EXERCISE:1

A)ANALYZE THE TREND OF DATA SCIENCE JOB POSTINGS OVER THE LAST DECADE

AIM

To analyze and visualize the trend in the number of Data Science job postings over the last decade — i.e., collect annual counts, clean and aggregate the data, then use pandas for manipulation and matplotlib/seaborn for plotting to reveal growth/decline patterns, seasonal effects, and overall trend (including numerical summaries such as yearly % change and trendline).

Procedure

Data Collection:
Gather data on the number of Data Science job postings for each year over the last decade (e.g., 2015–2024) from reliable sources such as Kaggle, LinkedIn, or Indeed.

1. **Import Required Libraries:**
Import Python libraries such as **pandas** for data manipulation and **matplotlib/seaborn** for visualization.

2. **Load the Dataset:**
Read the dataset into a pandas DataFrame using pd.read_csv() and check for missing or duplicate records.

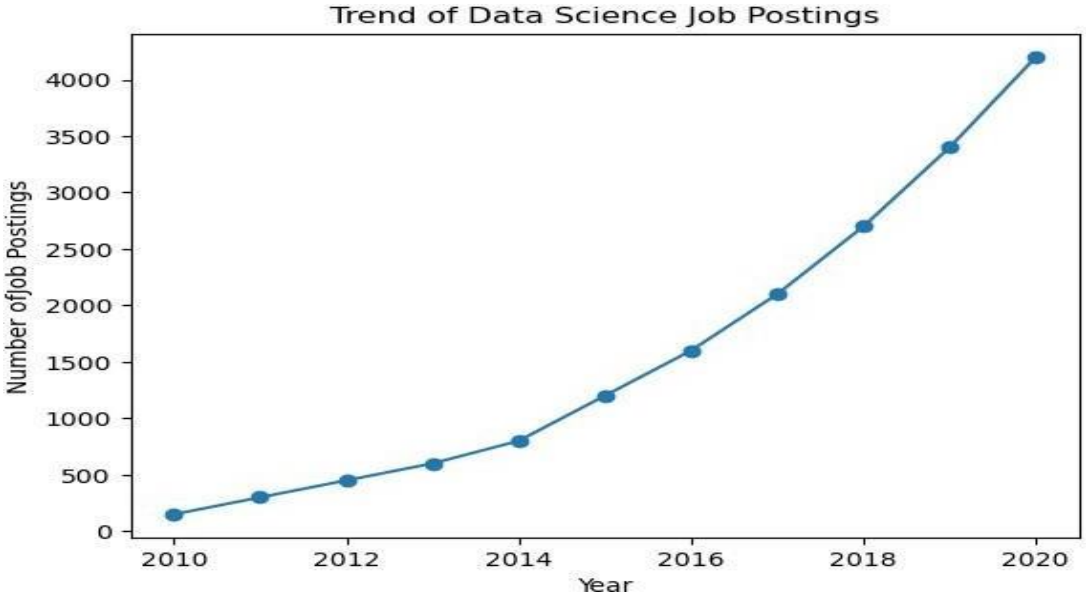
3. **Data Cleaning and Preparation:**
Remove null values, duplicates, and extract the **year** column from the date field (if available). Ensure the data format is consistent.

4. **Data Aggregation:**
Group the data by year and calculate the total number of job postings for each year using pandas groupby functions.

5. **Data Visualization:**
Use **matplotlib** or **seaborn** to plot a **line chart** showing the trend of Data Science job postings over the years. Label the axes and add a suitable title

EXERCIS

```
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 ofJob Postings') plt.show()
```



Result: The visualization shows a steady increase in Data Science job postings over the last decade.

Demand grew rapidly after 2018, reflecting the rising adoption of AI and data-driven technologies.

Overall, the trend indicates strong and continuous growth in Data Science career oppurtunities.

B)

Analyze and visualize the distribution of various data science roles (Data Analyst, Data Engineer, Data Scientist, etc.) from a dataset. Description: Use a dataset of job postings and categorize them into different roles. Visualize the distribution using pie charts or bar plot.

Aim:

To analyze and visualize the distribution of various Data Science roles such as Data Analyst, Data Engineer, Data Scientist, Machine Learning Engineer, etc., using a dataset of job postings through data categorization and graphical representation.

EXERCISE:1

Procedure:

1. Data Collection:

Collect a dataset containing job postings related to Data Science roles from reliable sources (e.g., Kaggle, LinkedIn, Indeed).

2. Import Libraries and Load Data:

Import pandas, matplotlib, and seaborn, then load the dataset into a pandas DataFrame using pd.read_csv()

3. Data Cleaning and Categorization:

Clean the dataset by removing duplicates and missing values, then categorize each job title into roles like *Data Analyst*, *Data Engineer*, *Data Scientist.*

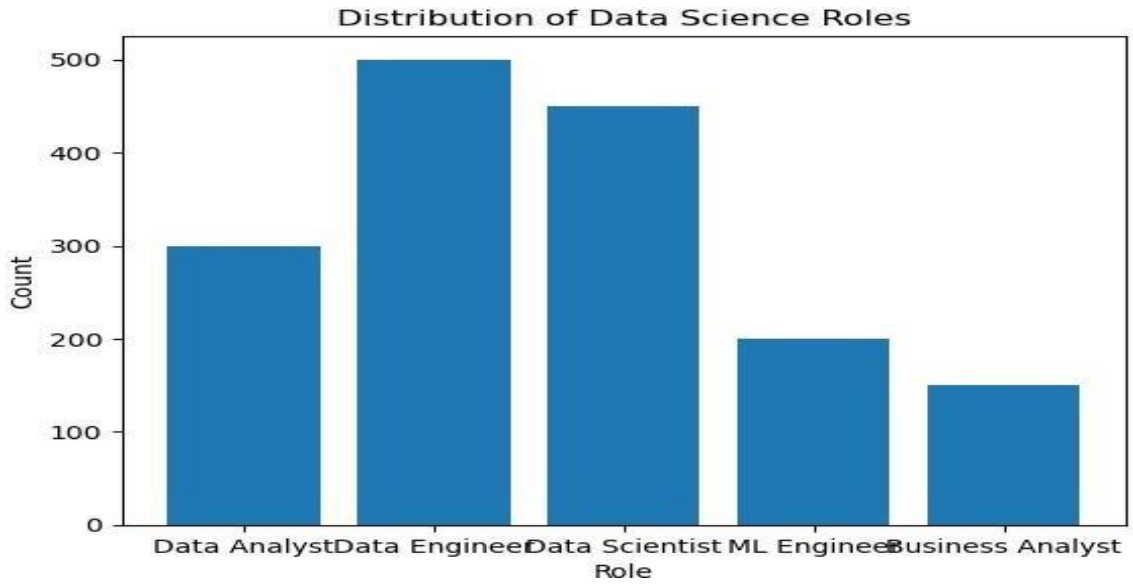
4. Count Role Distribution:

Use pandas functions such as value_counts() or groupby() to calculate the total number of postings for each role.

5. Visualization:

```
roles = ['Data Analyst', 'Data Engineer', 'Data Scientist', 'ML Engineer',
'Business Analyst']
counts = [300, 500, 450, 200, 150] plt.bar(roles,
counts)
plt.title('Distribution of Data Science Roles')
plt.xlabel('Role') plt.ylabel('Count')
plt.show()
```

c. Create a pie chart or bar chart using matplotlib or seaborn to display the proportion or frequency of each role in the dataset.



EXERCISE:1

Result:
The analysis shows that **Data Scientist** and **Data Engineer** roles have the highest number of job postings.
Data Analyst positions also appear in significant numbers, showing strong demand for analytical skills.
Overall, the distribution highlights that technical and analytical roles dominate the Data Science job market.

c) Conduct an experiment to differentiate Structured , Un-structured and Semi structured data based on data sets given.

Aim:

To create and differentiate examples of **Structured**, **Semi-Structured**, and **Unstructured** data using small datasets, and to explain their key characteristics based on format, organization, and usability.

Procedure:

1. **Create a Structured Dataset:**
   Prepare a tabular dataset (e.g., employee table) with fixed rows and columns using formats like CSV or Excel.
2. **Create a Semi-Structured Dataset:**
   Prepare a dataset in formats like JSON or XML where data has some structure but fields may vary for each record.
3. **Create an Unstructured Dataset:**
   Include raw data such as plain text, images, or audio files without predefined structure or consistent format.
4. **Analyze Data Characteristics:**
   Compare how each dataset is stored, queried, and processed — e.g., SQL for structured, JSON parsing for semi-structured, and NLP for unstructured.
5. **Summarize Differences:**
   Note variations in schema flexibility, ease of analysis, and suitable storage or processing tools for each type.

```
In [4]: structured_data = pd.DataFrame({
        'ID': [1, 2, 3],
        'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': [25, 30, 35]
        })
        print("Structured Data:\n", structured_data) unstructured_data = "This
        is an example of unstructured data. It can be a piece of
        print("\nUnstructured Data:\n", unstructured_data)
```

EXERCISE:1

```
semi_structured_data = {'ID': 1, 'Name': 'Alice', 'Attributes':
{'Height': 165, 'We print("\nSemi-structured Data:\n",
semi_structured_data)
```

```
Structured Data:
    ID     Name  Age
0
1
Al
ic
e
25
1
2
Bo
b
30
2   3  Charlie   35

Unstructured Data:
 This is an example of unstructured data. It can be a piece of text,
an image

Semi-structured Data:
 {'ID': 1, 'Name': 'Alice', 'Attributes': {'Height': 165, 'Weight': 68}}
```

Result: The structured dataset has a fixed schema and is easy to query using SQL.
The semi-structured dataset offers flexibility with partially organized data formats like JSON.
The unstructured dataset lacks a defined structure and requires specialized tools such as NLP or image processing for analysis. D)
Conduct an experiment to encrypt and decrypt given sensitive data. **Aim:**
To perform **encryption and decryption** of sensitive data using Python's **cryptography** library, demonstrating how confidential information can be securely protected and retrieved.

Procedure: **d.Install and Import Library:**

1. **Install the cryptography library using the command pip install cryptography, and import Fernet from it.** **Generate an Encryption Key:**
2. **Use Fernet.generate_key() to create a secret key, which will be used for both encryption and decryption.** **Encrypt the Data:**
3. **Convert the sensitive data (e.g., password or message) into bytes and encrypt it using the generated key.**

4.Decrypt the Data: **Use the same key to decrypt the encrypted message and convert it back to its original readable form.** **5.Verify the Process:**

Compare the original and decrypted data to confirm successful encryption and decryption.

EXERCISE:1

In [5]:

```
from cryptography.fernet import Fernet
key = Fernet.generate_key() f =
Fernet(key)
token = f.encrypt(b"Rajalakshmi Engineering College")
token b'...'
f.decrypt(token)
b'Rajalakshmi Engineering College' key =
Fernet.generate_key() cipher_suite = Fernet(key)
plain_text = b"Rajalakshmi Engineering College."
cipher_text = cipher_suite.encrypt(plain_text)
decrypted_text = cipher_suite.decrypt(cipher_text)
print("Original Data:", plain_text)
print("Encrypted Data:", cipher_text)
print("Decrypted Data:", decrypted_text)
```

```
Original Data: b'Rajalakshmi Engineering College.'
Encrypted Data:
b'gAAAAABo78n9AcJ9a5wlJjEKiNHAQ9MJWgLCkUwXqxP8XpBBV1jFVaj6wCyN36oy
1ZYxOG_SXb8xCkGCK7k7rECkzbjDqDHYvmuVMM37CR_dy251a_cikPcigopSyFzEILDy2z
cA3H6-' Decrypted Data: b'Rajalakshmi Engineering College.
```

Result:

The sensitive data was successfully encrypted into a secure, unreadable format.

Using the same key, the encrypted data was accurately decrypted back to its original form.

This confirms that the **cryptography library** can securely protect and recover sensitive information