Exploratory data analysis (EDA)

- ✓ Exploratory Data Analysis (EDA) with Python involves analyzing and summarizing data to gain insights and understand its underlying patterns, relationships, and distributions using Python programming language.
- ✓ Exploratory Data Analysis (EDA) is a crucial step in the data analysis process that involves examining and visualizing data to gain a better understanding of its characteristics, uncover patterns, identify outliers, and formulate hypotheses.
- ✓ Python is a popular programming language for performing EDA due to its rich ecosystem of data analysis and visualization libraries.

Advantages of Exploratory Data Analysis (EDA)

Advantage	Description
Data Understanding	Provides a deep understanding of data characteristics and patterns.
Detects Data Issues	Identifies missing values, outliers, and data quality problems.
Feature Selection	Helps in selecting relevant features for modeling, improving model performance.
Hypothesis Formulation	Generates hypotheses for further testing and research.
Data Visualization	Uses charts and plots to communicate insights visually.
Enhances Data Quality	Leads to improved data quality through cleaning and preprocessing.
Reduces Modeling Errors	Reduces the risk of incorrect modeling assumptions.
Validates Assumptions	Validates assumptions made during data collection and analysis.
Optimizes Model Selection	Assists in choosing the most suitable model by understanding data behavior.
Saves Time and Resources	Prevents wasted efforts on inappropriate modeling or data issues.
Enhances Decision-Making	Provides valuable insights for informed decision-making.
Identifies Business Opportunities	Reveals potential opportunities or areas for improvement.
Promotes Data-Driven Culture	Encourages organizations to rely on data for decision-making.
Improves Communication	Enables better communication of data insights within teams and stakeholders.
Enhances Interpretability	Increases the interpretability of model results.

How Exploratory Data Analysis (EDA) is conducted using Python.

Step	Description	Python Libraries
Data Collection	Gather data from various sources (e.g., CSV, databases).	**pandas**, **numpy**, **sqlite3**, **requests** (for web data)
Data Cleaning	Handle missing values, duplicates, data type issues.	**pandas**
Summary Statistics	Calculate descriptive statistics (mean, median, etc.).	**pandas**
Data Visualization	Create plots and charts to visualize data.	**matplotlib**, **seaborn**, **plotly**, **bokeh**, **altair**
Feature Engineering	Create or transform features for analysis.	**pandas**, **numpy**, **scikit-learn**
Correlation Analysis	Explore relationships between variables.	**pandas**, **seaborn**
Outlier Detection	Identify and handle outliers.	**pandas**, **seaborn**, **scipy**

Hypothesis Testing	Validate or reject hypotheses.	**scipy**, **statsmodels**
Interactive Dashboards	Create dynamic data exploration tools.	**dash**, **streamlit**, **bokeh**
Documentation	Document findings, cleaning, transformations, and code in Jupyter Notebooks or other formats.	Markdown, Jupyter Notebooks
Reports/Presentations	Communicate insights through reports or presentations, incorporating visualizations.	Markdown, Jupyter Notebooks, **matplotlib**, **plotly**

Practical Case study to illustrate how to conduct Exploratory data analysis (EDA)

Using Python Language.

Some steps used to investigate data

1. Load the Data.
2. Duplicate values.
3. Find and Replace the Null values.
4. Basic information about data – EDA / Summary statistics i.e *mean, count, standard deviation, etc.* / Unique values in the data.
5. Visualize the Unique counts.

1. Load the Data.

This dataset contains hypothetical information about students, including their ID, age, gender, test scores, and whether they passed the exam:

	A	B	C	D	E
1	StudentID	Age	Gender	Test Score	PassedExam
2	1	18	Male	85	Yes
3	2	19	Female	92	Yes
4	3	20	Male	78	No
5	4	19	Female	88	Yes
6	5	20	Male	76	No
7	6	18	Male	90	Yes
8	7	19	Female	82	Yes
9	8	20	Female	68	No
10	9	18	Male	79	No
11	10	19	Male	94	Yes

```
import pandas as pd
wainaina=pd.read_csv('/content/wainaina.csv')
wainaina.head(10)
```

Line Number	Code	Explanation
1	import pandas as pd	Imports the Pandas library and assigns it the alias "pd" for easier use. Pandas is a Python library used for data manipulation and analysis.
2	wainaina =pd.read_csv('/content/wainaina.csv')	Reads a CSV (Comma-Separated Values) file named "wainaina.csv" located at the '/content/' path and stores it in a Pandas DataFrame called "wainaina." This line of code is used to load data from a CSV file into a DataFrame for further analysis.
3	wainaina.head(10)	Displays the first 10 rows of the "wainaina" DataFrame, showing a preview of the data. The head() method is used to view the top rows of a DataFrame.

+ Code + Text

	StudentID	Age	Gender	Test Score	PassedExam
0	1	18	Male	85	Yes
1	2	19	Female	92	Yes
2	3	20	Male	78	No
3	4	19	Female	88	Yes
4	5	20	Male	76	No
5	6	18	Male	90	Yes
6	7	19	Female	82	Yes
7	8	20	Female	68	No
8	9	18	Male	79	No
9	10	19	Male	94	Yes

2. Duplicate values.

```
import pandas as pd
wainaina=pd.read_csv('/content/wainaina.csv')
duplicate_rows = wainaina[wainaina.duplicated()
print("Duplicate Rows:")
print(duplicate_rows)
```

```
Duplicate Rows:
Empty DataFrame
Columns: [StudentID , Age, Gender, Test Score, PassedExam]
Index: []
```

✓ This code will identify and print any rows in the DataFrame that are exact duplicates of other rows.

- ✓ In the sample dataset provided, there are no duplicates, so running this code should result in an empty DataFrame for duplicate_rows.
- ✓ If you want to check for duplicates based on specific columns, you can pass column names to the subset parameter of the duplicated() method. For example:

```
[29]
import pandas as pd
wainaina=pd.read_csv('/content/wainaina.csv')
AgeDuplicate = wainaina[wainaina['Age'].duplicated()]
print("Duplicate Age:")
print(AgeDuplicate)
```

```
Duplicate Age:
   StudentID  Age  Gender  Test Score PassedExam
3          4   19  Female          88        Yes
4          5   20    Male          76         No
5          6   18    Male          90        Yes
6          7   19  Female          82        Yes
7          8   20  Female          68         No
8          9   18    Male          79         No
9         10   19    Male          94        Yes
```

Line Number	Code	Explanation
1	import pandas as pd	Imports the Pandas library and assigns it the alias "pd" for easier use. Pandas is a Python library used for data manipulation and analysis.
2	wainaina = pd.read_csv('/content/wainaina.csv')	Reads a CSV (Comma-Separated Values) file named "wainaina.csv" located at the '/content/' path and stores it in a Pandas DataFrame called "wainaina." This line of code loads data from a CSV file into a DataFrame.
3	AgeDuplicate = wainaina[wainaina['Age'].duplicated()]	Creates a new DataFrame called "AgeDuplicate" by filtering the "wainaina" DataFrame. It selects rows where the 'Age' column has duplicate values.
4	print("Duplicate Age:")	Prints a message indicating that the following lines will display duplicate values in the 'Age' column.
5	print(AgeDuplicate)	Displays the "AgeDuplicate" DataFrame, which contains rows with duplicate values in the 'Age' column.

To display the count of each duplicated value in a specific column of a Pandas DataFrame, you can use the value_counts() method.

```
print("\nUnique Values in 'AGE' Column:")
print(wainaina['Age'].value_counts())
```

```
Unique Values in 'AGE' Column:
19    4
18    3
20    3
Name: Age, dtype: int64
```

```
import pandas as pd
wainaina=pd.read_csv('/content/wainaina.csv')
AgeDuplicateCount = wainaina['Age'].duplicated(keep=False).value_counts()
print("Counts of Duplicated Student IDs:")
print(AgeDuplicateCount)
```

```
Counts of Duplicated Student IDs:
True    10
Name: Age, dtype: int64
```

Line Number	Code	Explanation
1	import pandas as pd	Imports the Pandas library and assigns it the alias "pd" for easier use. Pandas is a Python library used for data manipulation and analysis.
2	wainaina = pd.read_csv('/content/wainaina.csv')	Reads a CSV (Comma-Separated Values) file named "wainaina.csv" located at the '/content/' path and stores it in a Pandas DataFrame called "wainaina." This line of code loads data from a CSV file into a DataFrame.
3	AgeDuplicateCount = wainaina['Age'].duplicated(keep=False).value_counts()	Calculates the counts of duplicated values in the 'Age' column. The duplicated(keep=False) method marks all duplicates as True, and then value_counts() counts the occurrences of True (i.e., the duplicated values) and False (non-duplicated values). The result is a Series with counts.
4	print("Counts of Duplicated Student IDs:")	Prints a message indicating that the following lines will display the counts of duplicated student IDs in the 'Age' column.
5	print(AgeDuplicateCount)	Displays the counts of duplicated student IDs in the 'Age' column, showing how many times each duplicate age value appears in the dataset.

3. Find and Replace the Null values.

```
+ Code  + Text

import pandas as pd
wainaina=pd.read_csv('/content/wainaina.csv')
null_values = wainaina.isnull().sum()
# Display the null value counts for each column
print("Null Values in the DataSet:")
print(null_values)
```

```
Null Values in the DataSet:
StudentID     0
Age           0
Gender        0
Test Score    0
PassedExam    0
dtype: int64
```

Line Number	Code	Explanation
1	import pandas as pd	Imports the Pandas library and assigns it the alias "pd" for easier use. Pandas is a Python library used for data manipulation and analysis.
2	wainaina=pd.read_csv('/content/wainaina.csv')	Reads a CSV (Comma-Separated Values) file named "wainaina.csv" located at the '/content/' path and stores it in a Pandas DataFrame called "wainaina." This line of code loads data from a CSV file into a DataFrame.
3	null_values = wainaina.isnull().sum()	Calculates the number of null (missing) values in each column of the "wainaina" DataFrame using the isnull() method, and then sums up these counts using the sum() method. The result is a Series object with the count of null values for each column.
4	print("Null Values in the DataSet:")	Prints a message indicating that the following lines will display the count of null values in the dataset.
5	print(null_values)	Displays the count of null values in each column of the "wainaina" DataFrame, providing information about missing data in the dataset.

LOAD dataset below with missing values (I have named it wainaina2.csv)

	A	B	C	D	E
1	StudentID	Age	Gender	Test Score	PassedExam
2	1	18	Male	85	Yes
3	2	19	Female	92	Yes
4	3	20	Male	78	No
5	4	19	Female		Yes
6	5	20	Male	76	No
7	6	18	Male	90	Yes
8	7	19	Female	82	Yes
9	8		Female	68	No
10	9	18	Male	79	No
11	10	19	Male	94	Yes

```
[34] import pandas as pd
     wainaina=pd.read_csv('/content/wainaina2.csv')
     null_values = wainaina.isnull().sum()
     # Display the null value counts for each column
     print("Null Values in the DataSet:")
     print(null_values)
```

```
Null Values in the DataSet:
StudentID     0
Age           1
Gender        0
Test Score    1
PassedExam    0
dtype: int64
```

Check output Below (NaN): "NaN" stands for "Not a Number."

```
import pandas as pd
wainaina=pd.read_csv('/content/wainaina2.csv')
wainaina.head(10)
```

	StudentID	Age	Gender	TestScore	PassedExam
0	1	18.0	Male	85.0	Yes
1	2	19.0	Female	92.0	Yes
2	3	20.0	Male	78.0	No
3	4	19.0	Female	NaN	Yes
4	5	20.0	Male	76.0	No
5	6	18.0	Male	90.0	Yes
6	7	19.0	Female	82.0	Yes
7	8	NaN	Female	68.0	No
8	9	18.0	Male	79.0	No
9	10	19.0	Male	94.0	Yes

Now we replace the missing Values with a default data "0":

```
+ Code   + Text

import pandas as pd
wainaina=pd.read_csv('/content/wainaina2.csv')
FilledData = wainaina.fillna(0)
print("DataSet with Missing Values Replaced:")
print(FilledData)
```

```
DataSet with Missing Values Replaced:
   StudentID    Age  Gender  TestScore PassedExam
0          1  18.0    Male       85.0        Yes
1          2  19.0  Female       92.0        Yes
2          3  20.0    Male       78.0         No
3          4  19.0  Female        0.0        Yes
4          5  20.0    Male       76.0         No
5          6  18.0    Male       90.0        Yes
6          7  19.0  Female       82.0        Yes
7          8   0.0  Female       68.0         No
8          9  18.0    Male       79.0         No
9         10  19.0    Male       94.0        Yes
```

To fill in a specific row and colum value using at (Pandas DataFrame but use fillna() in NumPy float64 object) from the data I have loaded (Let us update TestScore):

```
import pandas as pd
wainaina=pd.read_csv('/content/wainaina2.csv')
print("Original DataFrame:")
print(wainaina)
row_index = 3  # Row index (0-based)
column_name = "TestScore"
new_value = "88"
wainaina.at[row_index, column_name] = new_value
print("\nModified DataFrame:")# Display the modified DataFrame
print(wainaina)
```

Line Number	Code	Explanation
1	import pandas as pd	Imports the Pandas library and assigns it the alias "pd" for easier use. Pandas is a Python library used for data manipulation and analysis.
2	wainaina=pd.read_csv('/content/wainaina2.csv')	Reads a CSV (Comma-Separated Values) file named "wainaina2.csv" located at the '/content/' path and stores it in a Pandas DataFrame called "wainaina." This line of code loads data from a CSV file into a DataFrame.
3	print("Original DataFrame:")	Prints a message indicating that the following lines will display the original DataFrame.
4	print(wainaina)	Displays the original "wainaina" DataFrame, showing its contents before any modifications.
5	row_index = 3	Assigns the value 3 to the variable row_index. This variable represents the row index (0-based) of the DataFrame where a modification will be made.

6	column_name = "TestScore"	Assigns the string "TestScore" to the variable column_name. This variable represents the name of the column where a modification will be made.
7	new_value = "88"	Assigns the string "88" to the variable new_value. This is the new value that will replace an existing value in the specified cell.
8	wainaina.at[row_index, column_name] = new_value	Modifies the DataFrame "wainaina" by setting the value in the cell at the specified row index (3) and column name ("TestScore") to the new value ("88"). This line updates a specific cell in the DataFrame.
9	print("\nModified DataFrame:")	Prints a message indicating that the following lines will display the modified DataFrame. The "\n" is used for a new line before the message.
10	print(wainaina)	Displays the modified "wainaina" DataFrame, showing its contents after the specified modification.

```
+ Code   + Text

0s
Original DataFrame:
   StudentID    Age  Gender  TestScore PassedExam
0          1  18.0    Male       85.0        Yes
1          2  19.0  Female       92.0        Yes
2          3  20.0    Male       78.0         No
3          4  19.0  Female        NaN        Yes
4          5  20.0    Male       76.0         No
5          6  18.0    Male       90.0        Yes
6          7  19.0  Female       82.0        Yes
7          8   NaN  Female       68.0         No
8          9  18.0    Male       79.0         No
9         10  19.0    Male       94.0        Yes

Modified DataFrame:
   StudentID    Age  Gender TestScore PassedExam
0          1  18.0    Male      85.0        Yes
1          2  19.0  Female      92.0        Yes
2          3  20.0    Male      78.0         No
3          4  19.0  Female        88        Yes
4          5  20.0    Male      76.0         No
5          6  18.0    Male      90.0        Yes
6          7  19.0  Female      82.0        Yes
7          8   NaN  Female      68.0         No
8          9  18.0    Male      79.0         No
9         10  19.0    Male      94.0        Yes
```

EXERCISE:

- Write a Python code to insert the Age of a student whose ID is 8.

4. Basic information about data – EDA / Summary statistics i.e mean, count, standard deviation, etc. / Unique values in the data.

```
import pandas as pd
wainaina=pd.read_csv('/content/wainaina.csv')
print("\nFirst 10 Rows of the DataFrame:")
print(wainaina.head(10))
# Display summary statistics for numeric columns
print("\nSummary Statistics:")
print(wainaina.describe())
# Display data types and non-null counts
print("\nData Types and Non-Null Counts:")
print(wainaina.info())
# Count unique values in the 'Passed Exam' column
print("\nUnique Values in 'Passed Exam' Column:")
print(wainaina['PassedExam'].value_counts())
# Count unique values in the 'Gender' column
print("\nUnique Values in 'Gender' Column:")
print(wainaina['Gender'].value_counts())
```

Line Number	Code	Explanation
1	import pandas as pd	Imports the Pandas library and assigns it the alias "pd" for easier use. Pandas is a Python library used for data manipulation and analysis.
2	wainaina = pd.read_csv('/content/wainaina.csv')	Reads a CSV (Comma-Separated Values) file named "wainaina.csv" located at the '/content/' path and stores it in a Pandas DataFrame called "wainaina." This line of code is used to load data from a CSV file into a DataFrame for further analysis.
4-5	print("\nFirst 10 Rows of the DataFrame:")	Prints a message indicating that the following lines will display the first 10 rows of the DataFrame. The "\n" is used for a new line before the message.
6	print(wainaina.head(10))	Displays the first 10 rows of the "wainaina" DataFrame using the head() method, which provides a preview of the data.
8-9	print("\nSummary Statistics:")	Prints a message indicating that the following lines will display summary statistics for numeric columns. The "\n" is used for a new line before the message.
10	print(wainaina.describe())	Displays summary statistics for numeric columns in the "wainaina" DataFrame using the describe() method. This includes statistics like count, mean, standard deviation, min, 25%, 50%, 75%, and max.
12-13	print("\nData Types and Non-Null Counts:")	Prints a message indicating that the following lines will display data types and non-null counts for each column. The "\n" is used for a new line before the message.
14	print(wainaina.info())	Displays data types and non-null counts for each column in the "wainaina" DataFrame using the info() method. It provides information on the number of non-null entries and the data type of each column.
16-17	print("\nUnique Values in 'Passed Exam' Column:")	Prints a message indicating that the following lines will count and display unique values in the 'Passed Exam' column. The "\n" is used for a new line before the message.

18	print(wainaina['Passed Exam'].value_counts())	Counts and prints unique values in the 'Passed Exam' column of the "wainaina" DataFrame using the value_counts() method. This is useful for categorical data to see how many times each unique value appears.
20-21	print("\nUnique Values in 'Gender' Column:")	Prints a message indicating that the following lines will count and display unique values in the 'Gender' column. The "\n" is used for a new line before the message.
22	print(wainaina['Gender'].value_counts())	Counts and prints unique values in the 'Gender' column of the "wainaina" DataFrame using the value_counts() method. This is useful for categorical data to see how many times each unique value appears.

OUTPUTS:

```
First 10 Rows of the DataFrame:
   StudentID   Age  Gender  Test Score PassedExam
0          1    18    Male          85        Yes
1          2    19  Female          92        Yes
2          3    20    Male          78         No
3          4    19  Female          88        Yes
4          5    20    Male          76         No
5          6    18    Male          90        Yes
6          7    19  Female          82        Yes
7          8    20  Female          68         No
8          9    18    Male          79         No
9         10    19    Male          94        Yes
```

```
Summary Statistics:
       StudentID        Age  Test Score
count   10.00000  10.000000   10.000000
mean     5.50000  19.000000   83.200000
std      3.02765   0.816497    8.134973
min      1.00000  18.000000   68.000000
25%      3.25000  18.250000   78.250000
50%      5.50000  19.000000   83.500000
75%      7.75000  19.750000   89.500000
max     10.00000  20.000000   94.000000
```

Explanation of each statistic:

1. **Count**: This represents the number of non-null values in each column. In this case, there are 10 non-null values for each column, indicating that there are 10 rows of data in your dataset.
2. **Mean (Average)**: The mean is the sum of all values in the column divided by the count. It represents the central tendency of the data. For "Student ID," the mean is 5.5; for "Age," the mean is 19.0; and for "Test Score," the mean is 83.2.
3. **Standard Deviation (std)**: The standard deviation measures the amount of variation or dispersion in the data. A higher standard deviation indicates more spread in the data. For "Student ID," the std is approximately 3.03; for "Age," the std is approximately 0.82; and for "Test Score," the std is approximately 8.13.
4. **Minimum (min)**: This is the smallest value in each column. For "Student ID," the smallest value is 1; for "Age," it's 18; and for "Test Score," it's 68.
5. **25th Percentile (25%)**: This is the value below which 25% of the data falls. It's also known as the first quartile. For "Student ID," the 25th percentile is 3.25; for "Age," it's 18.25; and for "Test Score," it's 78.25.
6. **50th Percentile (Median)**: This is the middle value of the data when it's sorted. It separates the higher half from the lower half of the data. For "Student ID," the median is 5.5; for "Age," it's 19.0; and for "Test Score," it's 83.5.
7. **75th Percentile (75%)**: This is the value below which 75% of the data falls. It's also known as the third quartile. For "Student ID," the 75th percentile is 7.75; for "Age," it's 19.75; and for "Test Score," it's 89.5.
8. **Maximum (max)**: This is the largest value in each column. For "Student ID," the largest value is 10; for "Age," it's 20; and for "Test Score," it's 94.

```
Data Types and Non-Null Counts:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   StudentID    10 non-null     int64
 1   Age          10 non-null     int64
 2   Gender       10 non-null     object
 3   Test Score   10 non-null     int64
 4   PassedExam   10 non-null     object
dtypes: int64(3), object(2)
memory usage: 528.0+ bytes
None
```

Explanation of output above:

- <class 'pandas.core.frame.DataFrame'>: This line indicates that the object being described is a Pandas DataFrame.
- RangeIndex: 10 entries, 0 to 9: This line provides information about the index of the DataFrame. It tells you that the DataFrame has a range index with 10 entries, ranging from 0 to 9.
- Data columns (total 5 columns):: This line indicates that there are a total of 5 columns in the DataFrame.
- The table below this line provides detailed information about each column:
    - Column: This column lists the names of the DataFrame's columns.
    - Non-Null Count: This column shows the number of non-null (non-missing) values in each column.
    - Dtype: This column displays the data type of each column.
- In your DataFrame, you have the following columns:
    1. Student ID: This column contains integer values, and all 10 entries are non-null.
    2. Age: This column also contains integer values, and all 10 entries are non-null.
    3. Gender: This column contains object data type, which typically indicates string or categorical values. All 10 entries are non-null.
    4. Test Score: This column contains integer values, and all 10 entries are non-null.
    5. PassedExam: This column contains object data type, indicating string or categorical values. All 10 entries are non-null.
- dtypes: int64(3), object(2): This line summarizes the data types present in the DataFrame. It tells you that there are three columns with data type int64 (integer) and two columns with data type object.
- memory usage: 528.0+ bytes: This line provides an estimate of the memory usage of the DataFrame, which can be helpful in understanding the memory footprint of your data.
- None: This is the return value of the wainaina.info() method. In a Jupyter Notebook or interactive environment, the method's output is displayed, but when executed in a script, it returns None.

```
Unique Values in 'Passed Exam' Column:
Yes    6
No     4
Name: PassedExam, dtype: int64

Unique Values in 'Gender' Column:
Male      6
Female    4
Name: Gender, dtype: int64
```

Explanation of output above:

Unique Values in 'Passed Exam' Column:

- Yes: This is one of the unique values found in the 'Passed Exam' column.
- No: This is another unique value found in the 'Passed Exam' column.

For the 'Passed Exam' column:

- There are 6 occurrences of the value 'Yes'.
- There are 4 occurrences of the value 'No'.
- The data type of this column is shown as int64, indicating that it's likely stored as numeric values, such as 1 for 'Yes' and 0 for 'No'.

This information tells you that in the 'Passed Exam' column, there are two unique categories: 'Yes' and 'No', and it shows how many times each category appears in the dataset.

Unique Values in 'Gender' Column:

- Male: This is one of the unique values found in the 'Gender' column.
- Female: This is another unique value found in the 'Gender' column.

For the 'Gender' column:

- There are 6 occurrences of the value 'Male'.
- There are 4 occurrences of the value 'Female'.
- The data type of this column is shown as int64, which may indicate that it's stored as numeric values (e.g., 1 for 'Male' and 0 for 'Female') or that it has been misclassified as int64 when it should be object (string).

This information tells you that in the 'Gender' column, there are two unique categories: 'Male' and 'Female', and it shows how many times each category appears in the dataset.

These counts of unique values can be helpful for understanding the distribution of categorical data within your dataset.

5. Visualize the Unique counts.

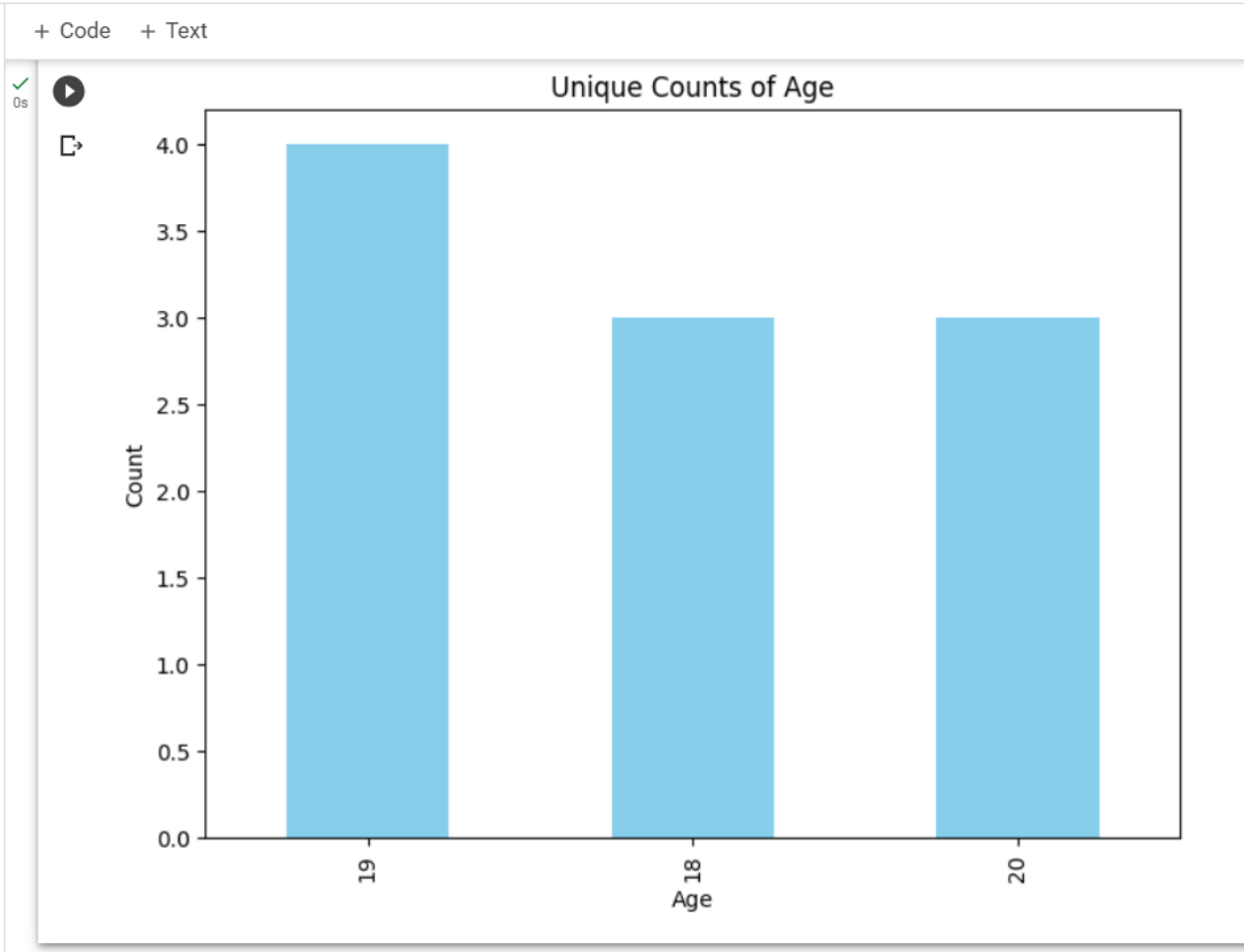
Using AGE column (BAR CHART):

```
# Import the necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
# Load the dataset from a CSV file
wainaina=pd.read_csv('/content/wainaina.csv')
# Specify the column you want to visualize
column_name = 'Age'
```

```
# Calculate unique counts of the specified column
unique_counts = wainaina[column_name].value_counts()
# Create a bar chart to visualize the unique counts
plt.figure(figsize=(8, 6))
unique_counts.plot(kind='bar', color='skyblue')
plt.title(f'Unique Counts of {column_name}')
plt.xlabel(column_name)
plt.ylabel('Count')
# Display the plot
plt.show()
```

Line Number	Code	Explanation
1	import pandas as pd	Imports the Pandas library and assigns it the alias "pd" for easier use. Pandas is a Python library used for data manipulation and analysis.
2	import matplotlib.pyplot as plt	Imports the Matplotlib library's pyplot module and assigns it the alias "plt" for creating plots and visualizations. Matplotlib is a popular data visualization library in Python.
3	wainaina=pd.read_csv('/content/wainaina.csv')	Reads a CSV (Comma-Separated Values) file named "wainaina.csv" located at the '/content/' path and stores it in a Pandas DataFrame called "wainaina." This line of code loads data from a CSV file into a DataFrame.
4	column_name = 'Age'	Assigns the string 'Age' to the variable column_name. This variable specifies the name of the column you want to visualize.
5	unique_counts = wainaina[column_name].value_counts()	Calculates the unique counts of values in the 'Age' column of the "wainaina" DataFrame using the value_counts() method. The result is stored in the variable unique_counts.
6	plt.figure(figsize=(8, 6))	Creates a new figure for the plot with a specified figure size of 8 inches in width and 6 inches in height.
7	unique_counts.plot(kind='bar', color='skyblue')	Creates a bar chart to visualize the unique counts of values in the 'Age' column. The kind='bar' argument specifies that a bar chart should be created, and color='skyblue' sets the color of the bars to sky blue.
8	plt.title(f'Unique Counts of {column_name}')	Sets the title of the plot dynamically based on the value of the column_name variable. In this case, it will be "Unique Counts of Age.". In Python, the f character before a string, as seen in f'Unique Counts of {column_name}', indicates an f-string. F-strings are a feature introduced in Python 3.6 and later versions for string formatting. They are used to embed expressions inside string literals, making it easier to create formatted strings with variables or values.
9	plt.xlabel(column_name)	Sets the label for the x-axis of the plot to the value of the column_name variable, which is "Age" in this case.
10	plt.ylabel('Count')	Sets the label for the y-axis of the plot as "Count," indicating the count of unique values.
11	plt.show()	Displays the plot on the screen.

OUTPUT:

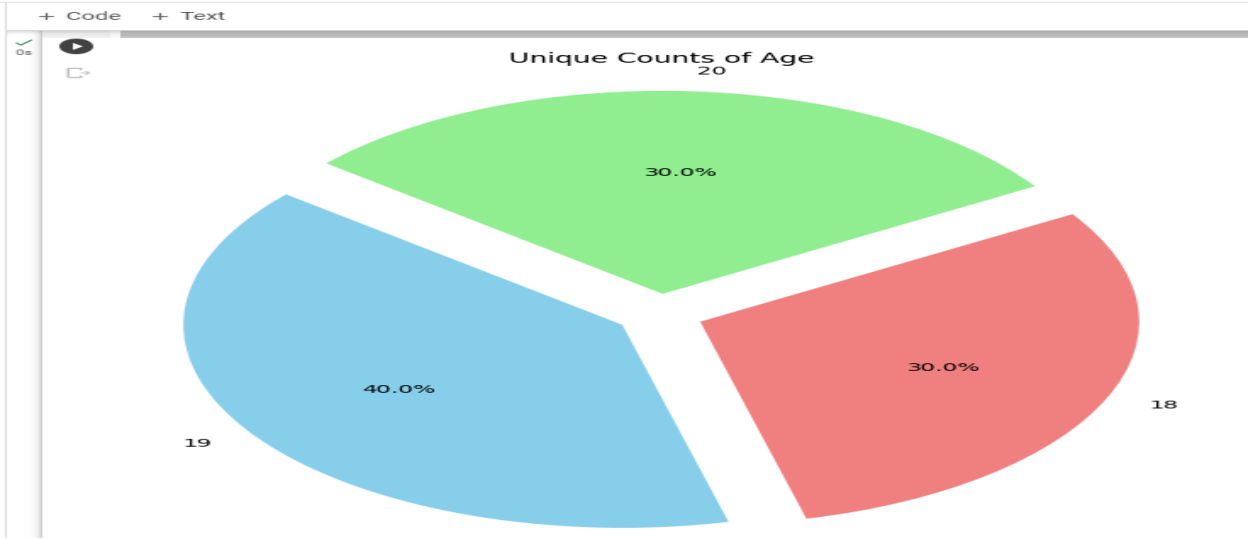


Using AGE column (PIE CHART):

```
# Import the necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
# Load the dataset from a CSV file
wainaina=pd.read_csv('/content/wainaina.csv')
# Specify the column you want to visualize
column_name = 'Age'  # Replace with the name of your categorical column
# Calculate unique counts of the specified column
unique_counts = wainaina[column_name].value_counts()
# Create a pie chart to visualize the unique counts
plt.figure(figsize=(8, 8))
plt.pie(unique_counts, labels=unique_counts.index, explode = (0.1, 0.1, 0.1),autopct='%1.1f%%',
startangle=140, colors=['skyblue', 'lightcoral', 'lightgreen'])
plt.title(f'Unique Counts of {column_name}')
# Display the plot
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```

Line Number	Code	Explanation
1	import pandas as pd	Imports the Pandas library and assigns it the alias "pd" for easier use. Pandas is a Python library used for data manipulation and analysis.
2	import matplotlib.pyplot as plt	Imports the Matplotlib library's pyplot module and assigns it the alias "plt" for creating plots and visualizations. Matplotlib is a popular data visualization library in Python.
3	wainaina=pd.read_csv('/content/wainaina.csv')	Reads a CSV (Comma-Separated Values) file named "wainaina.csv" located at the '/content/' path and stores it in a Pandas DataFrame called "wainaina." This line of code loads data from a CSV file into a DataFrame.
4	column_name = 'Age'	Specifies the name of the column you want to visualize, in this case, 'Age'. You can replace 'Age' with the name of your categorical column.
5	unique_counts = wainaina[column_name].value_counts()	Calculates the unique counts of values in the specified column ('Age') of the "wainaina" DataFrame using the value_counts() method. The result is stored in the unique_counts variable.
6	plt.figure(figsize=(8, 8))	Creates a new figure for the pie chart with a specified figure size of 8 inches in width and 8 inches in height.
7	plt.pie(unique_counts, labels=unique_counts.index, explode=(0.1, 0.1, 0.1), autopct='%1.1f%%', startangle=140, colors=['skyblue', 'lightcoral', 'lightgreen'])	Creates a pie chart to visualize the unique counts of values. It uses the unique_counts variable as the data, specifies labels, adds an explode effect (separating slices), displays the percentage labels with one decimal point, sets the starting angle of the pie chart, and assigns colors to the slices.
8	plt.title(f'Unique Counts of {column_name}')	Sets the title of the pie chart dynamically based on the value of the column_name variable. In this case, it will be "Unique Counts of Age."
9	plt.axis('equal')	Ensures that the aspect ratio of the pie chart is equal, so it appears as a circle.
10	plt.show()	Displays the pie chart on the screen.

OUTPUT:



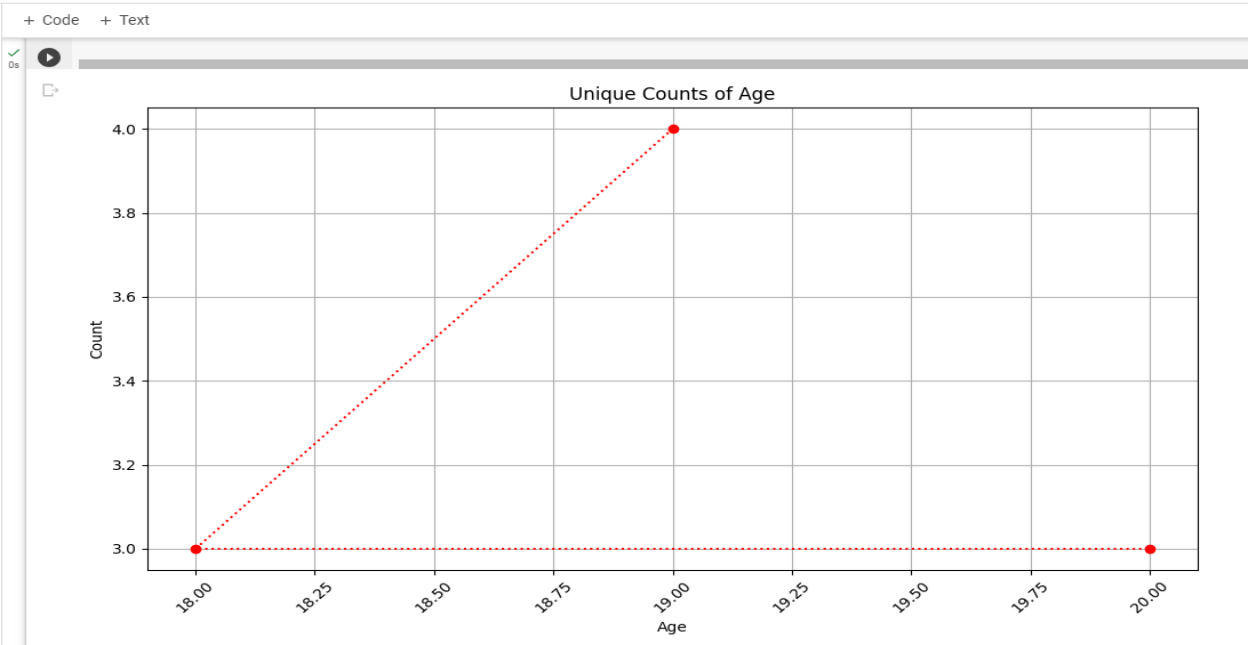
Using AGE column (LINE GRAPH):

```
# Import the necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
# Load the dataset from a CSV file
wainaina=pd.read_csv('/content/wainaina.csv')
# Specify the column you want to visualize
column_name = 'Age'  # Replace with the name of your categorical column
# Calculate unique counts of the specified column
unique_counts = wainaina[column_name].value_counts()
# Create a line graph to visualize the unique counts
plt.figure(figsize=(10, 6))
plt.plot(unique_counts.index, unique_counts.values, marker='o', linestyle='dotted',color='red')
plt.title(f'Unique Counts of {column_name}')
plt.xlabel(column_name)
plt.ylabel('Count')
# Rotate x-axis labels for better readability (optional)
plt.xticks(rotation=45)
# Display the plot
plt.grid(True)
plt.tight_layout()
plt.show()
```

Line Number	Code	Explanation
1	import pandas as pd	Imports the Pandas library and assigns it the alias "pd" for easier use. Pandas is a Python library used for data manipulation and analysis.
2	import matplotlib.pyplot as plt	Imports the Matplotlib library's pyplot module and assigns it the alias "plt" for creating plots and visualizations. Matplotlib is a popular data visualization library in Python.
4	wainaina=pd.read_csv('/content/wainaina.csv')	Reads a CSV (Comma-Separated Values) file named "wainaina.csv" located at the '/content/' path and stores it in a Pandas DataFrame called "wainaina." This line of code loads data from a CSV file into a DataFrame.
7	column_name = 'Age'	Specifies the name of the column you want to visualize, in this case, 'Age'. You can replace 'Age' with the name of your categorical column.
9	unique_counts = wainaina[column_name].value_counts()	Calculates the unique counts of values in the specified column ('Age') of the "wainaina" DataFrame using the value_counts() method. The result is stored in the unique_counts variable.
11-12	plt.figure(figsize=(10, 6))	Creates a new figure for the line graph with a specified figure size of 10 inches in width and 6 inches in height.
13	plt.plot(unique_counts.index, unique_counts.values, marker='o', linestyle='dotted', color='red')	Creates a line graph to visualize the unique counts of values. It uses the index of the unique_counts Series for the x-axis, the values for the y-axis, adds circular markers at data points, uses a dotted line style, and sets the line color to red.
14	plt.title(f'Unique Counts of {column_name}')	Sets the title of the line graph dynamically based on the value of the column_name variable. In this case, it will be "Unique Counts of Age."
15	plt.xlabel(column_name)	Sets the label for the x-axis of the plot to the value of the column_name variable, which is "Age" in this case.

16	plt.ylabel('Count')	Sets the label for the y-axis of the plot as "Count," indicating the count of unique values.
18	plt.xticks(rotation=45)	Optional: Rotates the x-axis labels by 45 degrees for better readability.
20	plt.grid(True)	Adds grid lines to the plot for reference.
21	plt.tight_layout()	Ensures that the plot layout is adjusted to fit all elements properly.
22	plt.show()	Displays the line graph on the screen.

OUTPUT:



OTHERS:

1. **Know the datatypes.**

```
+ Code  + Text
import pandas as pd
wainaina=pd.read_csv('/content/wainaina.csv')
df = pd.DataFrame(wainaina)
# Get the data types of each column
data_types = df.dtypes
# Display the data types
print("Data Types of DataFrame Columns:")
print(data_types)
```

```
Data Types of DataFrame Columns:
StudentID       int64
Age             int64
Gender         object
Test Score      int64
PassedExam     object
dtype: object
```

Line Number	Code	Explanation
1	import pandas as pd	Imports the Pandas library and assigns it the alias "pd" for easier use. Pandas is a Python library used for data manipulation and analysis.
2	wainaina=pd.read_csv('/content/wainaina.csv')	Reads a CSV (Comma-Separated Values) file named "wainaina.csv" located at the '/content/' path and stores it in a Pandas DataFrame called "wainaina." This line of code loads data from a CSV file into a DataFrame.
3	df = pd.DataFrame(wainaina)	Creates a new DataFrame called "df" by converting the existing "wainaina" DataFrame into a new DataFrame. This step is redundant and not necessary; you can work directly with the "wainaina" DataFrame without creating a new one.
5	data_types = df.dtypes	Uses the .dtypes attribute to retrieve and store the data types of each column in the DataFrame "df." The result is a Series object containing data types.
7-8	print("Data Types of DataFrame Columns:")	Prints a message indicating that the following lines will display the data types of DataFrame columns.
9	print(data_types)	Displays the data types of each column in the DataFrame. This provides information about the data type (e.g., int64, object) for each column in the DataFrame.
10	plt.show()	This line attempts to display a plot, but there is no preceding code to generate a plot. It seems to be unrelated to the previous DataFrame operations and can be removed unless it is part of a larger script that includes plotting.

2. **Filter the Data.**

+ Code + Text

```python
import pandas as pd
wainaina=pd.read_csv('/content/wainaina.csv')
# Filter data based on a condition (e.g., age greater than 25)
filtered= wainaina[wainaina['TestScore'] > 90]
# Display the filtered DataFrame
print("Filtered DataFrame:")
print(filtered)
```

```
Filtered DataFrame:
   StudentID   Age  Gender  TestScore PassedExam
1          2   19  Female         92        Yes
9         10   19    Male         94        Yes
```

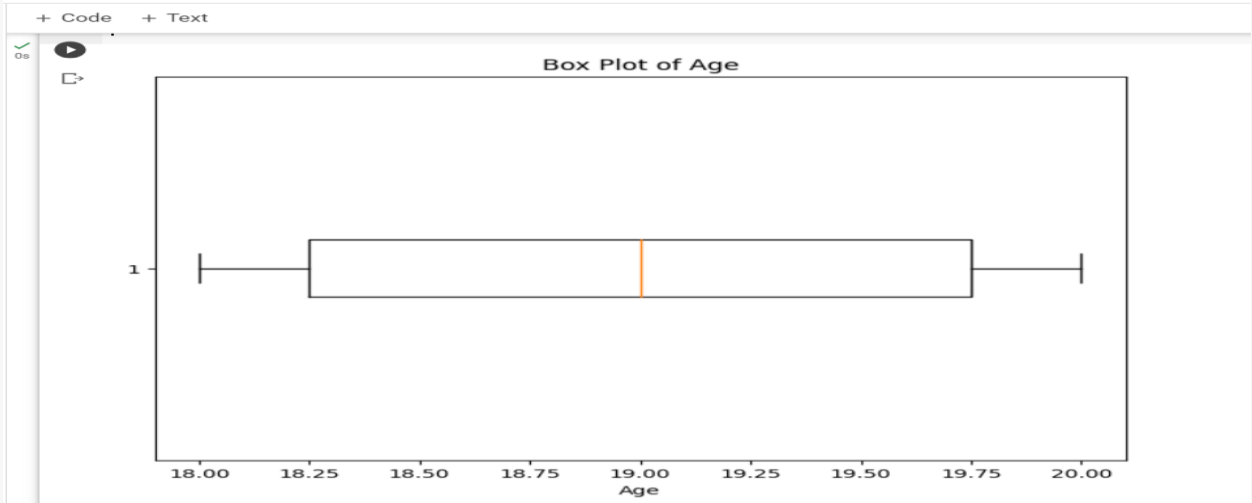
Line Number	Code	Explanation
1	import pandas as pd	Imports the Pandas library and assigns it the alias "pd" for easier use. Pandas is a Python library used for data manipulation and analysis.
2	wainaina=pd.read_csv('/content/wainaina.csv')	Reads a CSV (Comma-Separated Values) file named "wainaina.csv" located at the '/content/' path and stores it in a Pandas DataFrame called "wainaina." This line of code loads data from a CSV file into a DataFrame.

3	filtered= wainaina[wainaina['TestScore'] > 90]	Filters the data in the "wainaina" DataFrame based on a condition: in this case, selecting rows where the 'TestScore' column has values greater than 90. The filtered result is stored in a new DataFrame called "filtered."
4	print("Filtered DataFrame:")	Prints a message indicating that the following lines will display the filtered DataFrame.
5	print(filtered)	Displays the "filtered" DataFrame, which contains only the rows where the 'TestScore' is greater than 90. This allows you to view the subset of data that meets the specified condition.

3. **A quick box plots.**

```
# Import the necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
# Load the dataset from a CSV file
wainaina=pd.read_csv('/content/wainaina.csv')
# Create a box plot for a specific column (e.g., 'Age' in this case)
plt.figure(figsize=(8, 6))
plt.boxplot(wainaina['Age'], vert=False)
plt.title('Box Plot of Age')
plt.xlabel('Age')
plt.show()
```

Line Number	Code	Explanation
1	import pandas as pd	Imports the Pandas library and assigns it the alias "pd" for easier use. Pandas is a Python library used for data manipulation and analysis.
2	import matplotlib.pyplot as plt	Imports the Matplotlib library's pyplot module and assigns it the alias "plt" for creating plots and visualizations. Matplotlib is a popular data visualization library in Python.
3	wainaina=pd.read_csv('/content/wainaina.csv')	Reads a CSV (Comma-Separated Values) file named "wainaina.csv" located at the '/content/' path and stores it in a Pandas DataFrame called "wainaina." This line of code loads data from a CSV file into a DataFrame.
4	plt.figure(figsize=(8, 6))	Creates a new figure for the box plot with a specified figure size of 8 inches in width and 6 inches in height.
5	plt.boxplot(wainaina['Age'], vert=False)	Creates a box plot for the 'Age' column in the "wainaina" DataFrame. The vert=False argument specifies that the boxes should be horizontally oriented.
6	plt.title('Box Plot of Age')	Sets the title of the box plot as "Box Plot of Age."
7	plt.xlabel('Age')	Sets the label for the x-axis of the plot as "Age."
8	plt.show()	Displays the box plot on the screen.

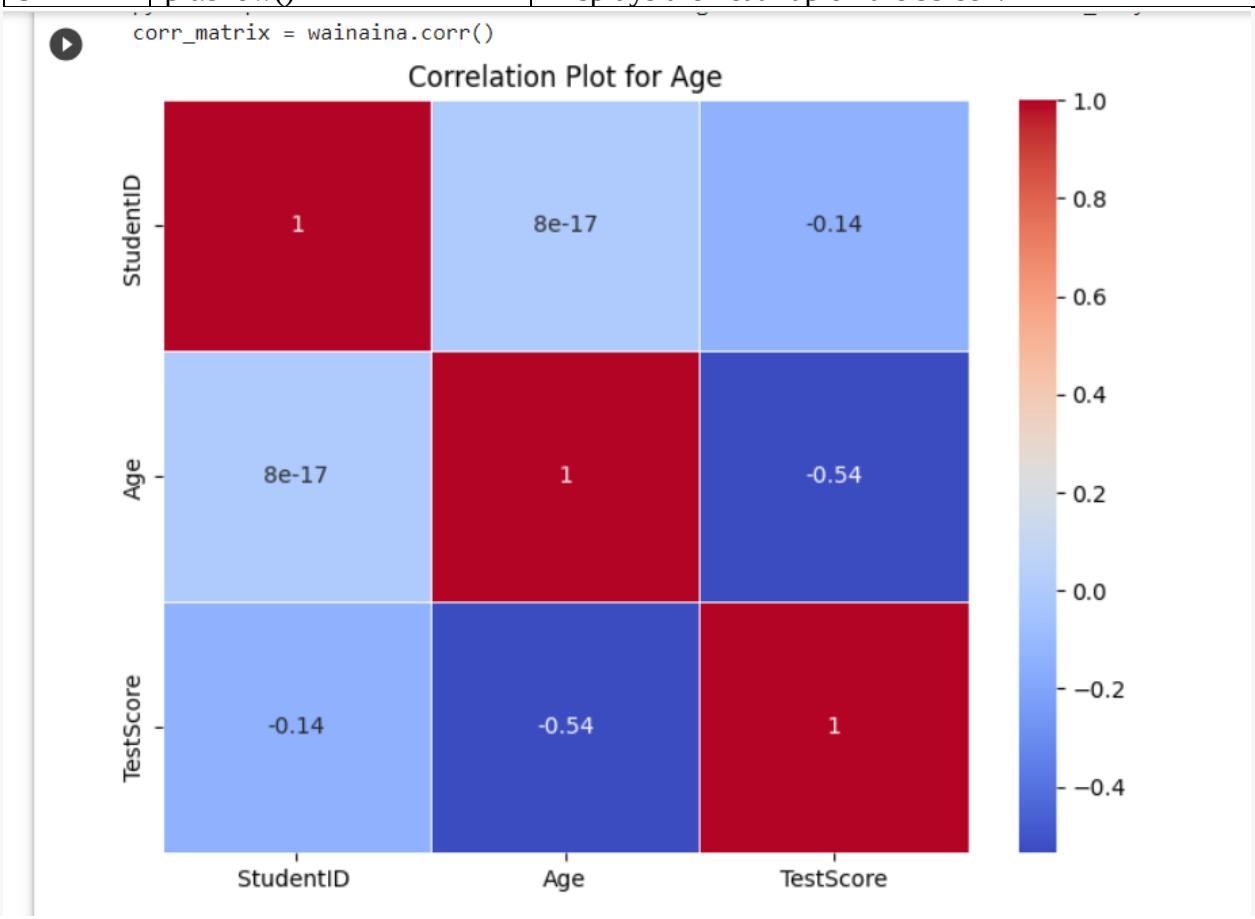


4. **Correlation Plot – EDA.**

```
# Import the necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# Load the dataset from a CSV file
wainaina=pd.read_csv('/content/wainaina.csv')
corr_matrix = wainaina.corr()
# Create a heatmap to visualize the correlation
plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=.5)
plt.title('Correlation Plot for My Data')
plt.show()
```

Line Number	Code	Explanation
1	import pandas as pd	Imports the Pandas library and assigns it the alias "pd" for easier use. Pandas is a Python library used for data manipulation and analysis.
2	import matplotlib.pyplot as plt	Imports the Matplotlib library's pyplot module and assigns it the alias "plt" for creating plots and visualizations. Matplotlib is a popular data visualization library in Python.
3	import seaborn as sns	Imports the Seaborn library and assigns it the alias "sns." Seaborn is a data visualization library that works well with Pandas and Matplotlib, providing high-level functions for creating attractive plots.
4	wainaina=pd.read_csv('/content/wainaina.csv')	Reads a CSV (Comma-Separated Values) file named "wainaina.csv" located at the '/content/' path and stores it in a Pandas DataFrame called "wainaina." This line of code loads data from a CSV file into a DataFrame.
5	corr_matrix = wainaina.corr()	Calculates the correlation matrix for the columns in the "wainaina" DataFrame using the .corr() method. The result is stored in the "corr_matrix" DataFrame. This matrix shows how each numeric column relates to every other numeric column in terms of correlation.

6	plt.figure(figsize=(8, 6))	Creates a new figure for the heatmap with a specified figure size of 8 inches in width and 6 inches in height.
7	sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=.5)	Creates a heatmap to visualize the correlation matrix. The corr_matrix is used as data, and several optional parameters are set: annot=True adds numeric annotations to the cells, cmap='coolwarm' sets the color map, and linewidths=.5 adds lines between cells for better separation.
8	plt.title('Correlation Plot for My Data')	Sets the title of the heatmap as "Correlation Plot for My Data."
9	plt.show()	Displays the heatmap on the screen.



This is the correlation matrix with the range from +1 to -1 where +1 is highly and positively correlated and -1 will be highly negatively correlated.

NB/ # Create a correlation table for selected columns (including 'Age')

```
corr_matrix = wainaina[['Age']].corr()
```

```
+ Code  + Text

[82] # Import the necessary libraries
import pandas as pd
# Load your dataset from a CSV file
wainaina=pd.read_csv('/content/wainaina.csv')
correlationtable = wainaina.corr()
# Display the correlation table
print("Correlation Table for My Data:")
print(correlationtable)
```

```
Correlation Table for My Data:
              StudentID           Age  TestScore
StudentID   1.000000e+00  7.984130e-17   -0.14436
Age         7.984130e-17  1.000000e+00   -0.53530
TestScore  -1.443596e-01 -5.352997e-01    1.00000
```

Line Number	Code	Explanation
1	import pandas as pd	Imports the Pandas library and assigns it the alias "pd" for easier use. Pandas is a Python library used for data manipulation and analysis.
2	wainaina=pd.read_csv('/content/wainaina.csv')	Reads a CSV (Comma-Separated Values) file named "wainaina.csv" located at the '/content/' path and stores it in a Pandas DataFrame called "wainaina." This line of code loads data from a CSV file into a DataFrame.
3	correlationtable = wainaina.corr()	Calculates the correlation matrix for the columns in the "wainaina" DataFrame using the .corr() method. The result is stored in the "correlationtable" DataFrame. This matrix shows how each numeric column relates to every other numeric column in terms of correlation.
4	print("Correlation Table for My Data:")	Prints a message indicating that the following lines will display the correlation table.
4	print(correlationtable)	Displays the correlation table, which contains the correlation coefficients between all pairs of numeric columns in the DataFrame. This table provides insights into the relationships between variables.

Exploratory Data Analysis – EDA Summary

- EDA is applied to **investigate** the data and **summarize** the key insights.
- It will give you the basic understanding of your data, it's **distribution**, **null values** and much more.
- You can either explore data **using graphs** or through some python functions.
- There will be two type of analysis. **Univariate and Bivariate**. In the **univariate**, *you will be analyzing a single attribute*. But in the **bivariate**, you *will be analyzing an attribute with the target attribute.*
- In the **non-graphical approach**,

- You will be using functions such as *shape, summary, describe, is null, info, datatypes and more*.

- In the **graphical approach,**
  - You will *be using plots such as scatter, box, bar, density and correlation plots*

Revision Questions Exploratory Data Analysis (EDA)

1) What is the Difference between Univariate, Bivariate, and Multivariate analysis? in EDA analysis.
2) During the data preprocessing step, how should one treat missing/null values? How will you deal with them?
3) What is an outlier and how to identify them?
4) Using the data set below, implement the concepts discussed in this lesson:

	A	B	C	D
1	Employee	Gender	Department	Salary
2	Emily Johnson	Female	Human Resources Department	6747
3	Benjamin Smith	Male	Marketing Department	1011
4	Sophia Williams	Female	Finance Department	2109
5	William Brown	Male	IT Department	1260
6	Olivia Davis	Female	Sales Department	2583
7	James Wilson	Male	Human Resources Department	8284
8	Ava Martinez	Female	Marketing Department	3989
9	Liam Anderson	Male	Finance Department	7416
10	Mia Thompson	Female	IT Department	7625
11	Ethan Rodriguez	Male	Sales Department	7054
12	Isabella Garcia	Female	Human Resources Department	4522
13	Noah Martinez	Male	Marketing Department	2028
14	Emma Johnson	Female	Finance Department	8770
15	Michael Lee	Male	IT Department	5708
16	Harper Taylor	Female	Sales Department	7693