M.Tech Program

Advanced Industry Integrated Programs

Jointly offered by University and LTIMindTree

Python for Data Science

Knowledge partner

LTIMindtree

Implementation partner

L&T EduTech

Modules to be covered

1. Python - Data Structures, OOPS & Modules
2. **Python - Numpy, Pandas & DS Libraries**
3. Scala – Data Structures, OOPS & Modules
4. Scala – DS Libraries & Spark
5. Factors to be considered for choosing Languages

Python - Numpy, Pandas & DS Libraries

Topics Covered

Pandas
Introduction to Pandas

Pandas Series
- Pandas Series
- Pandas Math Functions

Pandas DataFrame
- Pandas Dataframe
- Methods and Attributes

Pandas Functions and Methods
- Data Manipulation Function
- Handling Missing and Data Types
- Data Extraction and Ranking

Text Data in Pandas
- Working with Text Data
- Common string methods
- The split method

Advanced Pandas Techniques
- Multi Index, Transpose(), Stack(), Unstack(), Melt(), Groupby()

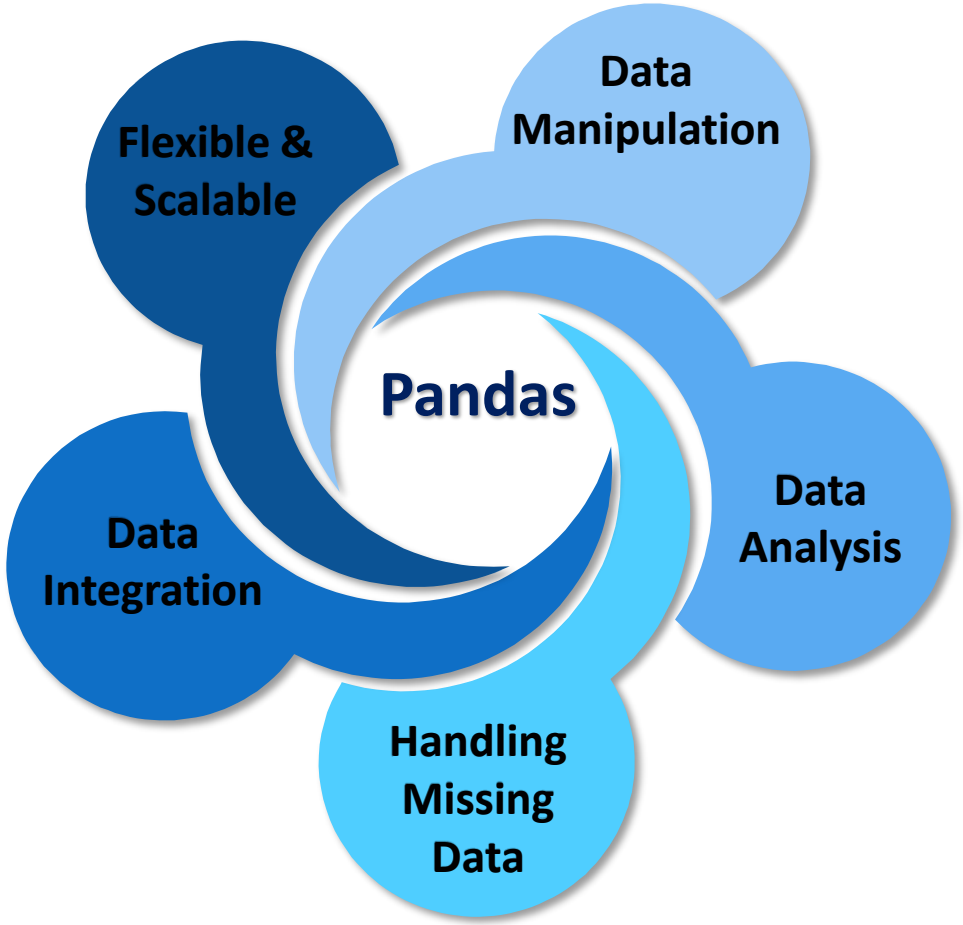
Data Mining
- Scrapy – Beautiful Soup

Data Processing, Modelling, and Visualization
- Numpy – Scipy – Pandas – Keras
- Matplotlib – Seaborn – Bokeh – Plotly – Folium.

Python - Numpy, Pandas & DS Libraries

Introduction to Pandas

- Open-source library , built on top of the NumPy library.
- Used for data manipulation and analysis.
- Pandas is a powerful and versatile library that simplifies the tasks of data manipulation in Python.



Python - Numpy, Pandas & DS Libraries

Pandas Series

- A Series in Pandas is a One-Dimensional Labeled Array which comprises any datatypes such as integer, float, strings, python object and so on.

	Series
a	10
b	20
c	30
d	40

Series Name

Series Value

Python - Numpy, Pandas & DS Libraries

Python Built-In Functions

- **sort_values()** : Sorts a pandas DataFrame by the specified column(s).
- **sort_index()** : Sorts a pandas DataFrame or Series by its index.
- **get()** : Retrieves a value from a dictionary for a given key, with an optional default.
- **copy()** : Creates a shallow copy of an object.
- **apply()** : Applies a function along an axis of a pandas DataFrame or Series.
- **map()** : Applies a function to each item of a sequence or pandas Series.

Python - Numpy, Pandas & DS Libraries

Pandas Math Functions

1. **Arithmetic:** `add()`, `sub()`, `mul()`, `div()` for element-wise operations.
2. **Aggregation:** `sum()`, `mean()`, `median()`, `std()`, `var()` for summarizing data.
3. **Cumulative:** `cumsum()`, `cumprod()`, `cummin()`, `cummax()` for cumulative calculations.
4. **Descriptive:** `describe()`, `quantile()` for statistical summaries.
5. **Correlation and Covariance:** `corr()`, `cov()` for relationship analysis between columns.

Python - Numpy, Pandas & DS Libraries

Pandas Dataframe

- Pandas DataFrame is a two-dimensional, size-mutable, and heterogeneous tabular data structure with labeled axes (rows and columns).

	Lang
P1	Python
P2	Java
P3	C#

	Framework
F1	Django
F2	Spring Boot
F3	Dot Net

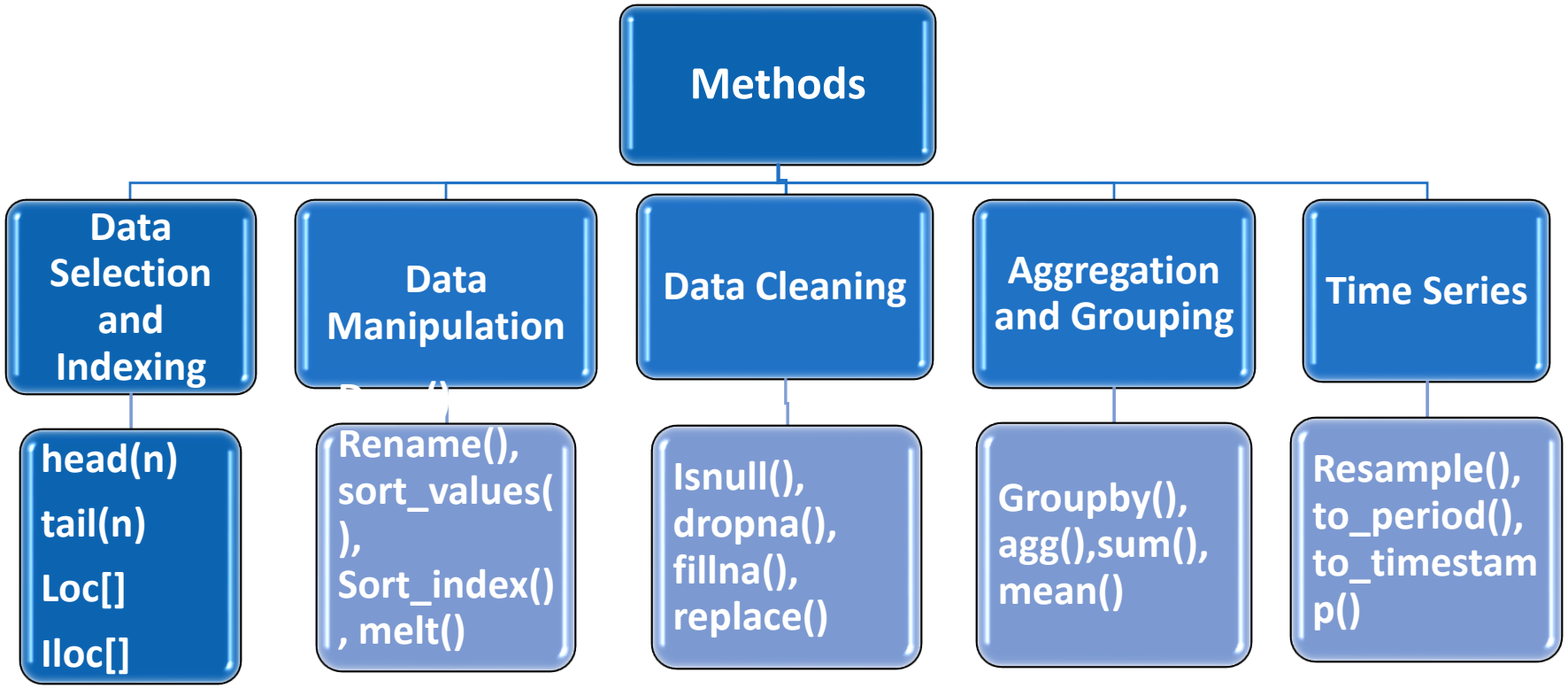
	Lang	Framework
0	Python	Django
1	Java	Spring Boot
2	C#	Dot Net

Series 1 + Series 2 = Data Frame

Python - Numpy, Pandas & DS Libraries

Methods and Attributes between Series and DataFrames

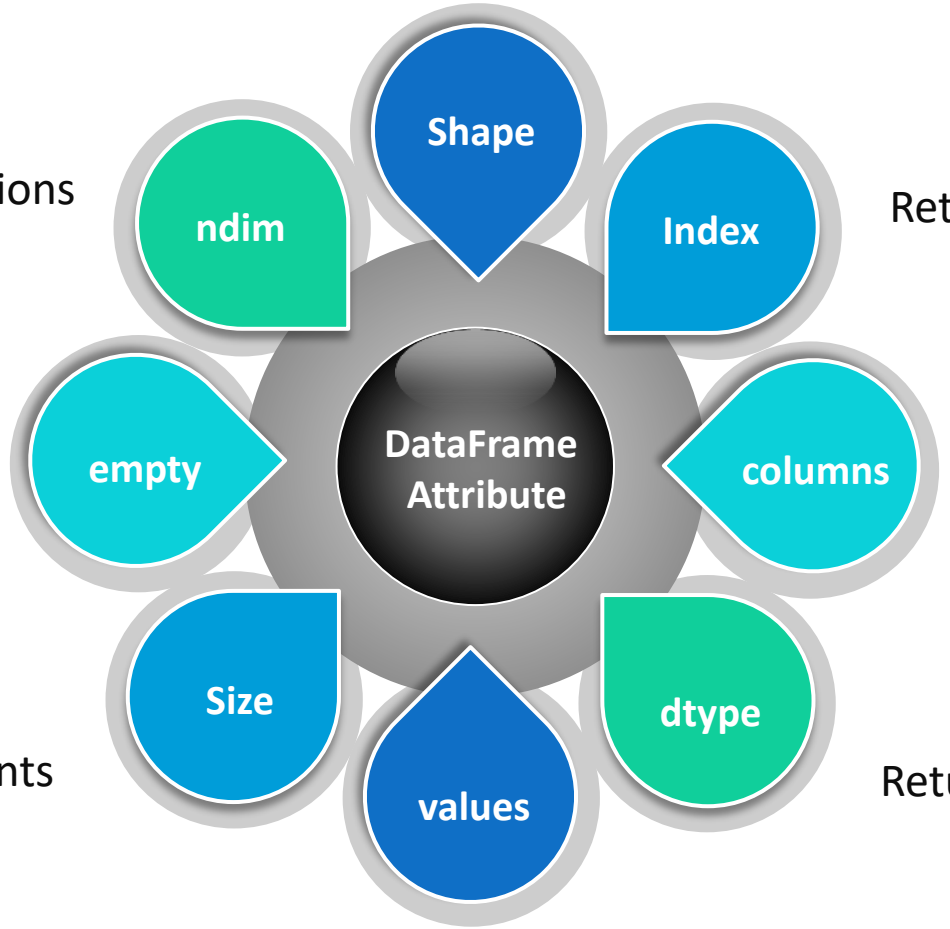
Methods



Python – Introduction to Pandas and Data Science Libraries

Methods and Attributes between Series and DataFrames

Attributes



Returns a tuple

Returns the number of dimensions

Returns the index

Returns True if the DataFrame is empty

Returns the column labels

Returns the number of elements

Returns the data types of each column

Returns the data as a 2D array

Python - Numpy, Pandas & DS Libraries

Pandas Missing Value- fillna() Method

- Impute missing values using statistical measures.
- Pandas .fillna handles missing values strategically.
- Data scientists replace NaN values with .fillna.

Python - Numpy, Pandas & DS Libraries

The astype() Method

- It type cast a Pandas object (such as a DataFrame or Series) to a specified data type.
- It's particularly useful when you need to change the data type of specific columns or multiple columns simultaneously
- df.astype(dtype, copy=None, errors='raise')

Python - Numpy, Pandas & DS Libraries

Rank Series Values with the rank Method

Some of the Data Manipulation and Transformation in Pandas

Filter Data

Used to filter data in a DataFrame.

Boolean Indexing
query Method
loc Method and iloc Method

fillna Method

fillna()

Impute missing values using statistical measures.

astype Method

astype()

Cast a Pandas object (such as a DataFrame or Series) to a specified data type.

Python - Numpy, Pandas & DS Libraries

Filtering data and methods in Dataframe

Some of the Data Manipulation and Transformation in Pandas

Rank Series Values

rank()

compute numerical data ranks along a specified axis

Python - Numpy, Pandas & DS Libraries

Data Extraction in Data Frames

Selecting Columns

- Extract specific columns using column names.
- **df['column_name']**

Filtering Rows

- Extract rows based on conditions using boolean indexing.
- **df[df['column_name'] > value]**

Locating Data

- Use loc for label-based indexing and slicing.
- **df.loc[row_label, 'column_name']**

Position-Based Extraction

- Use iloc for position-based indexing and slicing.
- **df.iloc[row_index, column_index]**

Querying Data

- Extract data using the query method with string expressions.
- **df.query('column_name > value')**

Python - Numpy, Pandas & DS Libraries

Working with Text Data

String Methods

df['column_name'].str.lower() for converting to lowercase.

Extracting Substrings

df['column_name'].str.extract(r'(\d+)') to extract digits.

Replacing Text

df['column_name'].str.replace('old', 'new').

Splitting Text

df['column_name'].str.split(' ', expand=True)

Finding Patterns

df[df['column_name'].str.contains('pattern')]

Python - Numpy, Pandas & DS Libraries

Common String Methods

Method	Description	Example
str.lower()	Converts all characters to lowercase	df['column_name'].str.lower()
str.upper()	Converts all characters to uppercase	df['column_name'].str.upper()
str.len()	Computes the length of each string	df['column_name'].str.len()
str.strip()	Removes leading and trailing whitespace	df['column_name'].str.strip()
str.replace(pattern, replacement)	Replaces occurrences of a pattern with a replacement string	df['column_name'].str.replace('old', 'new')

Python - Numpy, Pandas & DS Libraries

Common String Methods – Split Method

Method	Description	Example
str.contains(pattern)	Checks if each string contains a pattern	df['column_name'].str.contains('pattern')
str.split(separator)	Splits each string by the given separator	df['column_name'].str.split(' ')
str.startswith(prefix)	Checks if each string starts with the given prefix	df['column_name'].str.startswith('prefix')
str.endswith(suffix)	Checks if each string ends with the given suffix	df['column_name'].str.endswith('suffix')
str.extract(pattern)	Extracts substrings matching a regular expression pattern	df['column_name'].str.extract(r'(\d+)')

Python - Numpy, Pandas & DS Libraries

Multi Index Module

- MultiIndex in pandas allows organizing data with hierarchical row and column labels.
- `pd.MultiIndex` module provides functions for creating, manipulating, and sorting MultiIndex objects.
- Functions like `from_tuples()`, `from_arrays()`, and `from_product()` create MultiIndex from different data structures.
- Methods like `get_level_values()`, `set_levels()`, and `swaplevel()` allow manipulation of MultiIndex levels.
- MultiIndex is essential for analyzing complex datasets with multiple dimensions or categories efficiently.

Advanced Pandas Techniques

Python - Numpy, Pandas & DS Libraries

Transpose() Method

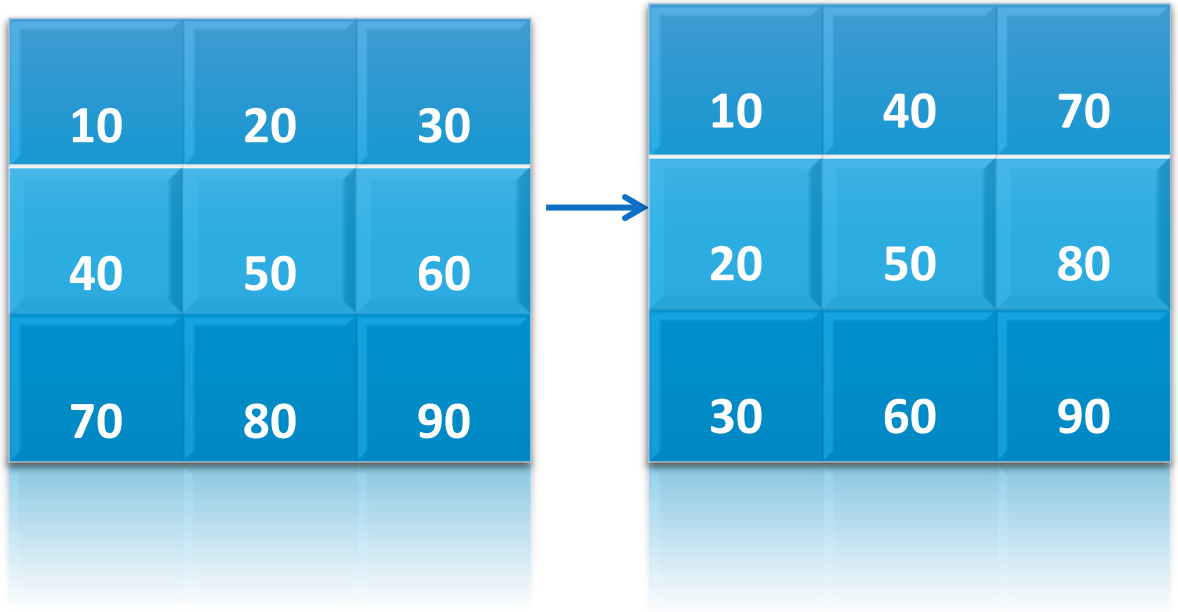
- The **transpose** method used to swap rows and columns in a DataFrame.

- **Interchanges Rows and Columns**
  - Converts rows into columns and vice versa.

- **Access Using .T Attribute**
  - Use .T for quick DataFrame transposition.

- **Useful for Data Reshaping**
  - Helpful in restructuring and analyzing data.

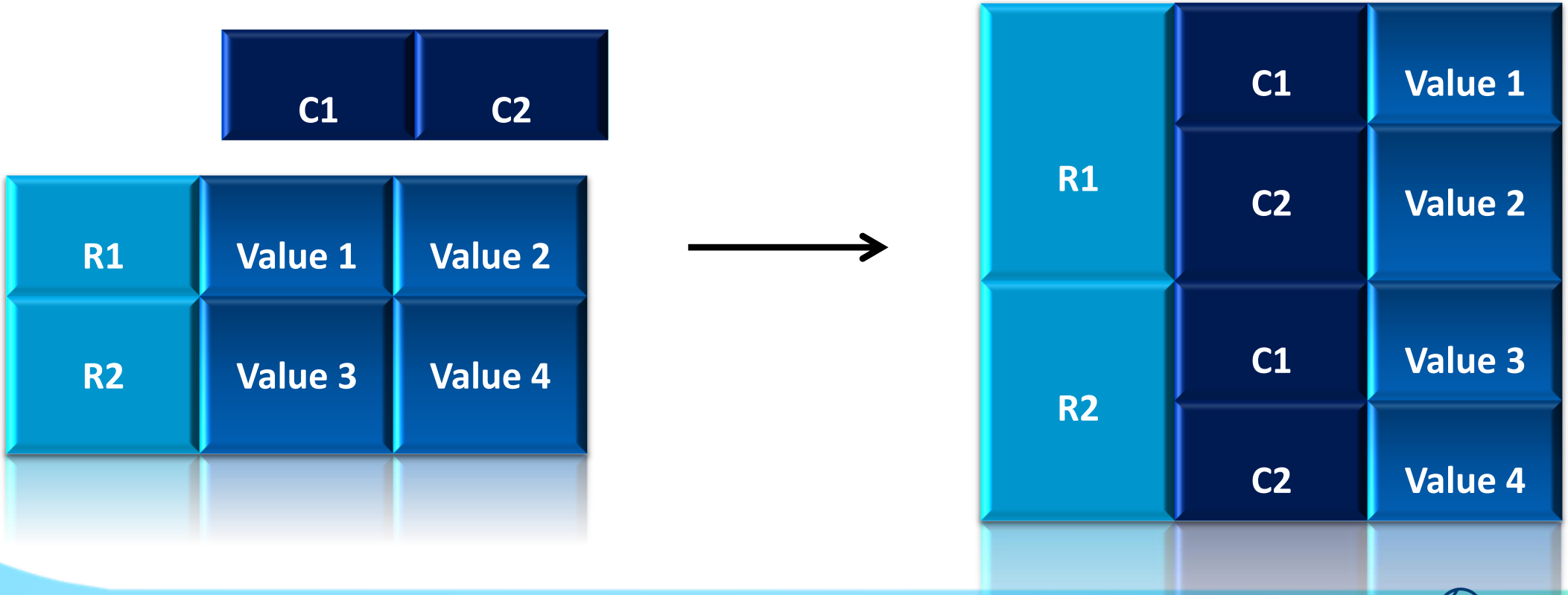
- **Retains Original Data Types**
  - Keeps data types intact during transposition.



Python - Numpy, Pandas & DS Libraries

Stack() Method

- Transform a DataFrame from a wide format to a long format.
- Easy to analyze and work with data.



Python - Numpy, Pandas & DS Libraries

Unstack() Method

Unstack() Method is reverse of the Stack() Method.

R1	C1	Value 1
	C2	Value 2
R2	C1	Value 3
	C2	Value 4

→

	C1	C2
R1	Value 1	Value 2
R2	Value 3	Value 4

Python - Numpy, Pandas & DS Libraries

Melt() Method

Reshapes DataFrame from wide to long format.

ID	NAME	MAT	PHY	CHEM
1	Berly	85	90	82
2	Jessy	90	95	92

ID	NAME	SUBJ	MARKS
1	Berly	MAT	85
2	Jessy	MAT	90
3	Berly	PHY	90
4	Jessy	PHY	95
5	Berly	CHEM	82
6	Jessy	CHEM	92

Python - Numpy, Pandas & DS Libraries

Knowledge Check

Which of the following optimizations does Pandas provide for handling large datasets?

A. Integration with Dask for out-of-core computations.

B. Automatically parallelizing operations across multiple CPU cores.

C. Utilizing GPU for data processing tasks.

D. Converting data frames to more memory-efficient structures like NumPy arrays.

Python - Numpy, Pandas & DS Libraries

Knowledge Check

Which of the following optimizations does Pandas provide for handling large datasets?

A. Integration with Dask for out-of-core computations.

B. Automatically parallelizing operations across multiple CPU cores.

C. Utilizing GPU for data processing tasks.

D. Converting data frames to more memory-efficient structures like NumPy arrays.

Ans : Integration with Dask for out-of-core computations.

Python - Numpy, Pandas & DS Libraries

Knowledge Check

Given a DataFrame df with a column 'A', which of the following statements correctly sorts the values of column 'A' in descending order and then returns the DataFrame with the sorted index?

A. df.sort_values('A', ascending=False).sort_index()

B. df.sort_index().sort_values('A', ascending=False)

C. df.sort_values('A', ascending=True).sort_index(ascending=False)

D. df.sort_values('A', ascending=False).set_index('A')

Python - Numpy, Pandas & DS Libraries

Knowledge Check

Given a DataFrame df with a column 'A', which of the following statements correctly sorts the values of column 'A' in descending order and then returns the DataFrame with the sorted index?

A. df.sort_values('A', ascending=False).sort_index()

B. df.sort_index().sort_values('A', ascending=False)

C. df.sort_values('A', ascending=True).sort_index(ascending=False)

D. df.sort_values('A', ascending=False).set_index('A')

Ans : df.sort_values('A', ascending=False).sort_index()

Python - Numpy, Pandas & DS Libraries

Knowledge Check

When converting a DataFrame column to a specific data type using the astype method, which of the following commands correctly converts column 'D' to a categorical type?

A. df.astype({'D': 'category’})

B. df['D'] = df['D'].astype('category')

C. df['D'].astype('categorical')

D. df.convert_dtype('D', 'category')

Python - Numpy, Pandas & DS Libraries

Knowledge Check

When converting a DataFrame column to a specific data type using the astype method, which of the following commands correctly converts column 'D' to a categorical type?

A. df.astype({'D': 'category’})

B. df['D'] = df['D'].astype('category')

C. df['D'].astype('categorical')

D. df.convert_dtype('D', 'category’)

Ans : df['D'] = df['D'].astype('category')

Python - Numpy, Pandas & DS Libraries

Groupby() Module

Groups DataFrame by specified column, allowing for aggregation operations on each group.

Python - Numpy, Pandas & DS Libraries

Groupby() Module

id	Species	Food_need
1	Cat	10
2	Cow	20
1	Cat	10
2	Cow	20
3	Dog	30
1	Cat	10
3	Dog	30

Groupby ("Id")

id	Species	Food_need
1	Cat	10
1	Cat	10
1	Cat	10

id	Species	Food_need
2	Cow	20
2	Cow	20

id	Species	Food_need
3	Dog	30
3	Dog	30

Aggregation

id	Species	Food_need
1	Cat	30

id	Species	Food_need
2	Cow	40

id	Species	Food_need
3	Dog	60

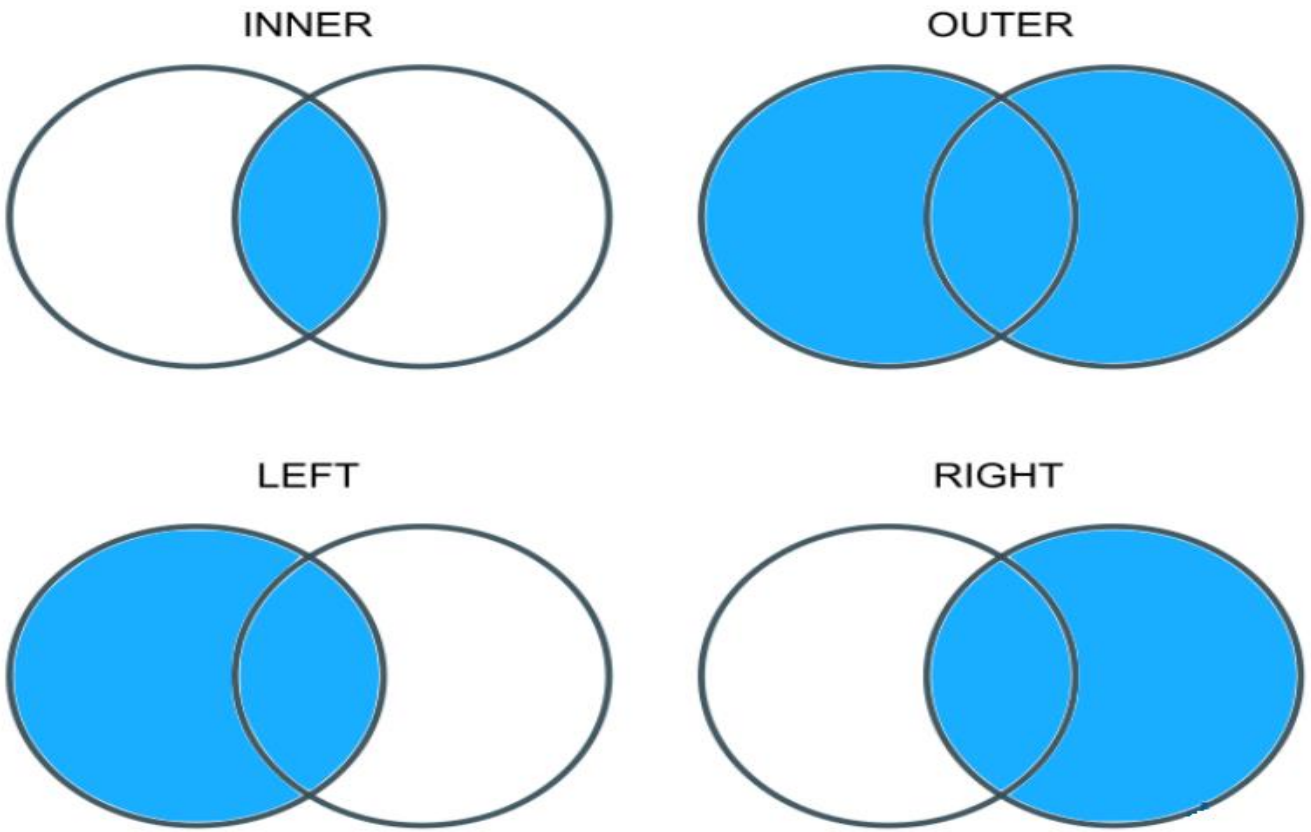
Combine

id	Species	Food_need
1	Cat	30
2	Cow	40
3	Dog	60

Python - Numpy, Pandas & DS Libraries

Merging Dataframes

Combines two or more DataFrames into a single DataFrame based on a common column or index.



Python - Numpy, Pandas & DS Libraries

Merging Dataframes

Combines two or more DataFrames into a single DataFrame based on a common column or index.

Example : Unique Key combination

Left

	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2
3	K3	A3	B3

Right

	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K2	C2	D2
3	K3	C3	D3

pd.merge(left, right, on="key") →

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K1	A1	B1	C1	D1
2	K2	A2	B2	C2	D2
3	K3	A3	B3	C3	D3

Python - Numpy, Pandas & DS Libraries

Merging Dataframes

Combines two or more DataFrames into a single DataFrame based on a common column or index.

Example : Multiple Join keys

Left

	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3

Right

	key1	key2	C	D
0	K0	K0	C0	D0
1	K1	K0	C1	D1
2	K1	K0	C2	D2
3	K2	K0	C3	D3

pd.merge(left, right, on=["key1", "key2"]) →

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K1	K0	A2	B2	C1	D1
2	K1	K0	A2	B2	C2	D2

Python - Numpy, Pandas & DS Libraries

Merging Dataframes

The how argument to merge specifies how to determine which keys are to be included in the resulting table.

Example : A key combination does not appear in either the left or right tables → the value in join table NaN

Left

	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3

pd.merge(left, right, how="left", on=["key1", "key2"])

Right

	key1	key2	C	D
0	K0	K0	C0	D0
1	K1	K0	C1	D1
2	K1	K0	C2	D2
3	K2	K0	C3	D3

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K0	K1	A1	B1	NaN	NaN
2	K1	K0	A2	B2	C1	D1
3	K1	K0	A2	B2	C2	D2
4	K2	K1	A3	B3	NaN	NaN

Python - Numpy, Pandas & DS Libraries

Merging Dataframes

The how argument to merge specifies how to determine which keys are to be included in the resulting table.

Example : A key combination does not appear in either the left or right tables → the value in join table NaN

Left

	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3

pd.merge(left, right, how="right", on=["key1", "key2"])

Right

	key1	key2	C	D
0	K0	K0	C0	D0
1	K1	K0	C1	D1
2	K1	K0	C2	D2
3	K2	K0	C3	D3

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K1	K0	A2	B2	C1	D1
2	K1	K0	A2	B2	C2	D2
3	K2	K0	NaN	NaN	C3	D3

Python - Numpy, Pandas & DS Libraries

Merging Dataframes

The how argument to merge specifies how to determine which keys are to be included in the resulting table.

Example : A key combination does not appear in either the left or right tables → the value in join table NaN

Left

	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3

pd.merge(left, right, how="outer", on=["key1", "key2"])

Right

	key1	key2	C	D
0	K0	K0	C0	D0
1	K1	K0	C1	D1
2	K1	K0	C2	D2
3	K2	K0	C3	D3

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K0	K1	A1	B1	NaN	NaN
2	K1	K0	A2	B2	C1	D1
3	K1	K0	A2	B2	C2	D2
4	K2	K1	A3	B3	NaN	NaN
5	K2	K0	NaN	NaN	C3	D3

Python - Numpy, Pandas & DS Libraries

Merging Dataframes

The how argument to merge specifies how to determine which keys are to be included in the resulting table.

Example : A key combination does not appear in either the left or right tables → the value in join table NaN

Left

	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3

pd.merge(left, right, how="inner", on=["key1", "key2"])

Right

	key1	key2	C	D
0	K0	K0	C0	D0
1	K1	K0	C1	D1
2	K1	K0	C2	D2
3	K2	K0	C3	D3

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K1	K0	A2	B2	C1	D1
2	K1	K0	A2	B2	C2	D2

Python - Numpy, Pandas & DS Libraries

Knowledge Check

Which of the following best describes the effect of the stack() method on a DataFrame with hierarchical columns?

A. It merges all columns into a single column.

B. It pivots the columns to rows, removing one level of column hierarchy.

C. It creates a new DataFrame by combining rows with similar index values.

D. It transposes the DataFrame.

Python - Numpy, Pandas & DS Libraries

Knowledge Check

Which of the following best describes the effect of the stack() method on a DataFrame with hierarchical columns?

A. It merges all columns into a single column.

B. It pivots the columns to rows, removing one level of column hierarchy.

C. It creates a new DataFrame by combining rows with similar index values.

D. It transposes the DataFrame.

Ans: It pivots the columns to rows, removing one level of column hierarchy.

Python - Numpy, Pandas & DS Libraries

Knowledge Check

When performing a merge operation, which argument would you use to specify the type of join (e.g., inner, outer, left, right)?

A. how

B. method

C. type

D. join_type

Python - Numpy, Pandas & DS Libraries

Knowledge Check

When performing a merge operation, which argument would you use to specify the type of join (e.g., inner, outer, left, right)?

A. how

B. method

C. type

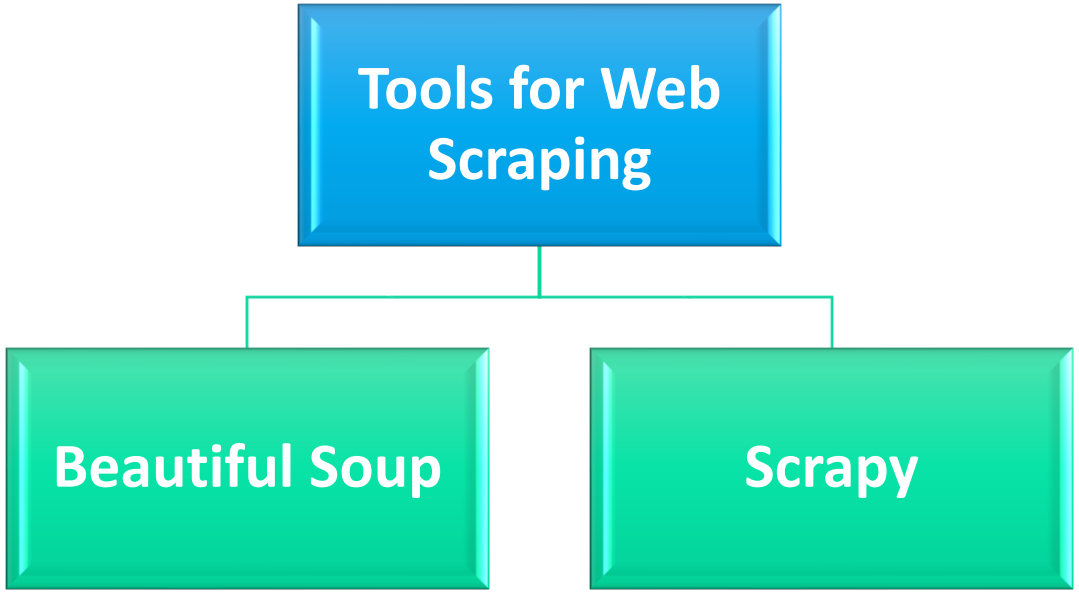
D. join_type

Ans: how

Python - Numpy, Pandas & DS Libraries

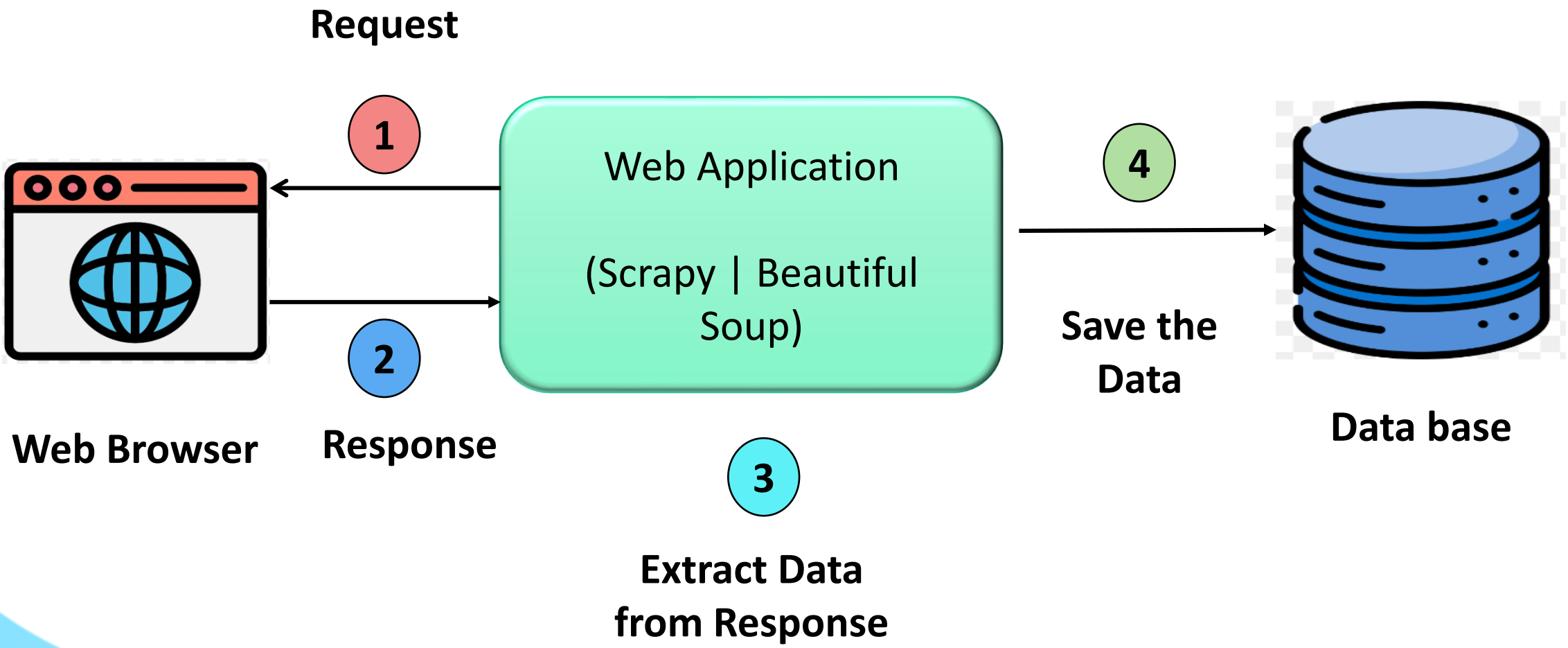
Data Mining(Scrapy – Beautiful Soup)

- Data mining is the process of discovering patterns and knowledge from large amounts of data.
- In the context of web scraping, it involves extracting useful information from websites.



Python - Numpy, Pandas & DS Libraries

Data Mining(Scrapy – Beautiful Soup)



Data Processing and Data Modeling

Python - Numpy, Pandas & DS Libraries

Data Processing and Data Modeling

Data Processing

The series of operations on data to retrieve, transform, or classify information.

Data Modelling

The process of creating a data model for the data to be stored in a database, representing the data structures and relationships.

Python - Numpy, Pandas & DS Libraries

Data Processing

01

DATA CLEANING
Handle missing values, remove duplicates, and correct inconsistencies

02

DATA TRANSFORMATION
Normalize, scale, and perform feature engineering

03

Data Integration:
Combine data from various sources into a single, coherent dataset.

DATA PROCESSING STEPS

01

HANDLING MISSING VALUES
fillna()
dropna()

02

NORMALIZATION AND SCALING
Standardize data for model compatibility(Min-Max scaling and Z-score)

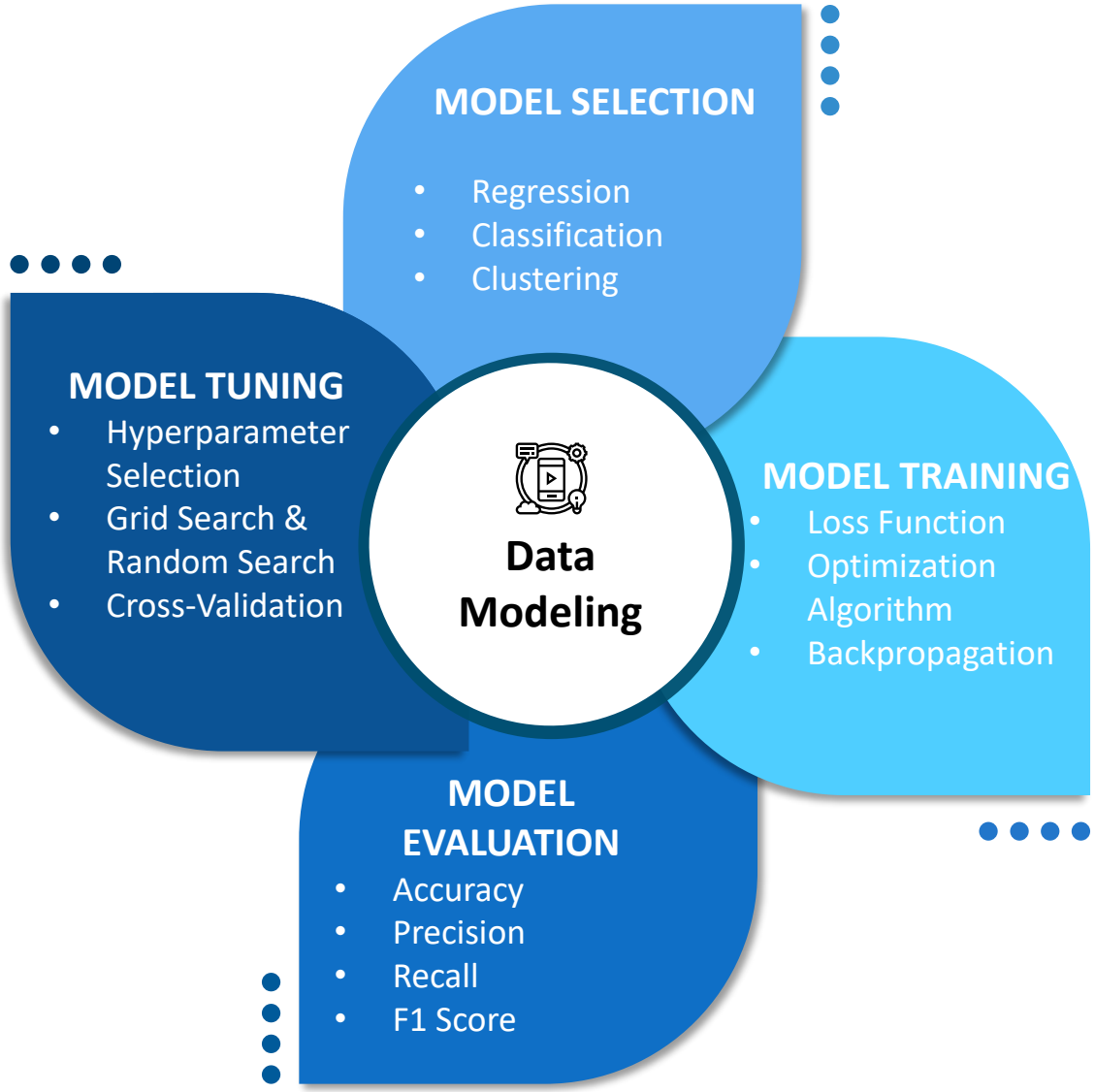
03

ENCODING CATEGORICAL DATA
Utilize one-hot encoding and label encoding

DATA PROCESSING TECHNIQUES

Python - Numpy, Pandas & DS Libraries

Data Modelling



Python - Numpy, Pandas & DS Libraries

Data Processing and Data Modeling

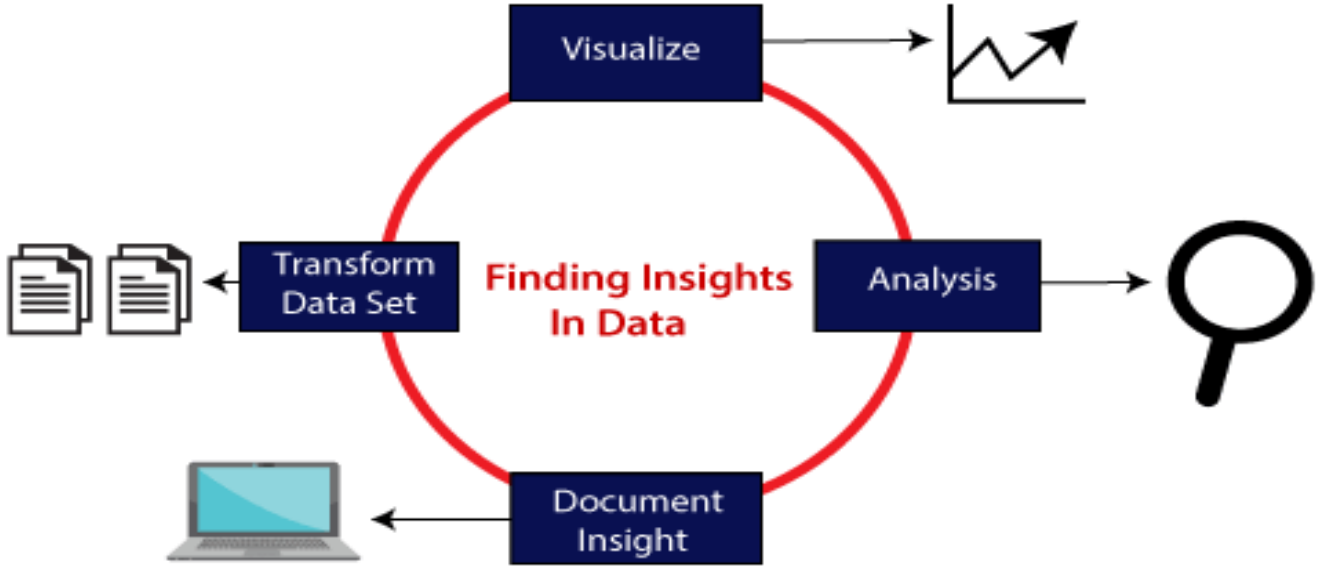
- **NumPy**: Numerical computing in Python.
- **SciPy:** Scientific computing with Python.
- **Pandas:** Data manipulation and analysis.
- **Keras:** High-level neural networks API.
- **Scikit-learn:** Machine learning in Python.
- **PyTorch:** Deep learning research platform.
- **TensorFlow:** Open-source machine learning framework.
- **XGBoost:** Optimized gradient boosting library.

Data Visualization

Python - Numpy, Pandas & DS Libraries

Data Visualization

Data visualization is the graphical representation of information and data using visual elements like charts, graphs, and maps.

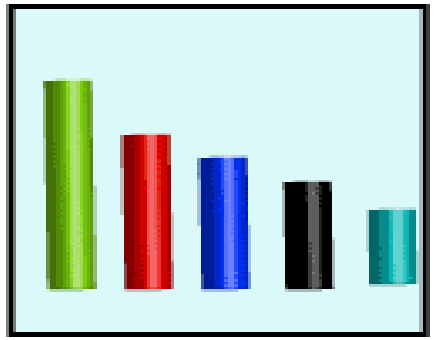


Python - Numpy, Pandas & DS Libraries

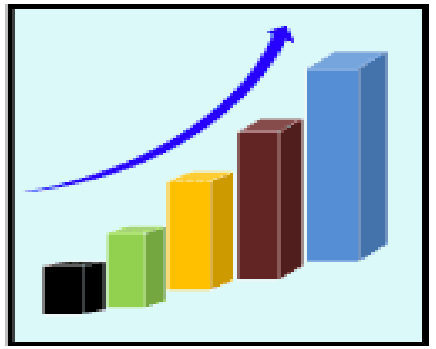
Data Visualization

- Graphics provides an excellent approach for exploring the data, which is essential for presenting results.

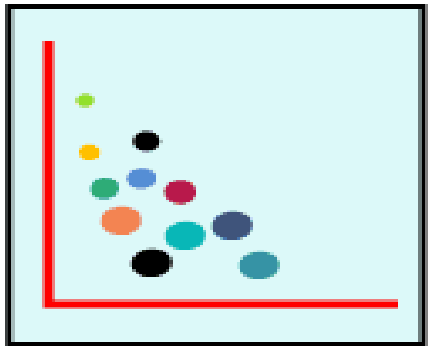
- There are five key plots that are used for data visualization:



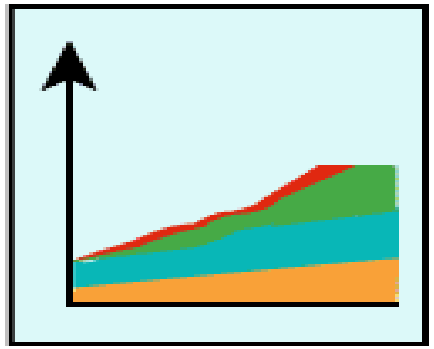
Bar Graph



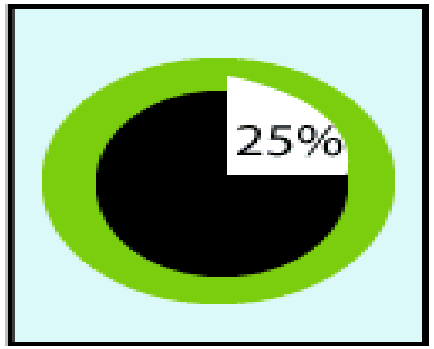
Histogram



Scatter Plot



Area Plot

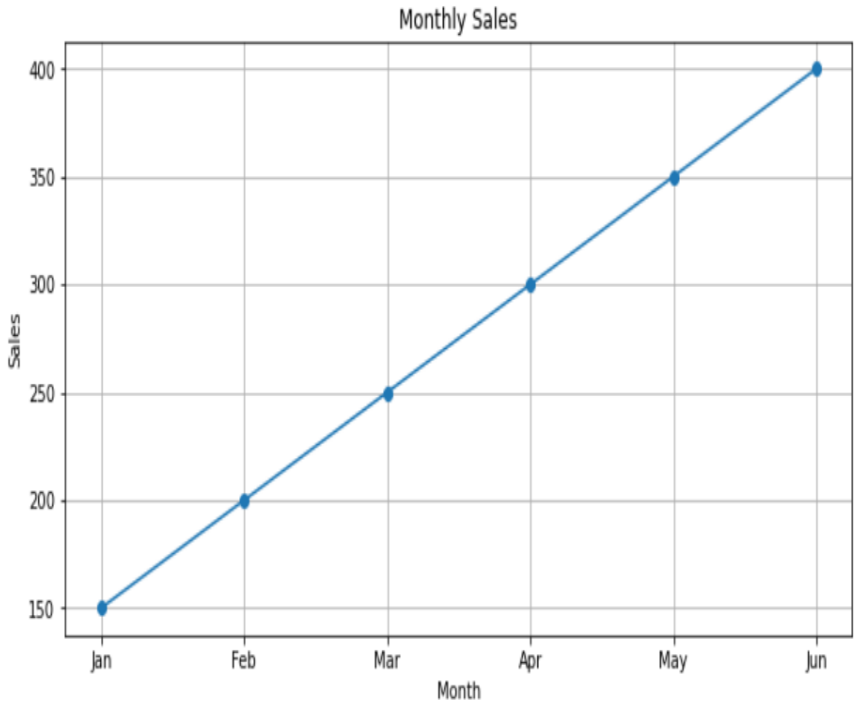


Pie Plot

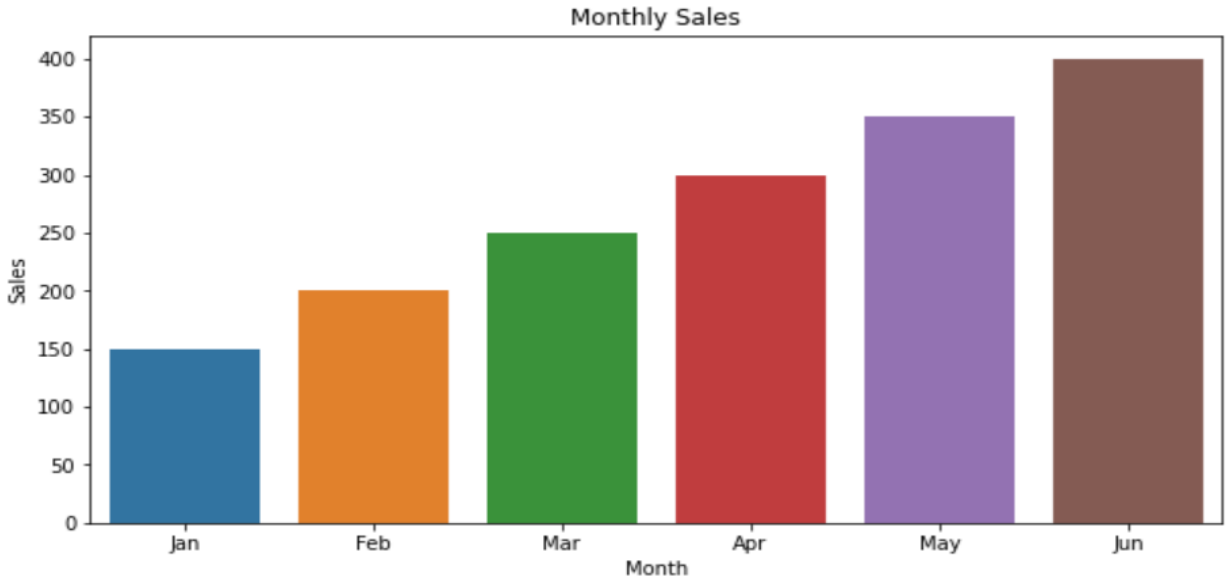
Python - Numpy, Pandas & DS Libraries

Data Visualization – Matplotlib and Seaborn

Matplotlib is a basic plotting library in Python that provides a wide range of static, animated, and interactive plots.



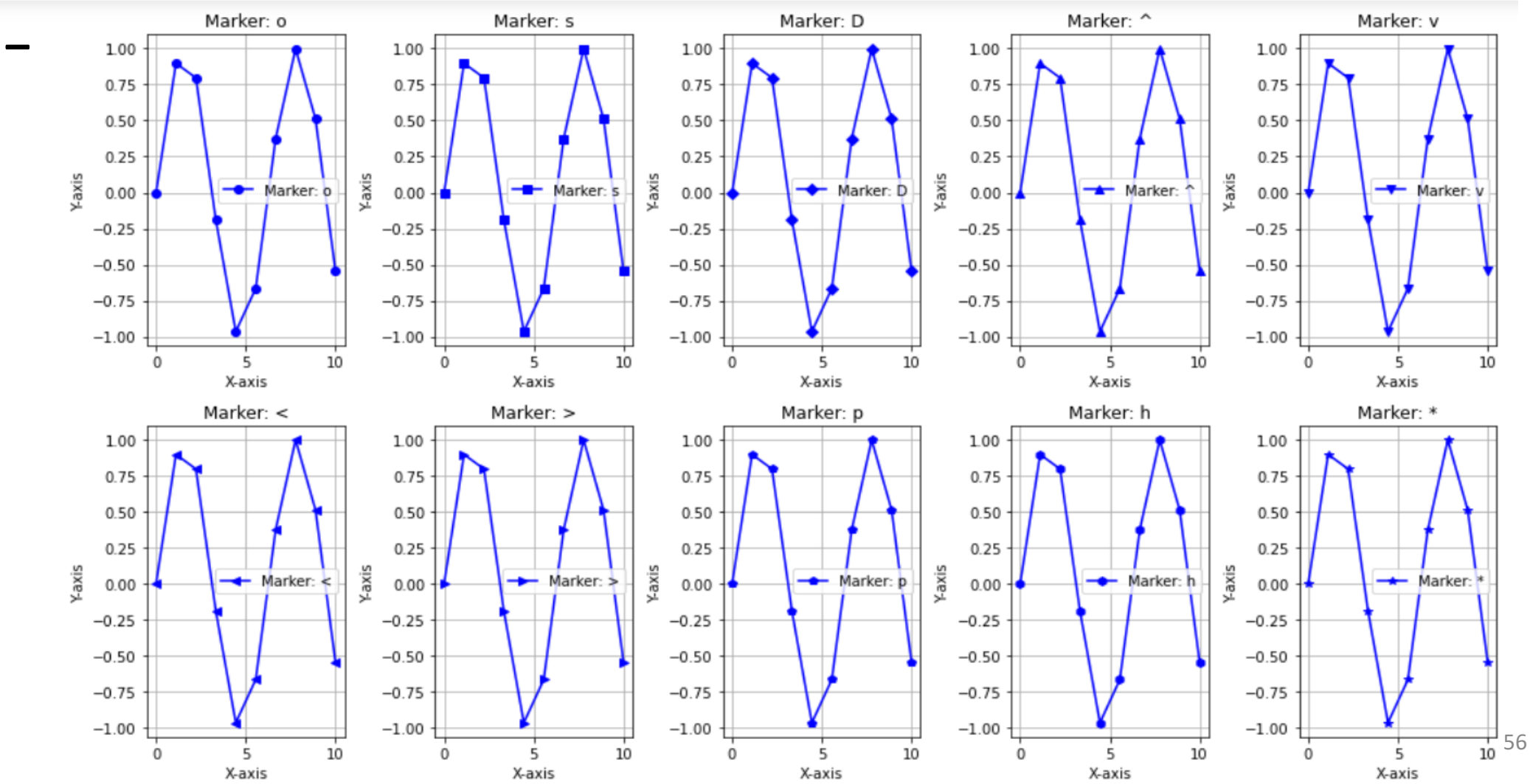
Seaborn is built on top of Matplotlib and provides a high-level interface for drawing attractive and informative statistical graphics



Python - Numpy, Pandas & DS Libraries

Data Visualization – Matplotlib and Seaborn

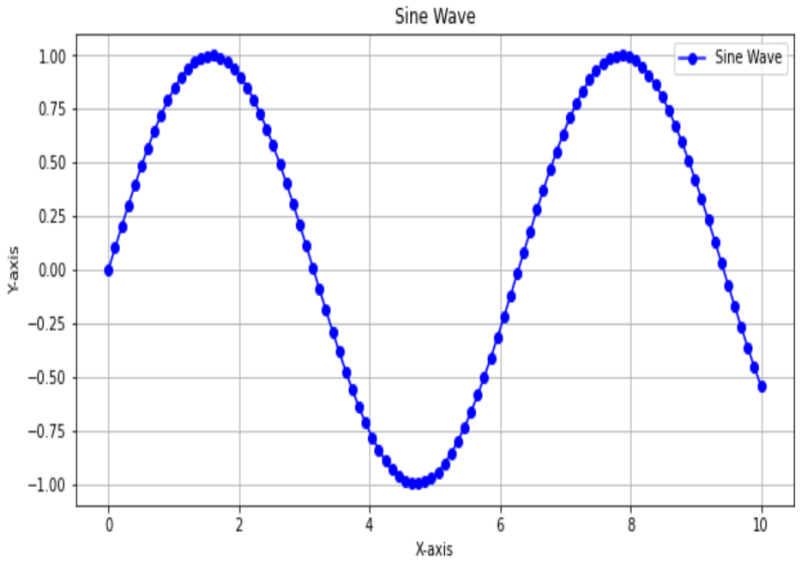
Matplotlib – Line Markers

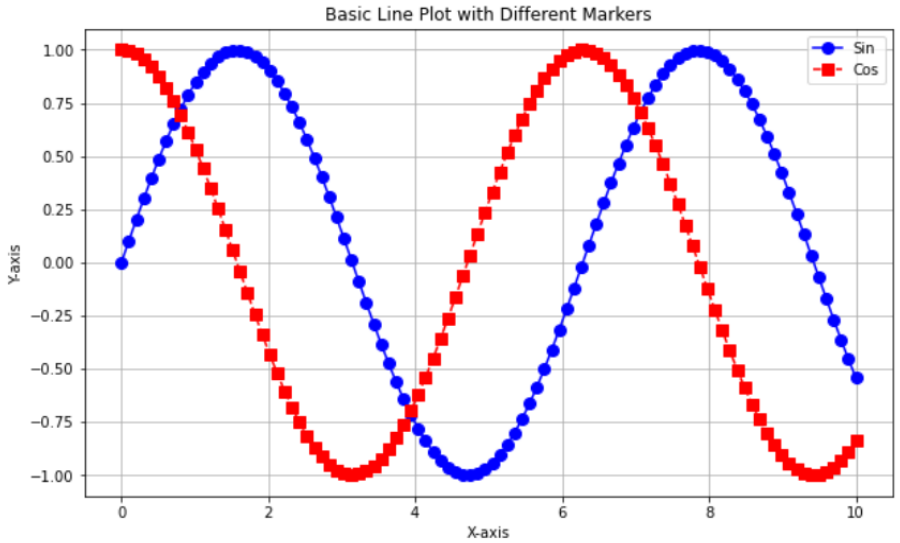


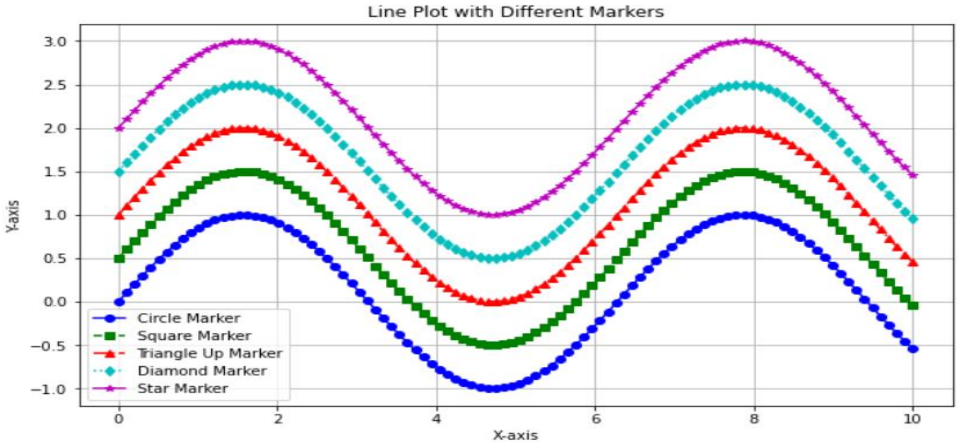
Python - Numpy, Pandas & DS Libraries

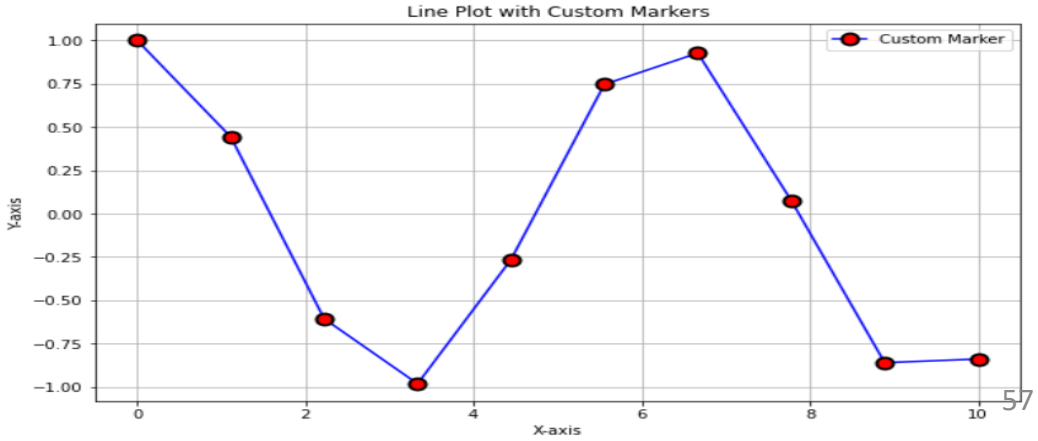
Data Visualization – Matplotlib and Seaborn

Matplotlib – Line Markers



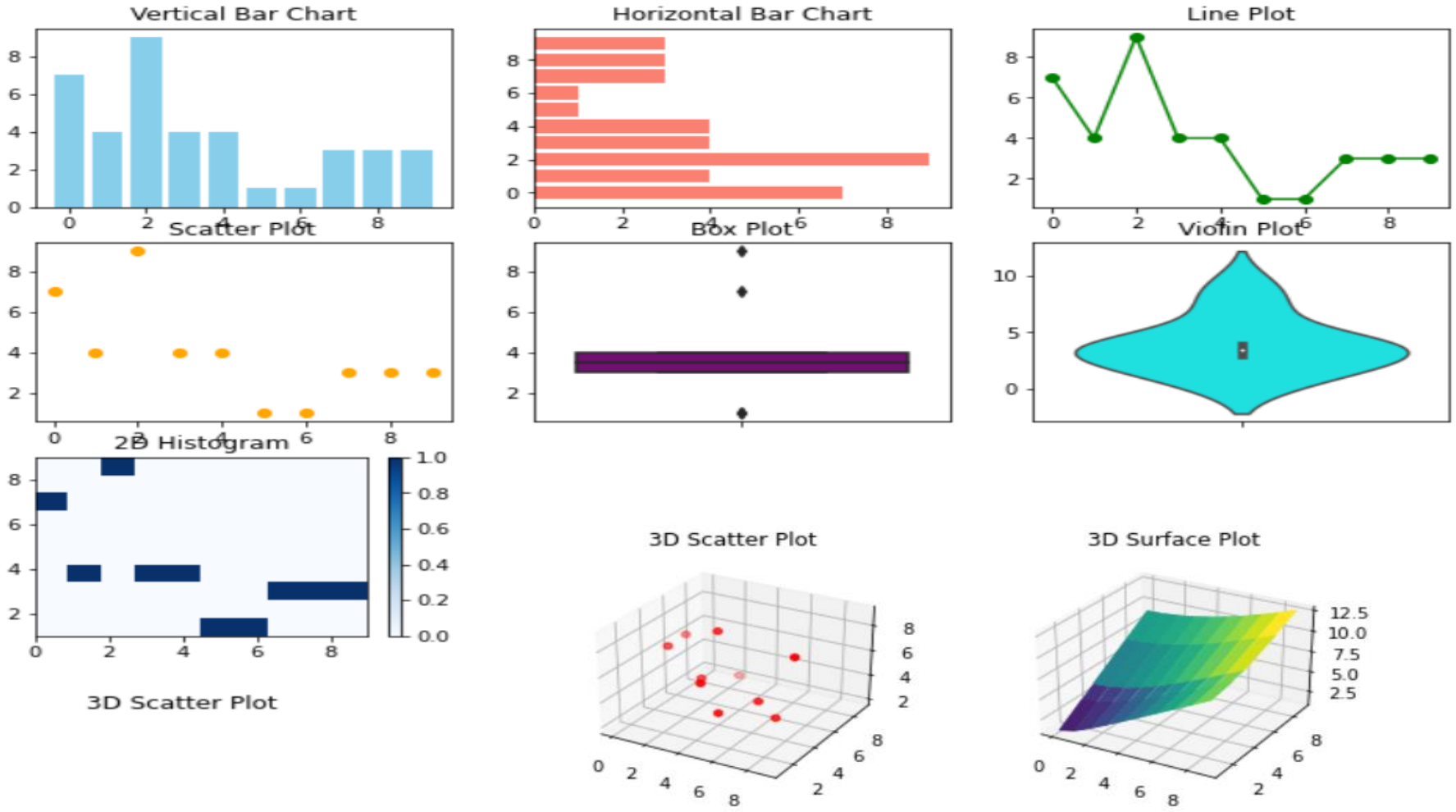






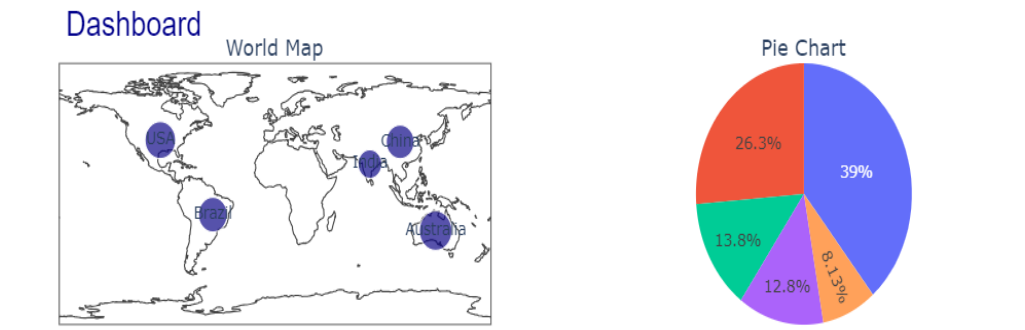
Python - Numpy, Pandas & DS Libraries

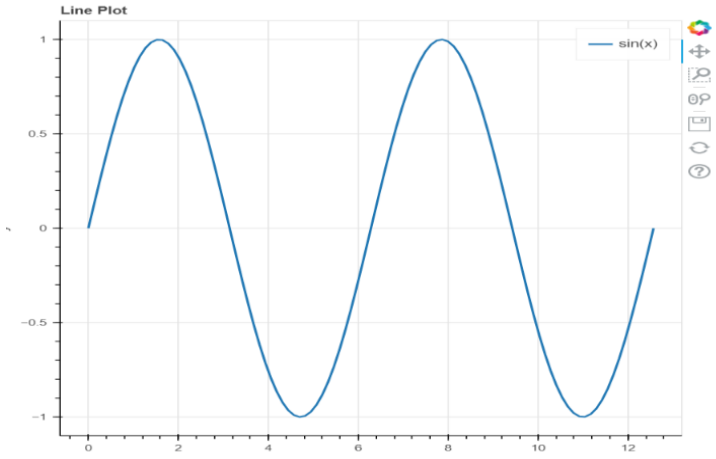
Data Visualization – Matplotlib and Seaborn

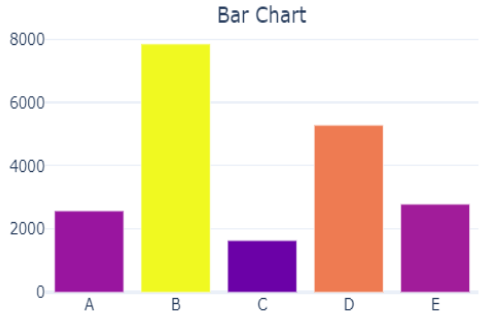


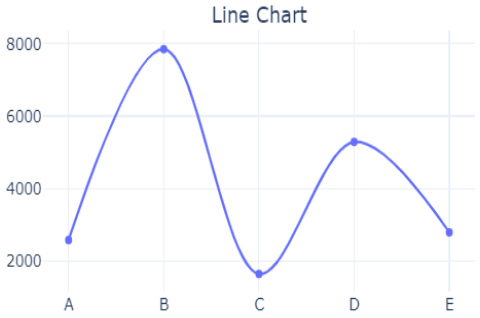
Python - Numpy, Pandas & DS Libraries

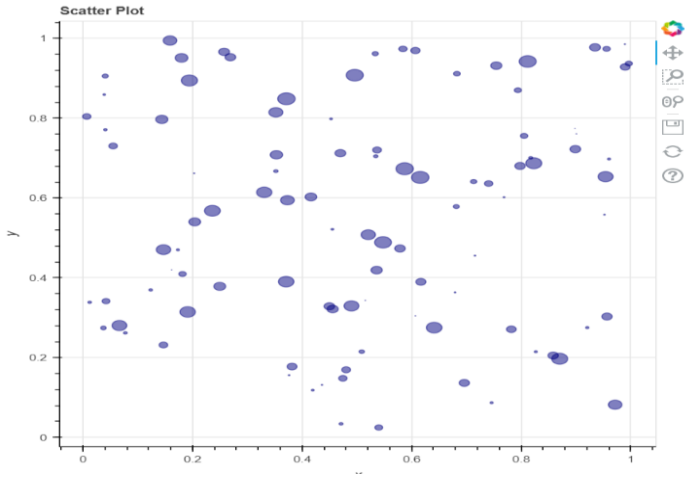
Data Visualization – Plotly, Folium, Bokeh





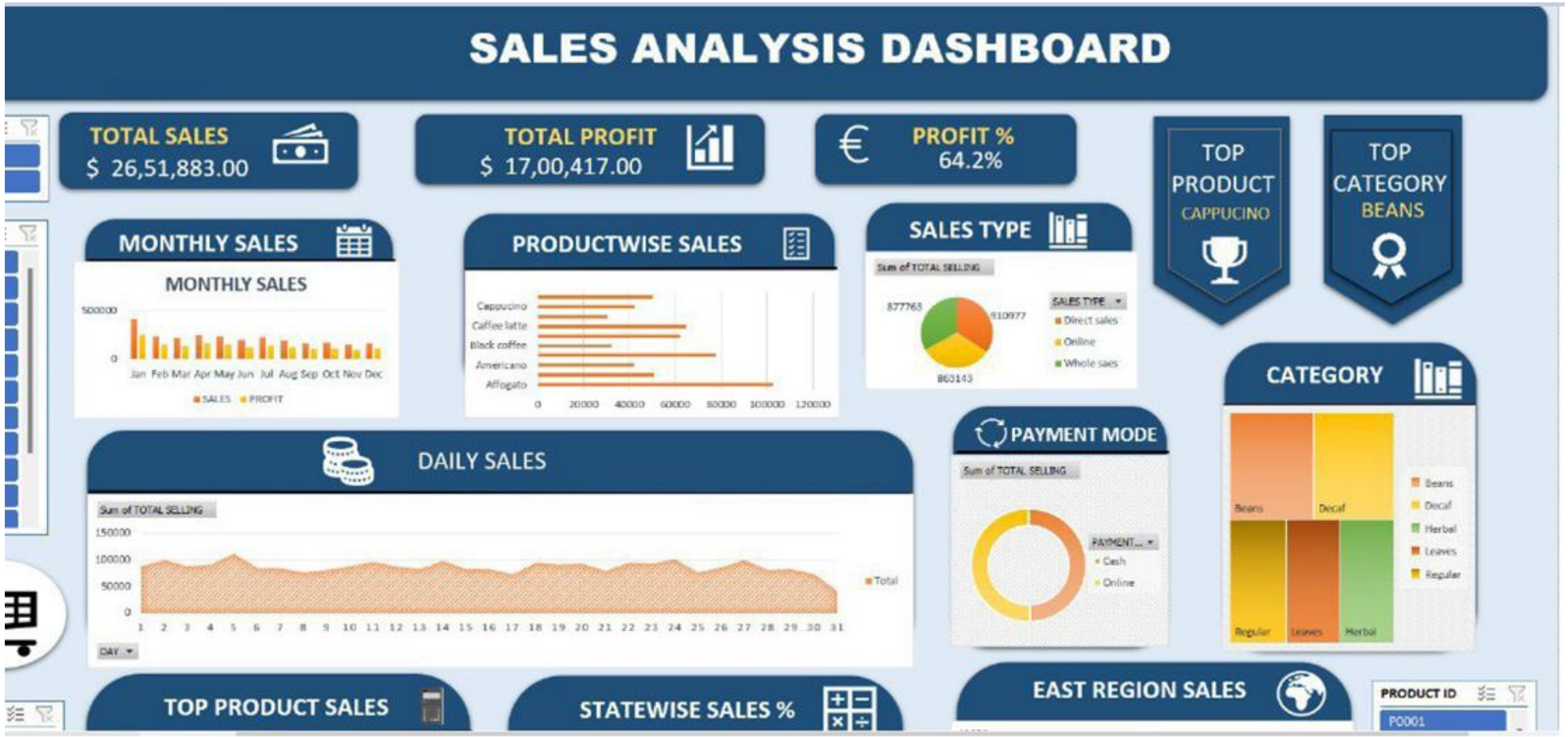






Python - Numpy, Pandas & DS Libraries

Data Visualization - Dashboard



Python - Numpy, Pandas & DS Libraries

Knowledge Check

Which method would you use to handle outliers in a DataFrame by capping them at a specified threshold?

A. df.clip(lower=lower_threshold, upper=upper_threshold)

B. df.limit(lower=lower_threshold, upper=upper_threshold)

C. df.truncate(lower=lower_threshold, upper=upper_threshold)

D. df.reduce(lower=lower_threshold, upper=upper_threshold)

Python - Numpy, Pandas & DS Libraries

Knowledge Check

Which method would you use to handle outliers in a DataFrame by capping them at a specified threshold?

A. df.clip(lower=lower_threshold, upper=upper_threshold)

B. df.limit(lower=lower_threshold, upper=upper_threshold)

C. df.truncate(lower=lower_threshold, upper=upper_threshold)

D. df.reduce(lower=lower_threshold, upper=upper_threshold)

Ans: df.clip(lower=lower_threshold, upper=upper_threshold)

Python - Numpy, Pandas & DS Libraries

Knowledge Check

Which of the following statements best describes the use of the pd.get_dummies() function in data modeling?

A. It normalizes the data to a range between 0 and 1.

B. It transforms categorical variables into a series of binary columns.

C. It fills missing values in the dataset.

D. It scales numerical features to a standard normal distribution.

Python - Numpy, Pandas & DS Libraries

Knowledge Check

Which of the following statements best describes the use of the pd.get_dummies() function in data modeling?

A. It normalizes the data to a range between 0 and 1.

B. It transforms categorical variables into a series of binary columns.

C. It fills missing values in the dataset.

D. It scales numerical features to a standard normal distribution.

Ans: It transforms categorical variables into a series of binary columns.

Python - Numpy, Pandas & DS Libraries

Knowledge Check

To create a bar chart in Plotly with grouped bars for different categories, which parameter should be used?

A. barmode='group'

B. barmode='stack'

C. groupmode='bar'

D. groupbar='mode'

Python - Numpy, Pandas & DS Libraries

Knowledge Check

To create a bar chart in Plotly with grouped bars for different categories, which parameter should be used?

A. barmode='group'

B. barmode='stack'

C. groupmode='bar'

D. groupbar='mode'

Ans: barmode='group'

?

Thank You !!!