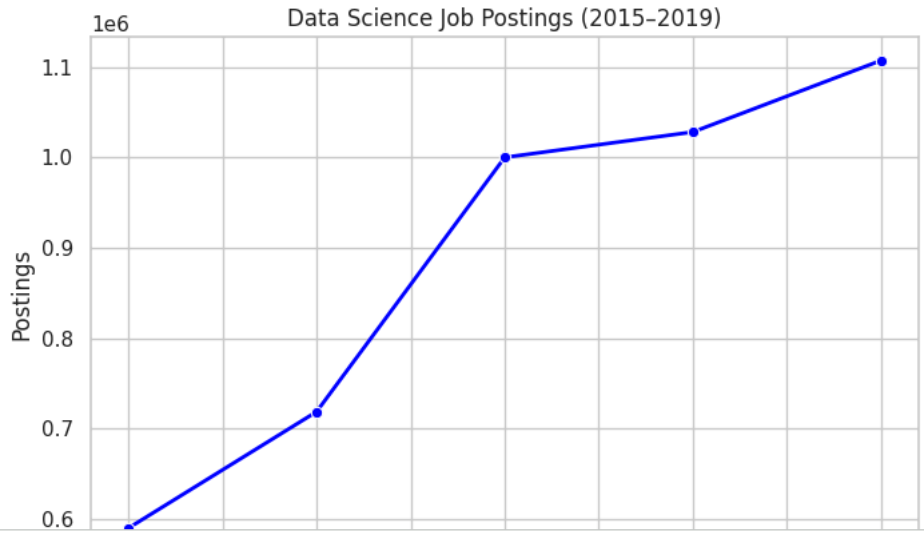
```
#1.a
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

data = {
    'Year': [2015, 2016, 2017, 2018, 2019],
        'Postings': [590184, 718846, 1000000, 1028056, 1107138]
        }

df = pd.DataFrame(data)
df['Growth_%'] = df['Postings'].pct_change() * 100
start_value = df['Postings'].iloc[0]
end_value = df['Postings'].iloc[-1]
years = df['Year'].iloc[-1] - df['Year'].iloc[0]
CAGR = ((end_value / start_value) ** (1 / years) - 1) * 100
print(f"CAGR: {CAGR:.2f}%")

sns.set_theme(style="whitegrid")
plt.figure(figsize=(8,5))
sns.lineplot(data=df, x='Year', y='Postings', marker='o', linewidth=2, color='blue')
plt.title("Data Science Job Postings (2015–2019)")
plt.show()
```

```
CAGR: 17.03%
```




```
#1.b
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

data = {
    'Job Title': [
            'Data Scientist', 'Data Analyst', 'Machine Learning Engineer', 'Data Engineer',
            'Data Analyst', 'Data Scientist', 'Business Intelligence Analyst',
            'Data Engineer', 'Data Scientist', 'Data Analyst', 'ML Engineer', 'Data Scientist'
      ]
}
df = pd.DataFrame(data)

def categorize_role(title):
      title = title.lower()
      if 'scientist' in title:
        return 'Data Scientist'
      elif 'engineer' in title and 'ml' not in title:
        return 'Data Engineer'
      elif 'ml' in title or 'machine learning' in title:
        return 'ML Engineer'
      elif 'analyst' in title:
        return 'Data Analyst'
      elif 'business intelligence' in title:
        return 'BI Analyst'
      else:
        return 'Other'

df['Role'] = df['Job Title'].apply(categorize_role)

role_counts = df['Role'].value_counts()
```

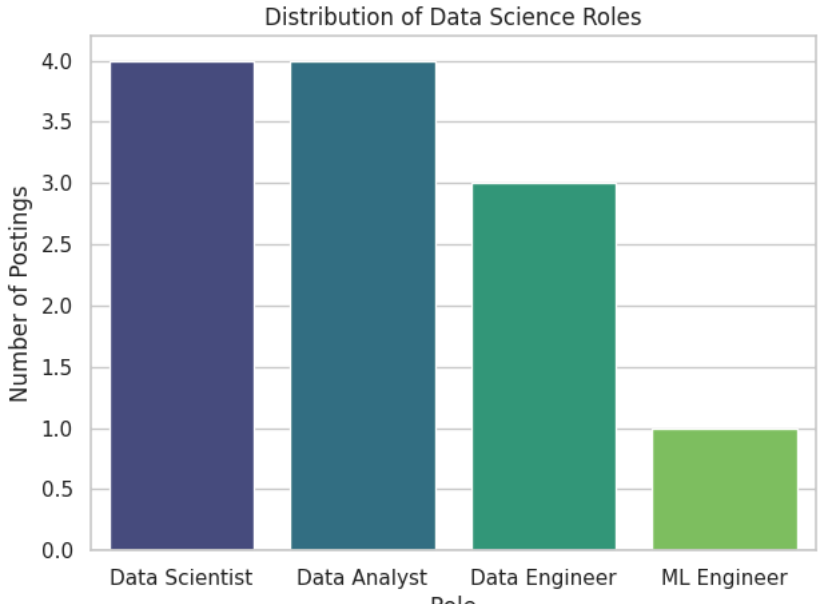
```
print(role_counts)

sns.set_theme(style="whitegrid")
plt.figure(figsize=(7,5))
sns.barplot(x=role_counts.index, y=role_counts.values, palette='viridis')
plt.title('Distribution of Data Science Roles')
plt.xlabel('Role')
plt.ylabel('Number of Postings')
plt.show()
```

```
Role
Data Scientist    4
Data Analyst      4
Data Engineer     3
ML Engineer       1
Name: count, dtype: int64
/tmp/ipython-input-2318413288.py:38: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and

  sns.barplot(x=role_counts.index, y=role_counts.values, palette='viridis')
```




```
#1.c
#structured data

import pandas as pd

structured_df = pd.DataFrame({
    'Employee_ID': [101, 102, 103],
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [29, 35, 41],
    'Department': ['Data Science', 'Engineering', 'Analytics']
    })

print("Structured Data:\n", structured_df)

#unstructured data

unstructured_data = [
      "Alice joined the Data Science team last year. She's great at Python and ML.",
      "Bob moved from Engineering. Loves working with AWS and building ML pipelines.",
      "Charlie wrote an amazing report on analytics trends and data visualization."
       ]
print("\nUnstructured Data (Text Documents):")
for doc in unstructured_data:
  print("-", doc)

#semi structured

import json

semi_structured_data = [
      {"Name": "Alice", "Skills": ["Python", "SQL"], "Experience": 3},
        {"Name": "Bob", "Skills": ["Java", "AWS"], "Experience": 5, "Certifications": ["AWS Developer"]},
            {"Name": "Charlie", "Skills": ["Tableau"], "Department": "Analytics"}
            ]
```

```
print("\nSemi-Structured Data (JSON):")
print(json.dumps(semi_structured_data, indent=2))
```

```
Structured Data:
    Employee_ID     Name  Age    Department
0          101    Alice   29  Data Science
1          102      Bob   35   Engineering
2          103  Charlie   41     Analytics

Unstructured Data (Text Documents):
- Alice joined the Data Science team last year. She's great at Python and ML.
- Bob moved from Engineering. Loves working with AWS and building ML pipelines.
- Charlie wrote an amazing report on analytics trends and data visualization.

Semi-Structured Data (JSON):
[
  {
    "Name": "Alice",
    "Skills": [
      "Python",
      "SQL"
    ],
    "Experience": 3
  },
  {
    "Name": "Bob",
    "Skills": [
      "Java",
      "AWS"
    ],
    "Experience": 5,
    "Certifications": [
      "AWS Developer"
    ]
  },
  {
    "Name": "Charlie",
    "Skills": [
      "Tableau"
    ],
    "Department": "Analytics"
  }
]
```

```
#1.d

!pip install cryptography

from cryptography.fernet import Fernet

key = Fernet.generate_key()
cipher = Fernet(key)

data = "Sensitive_Data_12345".encode()

encrypted_data = cipher.encrypt(data)

decrypted_data = cipher.decrypt(encrypted_data)

print("Encryption Key:", key.decode())
print("Original Data:", data.decode())
print("Encrypted Data:", encrypted_data.decode())
print("Decrypted Data:", decrypted_data.decode())
```

```
Requirement already satisfied: cryptography in /usr/local/lib/python3.12/dist-packages (43.0.3)
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.12/dist-packages (from cryptography) (2.0.0)
Requirement already satisfied: pycparser in /usr/local/lib/python3.12/dist-packages (from cffi>=1.12->cryptography) (2.23)
Encryption Key: m-aPpoK7WBvwntmkaJaNH8rXXOLCrmQhqrllppImFuw=
Original Data: Sensitive_Data_12345
Encrypted Data: gAAAAABpBeL2iZ5ZOKarggDY0-515odnbWbOhjCHHhgL_bwhbXptP7sA47NVQgSjHClIpmKFxL9BxFCTdM9dCi6ZhacyHRx7HYhAMp0vt_c8
Decrypted Data: Sensitive_Data_12345
```