Practical No. 1

Aim: - Installation and study of any one Data Analytics Tool Framework.

Step 1: Download Anaconda

1. **Visit the Anaconda Download Page**:
    - Go to the official Anaconda Distribution download page.
2. **Choose Your Operating System**:
    - Select your operating system: Windows, macOS, or Linux.
3. **Download the Installer**:
    - For Windows and macOS: Download the graphical installer.
    - For Linux: You can choose either the graphical installer or the command-line installer. **Step 2: Install Anaconda For Windows:**

1. **Run the Installer**:
    - Locate the .exe file you downloaded and double-click it to run the installer.
2. **Follow the Setup Instructions**:
    - Click "Next" to start the installation process.
    - Read and accept the license agreement.
    - Choose whether to install for "Just Me" or "All Users" (the default is "Just Me").
    - Select the installation location (the default is usually fine).
3. **Advanced Installation Options**:
    - **Add Anaconda to my PATH environment variable**: It's recommended to check this box so you can use Anaconda from the command line.
    - **Register Anaconda as my default Python**: It's recommended to check this option unless you have a specific reason not to.
4. **Finish the Installation**:
    - Click "Install" and wait for the installation to complete.
    - Once finished, click "Next" and then "Finish".

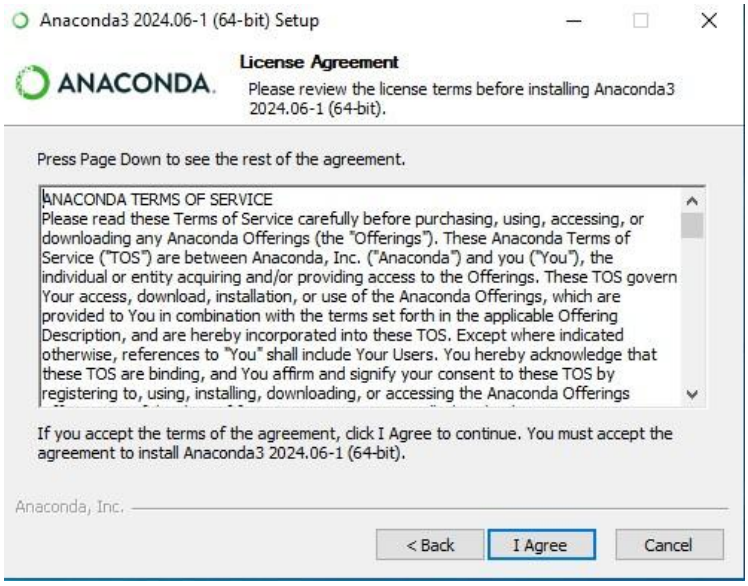
Step 3: Verify the Installation

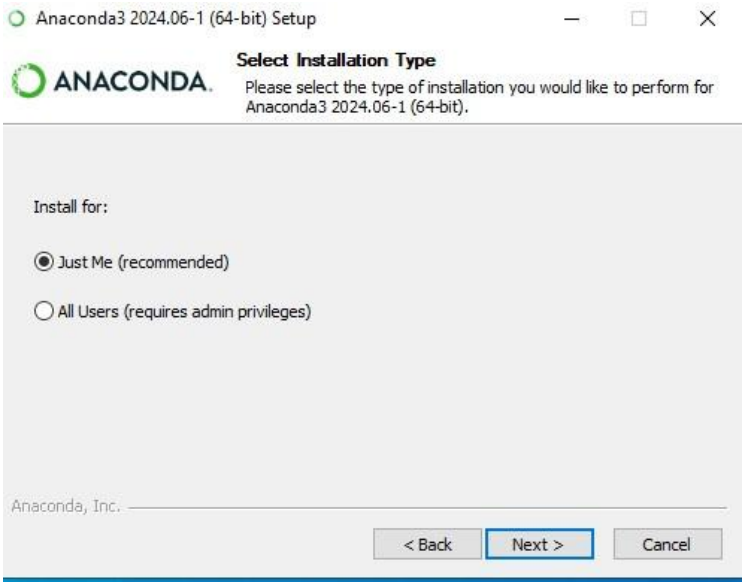
1. **Open the Anaconda Navigator**:

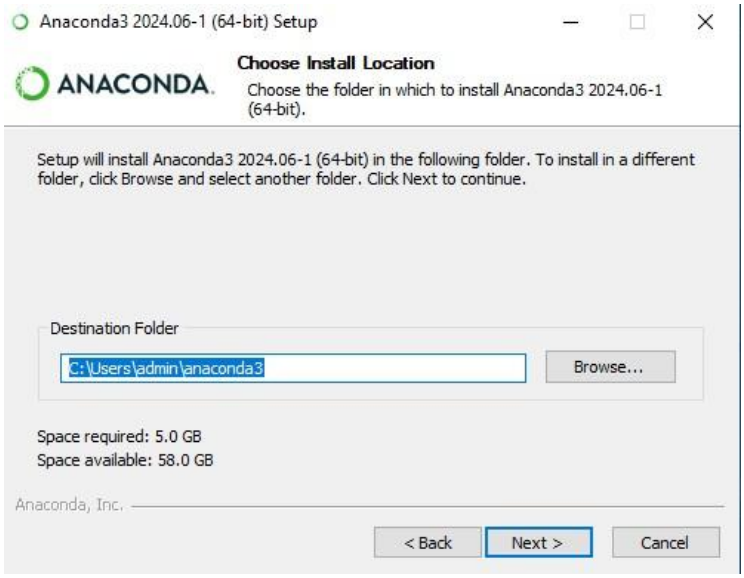
    - On Windows or macOS, search for "Anaconda Navigator" in your applications and launch it. **Step 4: Start Using Anaconda**

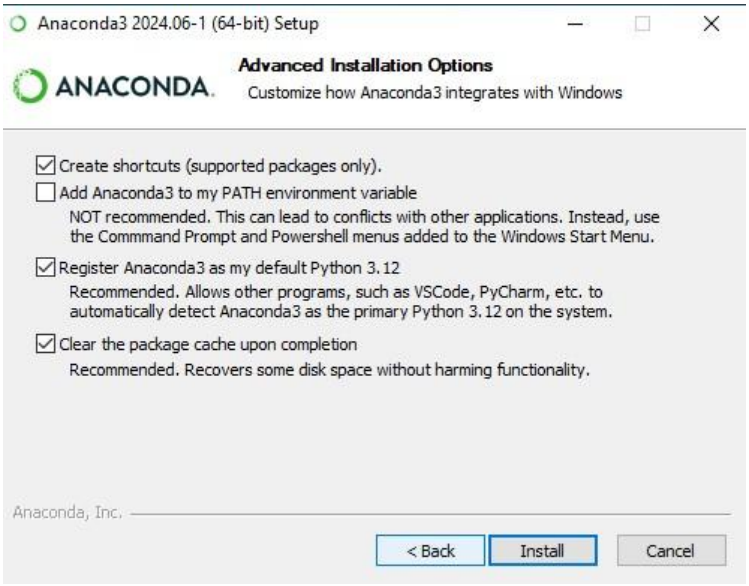
• **Jupyter Notebook**: A web-based interactive computing platform. You can launch it from Anaconda Navigator or by typing jupyter notebook in your terminal. **Output: -**

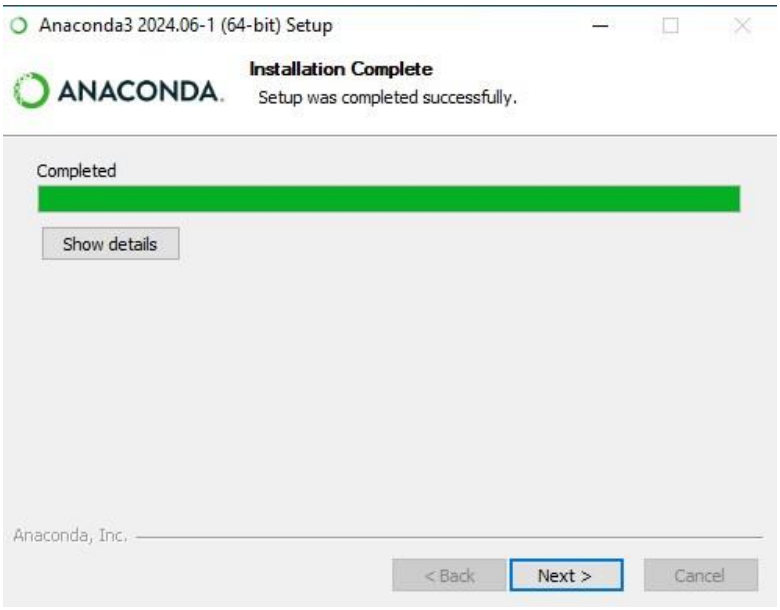

Anaconda3 2024.06-1 (64-bit) Setup
ANACONDA.
Welcome to Anaconda3 2024.06-1 (64-bit) Setup
Setup will guide you through the installation of Anaconda3 2024.06-1 (64-bit).
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
Next >
Cancel

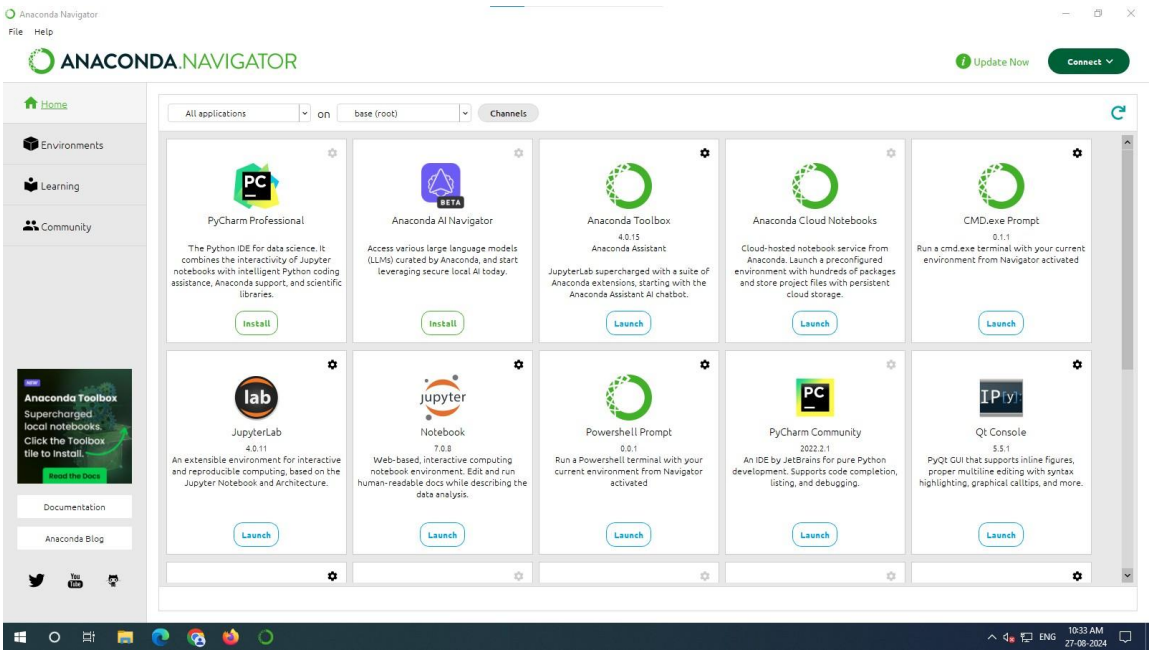

Anaconda3 2024.06-1 (64-bit) Setup
ANACONDA.
License Agreement
Please review the license terms before installing Anaconda3 2024.06-1 (64-bit).
Press Page Down to see the rest of the agreement.
ANACONDA TERMS OF SERVICE
Please read these Terms of Service carefully before purchasing, using, accessing, or downloading any Anaconda Offerings (the "Offerings"). These Anaconda Terms of Service ("TOS") are between Anaconda, Inc. ("Anaconda") and you ("You"), the individual or entity acquiring and/or providing access to the Offerings. These TOS govern Your access, download, installation, or use of the Anaconda Offerings, which are provided to You in combination with the terms set forth in the applicable Offering Description, and are hereby incorporated into these TOS. Except where indicated otherwise, references to "You" shall include Your Users. You hereby acknowledge that these TOS are binding, and You affirm and signify your consent to these TOS by registering to, using, installing, downloading, or accessing the Anaconda Offerings
If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Anaconda3 2024.06-1 (64-bit).
Anaconda, Inc.
< Back
I Agree
Cancel

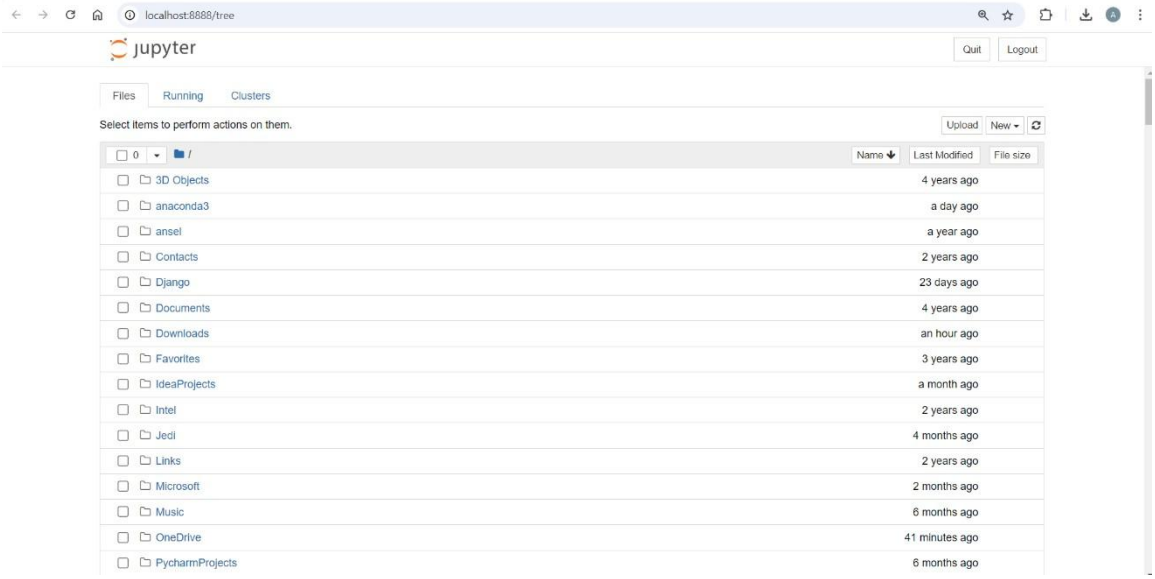

Anaconda3 2024.06-1 (64-bit) Setup
ANACONDA.
Select Installation Type
Please select the type of installation you would like to perform for Anaconda3 2024.06-1 (64-bit).
Install for:
Just Me (recommended)
All Users (requires admin privileges)
Anaconda, Inc.
< Back
Next >
Cancel


Anaconda3 2024.06-1 (64-bit) Setup
ANACONDA.
Choose Install Location
Choose the folder in which to install Anaconda3 2024.06-1 (64-bit).
Setup will install Anaconda3 2024.06-1 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.
Destination Folder
C:\Users\admin\anaconda3
Browse...
Space required: 5.0 GB
Space available: 58.0 GB
Anaconda, Inc.
< Back
Next >
Cancel


Anaconda3 2024.06-1 (64-bit) Setup
ANACONDA.
Advanced Installation Options
Customize how Anaconda3 integrates with Windows
Create shortcuts (supported packages only).
Add Anaconda3 to my PATH environment variable
NOT recommended. This can lead to conflicts with other applications. Instead, use the Commmand Prompt and Powershell menus added to the Windows Start Menu.
Register Anaconda3 as my default Python 3.12
Recommended. Allows other programs, such as VSCode, PyCharm, etc. to automatically detect Anaconda3 as the primary Python 3.12 on the system.
Clear the package cache upon completion
Recommended. Recovers some disk space without harming functionality.
Anaconda, Inc.
< Back
Install
Cancel


Anaconda3 2024.06-1 (64-bit) Setup
ANACONDA.
Installation Complete
Setup was completed successfully.
Completed
Show details
Anaconda, Inc.
< Back
Next >
Cancel


Anaconda Navigator
File Help
ANACONDA.NAVIGATOR
Update Now
Connect
Home
Environments
Learning
Community
All applications
on
base (root)
Channels
PyCharm Professional
The Python IDE for data science. It combines the interactivity of Jupyter notebooks with intelligent Python coding assistance, Anaconda support, and scientific libraries.
Install
Anaconda AI Navigator
BETA
Access various large language models (LLMs) curated by Anaconda, and start leveraging secure local AI today.
Install
Anaconda Toolbox
4.0.15
Anaconda Assistant
JupyterLab supercharged with a suite of Anaconda extensions, starting with the Anaconda Assistant AI chatbot.
Launch
Anaconda Cloud Notebooks
Cloud-hosted notebook service from Anaconda. Launch a preconfigured environment with hundreds of packages and store project files with persistent cloud storage.
Launch
CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated
Launch
Anaconda Toolbox
Supercharged local notebooks. Click the Toolbox tile to install.
Read the Docs
Documentation
Anaconda Blog
JupyterLab
4.0.11
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch
Notebook
7.0.8
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch
Powershell Prompt
0.0.1
Run a Powershell terminal with your current environment from Navigator activated
Launch
PyCharm Community
2022.2.1
An IDE by JetBrains for pure Python development. Supports code completion, listing, and debugging.
Launch
Qt Console
5.5.1
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Launch
ENG
10:33 AM
27-08-2024


localhost:8888/tree
jupyter
Quit
Logout
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
Name
Last Modified
File size
3D Objects 4 years ago
anaconda3 a day ago
ansel a year ago
Contacts 2 years ago
Django 23 days ago
Documents 4 years ago
Downloads an hour ago
Favorites 3 years ago
IdeaProjects a month ago
Intel 2 years ago
Jedi 4 months ago
Links 2 years ago
Microsoft 2 months ago
Music 6 months ago
OneDrive 41 minutes ago
PycharmProjects 6 months ago

```
import numpy as np

arr1 = np.array([1, 2, 3, 4]) arr2
= np.array([[1, 2], [3, 4]])

# Printing the arrays
print("Array 1:", arr1)
print("Array 2:\n", arr2)

# Adding 10 to each element of arr1
arr3 = arr1 + 10 print("Array 1 + 10:",
arr3)

# Calculating the mean of Array 1 print("Mean
of Array 1:", np.mean(arr1))

# Calculating the sum of Array 2 print("Sum
of Array 2:", np.sum(arr2))

# Reshaping Array 1 into a 2x2 matrix
reshaped = arr1.reshape(2, 2)
print("Reshaped Array 1:\n", reshaped)
```

```
Array 1: [1 2 3 4]
2:
 [[1 2]
 [3 4]]
Array 1 + 10: [11 12 13 14]
Mean of Array 1: 2.5
of Array 2: 10 Res
Array 1:
```

```
# Array Data structure import numpy as np

my_array = np.array([1, 2, 3, 4, 5])

print(my_array)  # Printing the NumPy array


# String Data structure string1 = "hello abc"

string2 = "welcome to numpy"

print(string1, string2)  # Printing the strings


# List Data structure my_list =

[10, 20, 30, 40] print(my_list)  #

Printing the list


# Tuple Data structure my_tuple =

(10, 20, 30, 40) print(my_tuple)  #

Printing the tuple


# Dictionary Data structure my_dict

= {

   'name': 'Alice',

   'age': 30,

   'city': 'New York'

}

print(my_dict)  # Printing the dictionary
```

```
[1 2 3 4 5]
hello abc welcome to numpy
[10, 20, 30, 40]
(10, 20, 30, 40)
{'name': 'Alice', 'age': 30, 'city': 'New York'}
```

Practical No. 4

Aim: - Design and develop at least 5 problem statements which demonstrate the use of Control Structure, Importing/Exporting Data in any data analytics tool.

Source code: - import

pandas as pd

```
data = {'Region': ['North', 'South', 'East', 'West', 'North', 'South', 'East', 'West'],
        'Sales': [100, 200, 300, 400, 500, 600, 700, 800],
        'Age': [25, 30, 35, 40, 45, 50, 55, 60],
'Purchase_History': [1, 0, 1, 1, 0, 1, 0, 1]}
df = pd.DataFrame(data)

# Problem Statement 1: Using If-Else Statements to Handle Missing Values
df['Sales'].fillna(df['Sales'].mean(), inplace=True)

# Problem Statement 2: Using For Loops to Iterate over Rows
total_sales = {} for index, row in df.iterrows():
   region = row['Region']     sales
=  row['Sales']          if  region  in
total_sales:
total_sales[region]     +=     sales
else:
        total_sales[region] = sales print("Total
Sales by Region:")
print(total_sales)

# Problem Statement 3: Using While Loops to Find the First Occurrence
target_value = 500 found = False
i = 0 while i < len(df) and not
found:
   if df.loc[i, 'Sales'] == target_value:
      found = True
      print(f"First occurrence of {target_value} found at row {i+1}")
i += 1

# Problem Statement 4: Using Conditional Statements to Filter Data threshold
= 400
filtered_df = df[df['Sales'] > threshold]
print("Filtered Data:") print(filtered_df)

# Problem Statement 5: Using Nested If-Else Statements to Categorize Data
categories = [] for index, row in df.iterrows():
```



```
   age = row['Age']
```

```
    purchase_history = row['Purchase_History']
if age < 40:            if purchase_history == 1:
category = "Young and Active"        else:
        category = "Young and Inactive"
else:       if purchase_history == 1:
        category = "Old and Active"
else:
        category = "Old and Inactive"
categories.append(category) df['Category'] =
categories print("Categorized Data:")
print(df)
```

Output: -

```
Total Sales by Region:
{'North': 600, 'South': 800, 'East': 1000, 'West': 1200}
First occurrence of 500 found at row 5
Filtered Data:
  Region  Sales  Age  Purchase_History
4  North    500   45                 0
5  South    600   50                 1
6   East    700   55                 0
7   West    800   60                 1
Categorized Data:
  Region  Sales  Age  Purchase_History            Category
0  North    100   25                 1   Young and Active
1  South    200   30                 0 Young and Inactive
2   East    300   35                 1   Young and Active
3   West    400   40                 1     Old and Active
4  North    500   45                 0   Old and Inactive
5  South    600   50                 1     Old and Active
6   East    700   55                 0   Old and Inactive
7   West    800   60                 1     Old and Active
```

Practical No. 5

Aim: - Implement KNN Classification techniques using any data analytics tool.

Source code: - import numpy as np

```
import pandas as pd      from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import
KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

iris = datasets.load_iris() X
= iris.data  # Features
y = iris.target  # Labels

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test) k = 3  #
Number of neighbors knn =
KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, y_train) y_pred
= knn.predict(X_test)

accuracy = accuracy_score(y_test, y_pred) conf_matrix
= confusion_matrix(y_test, y_pred) class_report =
classification_report(y_test, y_pred)

print("Accuracy:", accuracy)
print("\nConfusion Matrix:\n", conf_matrix) print("\nClassification
Report:\n", class_report)
```

Output: -

```
KNeighborsClassifier(n_neighbors=3)
Accuracy: 1.0

Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

Practical No. 6

Aim: - Implement Naive Bayes Classification technique using any Data Analytics tool

Source code: -

```
import numpy as np     import pandas as pd
from    sklearn.datasets    import    load_iris    from
sklearn.model_selection import train_test_split from
sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

iris = load_iris() X =
iris.data  # Features
y = iris.target  # Target variable

df = pd.DataFrame(data=np.c_[iris['data'], iris['target']],
columns=iris['feature_names'] + ['target'])

print("Iris Dataset:")
print(df.head())

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = GaussianNB()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred) conf_matrix
= confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print("\nModel Accuracy:", accuracy) print("\nConfusion
Matrix:\n", conf_matrix) print("\nClassification
Report:\n", class_report)
```

Output-

```
Iris Dataset:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   target
0     0.0
1     0.0
2     0.0
3     0.0
4     0.0

Model Accuracy: 1.0

Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]

Classification Report:
              precision    recall  f1-score   support
...
    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

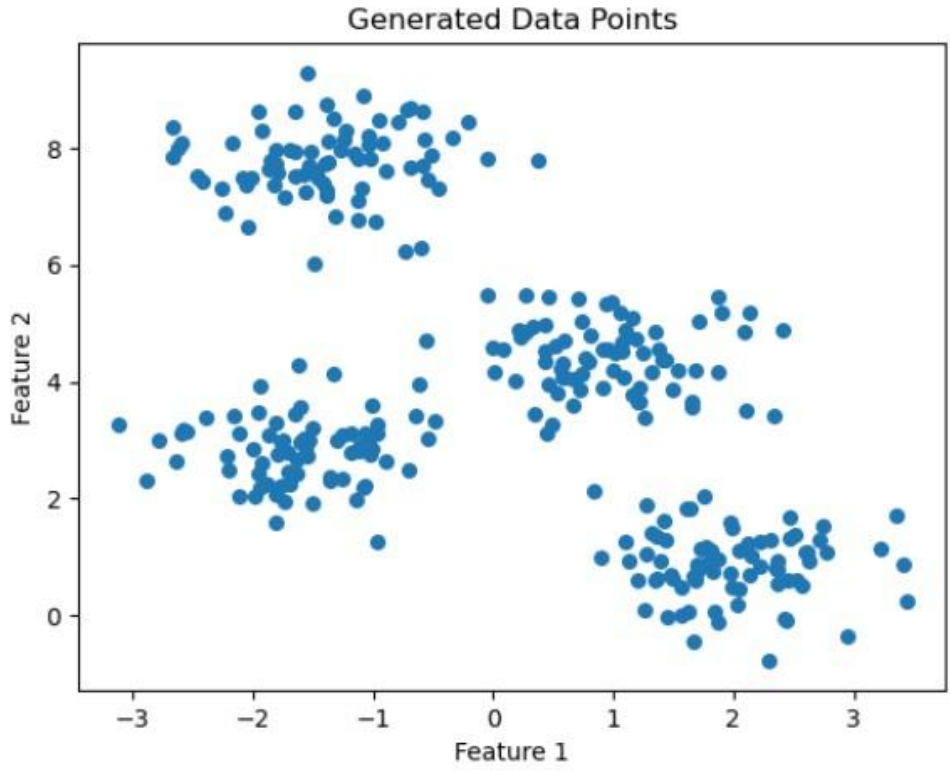
Practical No. 7

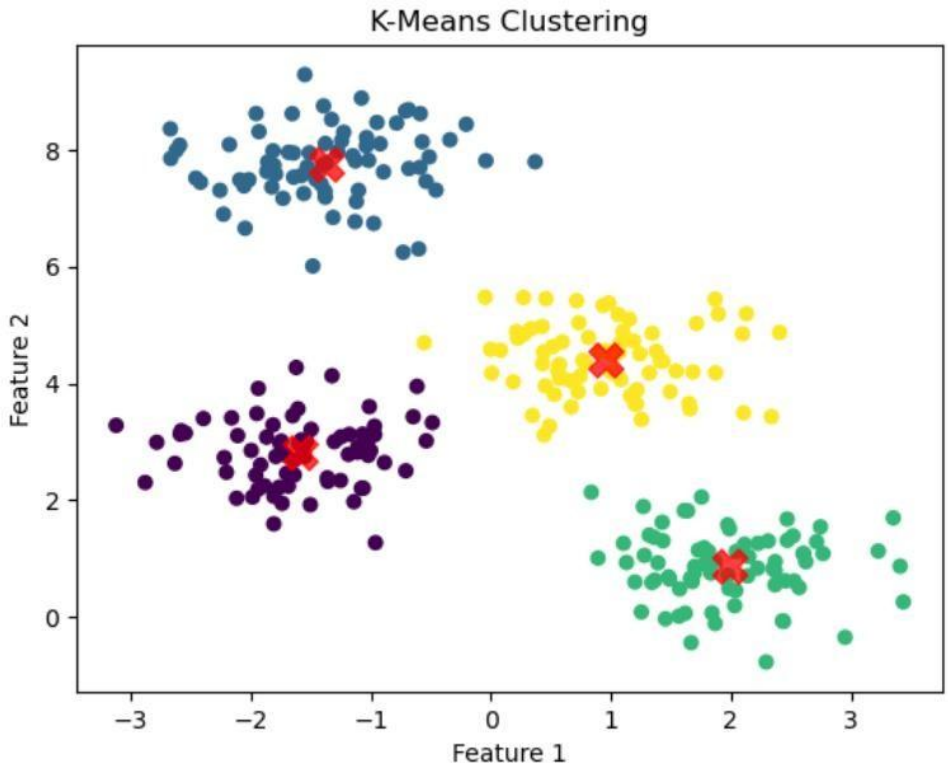
Aim: - Implement K means clustering technique using any Data Analytics tool.

Source code: -

```
import numpy as np import pandas as pd
import matplotlib.pyplot as plt from
sklearn.cluster import KMeans from
sklearn.datasets import make_blobs
import warnings

# Generate synthetic data
n_samples = 300
n_clusters = 4
X, y = make_blobs(n_samples=n_samples, centers=n_clusters, cluster_std=0.60,
random_state=0)

# Convert to DataFrame for easier analysis
data = pd.DataFrame(X, columns=['Feature1', 'Feature2'])

# Plot the synthetic data
plt.scatter(data['Feature1'], data['Feature2'],
s=30) plt.title('Generated Data Points')
plt.xlabel('Feature 1') plt.ylabel('Feature 2')
plt.show()

# Implement K-Means clustering kmeans
= KMeans(n_clusters=n_clusters)
kmeans.fit(X)

# Get cluster centroids and labels
centroids = kmeans.cluster_centers_
labels = kmeans.labels_

# Plotting the clustered data plt.scatter(data['Feature1'], data['Feature2'],
c=labels, s=30, cmap='viridis') plt.scatter(centroids[:, 0], centroids[:, 1], c='red',
s=200, alpha=0.75, marker='X') plt.title('K-Means Clustering')
plt.xlabel('Feature 1') plt.ylabel('Feature 2')
plt.show()
```

Output: -





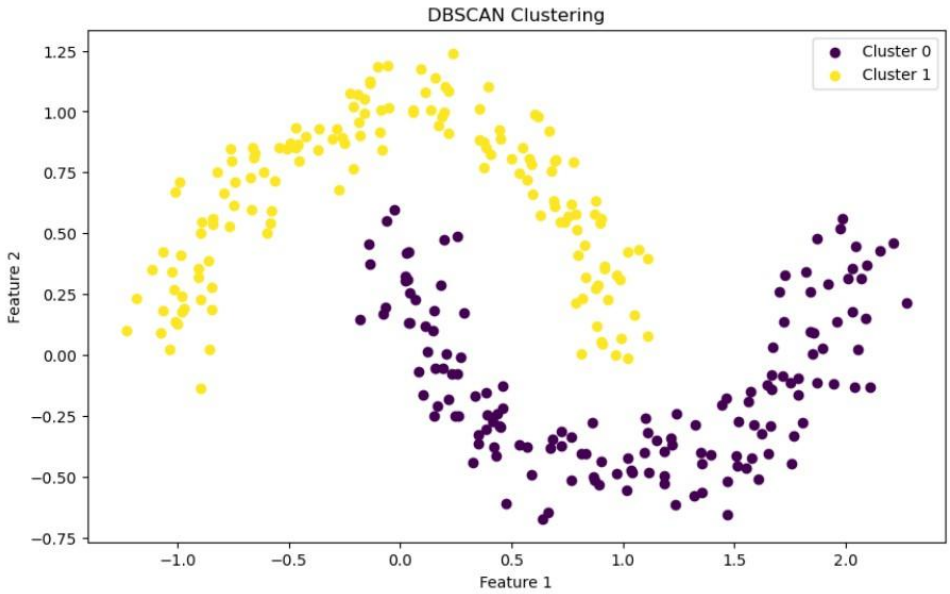
Practical No. 8

Aim: - Implement DBScan clustering technique using any Data Analytics tool

Source code: - import numpy as np import matplotlib.pyplot as plt from

```
sklearn.datasets import make_moons from sklearn.cluster import DBSCAN
import warnings

X, _ = make_moons(n_samples=300, noise=0.1, random_state=42)

dbscan = DBSCAN(eps=0.2, min_samples=5)
labels = dbscan.fit_predict(X) unique_labels
= set(labels)
colors = plt.cm.get_cmap('viridis', len(unique_labels))

plt.figure(figsize=(10,  6))
for label in unique_labels:
if label == -1:         color =
'k'     else:
        color = colors(label)

    plt.scatter(X[labels == label, 0], X[labels == label, 1], color=color, label=f'Cluster {label}'
if label != -1 else 'Noise')

plt.title('DBSCAN Clustering')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2') plt.legend()
plt.show()
```

Output: -


DBSCAN Clustering
Cluster 0
Cluster 1
Feature 2
Feature 1
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

Practical No. 9

Aim: - Implement Eclat Association Rule Mining technique using Data Analytics tool

Source code: -

```
import numpy
as np import matplotlib.pyplot
as plt import pandas as pd
from mlxtend.frequent_patterns import apriori,association_rules from
mlxtend.preprocessing import TransactionEncoder
df=pd.read_csv("/content/basket_analysis.csv",header=None)
#shape
df.shape
#head of data
df.head()

!pip install pyECLAT

from pyECLAT import ECLAT
eclat_instance = ECLAT(data=df, verbose=True)

eclat_instance.df_bin
eclat_instance.uniq_

get_ECLAT_indexes, get_ECLAT_supports =
eclat_instance.fit(min_support=0.08,min_combination=1,max_combination=3,separator=' &
',verbose=True)

get_ECLAT_supports
```

Output: -

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically
  and should_run_async(code)
Collecting pyECLAT
  Downloading pyECLAT-1.0.2-py3-none-any.whl.metadata (4.0 kB)
Requirement already satisfied: pandas>=0.25.3 in /usr/local/lib/python3.10/dist-packages (from pyECLAT) (2.2.2)
Requirement already satisfied: numpy>=1.17.4 in /usr/local/lib/python3.10/dist-packages (from pyECLAT) (1.26.4)
Requirement already satisfied: tqdm>=4.41.1 in /usr/local/lib/python3.10/dist-packages (from pyECLAT) (4.66.5)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.25.3->pyECLAT) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.25.3->pyECLAT) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.25.3->pyECLAT) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas>=0.25.3->pyECLAT) (1.16.0)
Downloading pyECLAT-1.0.2-py3-none-any.whl (6.3 kB)
Installing collected packages: pyECLAT
Successfully installed pyECLAT-1.0.2
100%|          | 1018/1018 [00:50<00:00, 20.23it/s]
100%|          | 1018/1018 [00:00<00:00, 17647.96it/s]
100%|          | 1018/1018 [00:00<00:00, 2029.26it/s]
Combination 1 by 1
2it [00:00,  2.06it/s]
Combination 2 by 2
1it [00:00,  1.52it/s]
Combination 3 by 3
0it [00:00, ?it/s]
{'False': 0.999, 'True': 0.999, 'False & True': 0.999}
```

Practical No. 10

Aim: - Implement Apriori Association Rule Mining technique using Data Analytics tool

Source code: -

```
pip install mlxtend pandas
import pandas as pd from mlxtend.frequent_patterns import
apriori, association_rules

data = {
   'TransactionID': [1, 2, 3, 4, 5, 6],
   'Items': [
      ['Milk', 'Bread', 'Diaper'],
      ['Bread', 'Diaper', 'Beer'],
      ['Milk', 'Diaper', 'Beer', 'Cola'],
      ['Milk', 'Bread'],
      ['Bread', 'Diaper', 'Cola'],
      ['Beer', 'Diaper']
   ]
}

df = pd.DataFrame(data)

from mlxtend.preprocessing import TransactionEncoder

te = TransactionEncoder()
te_ary = te.fit(df['Items']).transform(df['Items']) df_encoded
= pd.DataFrame(te_ary, columns=te.columns_)

frequent_itemsets = apriori(df_encoded, min_support=0.3, use_colnames=True)
print("Frequent Itemsets:") print(frequent_itemsets)

rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.5)
print("\nAssociation Rules:") print(rules)
```

Output: -

```
Frequent Itemsets:
    support         itemsets
0  0.500000          (Beer)
1  0.666667         (Bread)
2  0.333333          (Cola)
3  0.833333        (Diaper)
4  0.500000          (Milk)
5  0.500000   (Beer, Diaper)
6  0.500000  (Bread, Diaper)
7  0.333333    (Bread, Milk)
8  0.333333   (Diaper, Cola)
9  0.333333   (Diaper, Milk)

Association Rules:
  antecedents consequents  antecedent support  consequent support   support  \
0      (Beer)    (Diaper)            0.500000            0.833333  0.500000
1    (Diaper)      (Beer)            0.833333            0.500000  0.500000
2     (Bread)    (Diaper)            0.666667            0.833333  0.500000
3    (Diaper)     (Bread)            0.833333            0.666667  0.500000
4     (Bread)      (Milk)            0.666667            0.500000  0.333333
5      (Milk)     (Bread)            0.500000            0.666667  0.333333
6      (Cola)    (Diaper)            0.333333            0.833333  0.333333
7      (Milk)    (Diaper)            0.500000            0.833333  0.333333

   confidence  lift  leverage  conviction  zhangs_metric
...
4    0.500000   1.0  0.000000    1.000000       0.000000
5    0.666667   1.0  0.000000    1.000000       0.000000
6    1.000000   1.2  0.055556         inf       0.250000
7    0.666667   0.8 -0.083333    0.500000      -0.333333
```

Practical No. 11

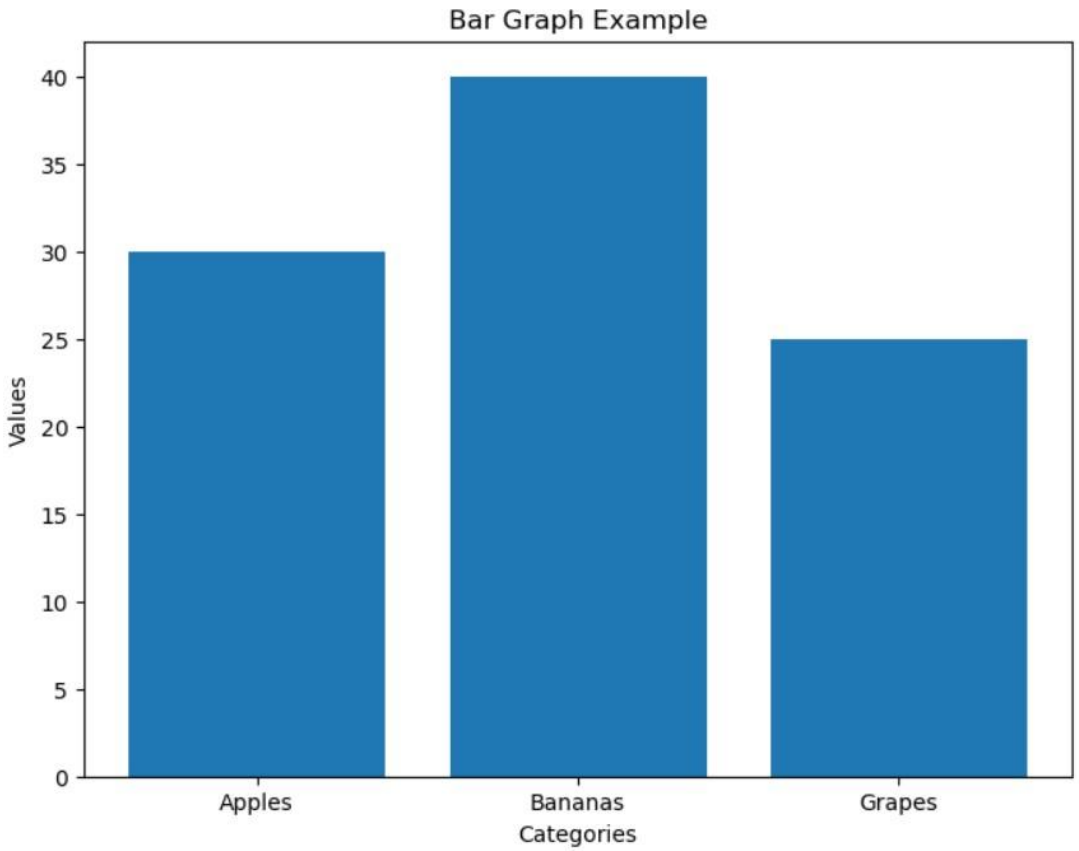
Aim: - Visualise all the statistical measures (mean, mode, median, range, inter quartile range etc.) using Histogram, Boxplots, scatter plots, etc.

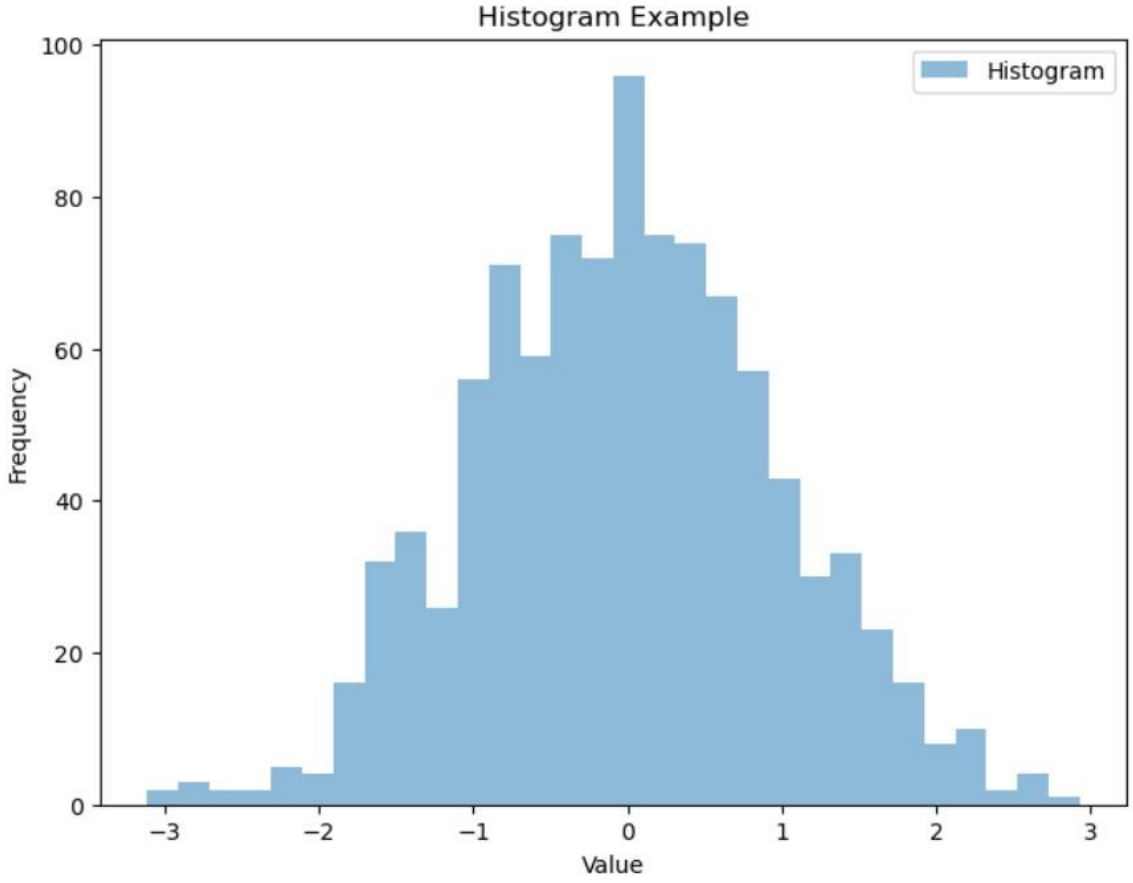
Source code: - import
matplotlib.pyplot as plt import
numpy as np

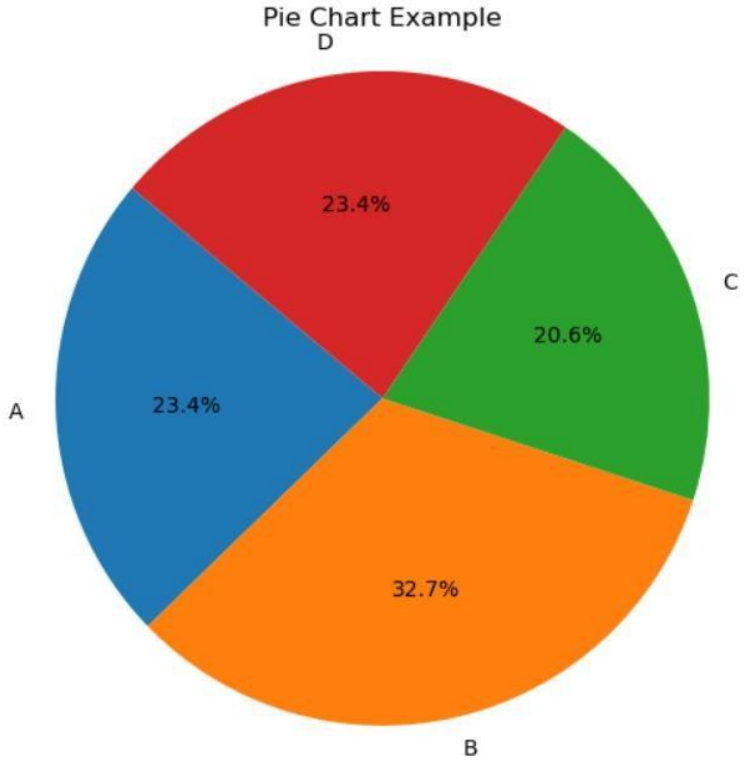
```
# Bar Graph Example
categories = ['Apples', 'Bananas', 'Grapes'] values
= [30, 40, 25]

plt.figure(figsize=(8, 6))
plt.bar(categories, values)
plt.xlabel('Categories')
plt.ylabel('Values') plt.title('Bar
Graph Example') plt.show()

# Histogram Example
data = np.random.randn(1000)

plt.figure(figsize=(8, 6))
plt.hist(data, bins=30, alpha=0.5, label='Histogram')
plt.xlabel('Value') plt.ylabel('Frequency')
plt.title('Histogram Example') plt.legend()
plt.show()

# Pie Chart Example sizes
= [25, 35, 22, 25]
labels = ['A', 'B', 'C', 'D']

plt.figure(figsize=(8, 6))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
plt.axis('equal') plt.title('Pie Chart Example') plt.show()

# Line Chart Example x
= [1, 2, 3, 4, 5]
y = [10, 15, 7, 10, 5]

plt.figure(figsize=(8, 6)) plt.plot(x,
y,  marker='o')  plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Line Chart Example')
```

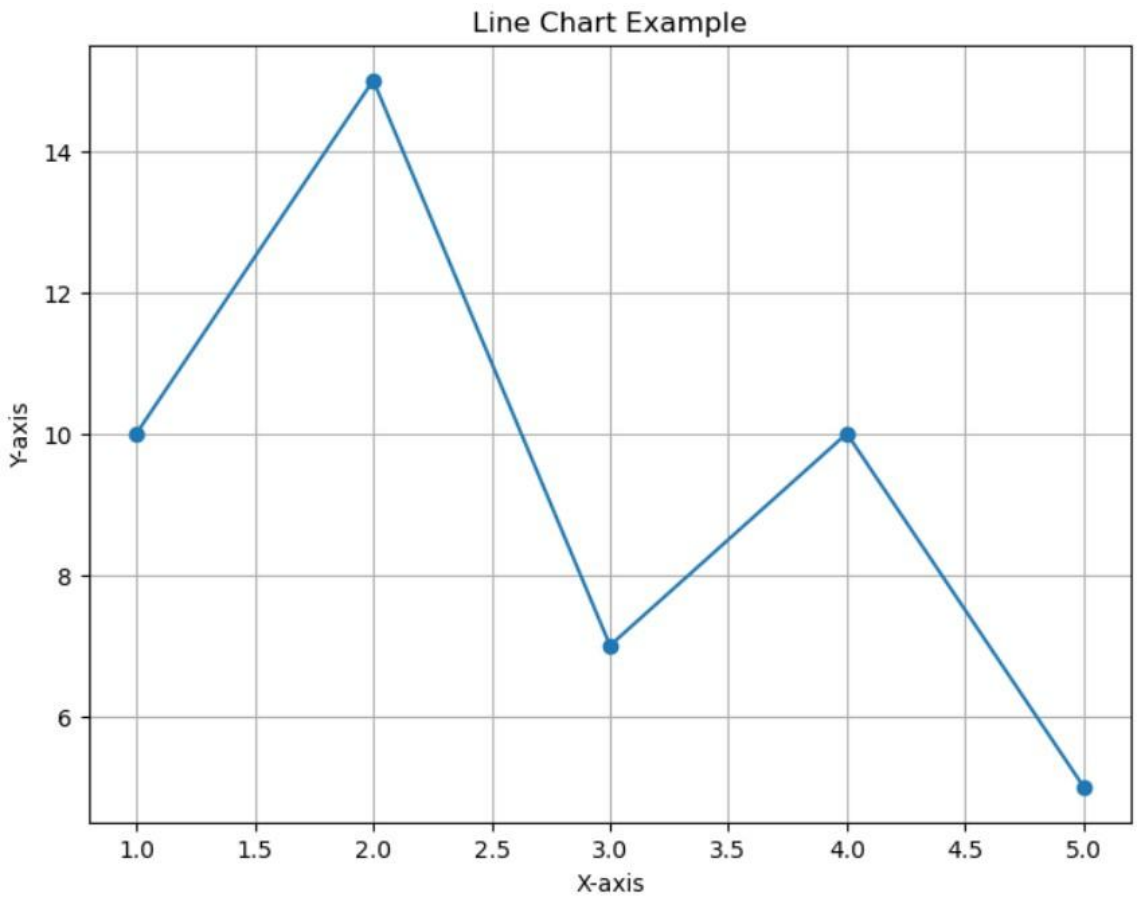
```
plt.grid(True) plt.show()

# Scatter Plot Example x
= np.random.rand(50) y
= np.random.rand(50)
sizes = np.random.rand(50) * 100

plt.figure(figsize=(8, 6))
plt.scatter(x, y, s=sizes, alpha=0.5)
plt.xlabel('X-axis') plt.ylabel('Y-
axis') plt.title('Scatter Plot
Example')
plt.show()
```

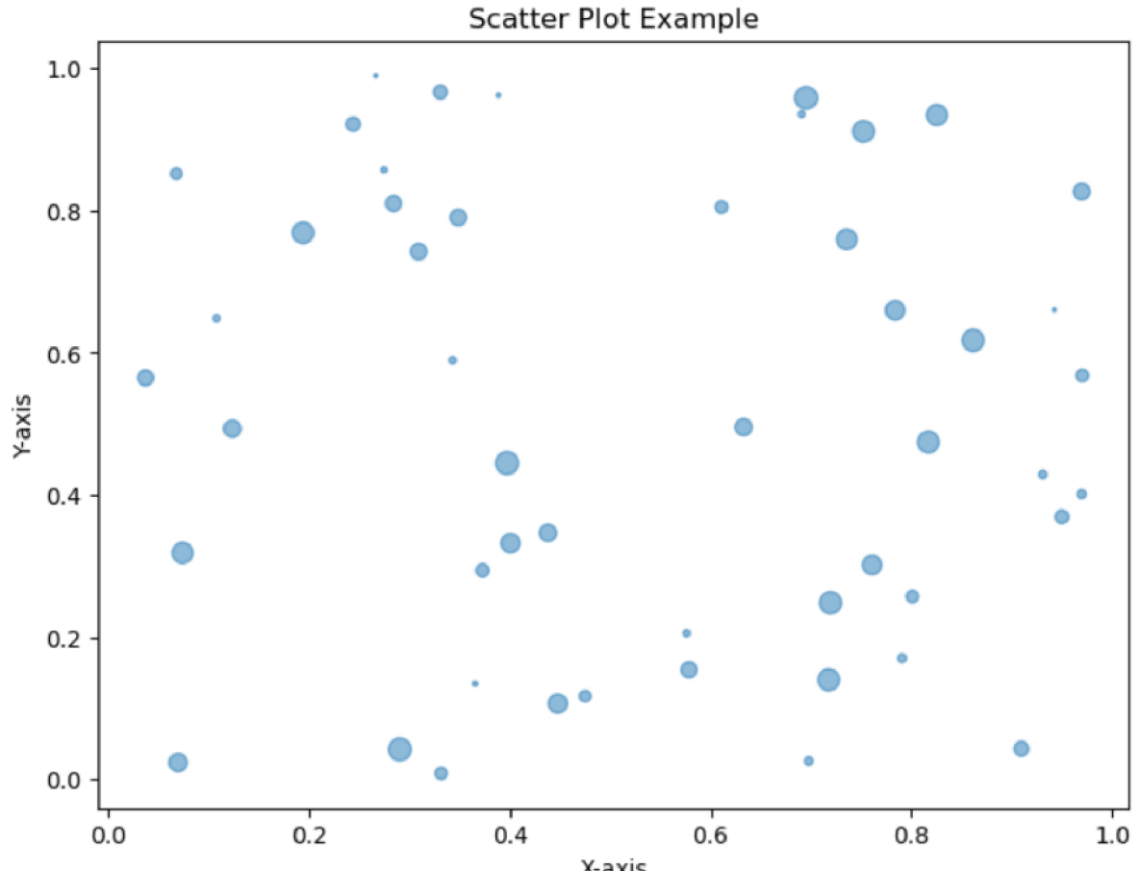
Output: -




Histogram Example
Histogram
Frequency
Value
0
20
40
60
80
100
−3
−2
−1
0
1
2
3


Pie Chart Example
D
C
A
B
23.4%
20.6%
23.4%
32.7%


Line Chart Example
Y-axis
X-axis
14
12
10
8
6
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Scatter Plot Example
Y-axis
X-axis
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0