EXPERIMENT 1A

Aim:
To analyze the trend of data science job postings over the last decade.

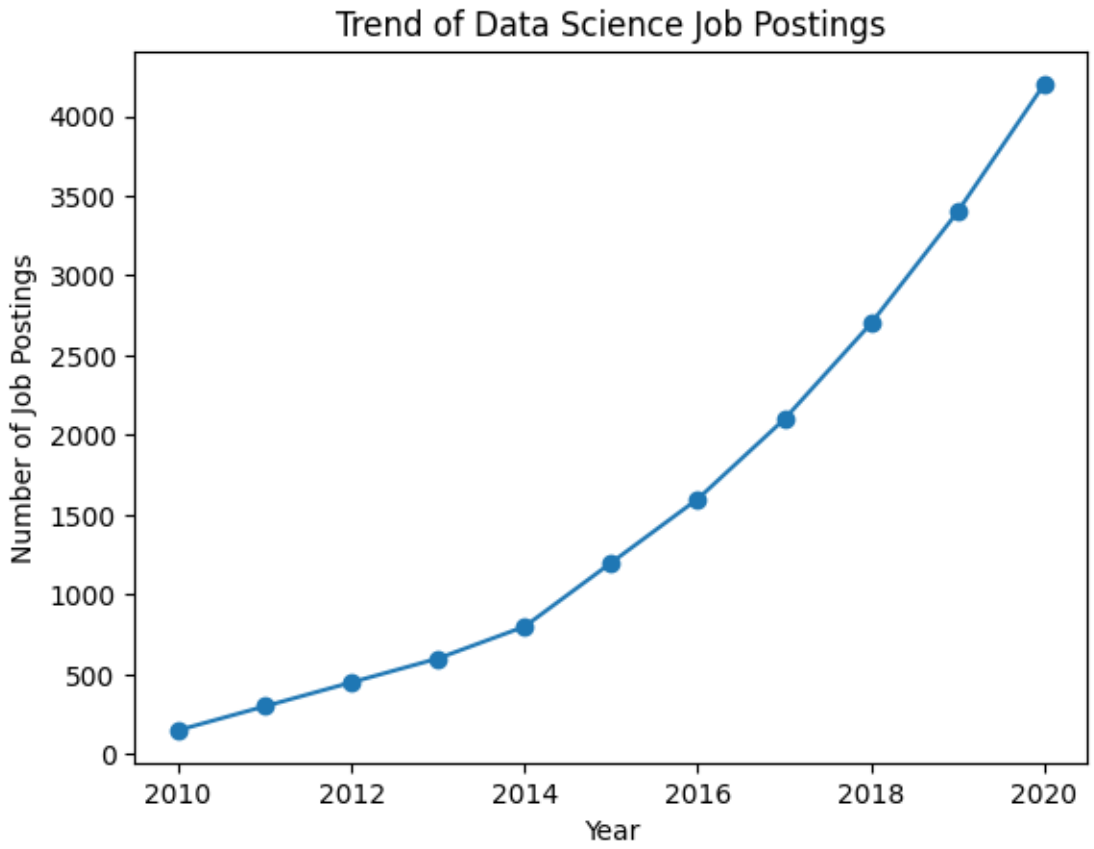
Algorithm:

1. Import pandas and matplotlib libraries.
2. Create a dataset with years and corresponding job postings.
3. Convert the dataset into a DataFrame using pandas.
4. Plot the data using matplotlib with years on the x-axis and job postings on the y-axis.
5. Label the axes and add a title to the graph.
6. Display the plotted graph.

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
data = {'Year': list(range(2010, 2021)),
    'Job Postings': [150, 300, 450, 600, 800, 1200, 1600, 2100, 2700, 3400, 4200]}
df = pd.DataFrame(data)
plt.plot(df['Year'], df['Job Postings'], marker='o')
plt.title('Trend of Data Science Job Postings')
plt.xlabel('Year')
plt.ylabel('Number of Job Postings')
plt.show()
```

Output:



Result:
Thus, the Python program to analyze the trend of data science job postings over the last decade was executed successfully, and the output was verified.

EXPERIMENT 1B

Aim:
To analyze and visualize the distribution of various data science roles such as Data Analyst, Data Engineer, Data Scientist, ML Engineer, and BI Developer from a dataset.

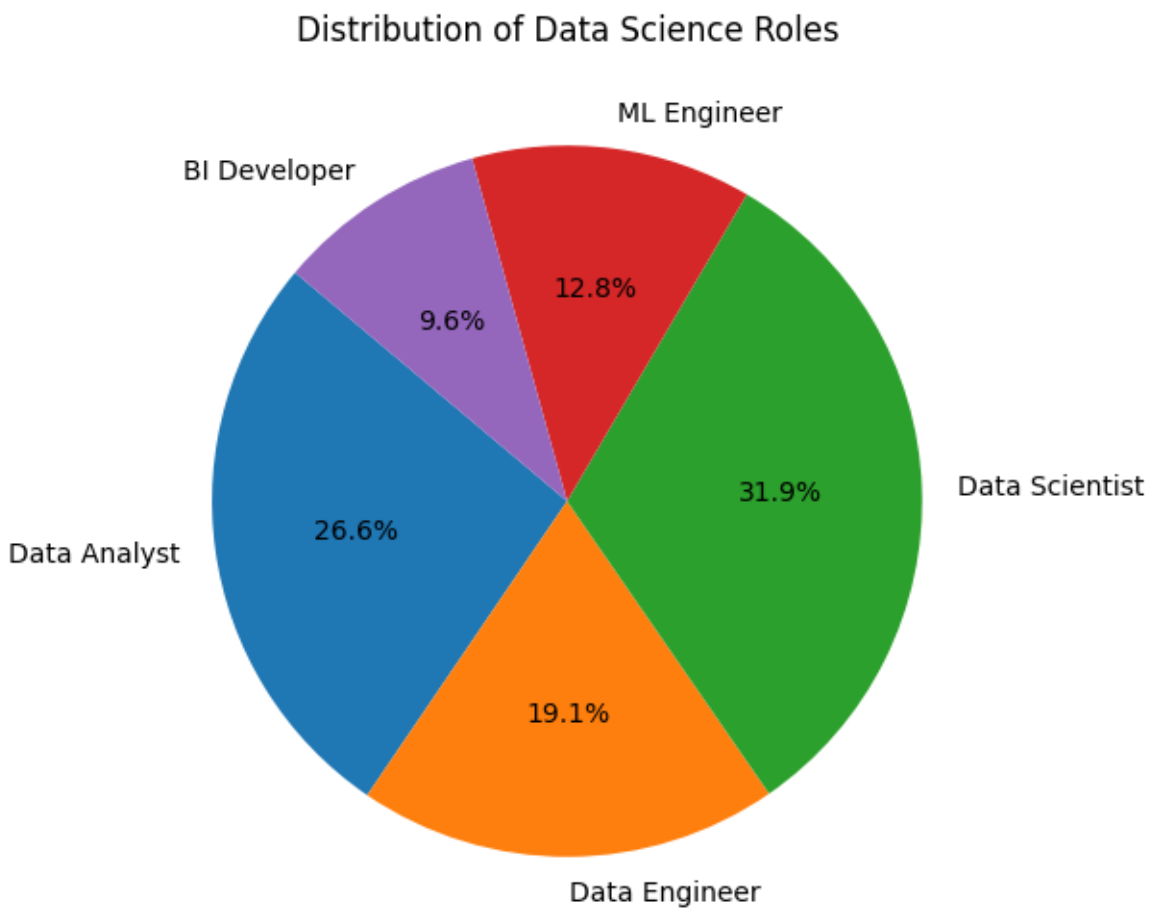
Algorithm:

1. Import pandas and matplotlib libraries.
2. Create a dataset containing different data science roles and their corresponding job postings.
3. Convert the dataset into a DataFrame using pandas.
4. Display the dataset to verify the data.
5. Plot a pie chart using matplotlib to visualize the distribution of roles.
6. Add appropriate labels, title, and display the chart.

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
data = {
   "Role": ["Data Analyst", "Data Engineer", "Data Scientist", "ML Engineer", "BI Developer"],
   "Job Postings": [2500, 1800, 3000, 1200, 900]
}
df = pd.DataFrame(data)
print(df)
plt.figure(figsize=(6, 6))
plt.pie(df["Job Postings"], labels=df["Role"], autopct="%1.1f%%", startangle=140)
plt.title("Distribution of Data Science Roles")
plt.show()
```

Output:



Result:
Thus, the Python program to analyze and visualize the distribution of various data science roles was executed successfully, and the output was verified.

EXPERIMENT 1C

Aim:
To conduct an experiment to differentiate between Structured, Unstructured, and Semi-structured data using Python.

Algorithm:

1. Import the pandas library.
2. Create structured data in tabular form using a pandas DataFrame.
3. Create unstructured data in the form of free text or media file descriptions.
4. Create semi-structured data in JSON-like format.
5. Display all three types of data to observe the differences.

Code:

```
import pandas as pd
structured_data = pd.DataFrame({
  "ID": [1, 2, 3],
  "Name": ["Alice", "Bob", "Charlie"],
  "Age": [25, 30, 35]
})
unstructured_data = [
  "Alice loves data science.",
  "An image file: photo.png",
  "Audio recording: interview.mp3"
]
semi_structured_data = [
  {"ID": 1, "Skills": ["Python", "SQL"]},
  {"ID": 2, "Skills": ["R", "Tableau"]},
  {"ID": 3, "Skills": ["Java", "Spark"]}
]
print("Structured Data:\n", structured_data)
```

```
print("\nUnstructured Data:\n", unstructured_data)

print("\nSemi-structured Data:\n", semi_structured_data)
```

Output:

```
Structured Data:

   ID    Name   Age

0   1    Alice     25

1   2     Bob      30

2   3  Charlie   35

Unstructured Data:

['Alice loves data science.', 'An image file: photo.png', 'Audio recording: interview.mp3']

Semi-structured Data:

[{'ID': 1, 'Skills': ['Python', 'SQL']}, {'ID': 2, 'Skills': ['R', 'Tableau']}, {'ID': 3, 'Skills': ['Java', 'Spark']}]
```

Result:
Thus, the Python program to differentiate between structured, unstructured, and semi-structured data was executed successfully, and the output was verified.

EXPERIMENT 1D

Exp No: 1.d

Aim:
To conduct an experiment to encrypt and decrypt given sensitive data using Python.

Algorithm:

1. Import the Fernet module from the cryptography library.
2. Generate a secret key using Fernet.generate_key().
3. Initialize a Fernet object with the generated key.
4. Define the original sensitive data to be encrypted.
5. Encrypt the data using the encrypt() function.
6. Decrypt the encrypted data using the decrypt() function.
7. Display the original, encrypted, and decrypted data to verify correctness.

Code:

```
from cryptography.fernet import Fernet

key = Fernet.generate_key()

fernet = Fernet(key)

data = "MyPassword123"
```

```
print("Original Data:", data)

encrypted = fernet.encrypt(data.encode())

print("Encrypted Data:", encrypted)

decrypted = fernet.decrypt(encrypted).decode()

print("Decrypted Data:", decrypted)
```

Output:

Original Data: MyPassword123

Encrypted Data:
b'gAAAAABovRxG0koAWG0eafrue5TFWw5Z0mbwxU5XBI_JAr46PrFNcajUvUKQ3VIKP_ZT2GWQEqGFQT_qhD5LlBiTLpRiHfj9SQ=='

Decrypted Data: MyPassword123

Result:
Thus, the Python program to encrypt and decrypt sensitive data was executed successfully, and the output was verified.