EXPLORATORY DATA ANALYSIS

A Complete Guide (Zero to Hero)

Table of Contents

Exploratory Data Analysis Complete Guide

Chapter 1: Introduction to EDA

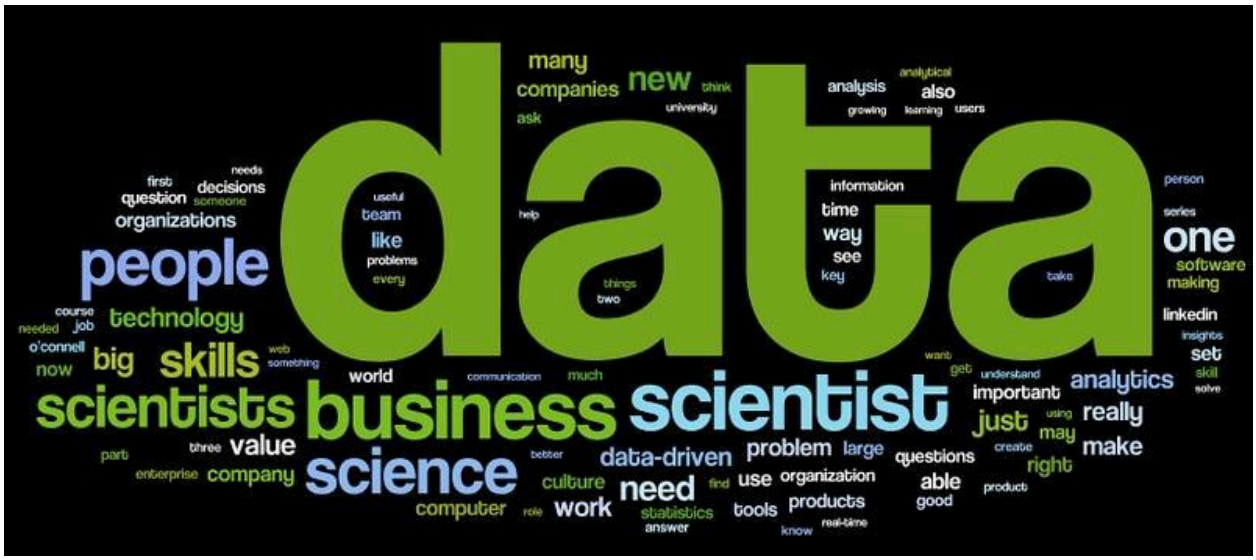
Exploratory Data Analysis (EDA)

Welcome to the exciting world of Exploratory Data Analysis (EDA)! Imagine you're a detective, and your data is the crime scene. EDA is your magnifying glass, helping you uncover hidden patterns, understand relationships, and ultimately solve the mystery your data holds.

This ebook will equip you with the tools and knowledge to become a data detective. Let's dive into each topic:

1. Introduction to Exploratory Data Analysis (EDA).

Exploratory Data Analysis (EDA) is a crucial initial step in data science projects. It involves analyzing and visualizing data to understand its key characteristics, uncovering patterns, and identifying relationships between variables. It refers to studying and exploring record sets to apprehend their predominant traits, discover patterns, locate outliers, and identify relationships between variables. EDA is normally carried out as a preliminary step before undertaking extra formal statistical analyses or modeling. This preliminary analysis ensures that your data is ready for more advanced modeling and analysis.



2. Importance of EDA in Data Science.

Just like a detective wouldn't jump straight to conclusions, EDA is crucial in data science. It lays the groundwork for everything that comes after. Through EDA, you can identify potential issues with the data, refine your research questions, and ultimately build more robust models. Think of it as building a solid foundation for your data science project. Without EDA, you might miss important insights or make decisions based on flawed data.

3. The Purposes and Objectives of EDA.

The main purposes of EDA are to:

- Summarize the main characteristics of the data.
- Identify patterns and relationships.
- Spot anomalies and outliers.

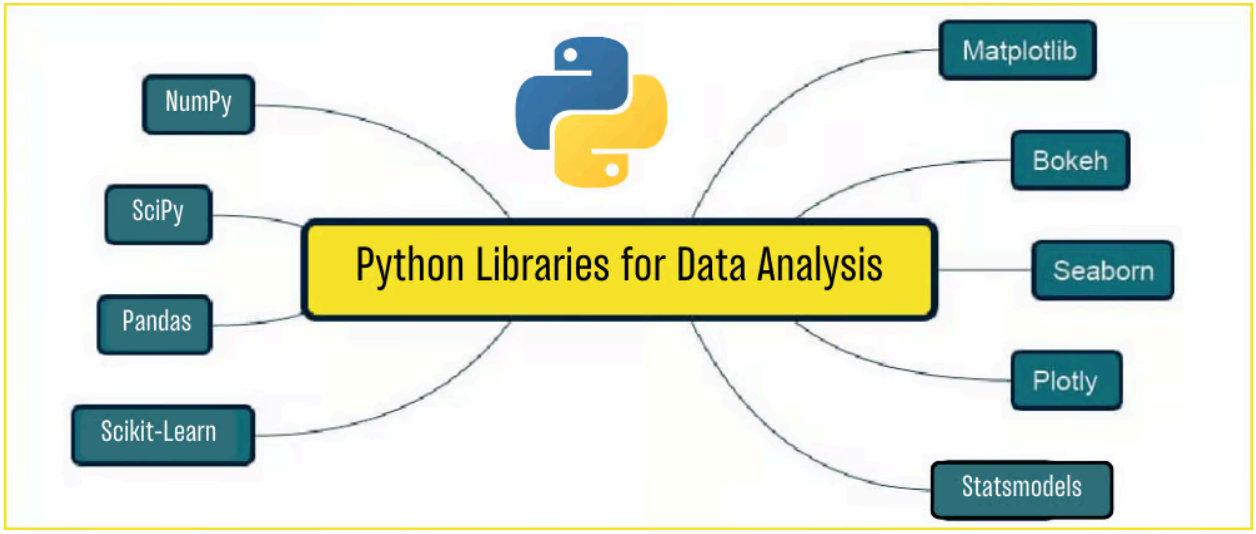
- Test initial hypotheses.
- Determine the data's structure and cleanliness.

The objectives include ensuring data quality, forming a clear understanding of the data, and preparing it for further analysis or modeling.

4. Types of EDA.

EDA can be divided into different types based on the techniques used:

- **Descriptive EDA:**
  Summarizes the data with statistics and visualizations.
- **Inferential EDA:**
  Makes inferences about the population based on sample data.
- **Univariate EDA:**
  Focuses on one variable at a time.
- **Bivariate EDA:**
  Examines relationships between two variables.
- **Multivariate EDA:**
  Examines relationships between two variables.



5. Tools and Libraries for EDA.

Several tools and libraries make EDA easier and more efficient:

- **Python Libraries:**
  Pandas, NumPy, Matplotlib, Seaborn, Plotly.
- **R Libraries:**
  ggplot2, dplyr, tidyr.
- **Software:**
  Jupyter Notebooks, RStudio, Tableau.

These tools help you manipulate data, perform statistical analysis, and create visualizations.

6. Statistical Concepts Used in EDA.

EDA relies on several key statistical concepts:

- **Mean, Median, Mode:**
  Measures of central tendency.
- **Variance and Standard Deviation:**
  Measures of spread.
- **Correlation and Covariance:**
  Measures of relationships between variables.
- **Probability Distributions:**
  Understanding the data's distribution.
- **Hypothesis Testing:**
  Making inferences about the data.

These concepts are your building blocks for interpreting your data's story.

7. Graphs and Charts Used in EDA.

A picture is worth a thousand words, and that's especially true in EDA. Some common types of charts include:

- **Histograms:**
  Show the distribution of a single variable.
- **Box Plots:**
  Highlight the distribution and outliers.
- **Scatter Plots:**
  Show relationships between two variables.
- **Bar Charts:**
  Compare different categories.
- **Heatmaps:**
  Visualize correlation matrices or other data patterns.

Remember, these visualizations are meant to help you understand your data, not just impress people with fancy charts.

8. Best Practices in EDA.

To get the most out of EDA, follow these best practices:

- **Start Simple:**
  Begin with cleaning your data, basic statistics, and plots.
- **Be Curious:**
  Explore different angles and ask questions.
- **Document Your Steps:**
  Keep track of what you do and why.
- **Clean Your Data:**
  Identify and handle missing values and outliers.

- **Use Visualizations**:
  They often reveal insights that numbers alone can't.

9. Common Pitfalls and How to Avoid Them.

Watch out for these common EDA mistakes:

- **Ignoring Data Quality:**
  Always check for and handle missing or erroneous data.
- **Overfitting to Noise:**
  Be cautious not to draw conclusions from random variations.
- **Confirmation Bias:**
  Avoid looking only for evidence that supports your hypotheses.
- **Overcomplicating:**
  Start simple and build up complexity as needed.
- **Neglecting Documentation:**
  Keep a clear record of your analysis process.

10. Future Trends in EDA.

EDA is evolving with advancements in technology. Some future trends include:

- Automated EDA Tools:
  Using AI to automate and enhance EDA processes.
- Interactive Visualizations:
  More powerful and user-friendly visualization tools.
- Integration with Machine Learning:
  Seamless integration with predictive modeling.
- Enhanced Collaboration:
  Tools that make it easier to share and collaborate on data analysis.
- Big Data Handling:
  Improved capabilities for handling and analyzing large datasets.

Exploratory Data Analysis Complete Guide

Chapter 2: Setting Up The Computer For EDA

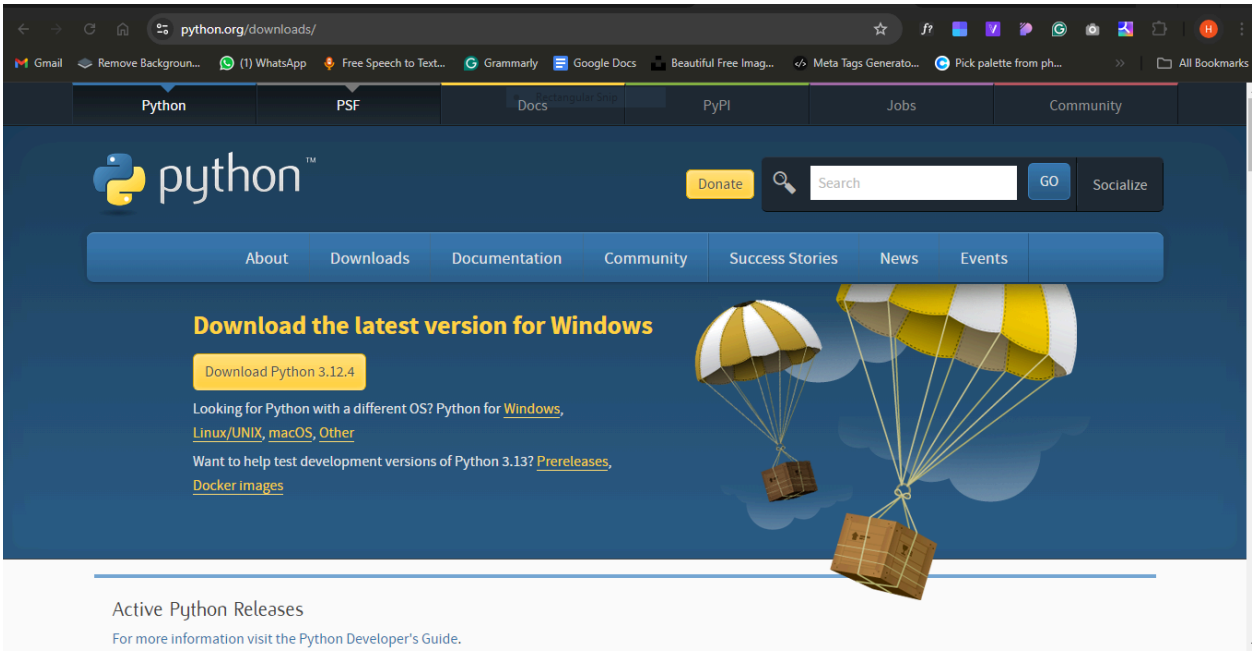
Preparing Your Computer for EDA.

To start with Exploratory Data Analysis (EDA) in Python, you'll need to set up your computer with the necessary software, extensions, and libraries. Here's a step-by-step guide to help you get everything ready.

Step 1: Install Python.

1. **Download Python**: Go to the [Python official website] and download the latest version of Python.

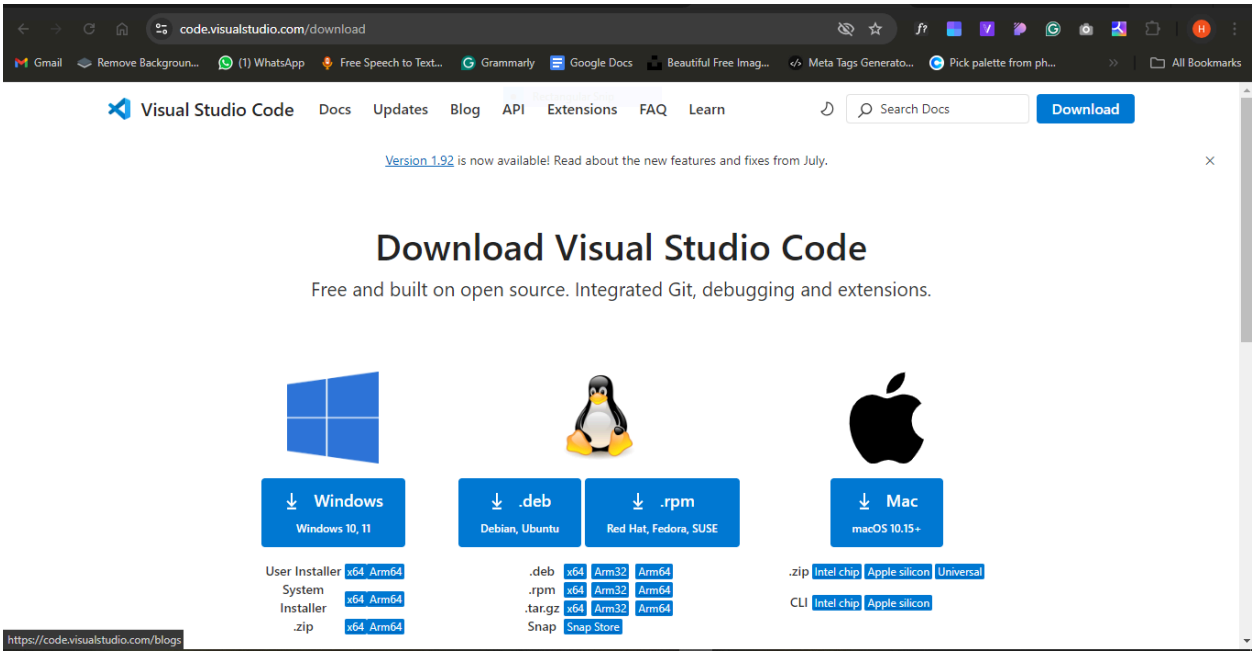
3



2. **Install Python**: Run the installer and make sure to check the box that says "Add Python to PATH". Follow the instructions to complete the installation.

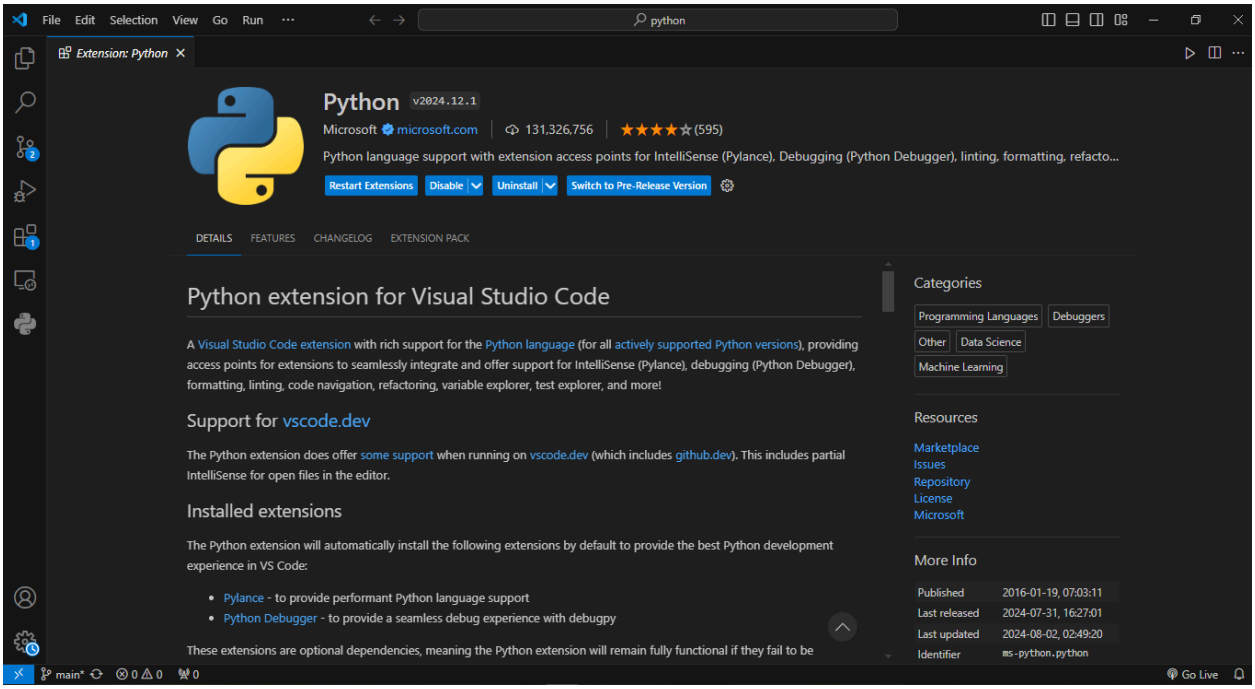
Step 2: Install Visual Studio Code (VS Code).

1. **Download VS Code**: Visit the [Visual Studio Code website] and download the installer for your operating system.
2. **Install VS Code**: Run the installer and follow the instructions to complete the installation.



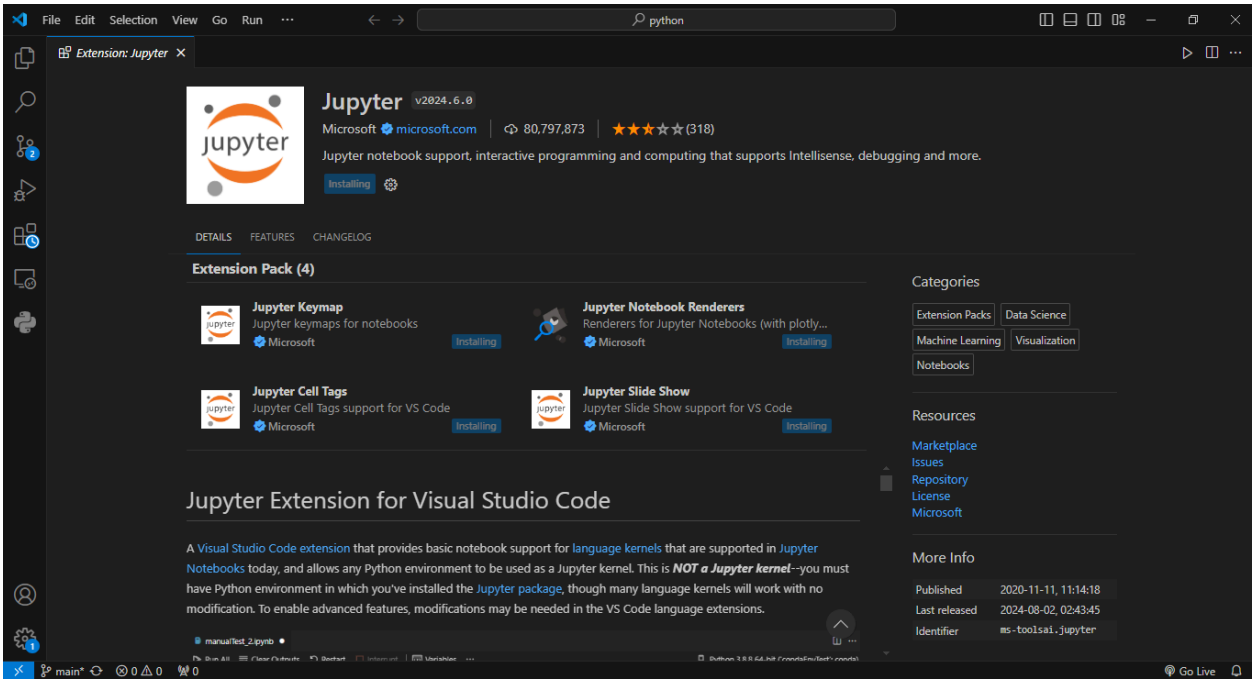
Step 3: Install Python Extension for VS Code.

1. **Open VS Code**.
2. **Go to Extensions**: Click on the Extensions view icon on the Sidebar or press Ctrl+Shift+X.
3. **Search for Python**: Type "Python" in the search bar.
4. **Install Python Extension**: Click on the Install button next to the Python extension by Microsoft. You can find it here.



Step 4: Install Jupyter Extension for VS Code.

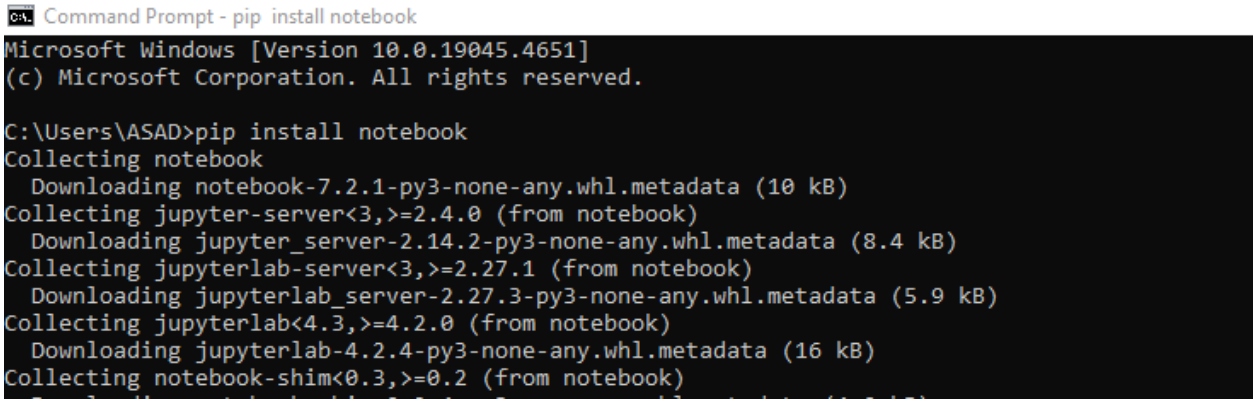
1. **Open Extensions**: Click on the Extensions view icon on the Sidebar or press Ctrl+Shift+X.
2. **Search for Jupyter**: Type "Jupyter" in the search bar.
3. **Install Jupyter Extension**: Click on the Install button next to the Jupyter extension by Microsoft. You can find it here.



Step 5: Install Jupyter via the Command Line.

1. **Open Command Prompt or Terminal**.
2. **Install Jupyter**: Type the following command and press Enter:

   pip install notebook

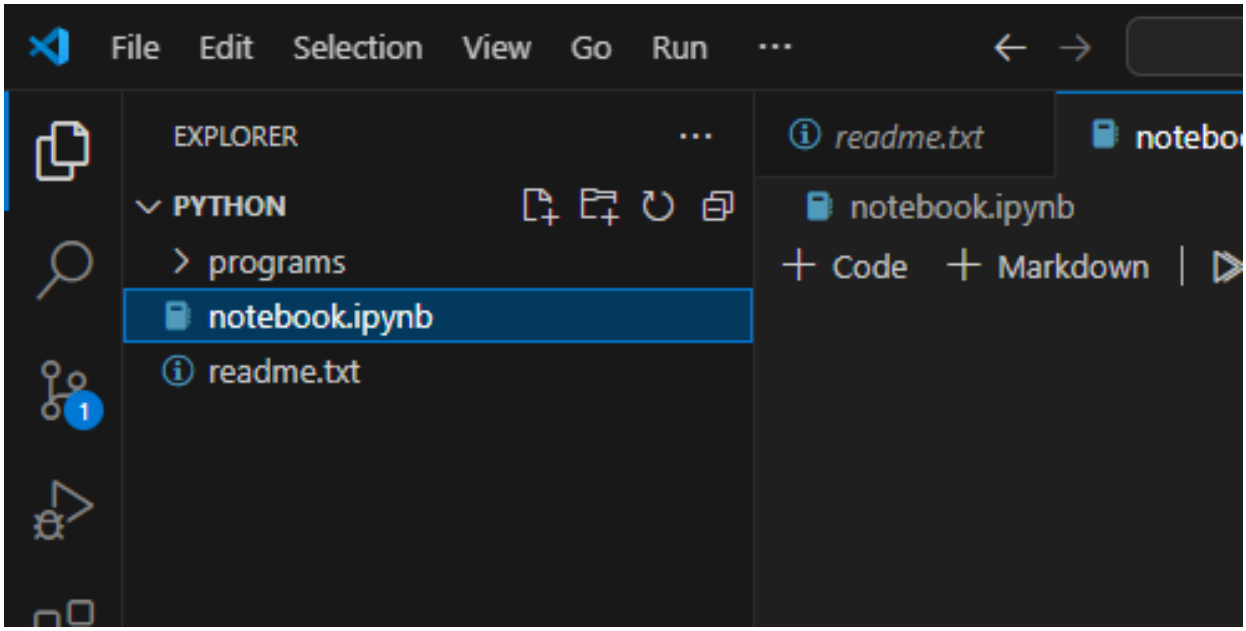


Step 6: Install Essential Python Libraries for EDA.

1. **Open Command Prompt or Terminal**.
2. **Install Libraries**: Use pip to install the essential libraries for EDA. Run the following commands:
   pip install pandas
   pip install numpy
   pip install matplotlib
   pip install seaborn
   pip install plotly
   pip install scipy

Step 8: Configure Jupyter Notebook in VS Code.

1. **Open VS Code**.
2. **Create or Open a Jupyter Notebook**: You can create a new Jupyter Notebook file by clicking on the File menu, selecting New File, and then changing the file extension to .ipynb. Alternatively, you can open an existing .ipynb file.



3. **Select Interpreter**: When you open a Jupyter Notebook file in VS Code for the first time, it will prompt you to select a Python interpreter. Choose the interpreter you installed in Step 1.
4. **Start Coding**: You should now see the Jupyter Notebook interface within VS Code, allowing you to create and run code cells.

By following these steps, your computer will be fully set up for performing Exploratory Data Analysis (EDA) using Python and Visual Studio Code.

Exploratory Data Analysis Complete Guide

Chapter 3: Exploring The Libraries

Exploring The Libraries And Tools.

In the world of data science, having the right tools and libraries at your disposal can significantly affect how efficiently and effectively you can analyze data. Exploratory Data Analysis (EDA) is no exception. In this section, we'll introduce you to some of the most essential libraries and tools used in EDA. These libraries will help you clean, manipulate, visualize, and analyze your data with ease, enabling you to uncover insights and make data-driven decisions.

1. Pandas

- **Description**: Pandas is a powerful data manipulation and analysis library for Python. It provides data structures like DataFrames, which are essential for handling and analyzing structured data.
- **Uses in EDA**:
    - Loading data from various file formats (CSV, Excel, SQL, etc.).
    - Cleaning and preprocessing data (handling missing values, filtering, etc.).
    - Aggregating and summarizing data.
    - Merging and joining datasets.

2. NumPy

- **Description**: NumPy is a fundamental library for numerical computing in Python. It provides support for arrays, matrices, and a large collection of mathematical functions to operate on these data structures.
- **Uses in EDA**:
    - Performing mathematical and statistical operations on arrays and matrices.
    - Efficient numerical computations.
    - Generating random samples and performing random sampling.

3. Matplotlib

- **Description**: Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It is highly customizable and provides a wide range of plotting functionalities.
- **Uses in EDA**:
    - Creating basic plots like line charts, bar charts, and histograms.
    - Customizing plot aesthetics (titles, labels, legends, etc.).
    - Visualizing distributions and trends in data.

4. Seaborn

- **Description**: Seaborn is built on top of Matplotlib and provides a high-level interface for drawing attractive and informative statistical graphics. It simplifies complex visualizations and is particularly useful for statistical plots.

- **Uses in EDA**:
    - Creating advanced visualizations like violin plots, box plots, and heatmaps.
    - Visualizing the distribution of data and relationships between variables.
    - Enhancing the aesthetics of Matplotlib plots.

5. Plotly

- **Description**: Plotly is a library for creating interactive, web-based visualizations. It supports a wide variety of plots and is particularly useful for creating dashboards and sharing interactive plots online.
- **Uses in EDA**:
    - Creating interactive plots that allow for zooming, panning, and hovering.
    - Building dashboards for exploratory analysis.
    - Visualizing complex datasets in an interactive manner.

6. SciPy

- **Description**: SciPy is a library used for scientific and technical computing. It builds on NumPy and provides a large number of higher-level functions for optimization, integration, interpolation, eigenvalue problems, and other advanced mathematical operations.
- **Uses in EDA**:
    - Performing statistical analysis and hypothesis testing.
    - Computing advanced mathematical functions.
    - Conducting signal processing and image processing.

7. Jupyter Notebooks

- **Description**: Jupyter Notebooks is an open-source web application that allows you to create and share documents containing live code, equations, visualizations, and narrative text. It is widely used in data science for exploratory analysis, data cleaning, and sharing results.
- **Uses in EDA**:
    - Writing and running Python code in an interactive environment.
    - Documenting the analysis process with markdown cells.
    - Visualizing data inline with the code.
    - Sharing notebooks with others for collaboration.

By understanding and utilizing these libraries and tools, you'll be well-equipped to perform comprehensive Exploratory Data Analysis (EDA). Pandas and NumPy will handle your data manipulation and numerical operations, while Matplotlib, Seaborn, and Plotly will help you create insightful visualizations. SciPy will enable advanced statistical analysis, and Jupyter Notebooks will provide an interactive platform to document and share your findings. Together, these tools form a robust toolkit for uncovering the hidden patterns and insights within your data.

Exploratory Data Analysis Complete Guide

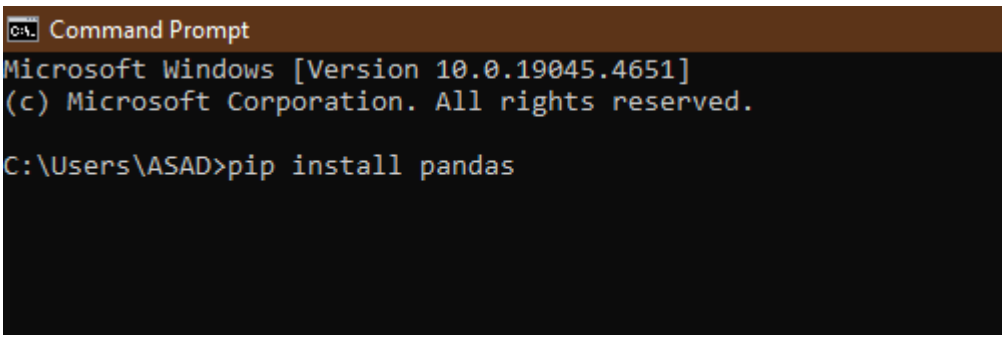
Chapter 4: Pandas Tutorial

Pandas Cheat Sheet.

Let’s dive into the world of Pandas and see what we can do with it!

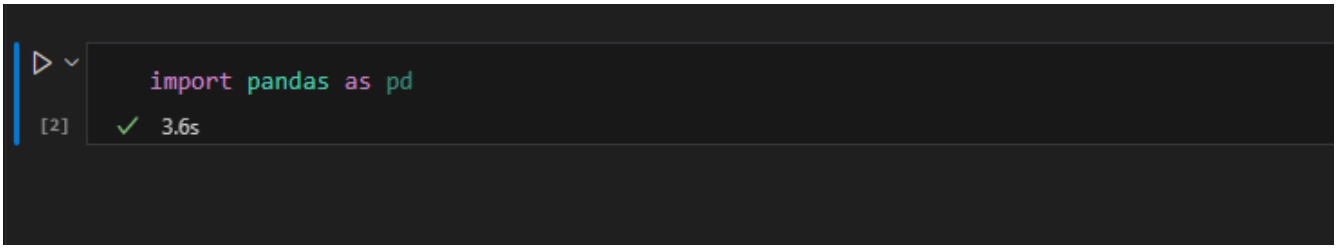
1. Installing Pandas

If you have Python installed, you can use the following command to install Pandas:
pip install pandas



2. Importing Pandas

Once Pandas is installed, you can import it into your Python script or Jupyter Notebook using the following import statement:
import pandas as pd

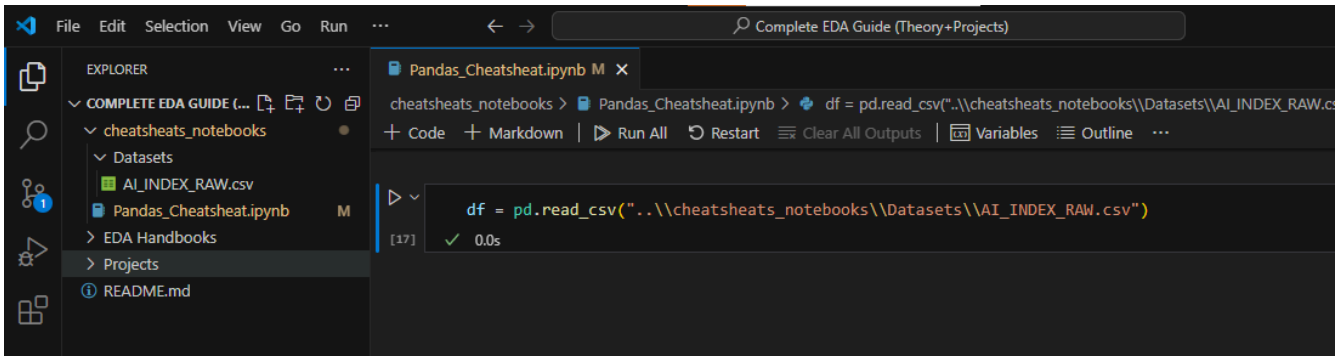


3. Reading/writing Files

- **Read the .csv file:**
  pd.read_csv('filename.csv')
- **Saving the .csv file:**
  df.to_csv('xyz.csv')

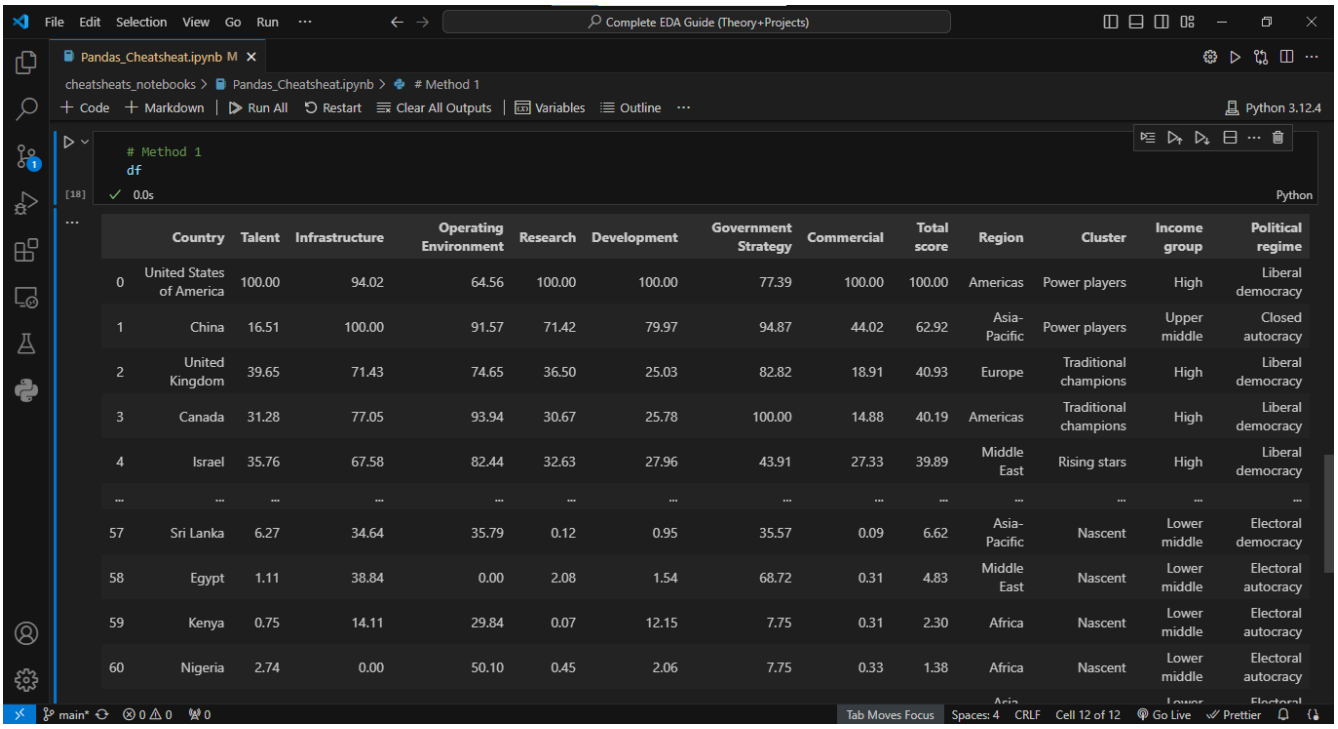
- **Read the Sheet1 of the Excel file ‘xyz.xls’ :**
  pd.read_excel(filename, 'Sheet1')
- **Saving the Sheet1 of the Excel file ‘xyz.xls’: df.to_excel('filename.xlsx', sheet_name='Sheet1')**

- **Read the xyz.json file:** `pd.read_json('filename.json')`
- **Read the xyz.sql file:** `pd.read_sql('filename.sql')`
- **Read the xyz.html file:** `pd.read_html(filename.html')`



4. Viewing The Data

- **Method 1 - Printing whole Data File:** df or dataframe-name



	Country	Talent	Infrastructure	Operating Environment	Research	Development	Government Strategy	Commercial	Total score	Region	Cluster	Income group	Political regime
0	United States of America	100.00	94.02	64.56	100.00	100.00	77.39	100.00	100.00	Americas	Power players	High	Liberal democracy
1	China	16.51	100.00	91.57	71.42	79.97	94.87	44.02	62.92	Asia-Pacific	Power players	Upper middle	Closed autocracy
2	United Kingdom	39.65	71.43	74.65	36.50	25.03	82.82	18.91	40.93	Europe	Traditional champions	High	Liberal democracy
3	Canada	31.28	77.05	93.94	30.67	25.78	100.00	14.88	40.19	Americas	Traditional champions	High	Liberal democracy
4	Israel	35.76	67.58	82.44	32.63	27.96	43.91	27.33	39.89	Middle East	Rising stars	High	Liberal democracy
...	...	...	...	...	...	...	...	...	...	...	...	...	...
57	Sri Lanka	6.27	34.64	35.79	0.12	0.95	35.57	0.09	6.62	Asia-Pacific	Nascent	Lower middle	Electoral democracy
58	Egypt	1.11	38.84	0.00	2.08	1.54	68.72	0.31	4.83	Middle East	Nascent	Lower middle	Electoral autocracy
59	Kenya	0.75	14.11	29.84	0.07	12.15	7.75	0.31	2.30	Africa	Nascent	Lower middle	Electoral autocracy
60	Nigeria	2.74	0.00	50.10	0.45	2.06	7.75	0.33	1.38	Africa	Nascent	Lower middle	Electoral autocracy

- **Method 2 - using .head() method:**
  The df.head() method shows rows from top to bottom. By default, it shows the first 5 rows. We can also change the number of rows by specifying them in the parentheses.
  df.head(8): it will show the first 8 rows of the dataframe.

```
# method 2

df.head(8)
```

[19] ✓ 0.0s Python

	Country	Talent	Infrastructure	Operating Environment	Research	Development	Government Strategy	Commercial	Total score	Region	Cluster	Income group	Political regime
0	United States of America	100.00	94.02	64.56	100.00	100.00	77.39	100.00	100.00	Americas	Power players	High	Liberal democracy
1	China	16.51	100.00	91.57	71.42	79.97	94.87	44.02	62.92	Asia-Pacific	Power players	Upper middle	Closed autocracy
2	United Kingdom	39.65	71.43	74.65	36.50	25.03	82.82	18.91	40.93	Europe	Traditional champions	High	Liberal democracy
3	Canada	31.28	77.05	93.94	30.67	25.78	100.00	14.88	40.19	Americas	Traditional champions	High	Liberal democracy
4	Israel	35.76	67.58	82.44	32.63	27.96	43.91	27.33	39.89	Middle East	Rising stars	High	Liberal democracy
5	Singapore	39.38	84.30	43.15	37.67	22.55	79.82	15.07	38.67	Asia-Pacific	Rising stars	High	Electoral democracy
6	South Korea	14.54	85.23	68.86	26.66	77.25	87.50	5.41	38.60	Asia-Pacific	Rising stars	High	Liberal democracy
7	The Netherlands	33.83	81.99	88.05	25.54	30.17	62.35	4.97	36.35	Europe	Rising stars	High	Liberal democracy

- **method 3- using tail method:**
  The df.tail() method is similar to the .head method but shows the last 5 rows of the dataframe.
  df.tail(8) **: It will show the last 8 rows of the dataframe.**

```
# Method 3

df.tail(8)
```

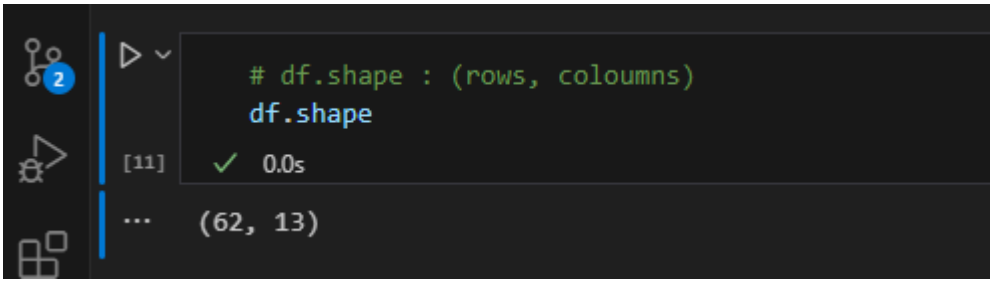
[20] ✓ 0.0s Python

	Country	Talent	Infrastructure	Operating Environment	Research	Development	Government Strategy	Commercial	Total score	Region	Cluster	Income group	Political regime
54	South Africa	4.61	45.73	58.43	0.83	7.52	0.00	2.03	9.71	Africa	Waking up	Upper middle	Electoral democracy
55	Morocco	3.36	44.88	60.17	1.46	0.05	15.90	0.10	8.87	Africa	Waking up	Lower middle	Closed autocracy
56	Armenia	6.69	37.84	58.40	0.28	0.33	14.40	1.37	8.49	Europe	Waking up	Upper middle	Electoral democracy
57	Sri Lanka	6.27	34.64	35.79	0.12	0.95	35.57	0.09	6.62	Asia-Pacific	Nascent	Lower middle	Electoral democracy
58	Egypt	1.11	38.84	0.00	2.08	1.54	68.72	0.31	4.83	Middle East	Nascent	Lower middle	Electoral autocracy
59	Kenya	0.75	14.11	29.84	0.07	12.15	7.75	0.31	2.30	Africa	Nascent	Lower middle	Electoral autocracy
60	Nigeria	2.74	0.00	50.10	0.45	2.06	7.75	0.33	1.38	Africa	Nascent	Lower middle	Electoral autocracy
61	Pakistan	8.00	2.43	12.48	2.17	1.09	13.92	0.27	0.00	Asia-Pacific	Nascent	Lower middle	Electoral autocracy

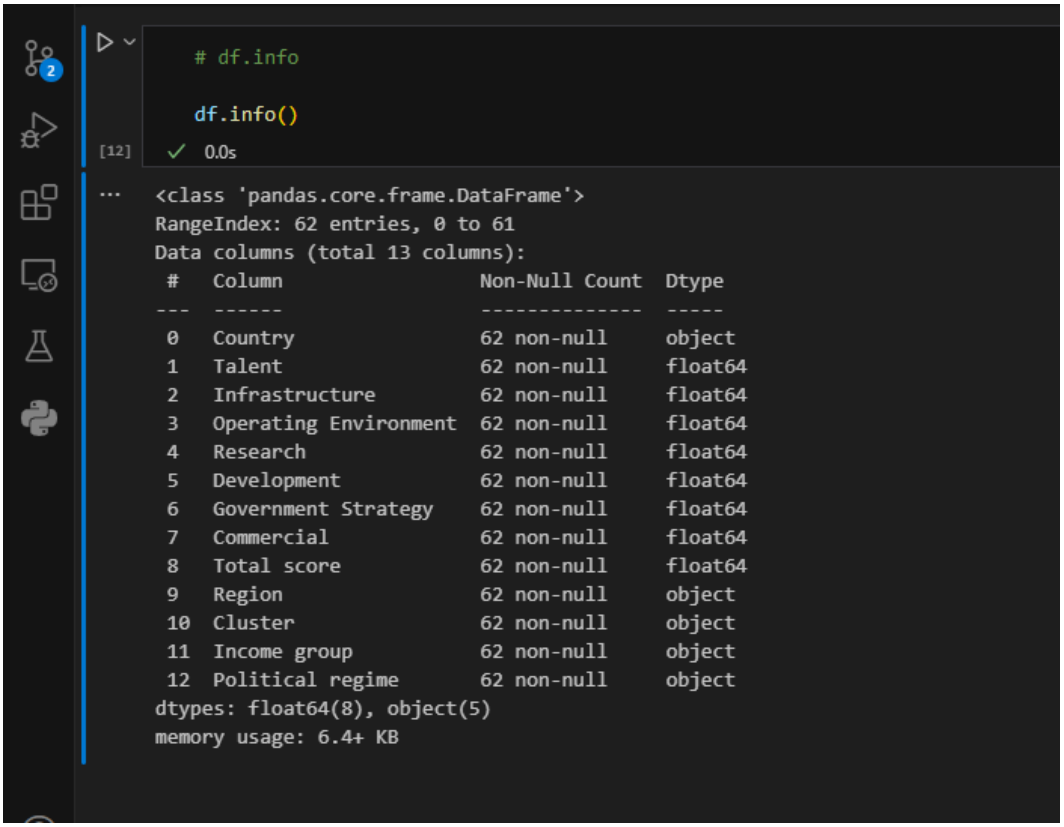
5. Exploring Data

- df.shape()
  The df.shape() function in Pandas is used to obtain the shape of a DataFrame. It returns a tuple representing the dimensionality of the DataFrame, where the first element of the tuple is the number of rows and the second element is the number of columns

  This function is particularly useful when you need to quickly understand the structure of the DataFrame, as it provides a concise way to retrieve the number of rows and columns without having to manually count them.



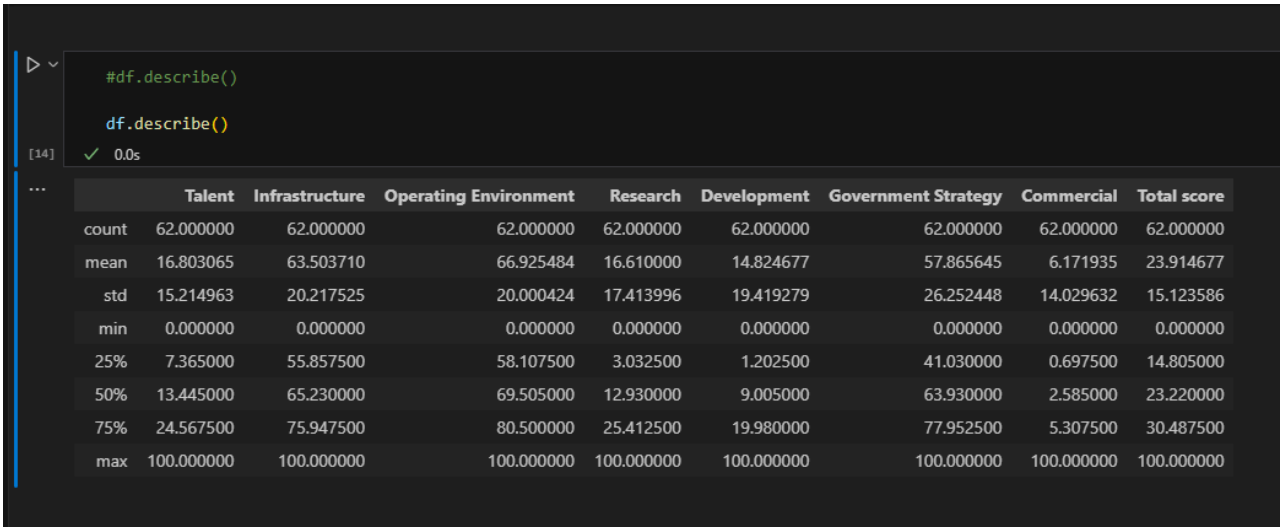
- df.info()
  By using df.info(), you can quickly gather essential details about the DataFrame, such as the number of columns, column labels, data types, memory usage, and the number of non-null values in each column. This can be particularly helpful for understanding the structure and content of the DataFrame at a glance.



- df.describe()

The df.describe() function in Pandas is used to generate descriptive statistics that summarize the central tendency, dispersion, and shape of a dataset's distribution. It provides a summary of the numerical columns in the DataFrame, including count, mean, standard deviation, minimum, maximum, and various quantiles.

This function is extremely useful for getting a quick overview of the distribution and characteristics of the numerical data within the DataFrame. It allows you to understand the range of values, the presence of outliers, and the general shape of the data distribution.



```
#df.describe()

df.describe()
```

[14] ✓ 0.0s

	Talent	Infrastructure	Operating Environment	Research	Development	Government Strategy	Commercial	Total score
count	62.000000	62.000000	62.000000	62.000000	62.000000	62.000000	62.000000	62.000000
mean	16.803065	63.503710	66.925484	16.610000	14.824677	57.865645	6.171935	23.914677
std	15.214963	20.217525	20.000424	17.413996	19.419279	26.252448	14.029632	15.123586
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	7.365000	55.857500	58.107500	3.032500	1.202500	41.030000	0.697500	14.805000
50%	13.445000	65.230000	69.505000	12.930000	9.005000	63.930000	2.585000	23.220000
75%	24.567500	75.947500	80.500000	25.412500	19.980000	77.952500	5.307500	30.487500
max	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000

- df.apply()
  The df.apply() function in Pandas is used to apply a function along an axis of the DataFrame. It allows you to apply a custom function to either the rows or columns of the DataFrame, depending on the specified axis. The function to be applied can be a built-in function, a lambda function, or a user-defined function.

  When using df.apply(), you can specify the axis along which the function should be applied (0 for index or row-wise, 1 for columns), and you can also pass additional positional and keyword arguments to the function if needed.

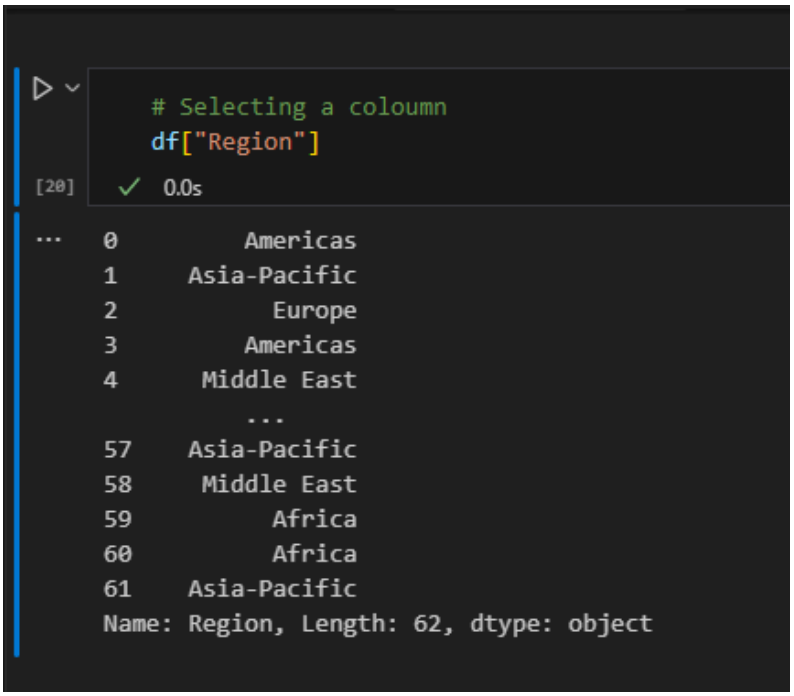
  Syntax: df.apply(function,axis) : axis (0 = rows) and (1 = coloumns)

- df.mean() : Returns the mean of every column
- df.corr() : Returns the correlation between columns in a DataFrame
- df.count() : Returns the number of non-null values in each DataFrame column
- df.max() : Returns the biggest value in every column
- df.min() : Returns the lowest value in every column
- df.median() : Returns the median of every column
- df.std() : Returns the standard deviation of every column

6. Selecting The Data

In order to review or conduct further analysis of the data, you may frequently need to choose only one piece or a specific subset of the data. These techniques will come in very handy:

- **Selecting a column**
  You can select a column by specifying its name within square brackets, like this: df['column_name'].



- **Selecting a group of columns**
  You can select a group of columns by specifying their names within a list and passing it to the DataFrame, like this: df[['column1', 'column2', 'column3']].

```
# Selecting a Group of Coloumns

df[["Region","Talent","Infrastructure"]]
```

[21] ✓ 0.0s

	Region	Talent	Infrastructure
0	Americas	100.00	94.02
1	Asia-Pacific	16.51	100.00
2	Europe	39.65	71.43
3	Americas	31.28	77.05
4	Middle East	35.76	67.58
...	...	...	...
57	Asia-Pacific	6.27	34.64
58	Middle East	1.11	38.84
59	Africa	0.75	14.11
60	Africa	2.74	0.00
61	Asia-Pacific	8.00	2.43

62 rows × 3 columns

- **Selecting a Row**
  You can Select a Row by df[row-no:row-no + 1]

```
#selecting a single row
df[0:1]
```

[18] ✓ 0.0s Python

	Country	Talent	Infrastructure	Operating Environment	Research	Development	Government Strategy	Commercial	Total score	Region	Cluster	Income group	Political regime
0	United States of America	100.0	94.02	64.56	100.0	100.0	77.39	100.0	100.0	Americas	Power players	High	Liberal democracy

- **Selecting a Group of Rows**
  You can Select a Row by df[start:end]

```
# selecting a group of rows
df[0:5]
```

[17] ✓ 0.0s

	Country	Talent	Infrastructure	Operating Environment	Research	Development	Government Strategy	Commerci
0	United States of America	100.00	94.02	64.56	100.00	100.00	77.39	100.0
1	China	16.51	100.00	91.57	71.42	79.97	94.87	44.0
2	United Kingdom	39.65	71.43	74.65	36.50	25.03	82.82	18.9
3	Canada	31.28	77.05	93.94	30.67	25.78	100.00	14.8
4	Israel	35.76	67.58	82.44	32.63	27.96	43.91	27.3

- **Using the .loc[] Method**
  The df.loc[] method in Pandas is used for label-based indexing or selecting data based on the labels of rows and columns. It allows for both single and multiple-label-based indexing. There are multiple ways in which you can use .loc[] method:

  1. **Selecting Single Row**

  2. **Selecting Multiple Rows**

```
# Selecting Multiple Rows
df.loc[0:2]
```

[28] ✓ 0.0s

	Country	Talent	Infrastructure	Operating Environment	Research	Development	Government Strategy	Commercial	Total score	Region
0	United States of America	100.00	94.02	64.56	100.00	100.00	77.39	100.00	100.00	Americas
1	China	16.51	100.00	91.57	71.42	79.97	94.87	44.02	62.92	Asia-Pacific
2	United Kingdom	39.65	71.43	74.65	36.50	25.03	82.82	18.91	40.93	Europe

3. Selecting Single Column

```
# Selecting Single Coloumn
df.loc[:,"Country"]
```

[29] ✓ 0.0s

```
0     United States of America
1                        China
2               United Kingdom
3                       Canada
4                       Israel
                ...
57                    Sri Lanka
58                        Egypt
59                        Kenya
60                      Nigeria
61                     Pakistan
Name: Country, Length: 62, dtype: object
```

4. Selecting Multiple Columns

```
# Selecting multiple Coloumns
df.loc[:,"Country":"Infrastructure"]
```

[31] ✓ 0.0s

	Country	Talent	Infrastructure
0	United States of America	100.00	94.02
1	China	16.51	100.00
2	United Kingdom	39.65	71.43
3	Canada	31.28	77.05
4	Israel	35.76	67.58
...	...	...	...
57	Sri Lanka	6.27	34.64
58	Egypt	1.11	38.84
59	Kenya	0.75	14.11
60	Nigeria	2.74	0.00
61	Pakistan	8.00	2.43

5. Selecting Multiple Rows and Columns

```
# Selecting multiple Coloumns and Rows
df.loc[3:6,"Country":"Research"]
```

[34] ✓ 0.0s

	Country	Talent	Infrastructure	Operating Environment	Research
3	Canada	31.28	77.05	93.94	30.67
4	Israel	35.76	67.58	82.44	32.63
5	Singapore	39.38	84.30	43.15	37.67
6	South Korea	14.54	85.23	68.86	26.66

6. Selection Based on Condition

- Selecting on One Condition

```python
# Selecting on One Condition

condition = df["Region"] == "Europe"
df[condition]
```

[36] ✓ 0.0s Python

	Country	Talent	Infrastructure	Operating Environment	Research	Development	Government Strategy	Commercial	Total score	Region	Cluster	Income group	Political regime
2	United Kingdom	39.65	71.43	74.65	36.50	25.03	82.82	18.91	40.93	Europe	Traditional champions	High	Liberal democracy
7	The Netherlands	33.83	81.99	88.05	25.54	30.17	62.35	4.97	36.35	Europe	Rising stars	High	Liberal democracy
8	Germany	27.63	77.22	70.22	35.84	24.79	84.65	8.29	36.04	Europe	Traditional champions	High	Liberal democracy
9	France	28.32	77.15	80.02	25.48	21.44	91.20	7.65	34.42	Europe	Traditional champions	High	Liberal democracy

- Selecting on More Than One Condition

```python
# Selecting on More Than One Condition

condition1 = df["Region"] == "Europe"
condition2 = df["Total score"] >= 35
df[condition1 & condition2]
```

[40] ✓ 0.0s Python

	Country	Talent	Infrastructure	Operating Environment	Research	Development	Government Strategy	Commercial	Total score	Region	Cluster	Income group	Political regime
2	United Kingdom	39.65	71.43	74.65	36.50	25.03	82.82	18.91	40.93	Europe	Traditional champions	High	Liberal democracy
7	The Netherlands	33.83	81.99	88.05	25.54	30.17	62.35	4.97	36.35	Europe	Rising stars	High	Liberal democracy
8	Germany	27.63	77.22	70.22	35.84	24.79	84.65	8.29	36.04	Europe	Traditional champions	High	Liberal democracy

- Selecting Specific Columns on a Condition

```python
# Selecting Specific Coloumns on a Condition

condition1 = df["Region"] == "Europe"
condition2 = df["Total score"] >= 35
df.loc[condition1 & condition2,["Country","Talent","Total score","Region"]]
```

[41] ✓ 0.0s

	Country	Talent	Total score	Region
2	United Kingdom	39.65	40.93	Europe
7	The Netherlands	33.83	36.35	Europe
8	Germany	27.63	36.04	Europe

- **Using the .iloc[] Method :**
  The df.iloc[] method in Pandas is used for integer-location-based indexing, allowing the selection of specific rows and columns using integer indices df.iloc[row-indexes, column indexes]

```
# .iloc[] Method

df.iloc[0:4,0:4]
```

[43] ✓ 0.0s

...

	Country	Talent	Infrastructure	Operating Environment
0	United States of America	100.00	94.02	64.56
1	China	16.51	100.00	91.57
2	United Kingdom	39.65	71.43	74.65
3	Canada	31.28	77.05	93.94