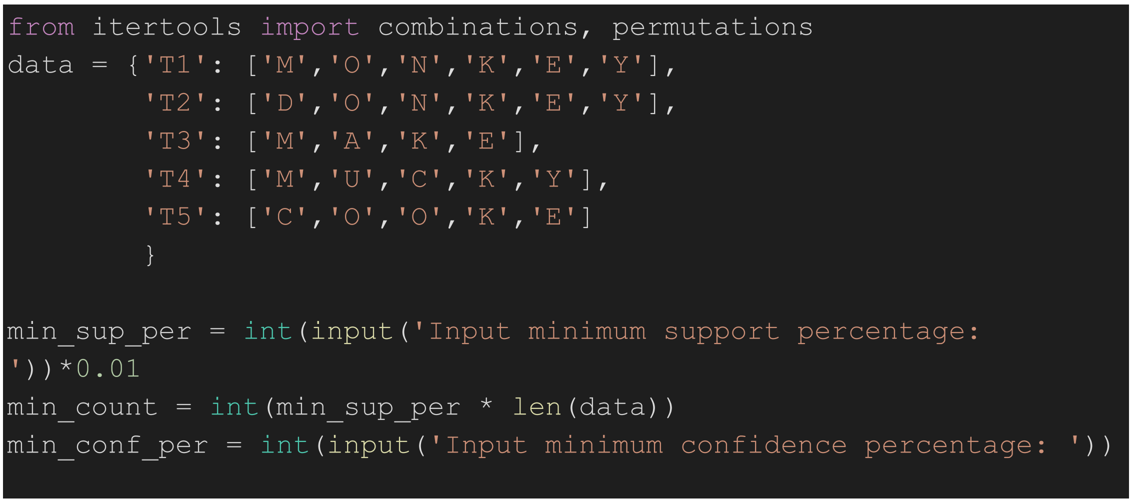
Name: Bhavvya Jain	**Sap Id:** 60018220108
Course: Data Mining and Analytics	**Roll No:** S019
Branch: Artificial Intelligence and Data Science	**Div:** S
Course Code: DJS22ADL403	**Batch:** A1

Experiment 06

Title	Implement association rule mining
Aim	Implement association rule mining using Apriori Algorithm
Software	Google Colab

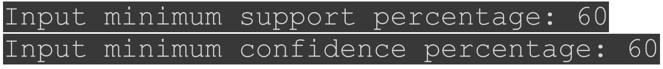
Implementation

Code:



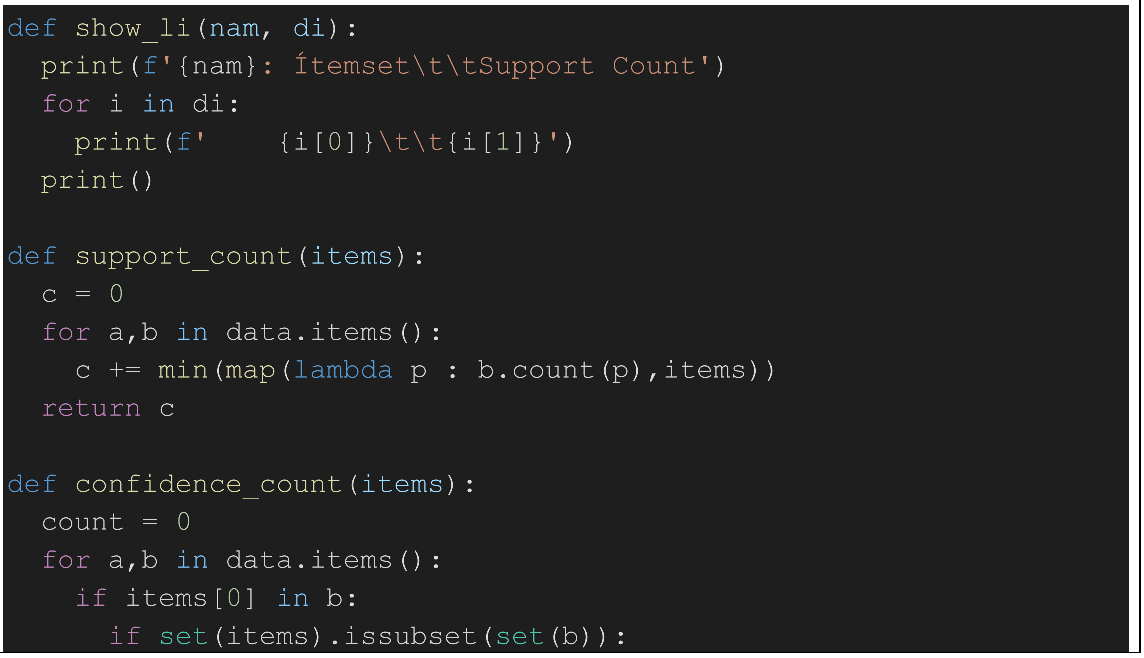
```
from itertools import combinations, permutations
data = {'T1': ['M','O','N','K','E','Y'],
        'T2': ['D','O','N','K','E','Y'],
        'T3': ['M','A','K','E'],
        'T4': ['M','U','C','K','Y'],
        'T5': ['C','O','O','K','E']
        }

min_sup_per = int(input('Input minimum support percentage: '))*0.01
min_count = int(min_sup_per * len(data))
min_conf_per = int(input('Input minimum confidence percentage: '))
```

Output:



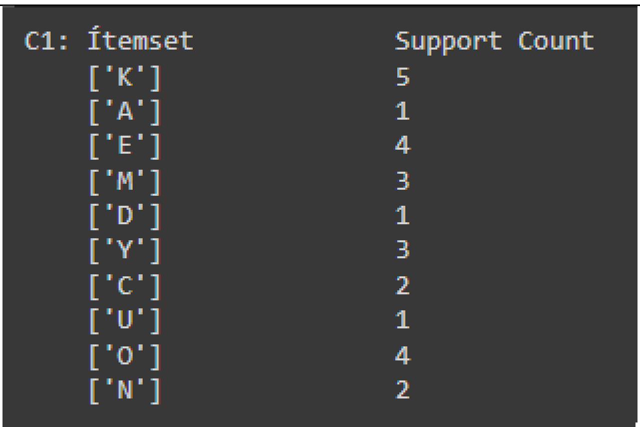
```
Input minimum support percentage: 60
Input minimum confidence percentage: 60
```

Code:

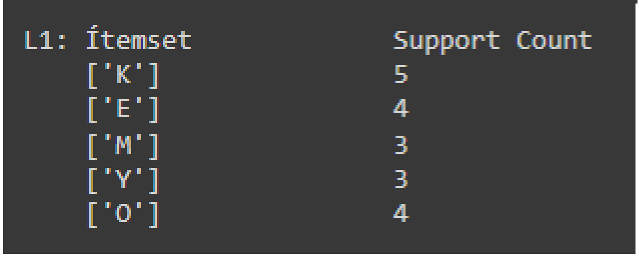


```
def show_li(nam, di):
  print(f'{nam}: Ítemset\t\tSupport Count')
  for i in di:
    print(f'    {i[0]}\t\t{i[1]}')
  print()

def support_count(items):
  c = 0
  for a,b in data.items():
    c += min(map(lambda p : b.count(p),items))
  return c

def confidence_count(items):
  count = 0
  for a,b in data.items():
    if items[0] in b:
      if set(items).issubset(set(b)):
        count+=1
  return count

def step(num, li):
  uniq = li if num < 2 else list(set(a for b,_ in li for a in b))
  c = [list(a) for a in combinations(uniq, num) if len(set(a)) ==
num]
  c = list(map(lambda a : [a, support_count(a)],c))

  show_li(f'C{num}',c)

  l = list(filter(lambda a: a[1] >= min_count, c))

  show_li(f'L{num}',l)

  return l

def rule(items):
  print("Rule\t\tSupport Count\tConfidence Percentage")
  for p in items:
    for i in range(2,4):
      for a in permutations(p[0], i):
        supp_count = support_count(a)
        conf_count = confidence_count(a)
        conf_per = conf_count/len(data)
        if supp_count >= min_count and conf_per >= min_sup_per:
          print(f"{a[0]} ->
{a[1:]}\t\t{supp_count}\t{conf_per*100:.2f} %")

#generating initial candiate list from given data
initial_list = list(set(value for values in data.values() for
value in values))

temp_list = initial_list

#Iteratively generating larger sets of reoccuring items
for i in range(1,4):
  temp_list = step(i,temp_list)

#giving the association rules found
print('Associatian Rules:')
rule(temp_list)
```

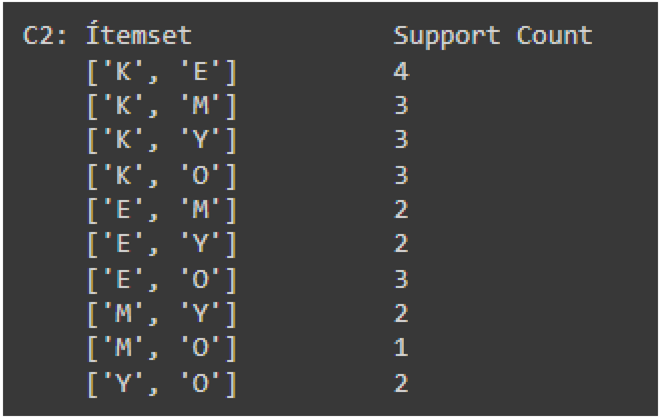
Output:



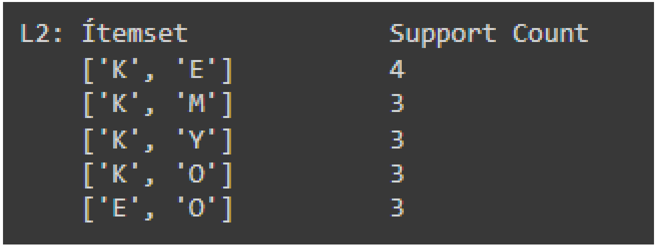
```
C1: Ítemset             Support Count
    ['K']               5
    ['A']               1
    ['E']               4
    ['M']               3
    ['D']               1
    ['Y']               3
    ['C']               2
    ['U']               1
    ['O']               4
    ['N']               2
```



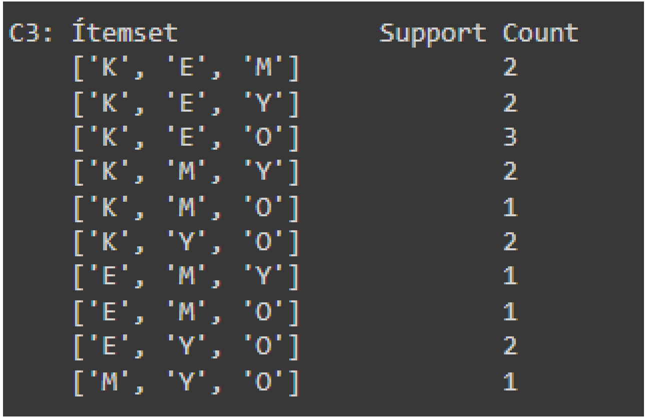
```
L1: Ítemset             Support Count
    ['K']               5
    ['E']               4
    ['M']               3
    ['Y']               3
    ['O']               4
```



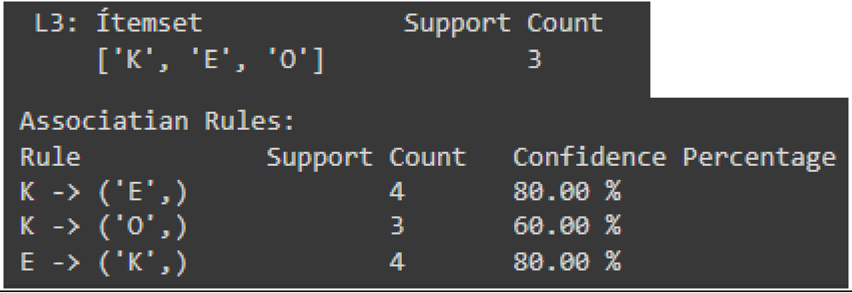
```
C2: Ítemset             Support Count
    ['K', 'E']          4
    ['K', 'M']          3
    ['K', 'Y']          3
    ['K', 'O']          3
    ['E', 'M']          2
    ['E', 'Y']          2
    ['E', 'O']          3
    ['M', 'Y']          2
    ['M', 'O']          1
    ['Y', 'O']          2
```



```
L2: Ítemset             Support Count
    ['K', 'E']          4
    ['K', 'M']          3
    ['K', 'Y']          3
    ['K', 'O']          3
    ['E', 'O']          3
```



```
C3: Ítemset             Support Count
    ['K', 'E', 'M']           2
    ['K', 'E', 'Y']           2
    ['K', 'E', 'O']           3
    ['K', 'M', 'Y']           2
    ['K', 'M', 'O']           1
    ['K', 'Y', 'O']           2
    ['E', 'M', 'Y']           1
    ['E', 'M', 'O']           1
    ['E', 'Y', 'O']           2
    ['M', 'Y', 'O']           1
```



```
L3: Ítemset             Support Count
    ['K', 'E', 'O']           3

Associatian Rules:
Rule            Support Count    Confidence Percentage
K -> ('E',)             4        80.00 %
K -> ('O',)             3        60.00 %
E -> ('K',)             4        80.00 %
```

Conclusion	Implementing the Apriori algorithm in Python allows for the extraction of association rules that meet specified thresholds, providing valuable insights into relationships within datasets for enhanced decision-making and data analysis.

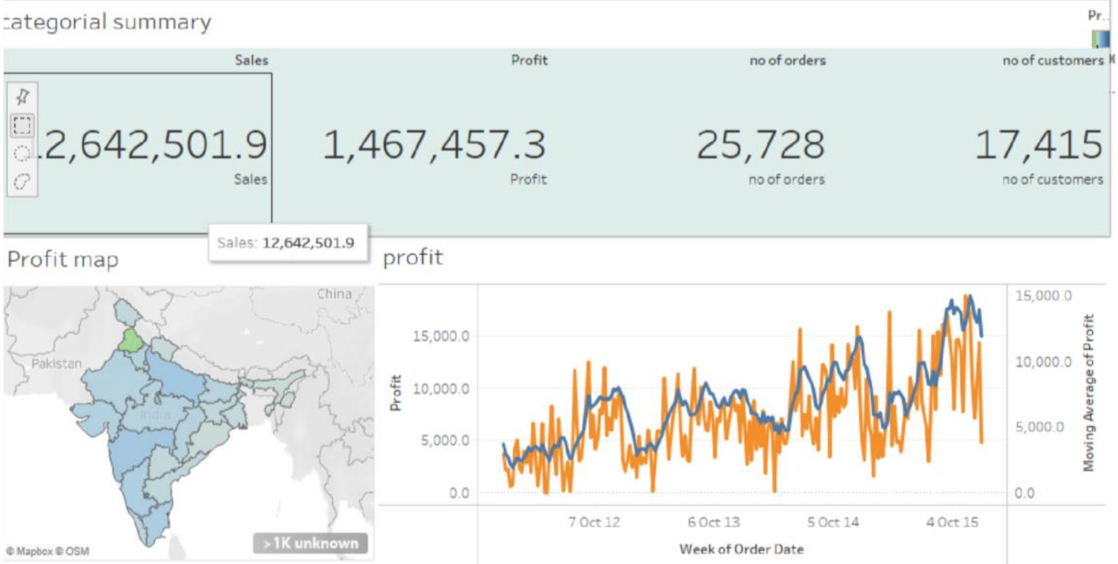
Signature of Faculty

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Academic Year: 2023-2024

Name: Bhavvya Jain	**Sap Id:** 60018220108
Course: Data Mining and Analytics	**Roll No:** S019
Branch: Artificial Intelligence and Data Science	**Div:** S
Course Code: DJS22ADL403	**Batch:** A1

Experiment 07

Title	Design interactive dashboard using tableau
Aim	Design interactive dashboard using tableau
Software	Tableau
Implementation	 :ategorial summary Pr.. Sales Profit no of orders no of customers .2,642,501.9 Sales 1,467,457.3 Profit 25,728 no of orders 17,415 no of customers Sales: 12,642,501.9 Profit map China Pakistan India © Mapbox © OSM > 1K unknown profit 15,000.0 10,000.0 5,000.0 0.0 Profit 7 Oct 12 6 Oct 13 5 Oct 14 4 Oct 15 Week of Order Date Moving Average of Profit
Conclusion	Creating dashboards in Tableau is a fundamental aspect of data analysis, allowing users to visualize and present data effectively through interactive and visually engaging displays.

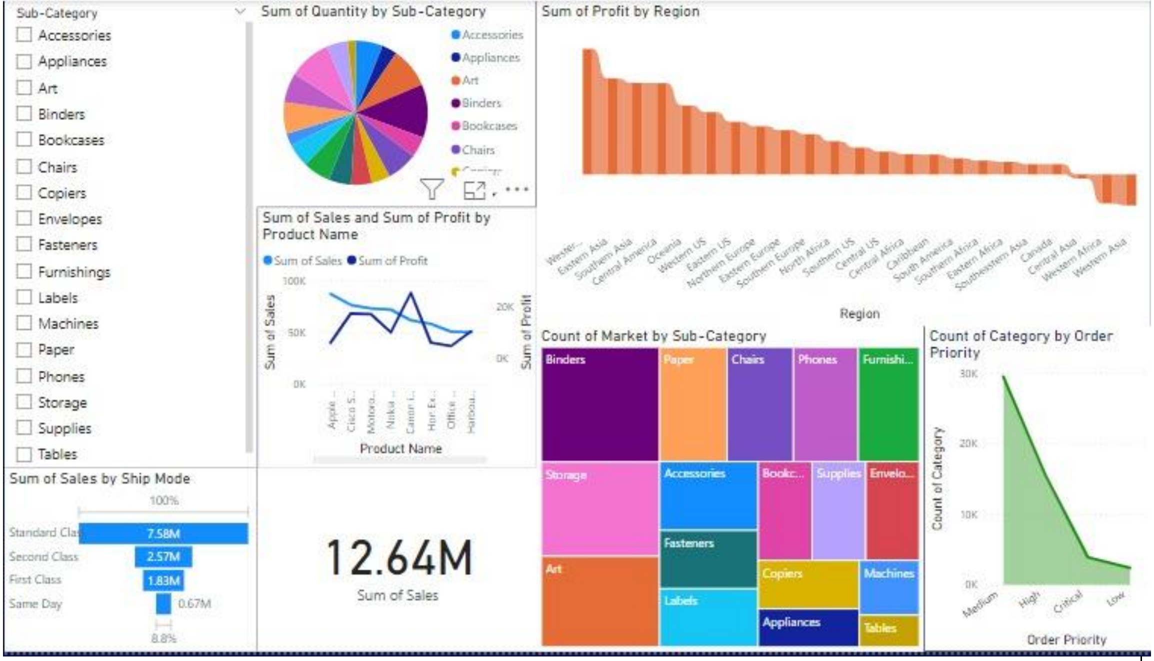
Signature of Faculty

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Academic Year: 2023-2024

Name: Bhavvya Jain	**Sap Id:** 60018220108
Course: Data Mining and Analytics	**Roll No:** S019
Branch: Artificial Intelligence and Data Science	**Div:** S
Course Code: DJS22ADL403	**Batch:** A1

Experiment 08

Title	Design interactive dashboard using PowerBI
Aim	Design interactive dashboard using PowerBI
Software	PowerBi
Implementation	 Sub-Category □ Accessories □ Appliances □ Art □ Binders □ Bookcases □ Chairs □ Copiers □ Envelopes □ Fasteners □ Furnishings □ Labels □ Machines □ Paper □ Phones □ Storage □ Supplies □ Tables Sum of Quantity by Sub-Category ● Accessories ● Appliances ● Art ● Binders ● Bookcases ● Chairs Sum of Profit by Region Wester... Eastern Asia Southern Asia Central America Oceania Western US Eastern US Northern Europe Eastern Europe Southern Europe North Africa Southern US Central US Central Africa Caribbean South America Southern Africa Eastern Africa Southeastern Asia Canada Central Asia Western Africa Western Asia Region Sum of Sales and Sum of Profit by Product Name ● Sum of Sales ● Sum of Profit 100K 50K 0K Sum of Sales 20K 0K Sum of Profit Apple ... Cisco S... Motoro... Nokia ... Canon i... Hon Ex... Office ... Harbou... Product Name Count of Market by Sub-Category Binders Paper Chairs Phones Furnishi... Storage Accessories Bookc... Supplies Envelo... Fasteners Art Copiers Machines Labels Appliances Tables Count of Category by Order Priority 30K 20K 10K 0K Count of Category Medium High Critical Low Order Priority Sum of Sales by Ship Mode 100% Standard Clas 7.58M Second Class 2.57M First Class 1.83M Same Day 0.67M 8.8% 12.64M Sum of Sales
Conclusion	PowerBI is not just a visualization tool; it offers a wide array of features beyond typical graphs and charts, enabling users to transform data into valuable insights for informed decision-making.

Signature of Faculty

Academic Year: 2023-2024

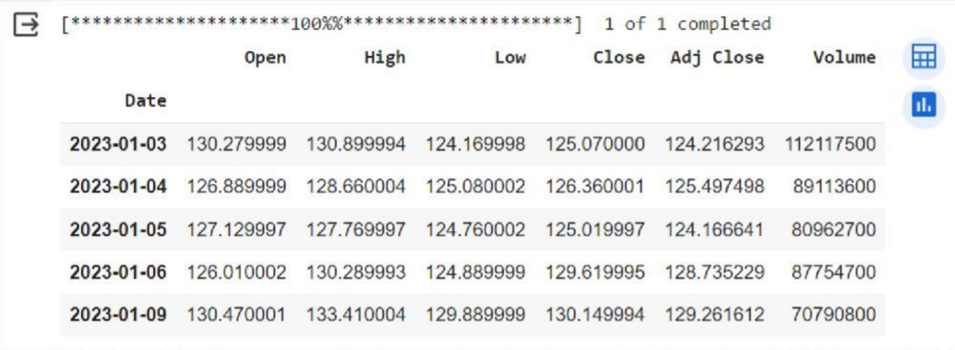
Name: Bhavvya Jain	**Sap Id:** 60018220108
Course: Data Mining and Analytics	**Roll No:** S019
Branch: Artificial Intelligence and Data Science	**Div:** S
Course Code: DJS22ADL403	**Batch:** A1

Experiment 09

Title	Perform Web Mining Analysis
Aim	To Perform Web Mining Analysis.
Software	Google Colab

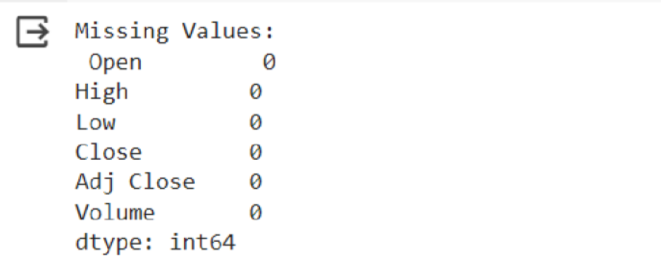
Implementation

```
import yfinance as yf
# Define the stock symbol and date range symbol =
'AAPL' start_date = '2023-01-01' end_date = '2024-01-
01'


# Retrieve historical stock price data from Yahoo Finance
stock_data =  yf.download(symbol, start=start_date, end=end_date)
stock_data.head()
```

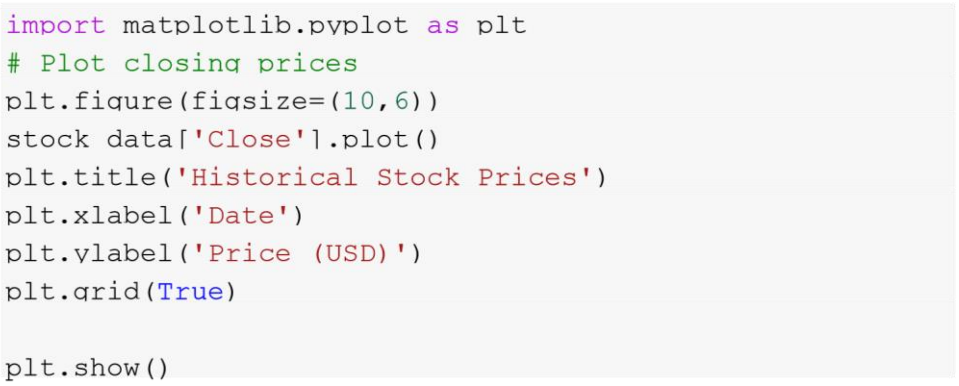


```
[*********************100%%**********************]  1 of 1 completed
                  Open        High         Low       Close   Adj Close     Volume

      Date
2023-01-03  130.279999  130.899994  124.169998  125.070000  124.216293  112117500
2023-01-04  126.889999  128.660004  125.080002  126.360001  125.497498   89113600
2023-01-05  127.129997  127.769997  124.760002  125.019997  124.166641   80962700
2023-01-06  126.010002  130.289993  124.889999  129.619995  128.735229   87754700
2023-01-09  130.470001  133.410004  129.889999  130.149994  129.261612   70790800
```

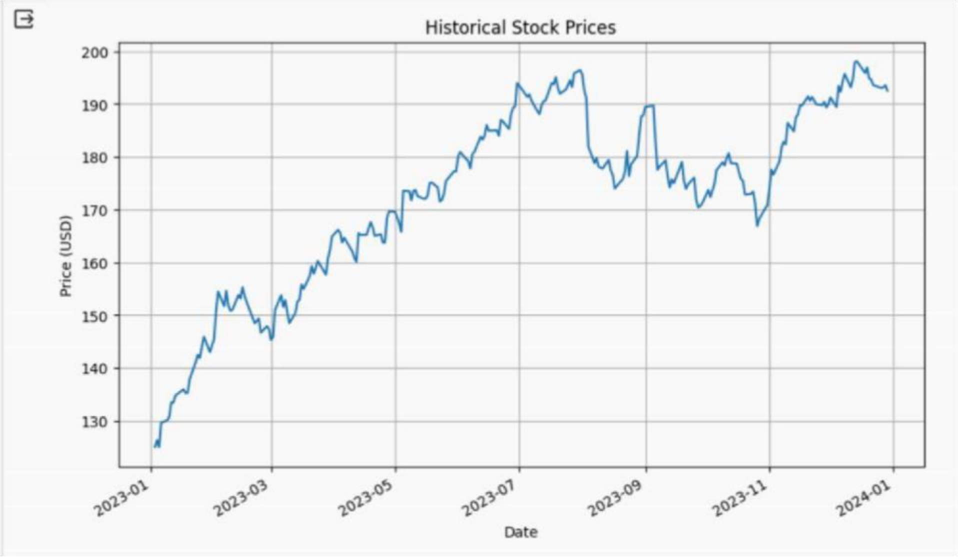
```
# Check for missing values  missing_values =
stock_data.isnull().sum() print("Missing
Values:\n", missing_values)


# Drop rows with missing values (if any)
stock_data.dropna(inplace=True)
```

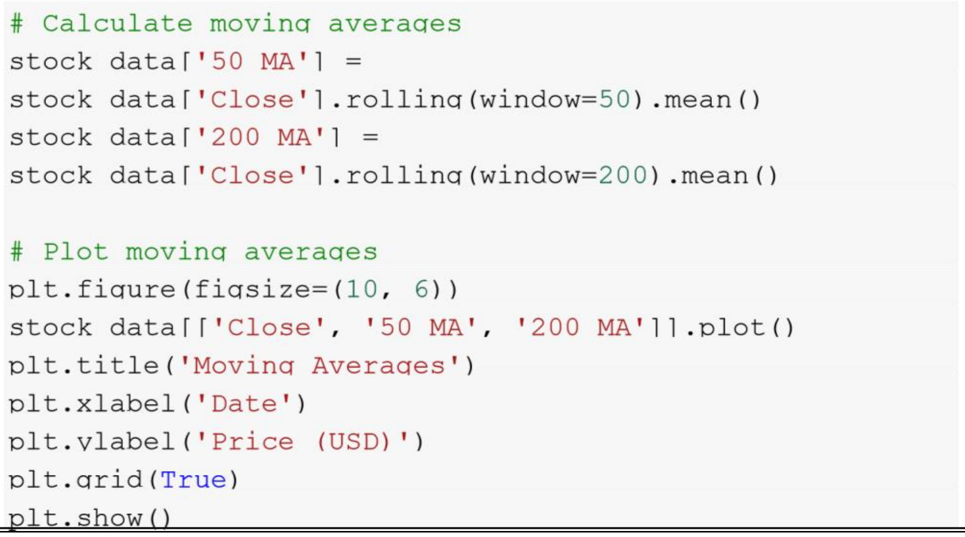


```
Missing Values:
 Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

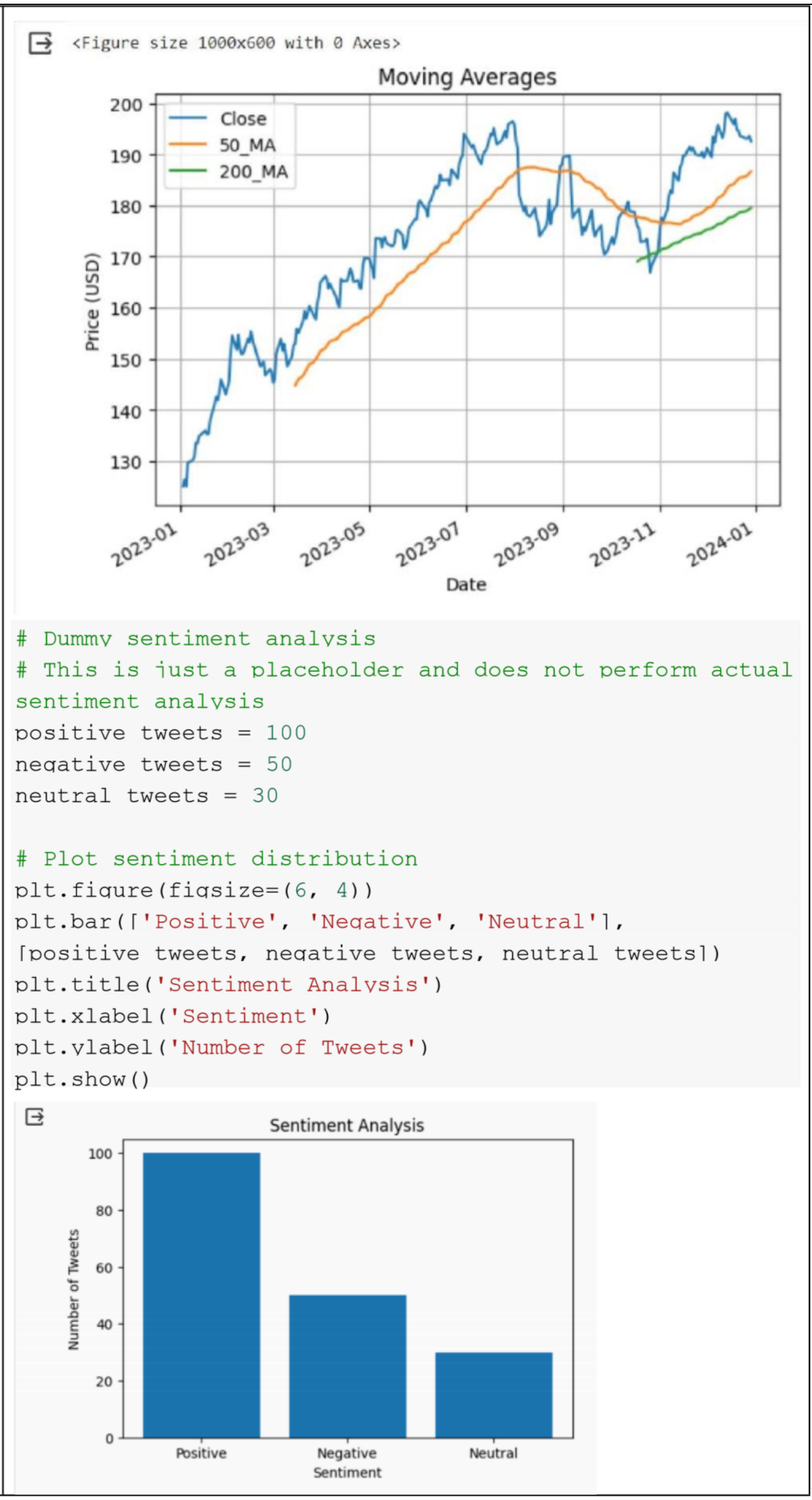


```
import matplotlib.pyplot as plt
# Plot closing prices
plt.figure(figsize=(10,6))
stock_data['Close'].plot()
plt.title('Historical Stock Prices')
plt.xlabel('Date')
plt.ylabel('Price (USD)')
plt.grid(True)

plt.show()
```





```
# Calculate moving averages
stock_data['50_MA'] =
stock_data['Close'].rolling(window=50).mean()
stock_data['200_MA'] =
stock_data['Close'].rolling(window=200).mean()

# Plot moving averages
plt.figure(figsize=(10, 6))
stock_data[['Close', '50_MA', '200_MA']].plot()
plt.title('Moving Averages')
plt.xlabel('Date')
plt.ylabel('Price (USD)')
plt.grid(True)
plt.show()
```

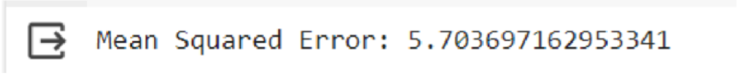


```
<Figure size 1000x600 with 0 Axes>
```

```
# Dummy sentiment analysis
# This is just a placeholder and does not perform actual
sentiment analysis
positive_tweets = 100
negative_tweets = 50
neutral_tweets = 30

# Plot sentiment distribution
plt.figure(figsize=(6, 4))
plt.bar(['Positive', 'Negative', 'Neutral'],
[positive_tweets, negative_tweets, neutral_tweets])
plt.title('Sentiment Analysis')
plt.xlabel('Sentiment')
plt.ylabel('Number of Tweets')
plt.show()
```

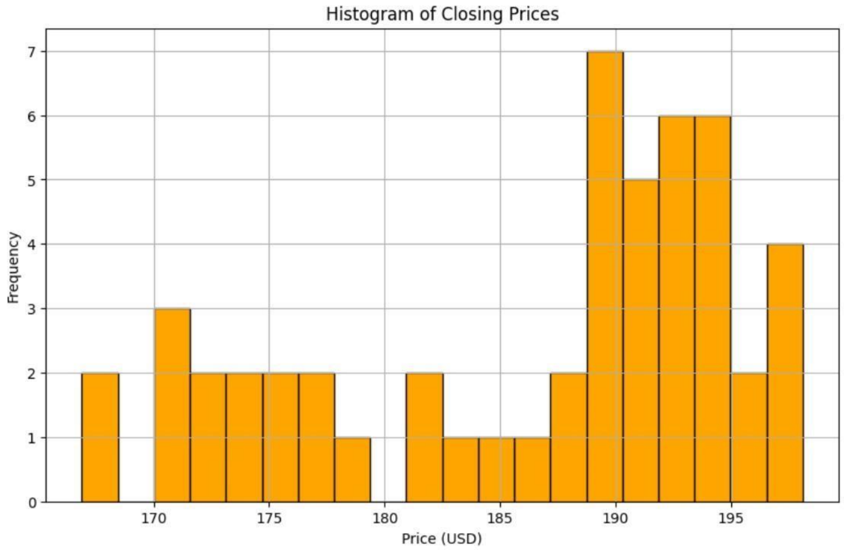
```
from sklearn.linear_model  import  LinearRegression  from
sklearn.model_selection     import  train_test_split
from sklearn.metrics  import mean_squared_error
# Ensure the  features and target variable have the same  length
```

```
stock_data = stock_data.dropna()   # Drop rows with
missing values
# Define features (moving averages) and target variable (closing
price)
X = stock_data[['50_MA', '200_MA']].values  y =
stock_data['Close'].values

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Train a linear regression model  model =
LinearRegression()  model.fit(X_train, y_train)

# Make predictions  y_pred =
model.predict(X_test)

# Calculate mean squared error  mse =
mean_squared_error(y_test, y_pred)

print("Mean Squared Error:", mse)
```



```
Mean Squared Error: 5.703697162953341
```

```
import matplotlib.pyplot as plt

# Assuming stock_data contains the 'Close' prices
plt.figure(figsize=(10, 6))
plt.hist(stock_data['Close'], bins    =20, color='orange',
edgecolor='black')
plt.title('Histogram of Closing Prices')
plt.xlabel('Price (USD)')  plt.ylabel(
'Frequency')
plt.grid(True)
plt.show()
```



Conclusion	Hence, we have studied handling of Categorical Data using One Hot Encoding.

Signature of Faculty