In [1]:
```python
#Create Pandas Series and DataFrames

# Importing Pandas
import pandas as pd

# Creating a Pandas Series
data_series = pd.Series([10, 20, 30, 40])
print("Pandas Series:")
print(data_series)

# Creating a Pandas DataFrame
data_frame = pd.DataFrame({
    'Column1': [1, 2, 3, 4],
    'Column2': ['A', 'B', 'C', 'D']
})
print("\nPandas DataFrame:")
print(data_frame)
```

```
Pandas Series:
0    10
1    20
2    30
3    40
dtype: int64

Pandas DataFrame:
   Column1 Column2
0        1       A
1        2       B
2        3       C
3        4       D
```

In [2]:
```python
#Load Data from Various File Formats (CSV, Excel, etc.)

# Creating a CSV file for demonstration
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'City': ['New York', 'Los Angeles', 'Chicago']
}

df = pd.DataFrame(data)
df.to_csv('sample_data.csv', index=False)

# Loading data from CSV
loaded_df = pd.read_csv('sample_data.csv')
print("\nLoaded DataFrame from CSV:")
print(loaded_df)
```

```
Loaded DataFrame from CSV:
      Name  Age         City
0    Alice   25     New York
1      Bob   30  Los Angeles
2  Charlie   35      Chicago
```

In [3]:
```python
#Perform Data Cleaning and Manipulation Tasks
# Creating a DataFrame with missing values
data_with_nan = {
    'Name': ['Alice', 'Bob', None, 'David'],
    'Age': [25, None, 30, 22],
    'City': ['New York', 'Los Angeles', 'Chicago', None]
}

df_with_nan = pd.DataFrame(data_with_nan)

print("\nDataFrame with NaN values:")
print(df_with_nan)

# Dropping rows with any NaN values
cleaned_df = df_with_nan.dropna()
print("\nDataFrame after dropping NaN values:")
print(cleaned_df)

# Filling missing values with a default value
filled_df = df_with_nan.fillna({'Name': 'Unknown', 'Age': 0, 'City': 'Unknown'})
print("\nDataFrame after filling NaN values:")
print(filled_df)
```

```
DataFrame with NaN values:
    Name   Age         City
0  Alice  25.0     New York
1    Bob   NaN  Los Angeles
2   None  30.0      Chicago
3  David  22.0         None

DataFrame after dropping NaN values:
    Name   Age      City
0  Alice  25.0  New York

DataFrame after filling NaN values:
      Name   Age         City
0    Alice  25.0     New York
1      Bob   0.0  Los Angeles
2  Unknown  30.0      Chicago
3    David  22.0      Unknown
```

In [7]:
```python
#Explore Data Analysis and Visualization Using Pandas

# Creating a DataFrame with missing values
data_with_nan = {
    'Name': ['Alice', 'Bob', None, 'David'],
    'Age': [25, None, 30, 22],
    'City': ['New York', 'Los Angeles', 'Chicago', None]
}

df_with_nan = pd.DataFrame(data_with_nan)

print("\nDataFrame with NaN values:")
print(df_with_nan)

# Dropping rows with any NaN values
```

```python
cleaned_df = df_with_nan.dropna()
print("\nDataFrame after dropping NaN values:")
print(cleaned_df)

# Filling missing values with a default value
filled_df = df_with_nan.fillna({'Name': 'Unknown', 'Age': 0, 'City': 'Unknown'})
print("\nDataFrame after filling NaN values:")
print(filled_df)
```

```
DataFrame with NaN values:
    Name   Age         City
0  Alice  25.0     New York
1    Bob   NaN  Los Angeles
2   None  30.0      Chicago
3  David  22.0         None

DataFrame after dropping NaN values:
    Name   Age      City
0  Alice  25.0  New York

DataFrame after filling NaN values:
      Name   Age         City
0    Alice  25.0     New York
1      Bob   0.0  Los Angeles
2  Unknown  30.0      Chicago
3    David  22.0      Unknown
```

In [8]:
```python
# Create Pivot Tables and Group Data for Analysis

# Creating a DataFrame for pivot table example
pivot_data = {
    'Date': ['2024-10-01', '2024-10-01', '2024-10-02', '2024-10-02'],
    'Category': ['A', 'B', 'A', 'B'],
    'Sales': [100, 150, 200, 250]
}

df_pivot = pd.DataFrame(pivot_data)

# Creating a pivot table
pivot_table = df_pivot.pivot_table(values='Sales', index='Date', columns='Category'
print("\nPivot Table:")
print(pivot_table)

# Grouping data
grouped_data = df_pivot.groupby('Category').sum()
print("\nGrouped Data by Category:")
print(grouped_data)
```

```
Pivot Table:
Category        A    B
Date
2024-10-01  100  150
2024-10-02  200  250

Grouped Data by Category:
                        Date  Sales
Category
A         2024-10-012024-10-02    300
B         2024-10-012024-10-02    400
```

In [ ]: