DATA ANALYTICS WITH PYTHON: FROM BASICS TO FINANCIAL INSIGHTS

Understanding Data Analytics, Its Applications, Python Libraries

CONTENTS

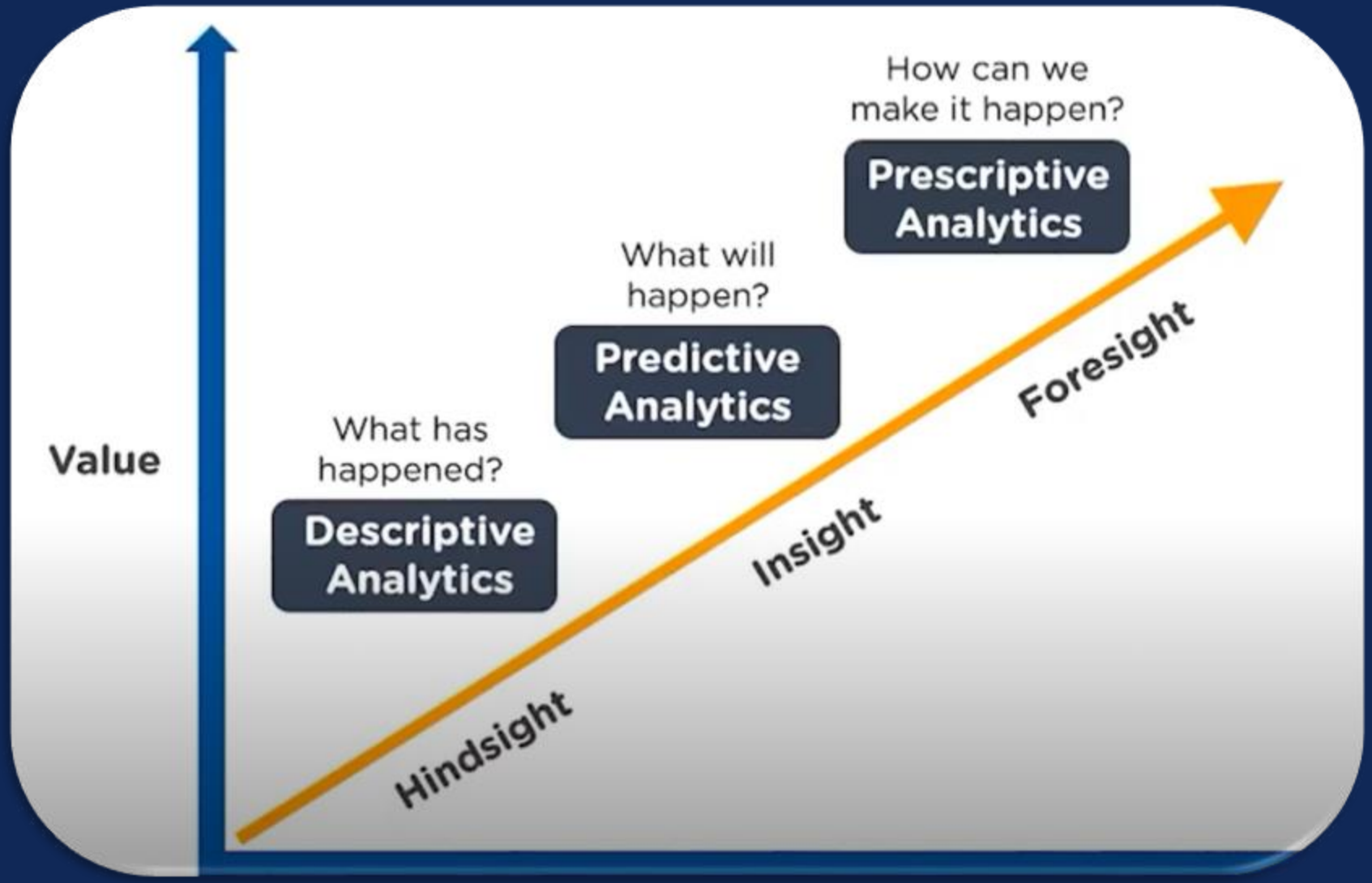
WHAT IS DATA ANALYTICS?

Data analytics is the process of extracting insights from raw data to make informed decisions. It involves collecting, processing, analyzing, and visualizing data to uncover patterns and trends.

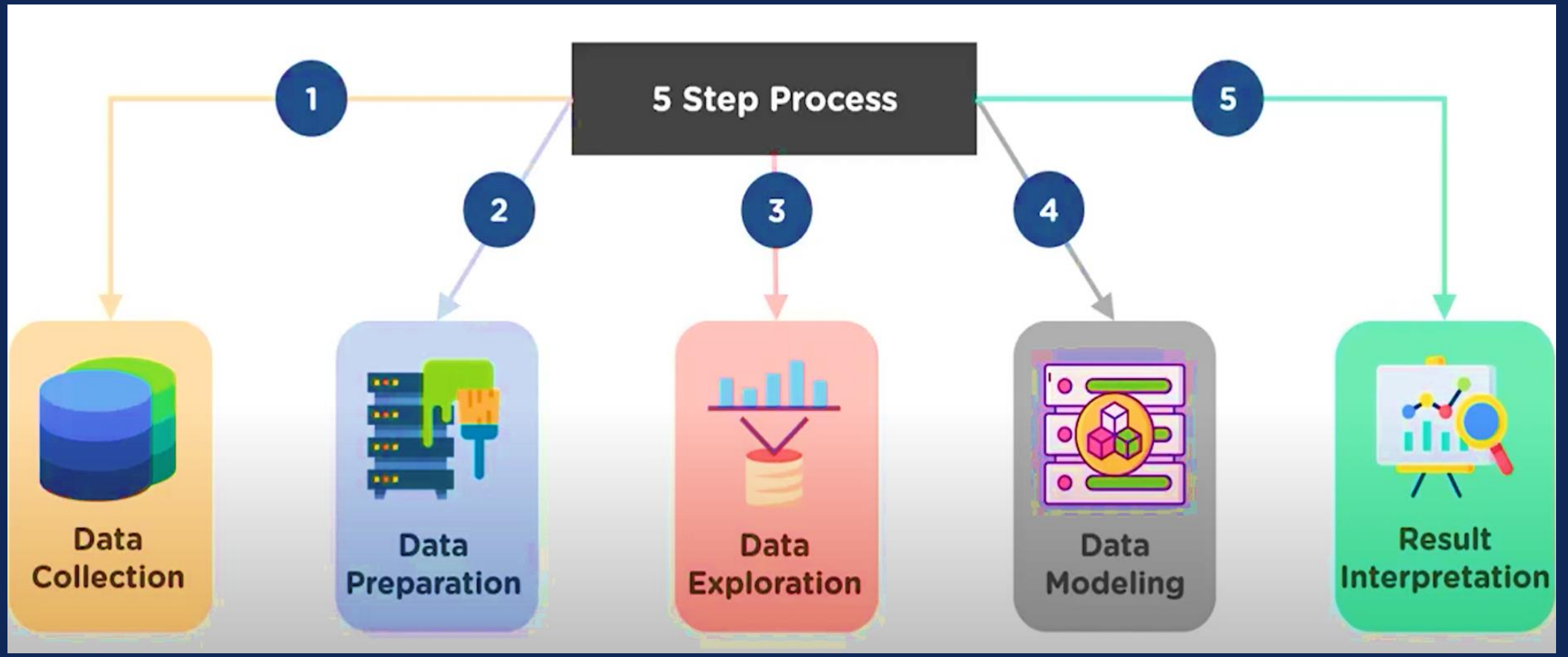
WHERE IS DATA ANALYTICS USED?

Fraud Analysis	Healthcare	Inventory Management	Delivery Logistics	Targeted Marketing
It detecting suspicious transactions in banking.	It predicting diseases, optimizing patient care.	It tracking stock levels & demand forecasting.	It optimizing routes, reducing delays.	It understanding customer behavior & improving ads.

TYPES OF DATA ANALYTICS



DATA ANALYTICS PROCESS STEPS



Gathering data from various sources.	Cleaning & formatting data.	Identifying trends & patterns.	Applying algorithms to analyze data.	Extracting insights for decision-making.

WHY PYTHON FOR DATA ANALYTICS?

Easy to Learn

Simple syntax & beginner-friendly.

Scalable & Flexible

Works with big data, AI, & ML.

Huge Collection of Libraries

Powerful tools for data analysis.

Graphics & Visualization

Supports Matplotlib & Seaborn.

Strong Community Support

Large developer base for help & guidance.

PYTHON LIBRARIES FOR DATA ANALYTICS

- **NumPy** – Supports numerical computing & multi-dimensional arrays.
- **Pandas** – Handles structured data, missing values, & mathematical operations.
- **Matplotlib** – Used for plotting & interactive visualizations.
- **SciPy** – Advanced scientific & technical computing.
- **Scikit-Learn** – Machine learning & predictive modelling.

NUMPY IN DATA ANALYTICS

Array Operations	• Type, Shape, Indexing, Slicing, 2D & 3D Arrays.
Random Numbers	• rand, randint, zeros, ones.
Mathematical Operations	• sum, min, max, mean, std, var.
Sorting & Reshaping	• sort, transpose, flatten, concatenate.
Set Operations	• intersect1d, union1d, setdiff1d.

ARRAY OPERATIONS

```
import numpy as np
a = np.array([1, 2, 3, 4])
print(type(a))
```

✓ 0.0s

```
<class 'numpy.ndarray'>
```

np.array() creates an array, and **type()** checks its type.

```
import numpy as np
a = np.array([1, 2, 3, 4])
print(a.shape)
```

✓ 0.0s

```
(4,)
```

.shape returns the number of rows and columns in an array.

Arrays use zero-based **indexing**

```
import numpy as np
a = np.array([1, 2, 3, 4])
print(a[2])
```

✓ 0.0s

```
3
```

Slicing extracts specific portions.

```
import numpy as np
a = np.array([1, 2, 3, 4])
print(a[1:3])
```

✓ 0.0s

```
[2 3]
```

```
import numpy as np
a = np.array([[1, 2, 3], [4, 5, 6]])
print(a[1,1])
```

✓ 0.0s

```
5
```

2D Array

```
import numpy as np
a = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
print(a.shape)
```

✓ 0.0s

```
(2, 2, 2)
```

3D Array

RANDOM NUMBERS

rand(n) generates **n** random numbers between 0 and 1.

```
np.random.rand(2)
```

✓ 0.0s

```
array([0.1897767, 0.1902594])
```

randint(min, max, size) generates random integers.

```
np.random.randint(10, 50, 3)
```

✓ 0.0s

```
array([18, 35, 25], dtype=int32)
```

ones() creates an array filled with 1s.

```
np.zeros((3, 2))
```

✓ 0.0s

```
array([[0., 0.],
       [0., 0.],
       [0., 0.]])
```

zeros() creates an array filled with 0s

```
np.ones((2, 3))
```

✓ 0.0s

```
array([[1., 1., 1.],
       [1., 1., 1.]])
```

MATHEMATICAL OPERATIONS

sum() adds all elements, **min()** finds the smallest, and **max()** finds the largest.

```
a = np.array([2, 4, 6, 8, 0])
print('Sum: ', np.sum(a))
print('Minimum: ', np.min(a))
print('Maximum: ', np.max(a))
```

✓ 0.0s

```
Sum:  20
Minimum:  0
Maximum:  8
```

mean() calculates average, **std()** measures spread, and **var()** measures variance.

```
a = np.array([2, 4, 6, 8, 0])
print('Mean: ', np.mean(a))
print('Standard Deviation: ', np.std(a))
print('Variance: ', np.var(a))
```

✓ 0.0s

```
Mean:  4.0
Standard Deviation:  2.8284271247461903
Variance:  8.0
```

SORTING & RESHAPING

sort() arranges numbers in ascending order.

```
a = np.array([5, 3, 8, 1])
print(np.sort(a))
```

✓ 0.0s

```
[1 3 5 8]
```

.T swaps rows and columns of a matrix.

```
a = np.array([[1, 2, 3], [4, 5, 6]])
print(a.T)
```

✓ 0.0s

```
[[1 4]
 [2 5]
 [3 6]]
```

flatten() converts a multi-dimensional array into a 1D array.

```
a = np.array([[1, 2, 3], [4, 5, 6]])
print(a.flatten())
```

✓ 0.0s

```
[1 2 3 4 5 6]
```

concatenate() joins multiple arrays together.

```
a1 = np.array([1, 2])
a2 = np.array([3, 4])
print(np.concatenate((a1, a2)))
```

✓ 0.0s

```
[1 2 3 4]
```

SET OPERATIONS

intersect1d() finds common elements in two arrays.

```
a = np.array([1, 2, 3, 4])
b = np.array([3, 4, 5, 6])
print(np.intersect1d(a, b))
```

✓ 0.0s

```
[3 4]
```

union1d() merges both arrays, removing duplicates.

```
a = np.array([1, 2, 3, 4])
b = np.array([3, 4, 5, 6])
print(np.union1d(a, b))
```

✓ 0.0s

```
[1 2 3 4 5 6]
```

setdiff1d(A, B) returns elements present in A but not in B.

```
a = np.array([1, 2, 3, 4])
b = np.array([3, 4, 5, 6])
print(np.setdiff1d(a, b))
```

✓ 0.0s

```
[1 2]
```

PANDAS IN DATA ANALYTICS

Reading Data

- read_csv(), read_excel(), read_json().

Basic Operations

- head(), tail(), columns(), shape().

Data Cleaning

- drop(), rename(), dropna(), isnull().sum().

Visualization

- sns.boxplot(), sns.pairplot(), sns.heatmap().

Statistical Analysis

- describe(), corr(), value_counts().

READING DATA

read_csv()

```
import pandas as pd

df = pd.read_csv("data.csv")
display(df.head())
```

✓ 0.0s

	Unnamed: 0	Name	Age	Employment_status	State
0	0	Anush	23	Emp	PB
1	1	Ankush	32	Unemp	PB
2	2	Alisha	21	Emp	PB
3	3	Rohit	34	Emp	HP
4	4	Komal	26	Unemp	HR

read_excel()

```
import pandas as pd

df = pd.read_excel("Diversity-Inclusion-Dataset.xlsx")
display(df.head())
```

✓ 0.1s

	Employee ID	Gender	Job Level after FY20 promotions	New hire FY20?	FY20 Performance Rating	Promotion in FY21?
0	1	Male	6 - Junior Officer	N	2.0	No
1	2	Female	4 - Manager	N	3.0	No
2	3	Male	2 - Director	N	2.0	No

read_json()

```
import pandas as pd

df = pd.read_json("data.json")
display(df.head())
```

✓ 0.0s

	Name	Age	Location
0	ABC	20	India
1	XYZ	35	Ahmedabad
2	DEF	19	Kota

BASIC OPERATIONS

head(n) displays the first n rows

```
import pandas as pd
df = pd.read_json("data.json")
print(df.head(2))
```

✓ 0.0s

```
  Name  Age   Location
0  ABC   20      India
1  XYZ   35  Ahmedabad
```

tail(n) shows the last n rows

```
import pandas as pd
df = pd.read_json("data.json")
print(df.tail(2))
```

✓ 0.0s

```
  Name  Age   Location
1  XYZ   35  Ahmedabad
2  DEF   19       Kota
```

columns lists all column names

```
import pandas as pd
df = pd.read_json("data.json")
print(df.columns)
```

✓ 0.0s

```
Index(['Name', 'Age', 'Location'], dtype='object')
```

shape returns the number of rows & columns in the dataset.

```
import pandas as pd
df = pd.read_json("data.json")
print(df.shape)
```

✓ 0.0s

```
(3, 3)
```

DATA CLEANING

rename() changes column names.

```
import pandas as pd
df = pd.read_json("data.json")
display(df.rename(columns={"Name": "First_Name"}))
```

✓ 0.0s

	First_Name	Age	Location
0	ABC	20	India
1	XYZ	35	Ahmedabad
2	DEF	19	Kota

drop() removes specified columns or rows (axis=1 for columns, axis=0 for rows)

```
import pandas as pd
df = pd.read_json("data.json")
df = df.drop(["Age"], axis=1)
display(df)
```

✓ 0.0s

	Name	Location
0	ABC	India
1	XYZ	Ahmedabad
2	DEF	Kota

isnull().sum() shows missing values in each column.

```
import pandas as pd
df = pd.read_json("data.json")
display(df.isnull().sum())
```

✓ 0.0s

```
Name        0
Age         0
Location    0
dtype: int64
```

dropna() removes rows containing NaN values.

```
import pandas as pd
df = pd.read_json("data.json")
display(df.dropna())
```

✓ 0.0s

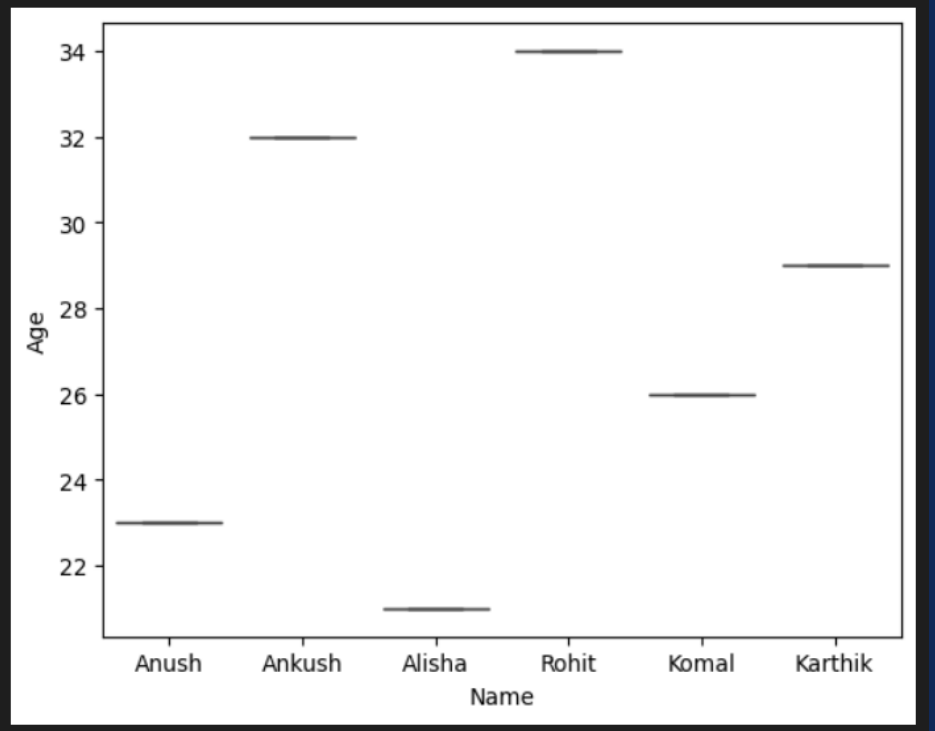
	Name	Age	Location
0	ABC	20	India
1	XYZ	35	Ahmedabad
2	DEF	19	Kota

DATA VISUALIZATION

sns.boxplot() creates a box plot to visualize outliers.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv("data.csv")
sns.boxplot(x="Name", y="Age", data=df)
plt.show()
```

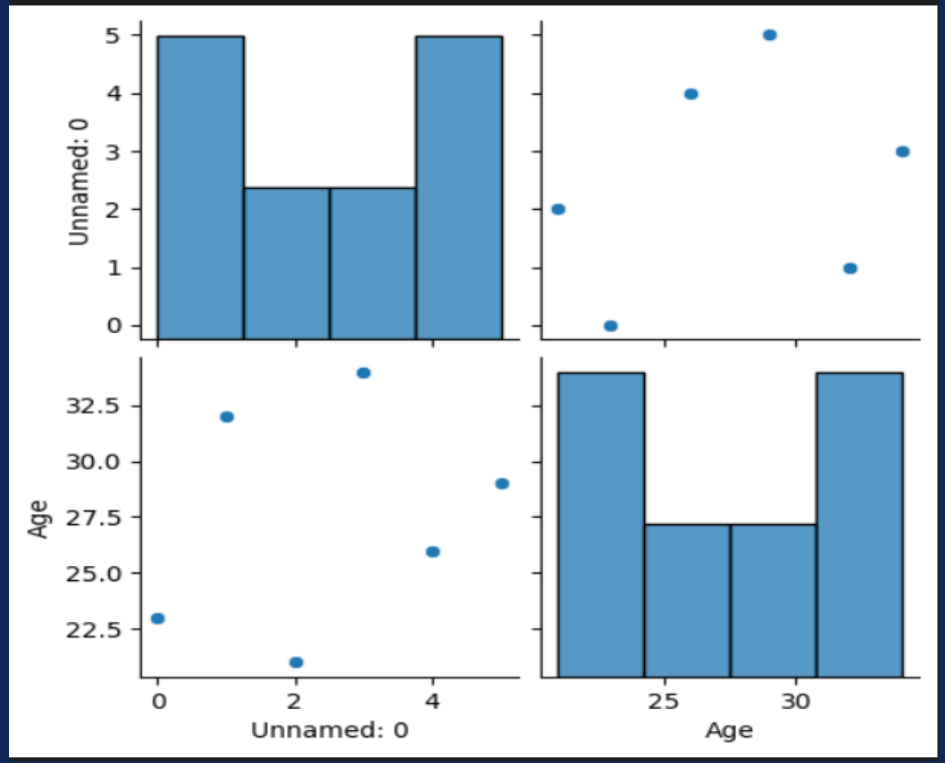
✓ 0.1s



sns.pairplot() plots relationships between all numeric columns.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv("data.csv")
sns.pairplot(df)
plt.show()
```

✓ 0.3s



STATISTICAL ANALYSIS

describe() provides count, mean, min, max, standard deviation, and percentiles.

```
import pandas as pd
df = pd.read_csv("data.csv")
print(df.describe())
```

✓ 0.0s

	Unnamed: 0	Age
count	6.000000	6.000000
mean	2.500000	27.500000
std	1.870829	5.089204
min	0.000000	21.000000
25%	1.250000	23.750000
50%	2.500000	27.500000
75%	3.750000	31.250000
max	5.000000	34.000000

value_counts() shows the frequency of unique values in a column.

```
import pandas as pd
df = pd.read_csv("data.csv")
print(df["State"].value_counts())
```

✓ 0.0s

```
State
PB    3
HR    2
HP    1
Name: count, dtype: int64
```

SUMMARY

- Data Analytics: Extracts insights from data for decision-making.
- Applications: Used in fraud detection, healthcare, inventory, logistics, and marketing.
- Types of Analytics:
  - Descriptive – What happened?
  - Predictive – What will happen?
  - Prescriptive – How to make it happen?
- Process Steps: Collect → Clean → Explore → Model → Interpret.
- Why Python? Easy, scalable, rich libraries, strong community.
- Key Libraries: NumPy, Pandas, Matplotlib, SciPy, Scikit-Learn.
- NumPy & Pandas: Handle arrays, math operations, data cleaning, and visualization.
- Python makes data analytics simple, efficient, and powerful!

THANK YOU