pip install pandas numpy matplotlib seaborn

* **pandas**: For data manipulation and analysis.
* **numpy**: For numerical operations.
* **matplotlib** and **seaborn**: For data visualization.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

Pandas can read various data formats, such as CSV, Excel, SQL databases, and more. For this example, let's use a CSV file. Place the CSV file in the same directory as your script or notebook.

data = pd.read\_csv('data.csv')

**Basic Data Exploration**

* **data.head()** or **data.tail()**: Display the first or last few rows of the dataset.
* **data.info()**: Show summary information about the dataset.
* **data.describe()**: Generate summary statistics for numeric columns.

**Data Selection and Filtering**

* **data['column\_name']**: Access a specific column.
* **data[['col1', 'col2']]**: Access multiple columns.
* **data.loc[row\_index]** or **data.iloc[row\_index]**: Access specific rows by label or index.
* **data[data['column\_name'] > 50]**: Filter rows based on a condition.

**5. Data Cleaning and Transformation**

**Handling Missing Values**

* **data.dropna()**: Remove rows with missing values.
* **data.fillna(value)**: Fill missing values with a specific value.
* **data.isna()**: Check for missing values.

**Data Transformation**

* **data['new\_column'] = data['col1'] + data['col2']**: Create a new column.
* **data['col1'].apply(function)**: Apply a function to a column.
* **data.groupby('category\_col').agg({'numeric\_col': 'mean'})**: Group and aggregate data.

**Matplotlib and Seaborn**

* **plt.plot(x, y)**: Create line plots.
* **plt.scatter(x, y)**: Create scatter plots.
* **sns.barplot(x='category', y='value', data=data)**: Create bar plots.

**7. Further Learning**

This is just the tip of the iceberg. Data analysis involves more complex tasks like merging datasets, handling time series data, and advanced visualization. Consider exploring more advanced topics and libraries like:

* **Time Series Analysis**: Pandas has tools for handling time series data. Check out **pd.to\_datetime()** and time-based indexing.
* **Machine Learning with Scikit-Learn**: Integrate machine learning algorithms to analyze and predict data patterns.
* **Jupyter Notebooks**: Use Jupyter for interactive data analysis and sharing your work.
* **Advanced Visualization**: Explore Plotly for interactive visualizations and Bokeh for creating interactive, web-ready visualizations.

**Basic Exploration**

* **data.head()**: Display the first few rows.
* **data.info()**: Get information about column data types and missing values.
* **data.describe()**: Summary statistics for numeric columns.
* **data['column\_name'].value\_counts()**: Count unique values in a column.

**Detect Missing Values**

* **data.isna()**: Identify missing values in the dataset.
* **data.isna().sum()**: Count missing values in each column.

data\_cleaned = data.dropna()

**Fill Missing Values**

* **data['column\_name'].fillna(value)**: Fill missing values in a specific column with a given value.
* **data.fillna(method='ffill')**: Fill missing values using forward-fill (propagate the last valid value forward).
* **data.fillna(method='bfill')**: Fill missing values using backward-fill (propagate the next valid value backward).

data\_cleaned = data.drop\_duplicates()

**Standardize Text Data**

* **data['column\_name'] = data['column\_name'].str.lower()**: Convert text to lowercase for consistency.
* **data['column\_name'] = data['column\_name'].str.strip()**: Remove leading and trailing whitespaces.

data['date\_column'] = pd.to\_datetime(data['date\_column'])

data['numeric\_column'] = pd.to\_numeric(data['numeric\_column'])

## Export Cleaned Data

Save the cleaned data to a new CSV or Excel file for further analysis:

data\_cleaned.to\_csv('cleaned\_data.csv', index=False)