NAME – BAGHEL VISHAL

ROLL NO. – 01

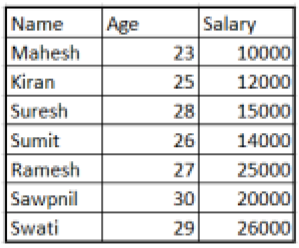
SEM - VII

SUBJECT – DATA ANALYSIS

COURSE - COMPUTER SCIENCE

ASSIGNMENT – 1 : UNIT 1 & 2

Q1 : Create a data frame of the below table.



Name	Age	Salary
Mahesh	23	10000
Kiran	25	12000
Suresh	28	15000
Sumit	26	14000
Ramesh	27	25000
Sawpnil	30	20000
Swati	29	26000

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt


# 1 - Create a data frame of the below table.

df = pd.DataFrame({

   'name': ['Mahesh', 'Kiran', 'Suresh', 'Sumit', 'Ramesh', 'Swapnil', 'Swati'],

   'age': [23, 25, 28, 26, 27, 30, 29],

   'salary': [10000, 12000, 15000, 14000, 25000, 20000, 26000]

})

print(df)
```

```
      name  age  salary
0   Mahesh   23   10000
1    Kiran   25   12000
2   Suresh   28   15000
3    Sumit   26   14000
4   Ramesh   27   25000
5  Swapnil   30   20000
6    Swati   29   26000
```

```
# 2 - Get the maximum values of salary from the data frame.

max_salary_index = df['salary'].idxmax()

print("Maximum Salary Person Details \nName :", df['name'][max_salary_index], ", Age :", df['age'][max_salary_index], ", Salary :", df['salary'][max_salary_index])
```

```
Maximum Salary Person Details 
Name : Swati , Age : 29 , Salary : 26000
```

```
# 3 - Get the minimum values of the salary from the data frame.

min_salary_index = df['salary'].idxmin()

print("Minimum Salary Person Details \nName :", df['name'][min_salary_index], ", Age :", df['age'][min_salary_index], ", Salary :", df['salary'][min_salary_index])
```

```
Minimum Salary Person Details 
Name : Mahesh , Age : 23 , Salary : 10000
```

```
# 4 - Write the statement which will sort the marks in the Data Frame, in descending order.

print("Sorted Values of Salary : ", df['salary'].sort_values(ascending=False).to_list() )
```

```
Sorted Values of Salary :  [26000, 25000, 20000, 15000, 14000, 12000, 10000]
```

```
# 5 - Check the shape of data frame.

print("The shape of Dataframe Is", df.shape )
```

```
The shape of Dataframe Is (7, 3)
```

```
# 6 - Add one row in the above data frame. (Name: Parita, Age: 32, Salary:25000)

length = len(df)

df.loc[length, 'name'] = 'Parita'

df.loc[length, 'age'] = 32

df.loc[length, 'salary'] = 25000

print(df)
```

```
      name   age   salary
0   Mahesh  23.0  10000.0
1    Kiran  25.0  12000.0
2   Suresh  28.0  15000.0
3    Sumit  26.0  14000.0
4   Ramesh  27.0  25000.0
5  Swapnil  30.0  20000.0
6    Swati  29.0  26000.0
7   Parita  32.0  25000.0
8   Parita  32.0  25000.0
```

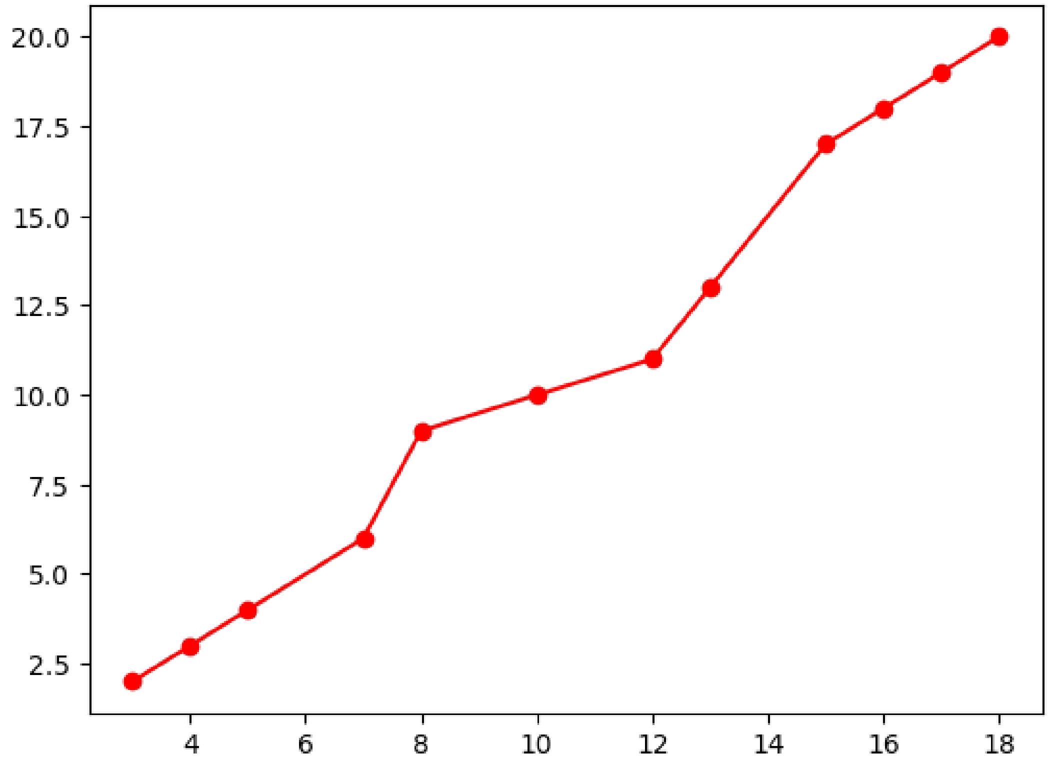
```
# 7 - Check the missing values in the data frame.

print("Missing Values Are :\n", df.isnull().any(), sep='' )
```

```
Missing Values Are :
name      False
age       False
salary    False
dtype: bool
```

```
# 8 - Display the data frame from Suresh to Parita by slicing.

print(df.loc[2:8])
```

```
name   age   salary
2   Suresh  28.0  15000.0
3    Sumit  26.0  14000.0
4   Ramesh  27.0  25000.0
5  Swapnil  30.0  20000.0
6    Swati  29.0  26000.0
7   Parita  32.0  25000.0
8   Parita  32.0  25000.0
```

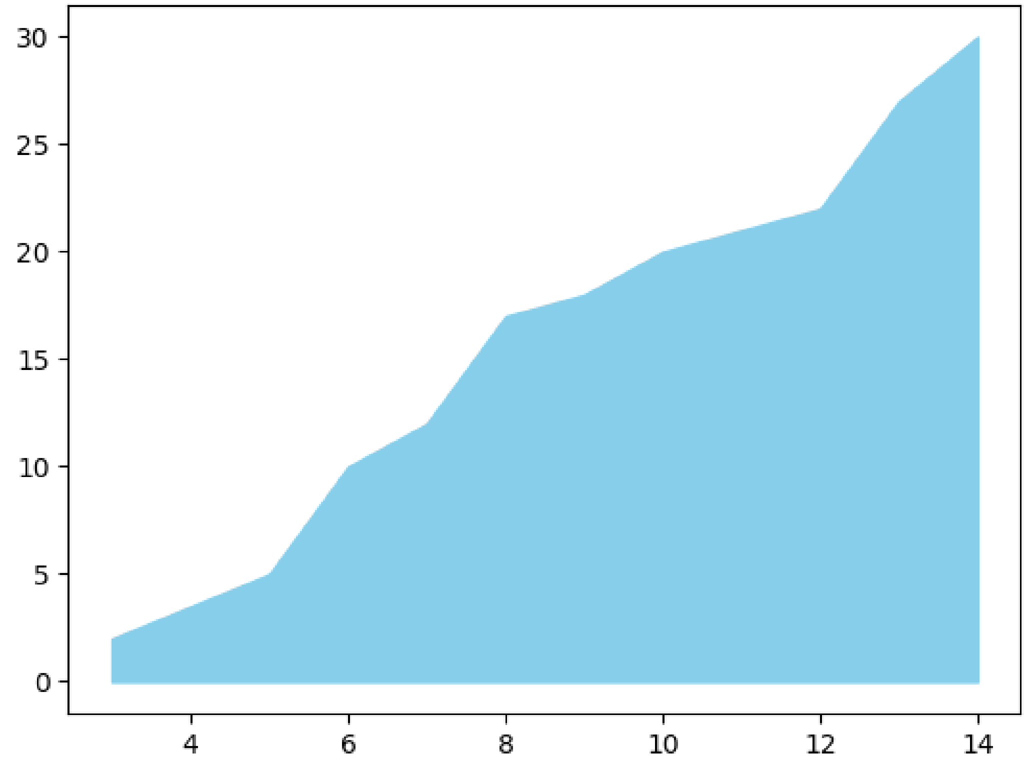
```
# 9 - Write a program to plot the ogive of random data from 1 to 20.

data = np.array([ np.random.randint(1, 20) for i in range(20) ])

data.sort()

counts = {}


for number in data:

    if number in counts:

        counts[number] += 1

    else:

        counts[number] = 1


last_index = -1

for index in counts:

    if last_index != -1:
        counts[index] = counts[last_index] + counts[index]
    last_index = index
counts
plt.plot(counts.keys(), counts.values(), marker='o', color='red')
plt.show()
```



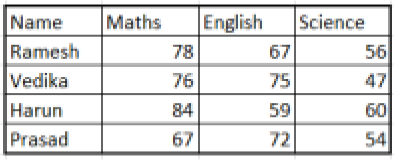
# 10 - Write a program to plot the area chart of the data from 1 to 30.

```
data = np.array([ np.random.randint(1, 15) for i in range(30) ])
data.sort()
counts = {}

for number in data:
    if number in counts:
        counts[number] += 1
    else:
        counts[number] = 1
```

```
last_index = -1
for index in counts:
    if last_index != -1:
        counts[index] = counts[last_index] + counts[index]
    last_index = index
counts
plt.fill_between(counts.keys(), counts.values(), color='skyblue')
plt.show()
```



Q11 : Create a data frame of the below table.



Name	Maths	English	Science
Ramesh	78	67	56
Vedika	76	75	47
Harun	84	59	60
Prasad	67	72	54

```
report = pd.DataFrame({
    'Name': ['Ramesh', 'Vedika', 'Harun', 'Prashant'],
    'Maths': [78, 76, 84, 67],
    'English': [67, 75, 59, 72],
    'Science': [56, 47, 60, 54]
})
print(report)
```

```
Name  Maths  English  Science
0    Ramesh     78       67       56
1    Vedika     76       75       47
2     Harun     84       59       60
3  Prashant     67       72       54
```

```
# 12 - Access the element in the 1st row in the 3rd column.
print("Element Value of 1st row and 3rd column is", report.loc[0,'English'])
```

```
Element Value of 1st row and 3rd column is 67
```

```
# 13 - Access all the element in 3rd column.
print("All the elements of 3rd column are", report.loc[:,'English'].to_list(), sep=' ')
```

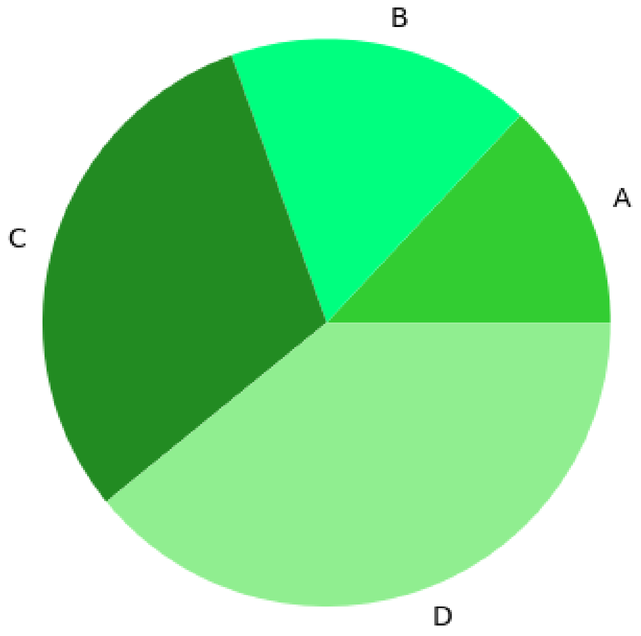
```
All the elements of 3rd column are [67, 75, 59, 72]
```

```
# 14 - Access elements of 2nd and 3rd row from 1st and 2nd column.
print("Elements Value of 2nd and 3rd row from 1st and 2nd column are", report.loc[1:2, ['Name','Maths']], sep="\n")
```

```
Elements Value of 2nd and 3rd row from 1st and 2nd column are
     Name  Maths
1  Vedika     76
2   Harun     84
```

# 15 - Write a program to plot a pie chart of values=[3,4,7,9] and categories=[‘A’, ‘B’, ‘C’, ‘D’].

```
values = [3, 4, 7, 9]

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


plt.pie(values, labels=categories, colors=['#32CD32','#00FF7F','#228B22','#90EE90'])

plt.show()
```



# 16 - Create a Series from Dictionary. {‘India’:’New Delhi’, ‘UK’: ‘London’, ‘Japan’:’Tokyo’}

```
series = pd.Series({

   'India': 'New Delhi',

   'UK': 'London',

   'Japan': 'Tokyo'

})

print(series)
```

```
India    New Delhi
UK          London
Japan        Tokyo
dtype: object
```

# 17 - Create a Series from 10 to 20 values using numpy and index should be of alphabet.

```
alphabetic_series = pd.Series( data = np.arange(10, 21), index= [chr(ord('A') + i ) for i in range(11) ] )

print(alphabetic_series)
```

```
A    10
B    11
C    12
D    13
E    14
F    15
G    16
H    17
I    18
J    19
K    20
dtype: int32
```

# 18 - Write the code for outlier removal from scratch.

```
# [ 4, 7, 10, 14, 36, 16, 18, 20, 67, 22, 2]

data = np.array([ 4, 7, 10, 14, 36, 16, 18, 20, 67, 22, 2] )

data.sort()

length = len(data)


Q1 = None

Q3 = None


index = length/4

if index % 1 == 0:

   Q1 = ( data[length // 4 - 1] + data[length // 4 ] ) / 2

else:

   Q1 = data[length // 4]
```

```
index = 3 * length/4
if index % 1 == 0:
    Q3 = ( data[ ( 3 * length // 4 ) - 1] + data[ 3 * length // 4 ] ) / 2
else:
    Q3 = data[3 * length // 4]

print("Data Is", data)
print(f'Q1 : {Q1} and Q3 : {Q3} ')

IQR = Q3 - Q1
lower_limit = Q1 - 1.5 * IQR
upper_limit = Q3 + 1.5 * IQR

print(f'Lower Limit Is {lower_limit} and Upper Limit Is {upper_limit}')
print('After Removing Outliers Data Is', data[ (data > lower_limit) & (data < upper_limit) ] )
```

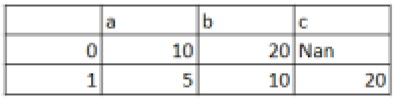
```
Data Is [ 2  4  7 10 14 16 18 20 22 36 67]
Q1 : 7 and Q3 : 22
Lower Limit Is -15.5 and Upper Limit Is 44.5
After Removing Outliers Data Is [ 2  4  7 10 14 16 18 20 22 36]
```

```
# 19 - Write the code of Finding Covariance from given data.
# X= [2, 4, 6, 8, 10] , Y= [1, 3, 5, 7, 9]
xi = [2, 4, 6, 8, 10]
yi = [1, 3, 5, 7, 9]

n = len(xi)

mean_x = sum(xi) / n
mean_y = sum(yi) / n
```

```
xi_x_yi_y = [ ( xi[index] - mean_x ) * ( yi[index] - mean_y ) for index in range(n) ]
covariance = sum(xi_x_yi_y) / ( n - 1 )
print("Covariance Is", covariance)
```

```
Covariance Is 10.0
```

Q20: Create a Data frame from Lists of Dictionary.

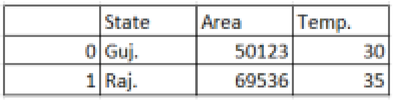


	a	b	c
0	10	20	Nan
1	5	10	20

```
example = pd.DataFrame({
    'a': [10, 5],
    'b': [20, 10],
    'c': [np.nan, 20]
})
print("Dataframe Example Is", example, sep='\n')
```

```
Dataframe Example Is
    a   b     c
0  10  20   NaN
1   5  10  20.0
```

Q21: Create a Data frame from Lists of Dictionary.

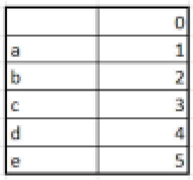


	State	Area	Temp.
0	Guj.	50123	30
1	Raj.	69536	35

```
states = pd.DataFrame({
    'State': ['Gujarat', 'Rajasthan'],
    'Area': [50123, 69536],
    'Temp': [30, 35]
})
print("Dataframe Example 1 Is", states, sep='\n')
```

```
Dataframe Example 1 Is
       State   Area  Temp
0    Gujarat  50123    30
1  Rajasthan  69536    35
```

Q22: Create a Data frame from Series.

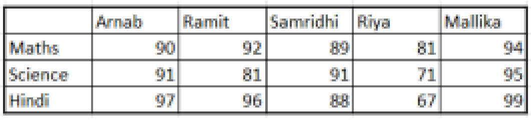


	0
a	1
b	2
c	3
d	4
e	5

```
series = pd.Series([1, 2, 3, 4, 5])

example = pd.DataFrame(data=series.values, index=[ chr( ord('a') + i) for i in range(len(series.values)) ] )

print(example)
```

```
   0
a  1
b  2
c  3
d  4
e  5
```

Q23: Create a Data frame from Dictionary of Series.



	Arnab	Ramit	Samridhi	Riya	Mallika
Maths	90	92	89	81	94
Science	91	81	91	71	95
Hindi	97	96	88	67	99

```
summary = pd.DataFrame({

   'Arnab': pd.Series([90, 91, 97]).values,

   'Ramit': pd.Series([92, 81, 96]).values,

   'Samridhi': pd.Series([89, 91, 88]).values,

   'Riya': pd.Series([81, 71, 67]).values,

   'Mallika': pd.Series([94, 95, 99]).values,

}, index=['Maths', 'Science', 'English'] )
```

```
print(summary)
```

```
        Arnab  Ramit  Samridhi  Riya  Mallika
Maths       90     92        89    81       94
Science     91     81        91    71       95
English     97     96        88    67       99
```

```
# 24 - Do the following pre- processing techniques on the different 10 data sets.

# a) Data Cleaning (Handling missing value for rows/columns, Handling Duplicates, Outliers detection and removal)

# b) Handling Categorical data

# c) Scaling of the data

# d) Data Normalization

# e) Identity insights which could be drawn from data and demonstrations of the same


import numpy as np

import pandas as pd

import sklearn


hotel_book= pd.read_csv('hotel_bookings.csv')

hotel_book


hotel_book.isnull().sum()

hotel_book=hotel_book.drop(['company','agent'],axis=1)

hotel_book



# a - Data Cleaning: Replace the null values with mode

hotel_book=hotel_book.fillna(hotel_book['country'].value_counts().index[0])

hotel_book

hotel_book.isnull().sum()
```

```
# b - Handling Categorical data: In the "hotel" column, replace the hotel names with "0" and "1" based on the condition that

# – if, "hotel" = "city_hotel", then "hotel" = "1"; else, "0"

hotel_book['hotel']=np.where(hotel_book['hotel'].str.contains('City Hotel'),1,0)

hotel_book

hotel_book.sample(5)



# c - Scaling of the data : Using the label encoder, assign a unique country code to each country

from sklearn.preprocessing import LabelEncoder

LE=LabelEncoder()

hotel_book['country code']=LE.fit_transform(hotel_book['country'])

hotel_book




# d, e - Data Normalization, Insights

# Identity insights which could be drawn from data and demonstrations of the same


from sklearn.preprocessing import OneHotEncoder

OHE=OneHotEncoder()

hotel_book['month']=OHE.fit_transform(hotel_book['arrival_date_month'])

hotel_book


OHE = OneHotEncoder(sparse=False, drop=None)


month_encoded = OHE.fit_transform(hotel_book[['arrival_date_month']])

month_df = pd.DataFrame(month_encoded, columns=OHE.get_feature_names_out(['arrival_date_month']))

hotel_book = pd.concat([hotel_book.drop('arrival_date_month', axis=1), month_df], axis=1)
```

UNIT – 2 : GRAPHICAL VISUALIZATION

Title	Content	Book Chapter
Data Visualization (Qualitative and Quantitative data)	Bar Charts, Pie Chart, Scatter Plots, Line Chart, Area Chart, Histogram, Ogive, Dot Plot	**2**
Descriptive statistics	Measures of location, Measures of variability, Measures of association between two variables, Measures of distribution	**3**

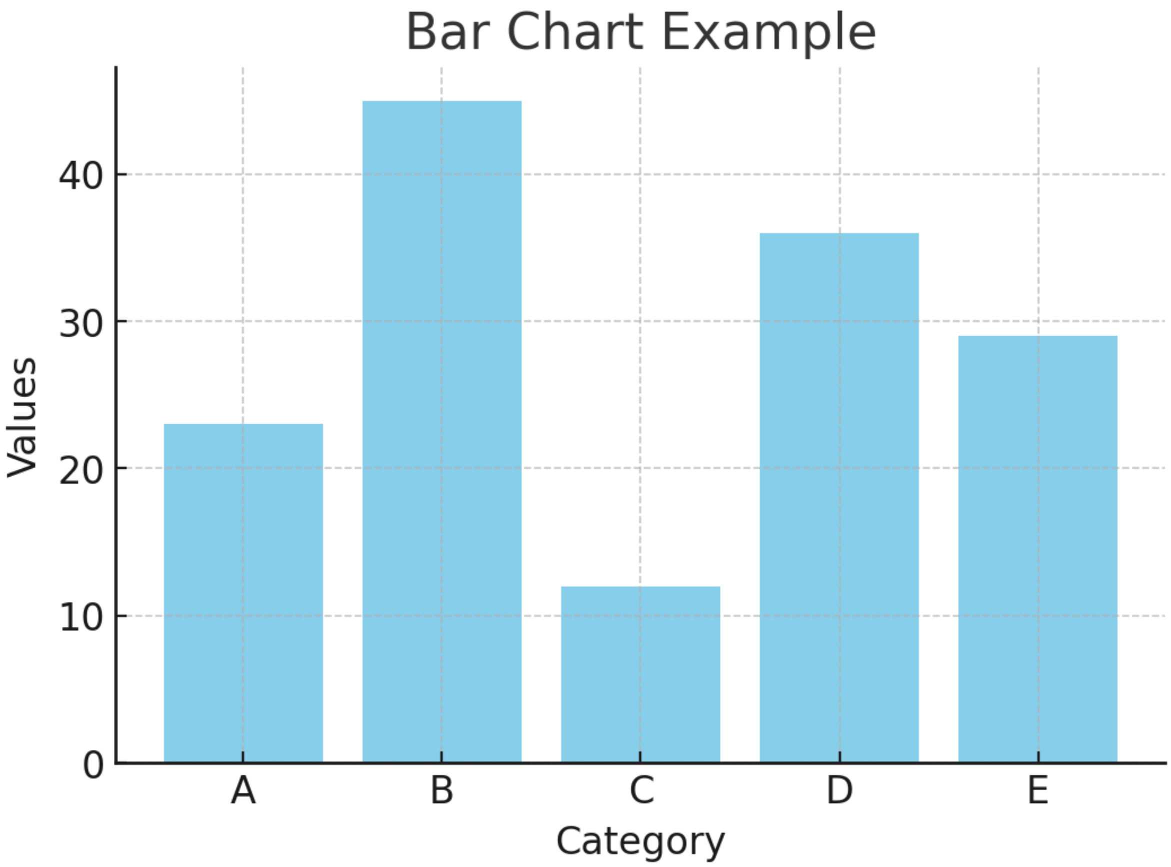
CHAPTER – 2: Data Visualization

(Qualitative and Quantitative data)

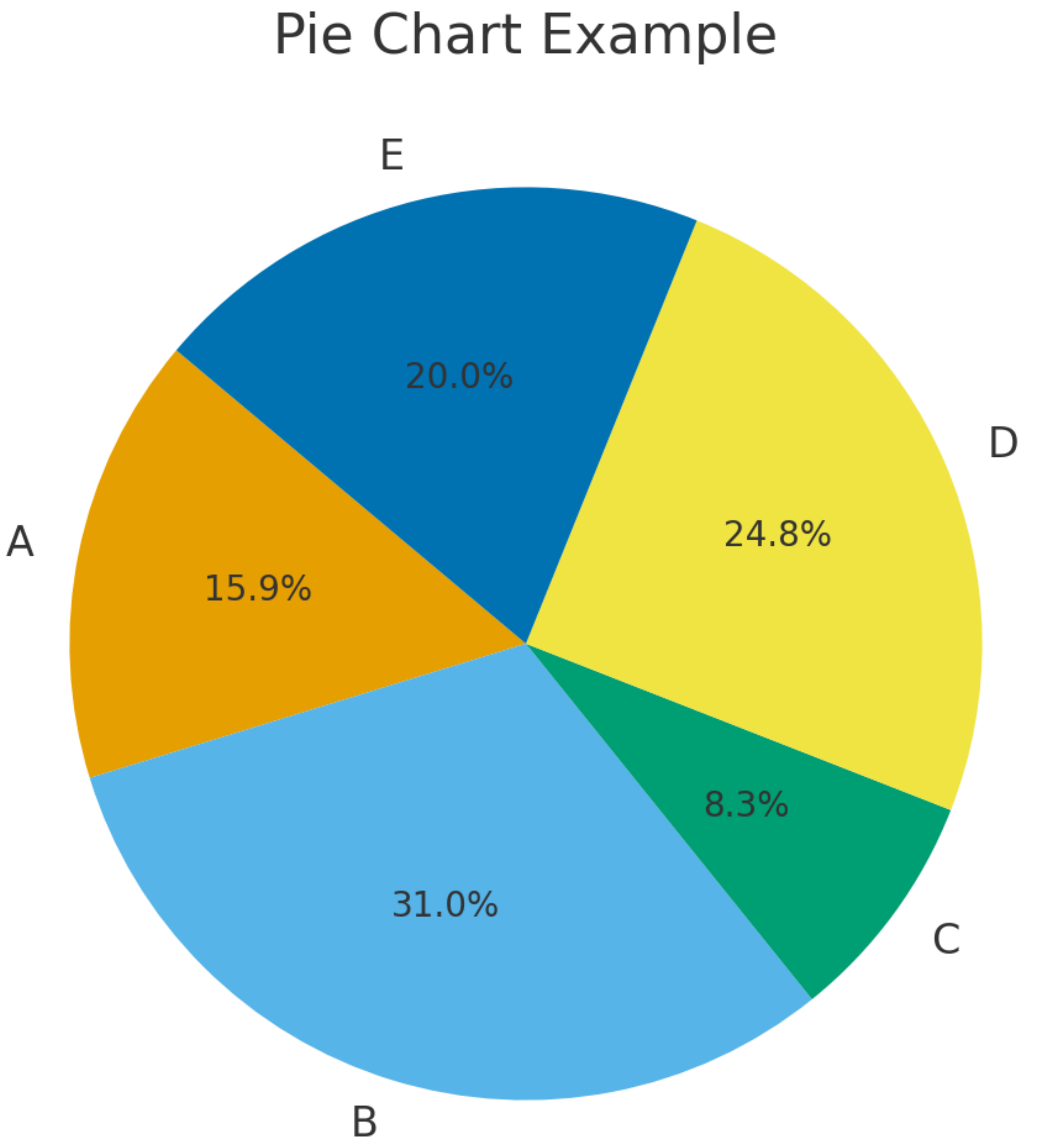
- **Bar Charts, Pie Chart, Scatter Plots, Line Chart**
- **Area Chart, Histogram, Ogive, Dot Plot**

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

# Sample dataset for exercises (inspired by textbook style data)
data = {
   "Category": ["A", "B", "C", "D", "E"],
   "Values": [23, 45, 12, 36, 29],
   "Sales": [200, 340, 150, 400, 300],
   "Profit": [50, 80, 40, 100, 70]
}



# Create DataFrame
df = pd.DataFrame(data)

# Save to CSV for demonstration
csv_path = "data.csv"
df.to_csv(csv_path, index=False)
```

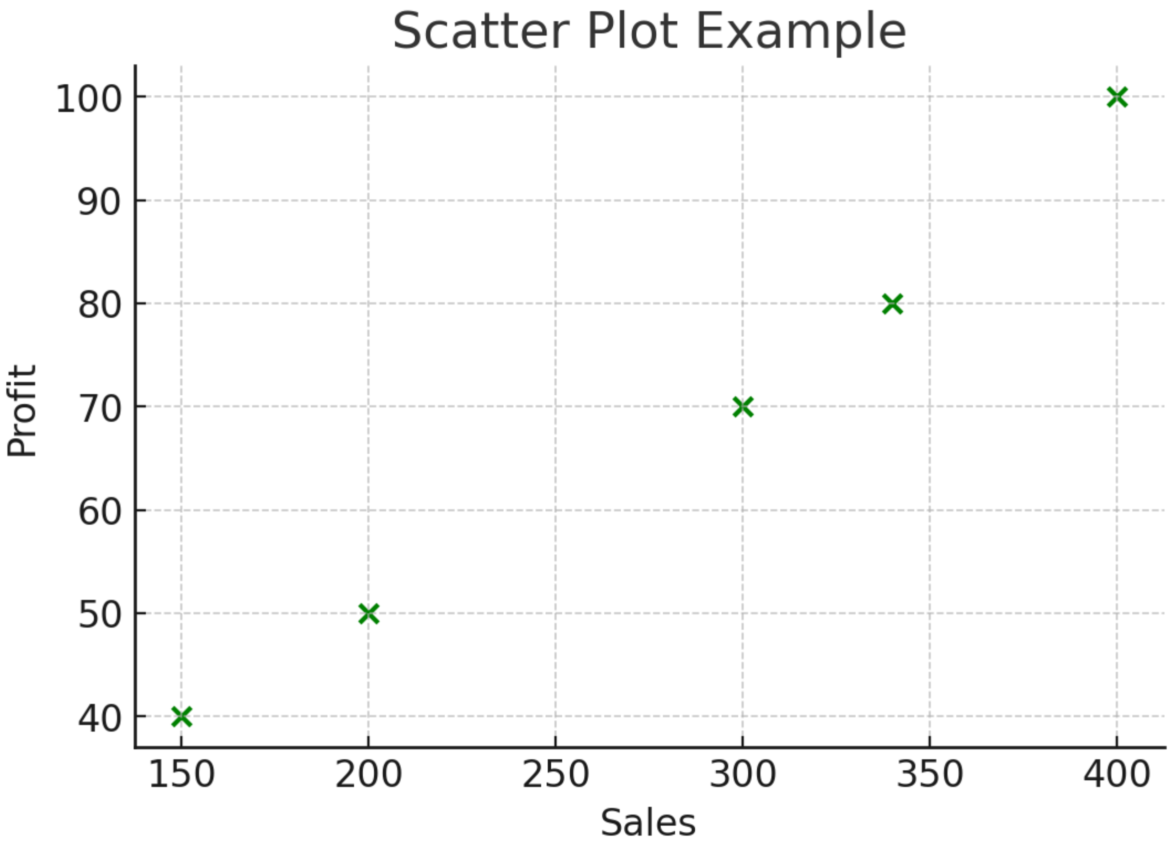
```
# Bar Chart
plt.figure(figsize=(6,4))
plt.bar(df["Category"], df["Values"], color="skyblue")
plt.title("Bar Chart Example")
plt.xlabel("Category")
plt.ylabel("Values")
plt.show()
```



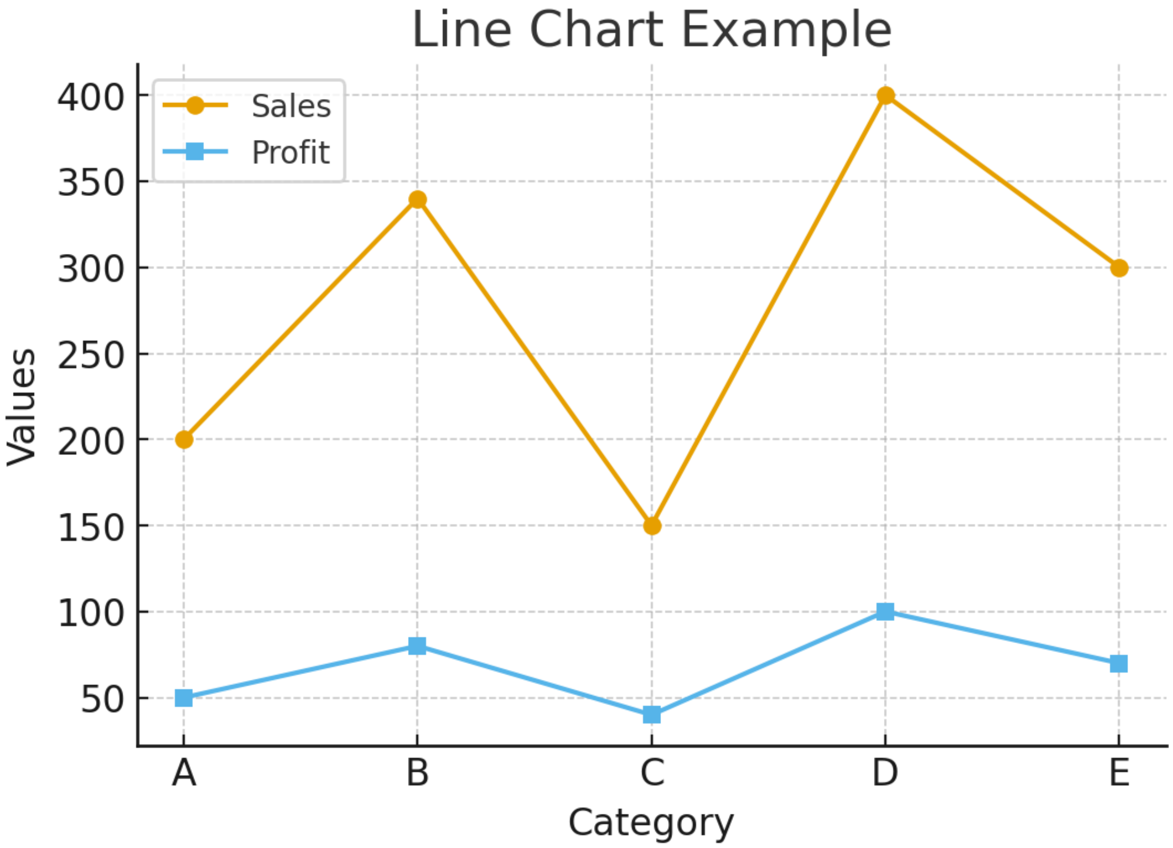
```
# Pie Chart
plt.figure(figsize=(6,6))
plt.pie(df["Values"], labels=df["Category"], autopct='%1.1f%%', startangle=140)
plt.title("Pie Chart Example")
plt.show()
```



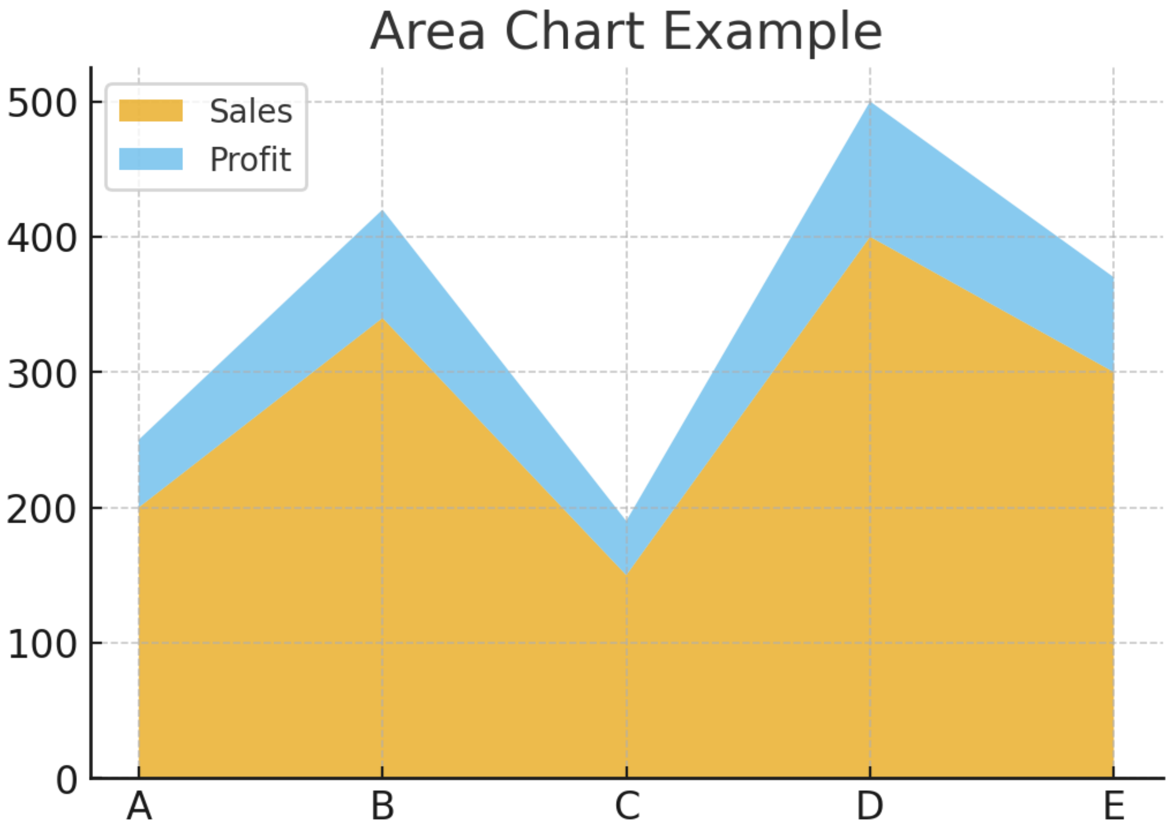
```
# Scatter Plot
plt.figure(figsize=(6,4))
plt.scatter(df["Sales"], df["Profit"], color="green")
plt.title("Scatter Plot Example")
plt.xlabel("Sales")
plt.ylabel("Profit")
plt.show()
```



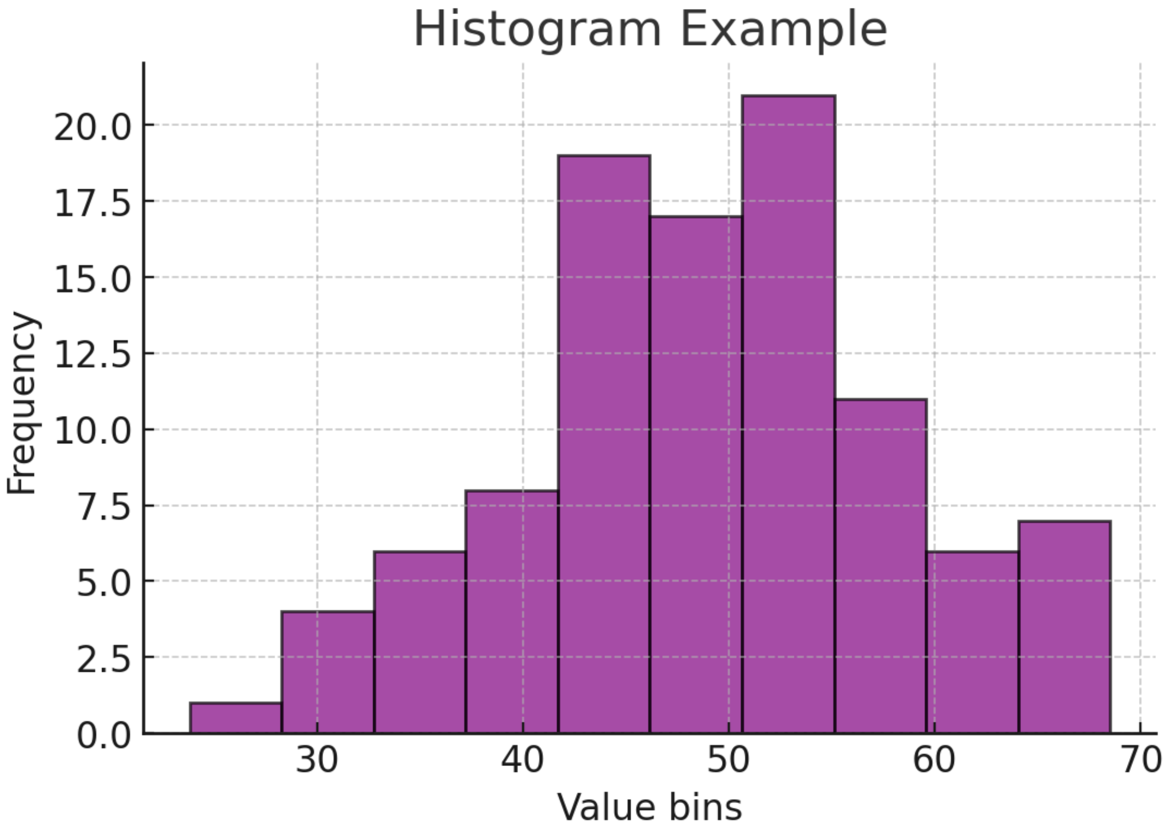
```
# Line Chart
plt.figure(figsize=(6,4))
plt.plot(df["Category"], df["Sales"], marker="o", label="Sales")
plt.plot(df["Category"], df["Profit"], marker="s", label="Profit")
plt.title("Line Chart Example")
plt.xlabel("Category")
plt.ylabel("Values")
plt.legend()
plt.show()
```



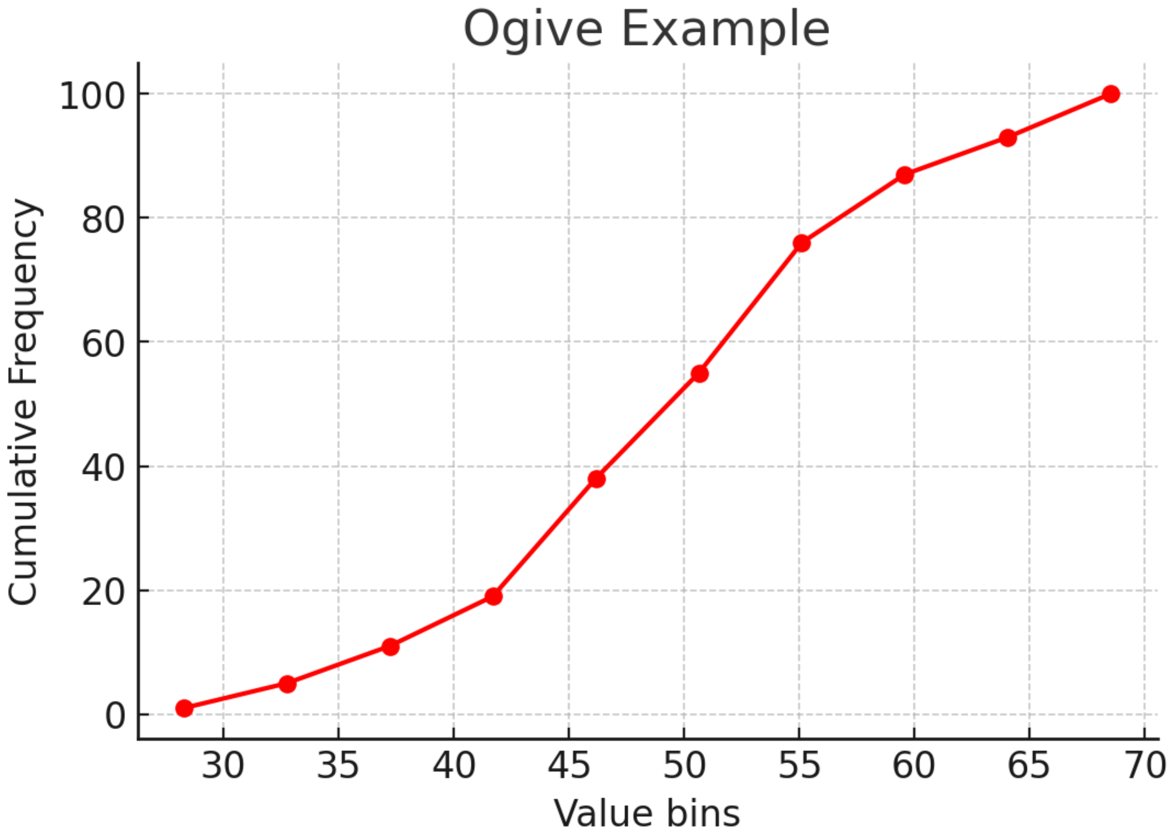
```
# Area Chart
plt.figure(figsize=(6,4))
plt.stackplot(df["Category"], df["Sales"], df["Profit"], labels=["Sales","Profit"],
alpha=0.7)
plt.title("Area Chart Example")
plt.legend(loc="upper left")
plt.show()
```



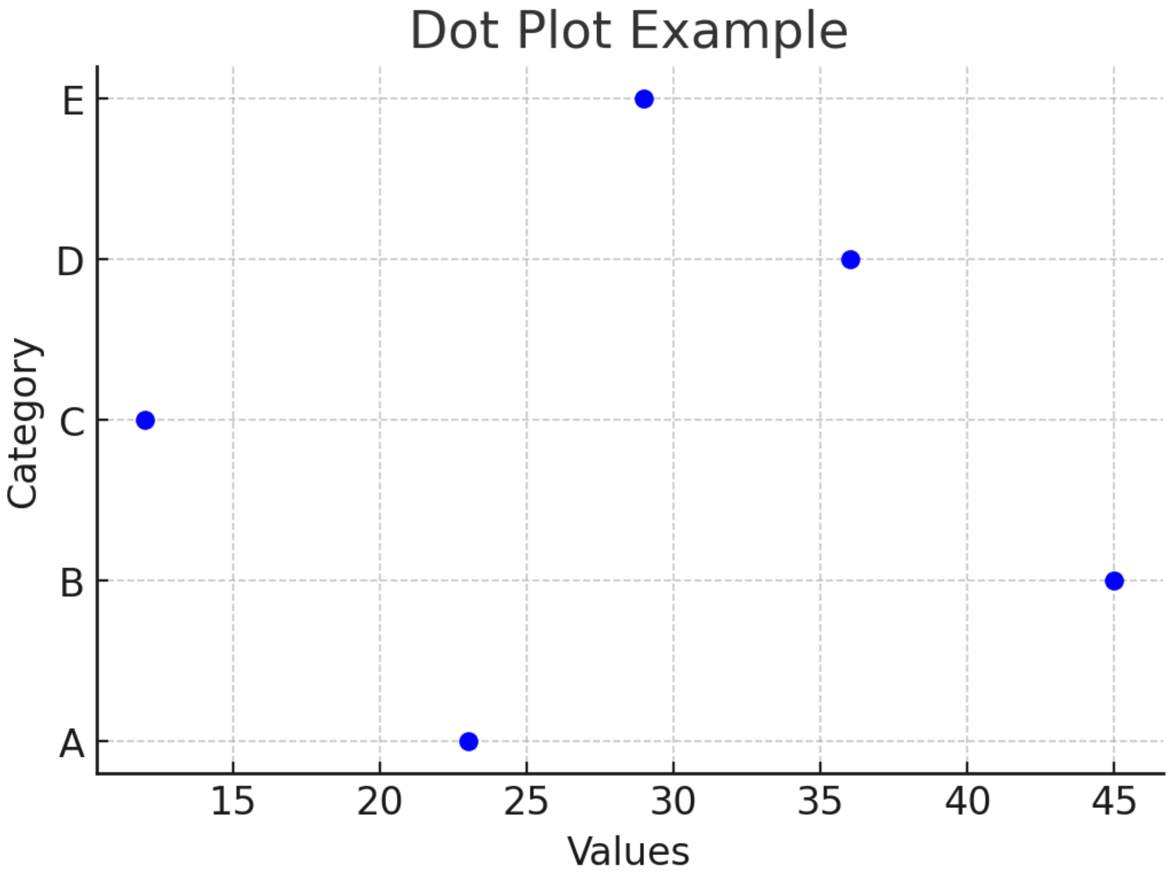
```
# Histogram
sample_data = np.random.randint(1, 75)
plt.figure(figsize=(6,4))
plt.hist(sample_data, bins=10, color="purple", alpha=0.7, edgecolor="black")
plt.title("Histogram Example")
plt.xlabel("Value bins")
plt.ylabel("Frequency")
plt.show()
```



```
# Ogive (Cumulative Histogram)
counts, bin_edges = np.histogram(sample_data, bins=10)
cum_counts = np.cumsum(counts)
plt.figure(figsize=(6,4))
plt.plot(bin_edges[1:], cum_counts, marker="o", color="red")
plt.title("Ogive Example")
plt.xlabel("Value bins")
plt.ylabel("Cumulative Frequency")
plt.show()
```



```
# Dot Plot
plt.figure(figsize=(6,4))
for i, val in enumerate(df["Values"]):
    plt.plot([val], [i], 'bo')
plt.yticks(range(len(df["Category"])), df["Category"])
plt.title("Dot Plot Example")
plt.xlabel("Values")
plt.ylabel("Category")
plt.show()
```



CHAPTER – 3: Descriptive statistics

- **Measures of location**
- **Measures of variability**
- **Measures of association between two variables**
- **Measures of distribution**

Q1: The following data were used to construct the histograms of the number of days required to fill orders for Dawson Supply, Inc., and J.C. Clark Distributors

(see Figure 3.2).

- **Dawson Supply Days for Delivery: 11 10 9 10 11 11 10 11 10 10**
- **Clark Distributors Days for Delivery: 8 10 13 7 10 11 10 7 15 12**

```
import pandas as pd


# Data

dawson = [11, 10, 9, 10, 11, 11, 10, 11, 10, 10]

clark = [8, 10, 13, 7, 10, 11, 10, 7, 15, 12]


# Create DataFrame

df = pd.DataFrame({

    "Dawson": dawson,

    "Clark": clark

})


# Measures of Location (Central Tendency)

location_measures = {

    "Mean": df.mean(),

    "Median": df.median(),

    "Mode": df.mode().iloc[0],

    "Min": df.min(),

    "Max": df.max(),
    "Range": df.max() - df.min(),
    "25th Percentile": df.quantile(0.25),
    "50th Percentile": df.quantile(0.50),
    "75th Percentile": df.quantile(0.75)
}
location_df = pd.DataFrame(location_measures)

# Measures of Variability (Dispersion)
variability_measures = {
    "Variance": df.var(),
    "Standard Deviation": df.std(),
    "IQR": df.quantile(0.75) - df.quantile(0.25),
    "Coefficient of Variation": df.std() / df.mean()
}
variability_df = pd.DataFrame(variability_measures)

# Measures of Association (Between Dawson & Clark)
association_measures = {
    "Covariance": df.cov().iloc[0, 1],
    "Correlation": df.corr().iloc[0, 1]
}
association_series = pd.Series(association_measures)

# Measures of Distribution (Shape)
distribution_measures = {
    "Skewness": df.skew(),
    "Kurtosis": df.kurt()
}
distribution_df = pd.DataFrame(distribution_measures)
```

```
print("Measures of Location:\n", location_df)

print("Measures of Variability:\n", variability_df)

print("Measures of Association:\n", association_series)

print("Measures of Distribution:\n", distribution_df)
```

```
Measures of Location:
         Mean  Median  Mode  Min  Max  Range  25th Percentile  50th Percentile
Dawson  10.3    10.0    10    9   11      2             10.0             10.0
Clark   10.3    10.0    10    7   15      8              8.5             10.0

        75th Percentile
Dawson            11.00
Clark             11.75

Measures of Variability:
         Variance  Standard Deviation   IQR  Coefficient of Variation
Dawson  0.455556            0.674949  1.00                  0.065529
Clark   6.677778            2.584140  3.25                  0.250887

Measures of Association:
 Covariance    -0.877778
Correlation   -0.503266
dtype: float64

Measures of Distribution:
         Skewness  Kurtosis
Dawson -0.433637 -0.282995
Clark   0.359289 -0.350865
```

Q2 : Scores turned in by an amateur golfer at the Bonita Fairways Golf Course in Bonita

Springs, Florida, during 2005 and 2006 are as follows:

- **2005 Season: 74 78 79 77 75 73 75 77**
- **2006 Season: 71 70 75 77 85 80 71 79**

```
import pandas as pd


season_2005 = [74, 78, 79, 77, 75, 73, 75, 77]

season_2006 = [71, 70, 75, 77, 85, 80, 71, 79]


df = pd.DataFrame({
    "2005": season_2005,
    "2006": season_2006
})

location_measures = {
    "Mean": df.mean(),
    "Median": df.median(),
    "Mode": df.mode().iloc[0],
    "Min": df.min(),
    "Max": df.max(),
    "Range": df.max() - df.min(),
    "25th Percentile": df.quantile(0.25),
    "50th Percentile": df.quantile(0.50),
    "75th Percentile": df.quantile(0.75)
}
location_df = pd.DataFrame(location_measures)

variability_measures = {
    "Variance": df.var(),
    "Standard Deviation": df.std(),
    "IQR": df.quantile(0.75) - df.quantile(0.25),
    "Coefficient of Variation": df.std() / df.mean()
}
variability_df = pd.DataFrame(variability_measures)

association_measures = {
    "Covariance": df.cov().iloc[0, 1],
    "Correlation": df.corr().iloc[0, 1]
}
association_series = pd.Series(association_measures)
```

```
distribution_measures = {
    "Skewness": df.skew(),
    "Kurtosis": df.kurt()
}
distribution_df = pd.DataFrame(distribution_measures)

print("\nMeasures of Location:\n", location_df)
print("\nMeasures of Variability:\n", variability_df)
print("\nMeasures of Association:\n", association_series)
print("\nMeasures of Distribution:\n", distribution_df)
```

```
Measures of Location:
       Mean  Median  Mode  Min  Max  Range  25th Percentile  50th Percentile  \
2005  76.0    76.0  75.0   73   79      6            74.75             76.0   
2006  76.0    76.0  71.0   70   85     15            71.00             76.0   

      75th Percentile  
2005            77.25  
2006            79.25  

Measures of Variability:
        Variance  Standard Deviation   IQR  Coefficient of Variation
2005   4.285714            2.070197  2.50                  0.027239
2006  27.714286            5.264436  8.25                  0.069269

Measures of Association:
 Covariance    -2.428571
Correlation   -0.222837
dtype: float64

Measures of Distribution:
       Skewness  Kurtosis
2005  0.000000 -1.204000
2006  0.462156 -0.683484
```

Q3 :The following times were recorded by the quarter-mile and mile runners of a university

- **Track team (times are in minutes).**
- **Quarter-Mile Times: .92 .98 1.04 .90 .99**
- **Mile Times: 4.52 4.35 4.60 4.70 4.50**

```
import pandas as pd

quarter_mile = [0.92, 0.98, 1.04, 0.90, 0.99]
mile = [4.52, 4.35, 4.60, 4.70, 4.50]

df = pd.DataFrame({
    "Quarter_Mile": quarter_mile,
    "Mile": mile
})

location_measures = {
    "Mean": df.mean(),
    "Median": df.median(),
    "Mode": df.mode().iloc[0],
    "Min": df.min(),
    "Max": df.max(),
    "Range": df.max() - df.min(),
    "25th Percentile": df.quantile(0.25),
    "50th Percentile": df.quantile(0.50),
    "75th Percentile": df.quantile(0.75)
}
location_df = pd.DataFrame(location_measures)

variability_measures = {
    "Variance": df.var(),
    "Standard Deviation": df.std(),
    "IQR": df.quantile(0.75) - df.quantile(0.25),
    "Coefficient of Variation": df.std() / df.mean()
}
variability_df = pd.DataFrame(variability_measures)

association_measures = {
    "Covariance": df.cov().iloc[0, 1],
    "Correlation": df.corr().iloc[0, 1]
}
association_series = pd.Series(association_measures)

distribution_measures = {
    "Skewness": df.skew(),
    "Kurtosis": df.kurt()
}
distribution_df = pd.DataFrame(distribution_measures)
```

```
print("\nMeasures of Location:\n", location_df)
print("\nMeasures of Variability:\n", variability_df)
print("\nMeasures of Association:\n", association_series)
print("\nMeasures of Distribution:\n", distribution_df)
```

```
Measures of Location:
                Mean  Median  Mode   Min   Max  Range  25th Percentile  \
Quarter_Mile  0.966    0.98  0.90  0.90  1.04   0.14             0.92   
Mile          4.534    4.52  4.35  4.35  4.70   0.35             4.50   

              50th Percentile  75th Percentile  
Quarter_Mile             0.98             0.99  
Mile                     4.52             4.60  

Measures of Variability:
               Variance  Standard Deviation   IQR  Coefficient of Variation
Quarter_Mile   0.00318            0.056391  0.07                  0.058376
Mile           0.01678            0.129538  0.10                  0.028570

Measures of Association:
 Covariance    -0.002205
Correlation   -0.301855
dtype: float64

Measures of Distribution:
               Skewness  Kurtosis
Quarter_Mile  0.085878 -1.348641
Mile         -0.270238  0.549927
```

Q4: Consider a sample with data values of 27, 25, 20, 15, 30, 34, 28

```
import pandas as pd

data = [27, 25, 20, 15, 30, 34, 28]

df = pd.DataFrame({"Sample": data})

location_measures = {
    "Mean": df.mean(),
    "Median": df.median(),
    "Mode": df.mode().iloc[0],
    "Min": df.min(),
    "Max": df.max(),
    "Range": df.max() - df.min(),
    "25th Percentile": df.quantile(0.25),
    "50th Percentile": df.quantile(0.50),
    "75th Percentile": df.quantile(0.75)
}
location_df = pd.DataFrame(location_measures)
```

```
variability_measures = {
    "Variance": df.var(),
    "Standard Deviation": df.std(),
    "IQR": df.quantile(0.75) - df.quantile(0.25),
    "Coefficient of Variation": df.std() / df.mean()
}
variability_df = pd.DataFrame(variability_measures)

distribution_measures = {
    "Skewness": df.skew(),
    "Kurtosis": df.kurt()
}
distribution_df = pd.DataFrame(distribution_measures)

print("\nMeasures of Location:\n", location_df)
print("\nMeasures of Variability:\n", variability_df)
print("\nMeasures of Distribution:\n", distribution_df)
```

```
Measures of Location:
              Mean  Median  Mode  Min  Max  Range  25th Percentile  \
Sample  25.571429    27.0    15   15   34     19             22.5

        50th Percentile  75th Percentile
Sample             27.0             29.0

Measures of Variability:
          Variance  Standard Deviation  IQR  Coefficient of Variation
Sample  40.285714            6.347103  6.5                  0.248211

Measures of Distribution:
         Skewness  Kurtosis
Sample -0.594676  0.041266
```

Q5: A bowler’s scores for six games were 182, 168, 184, 190, 170, and 174

```
import pandas as pd

scores = [182, 168, 184, 190, 170, 174]

df = pd.DataFrame({"Scores": scores})

location_measures = {
    "Mean": df.mean(),
    "Median": df.median(),
    "Mode": df.mode().iloc[0],
    "Min": df.min(),
    "Max": df.max(),
    "Range": df.max() - df.min(),
    "25th Percentile": df.quantile(0.25),
    "50th Percentile": df.quantile(0.50),
    "75th Percentile": df.quantile(0.75)
}

location_df = pd.DataFrame(location_measures)

variability_measures = {
    "Variance": df.var(),
    "Standard Deviation": df.std(),
    "IQR": df.quantile(0.75) - df.quantile(0.25),
    "Coefficient of Variation": df.std() / df.mean()
}
variability_df = pd.DataFrame(variability_measures)

distribution_measures = {
    "Skewness": df.skew(),
    "Kurtosis": df.kurt()
}
distribution_df = pd.DataFrame(distribution_measures)

print("\nMeasures of Location:\n", location_df)
print("\nMeasures of Variability:\n", variability_df)
print("\nMeasures of Distribution:\n", distribution_df)
```

```
Measures of Location:
          Mean  Median  Mode  Min  Max  Range  25th Percentile
Scores  178.0   178.0   168  168  190     22             171.0

        50th Percentile  75th Percentile
Scores            178.0            183.5

Measures of Variability:
         Variance  Standard Deviation   IQR  Coefficient of Variation
Scores      75.2            8.671793  12.5                  0.048718

Measures of Distribution:
         Skewness  Kurtosis
Scores  0.198737 -1.714577
```