

# Introduction to PANDAS

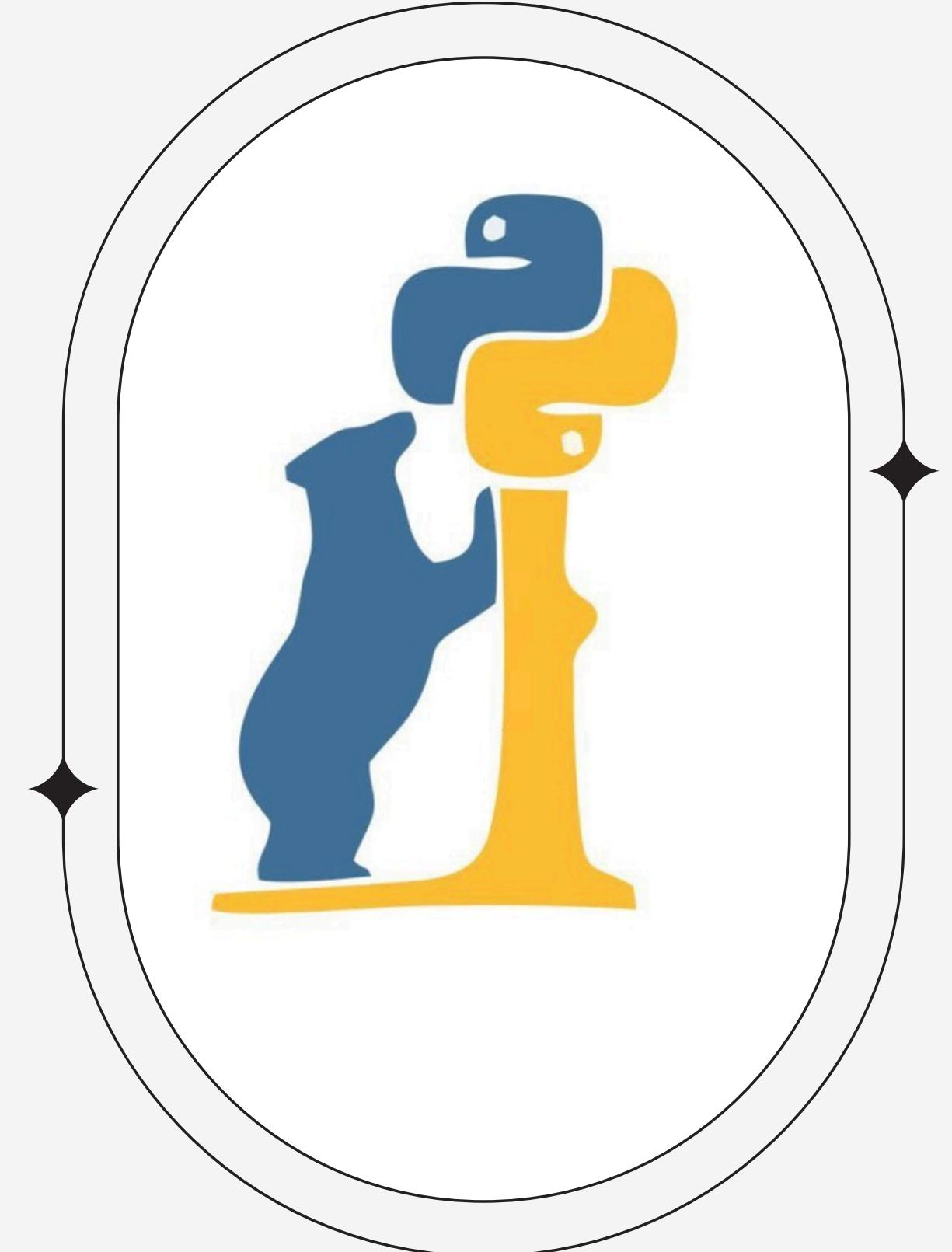
Presented by:

Yokshita Jaiswal



# What is Pandas?

- *Open-source data analysis & manipulation tool*
- *Built on top of NumPy*
- *Created by Wes McKinney in 2008*
- *Highly preferred for data cleaning, transformation, and exploration*



# Key Features

- 🎓 Fast and efficient DataFrame object
- 🎓 Tools for reading and writing data
- 🎓 Handling of missing data
- 🎓 Merge and join datasets



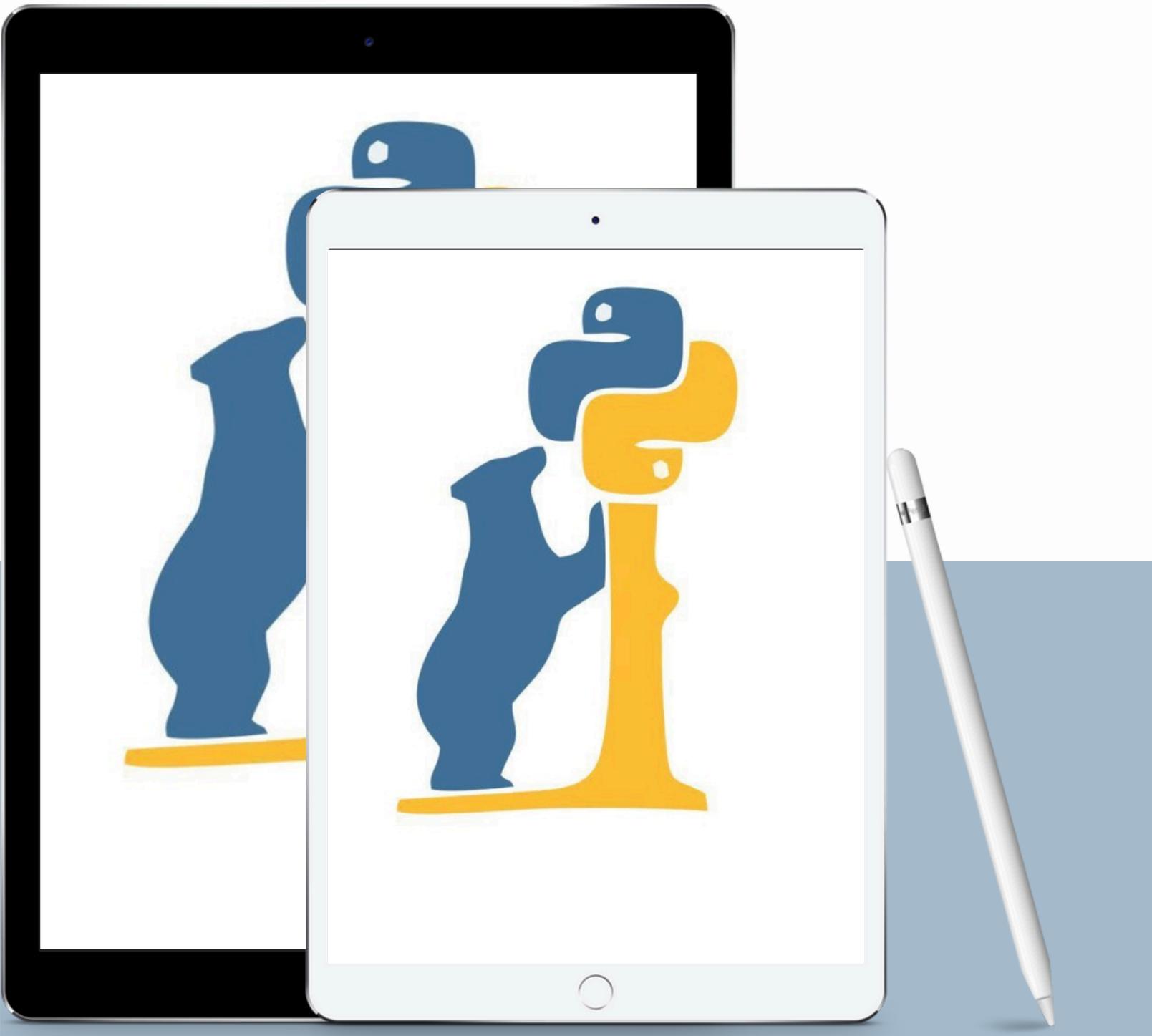
# Core Data Structures

## **Series**

- One-dimensional labeled array
- Example:  
`pd.Series([1, 2, 3])`

## **DataFrame**

- Two-dimensional labeled data structure
- Example:  
`pd.DataFrame({'Name': ['Alice', 'Bob'], 'Age': [25, 30]})`



# Reading Data

## From CSV

- df = pd.read\_csv("data.csv")

## From Excel

- df = pd.read\_excel("data.xlsx")



# Data Exploration

## Viewing Data

`df.head(n)` View first *n* rows (default: 5)

`df.tail(n)` View last *n* rows (default: 5)

## Data Summary

`df.info()` Overview of data types, non-null values, memory usage

`df.describe()` Statistical summary (mean, std, min, etc.)

## Data Structure

`df.shape` Returns (rows, columns)

`df.columns` List of column names

`df.index` Row index range



# Data Exploration

## Value Counts

*Frequency of unique values in a column*

**EXAMPLE** `df['column_name'].value_counts()`

## Checking for Missing Data

`df.isnull().sum()` Total missing values per column

`df.isnull().any()` Checks if any column has missing values



# Data Cleaning

## Handling Missing Values

```
df.dropna(inplace=True)
```

## Fill missing values

```
df.fillna(0, inplace=True)  
df.fillna(method='ffill') # Forward fill  
df.fillna(method='bfill') # Backward fill
```

## Renaming Columns

```
df.rename(columns={'oldName': 'newName'}, inplace=True)
```

## Removing Duplicates

```
df.drop_duplicates(inplace=True)
```

## Replacing Values

```
df['Column'].replace('old', 'new', inplace=True)
```



# Data Manipulation

## Adding Columns

```
df['New_Col'] = [value1, value2, ...]  
df['Double_Age'] = df['Age'] * 2
```

## Deleting Columns / Rows

```
df.drop('ColumnName', axis=1, inplace=True) # Drop column  
df.drop(index_number, axis=0, inplace=True) # Drop row
```

## Filtering Data

```
df[df['Age'] > 25]
```

## Sorting Data

```
df.sort_values(by='Age', ascending=False)
```



# Grouping & Aggregating

## Basic Syntax

```
df.groupby('Department')
```

## Aggregation Functions

```
df.groupby('Department')  
    ['Salary'].mean()
```

```
df.groupby('Department')  
    ['Salary'].sum()
```

```
df.groupby('Department')  
    ['Salary'].count()
```

## Multiple Aggregations

```
df.groupby('Department')  
    ['Salary'].agg(['mean', 'max',  
                   'min'])
```

## Grouping by Multiple Columns

```
df.groupby(['Department',  
           'Gender'])['Salary'].mean()
```

# Merging and Joining

## ***merge()***

Combines DataFrames based on column values

```
pd.merge(df1, df2, on='ID', how='inner')
```

## ***join()***

Simpler syntax than merge(), joins on index by default

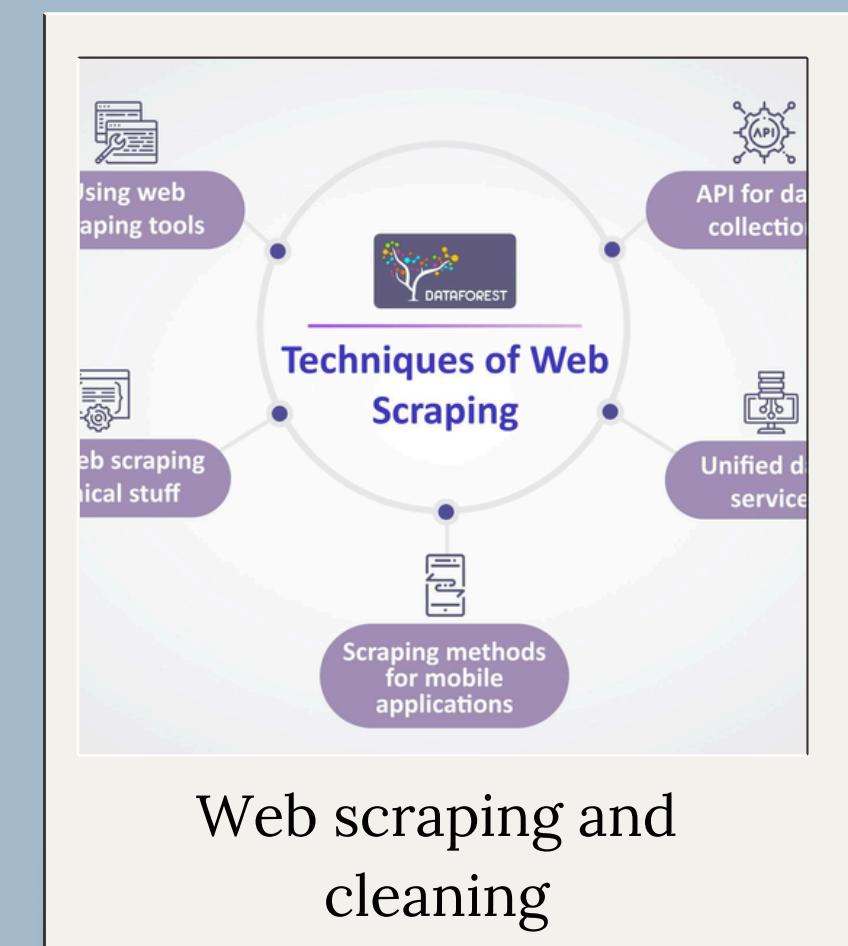
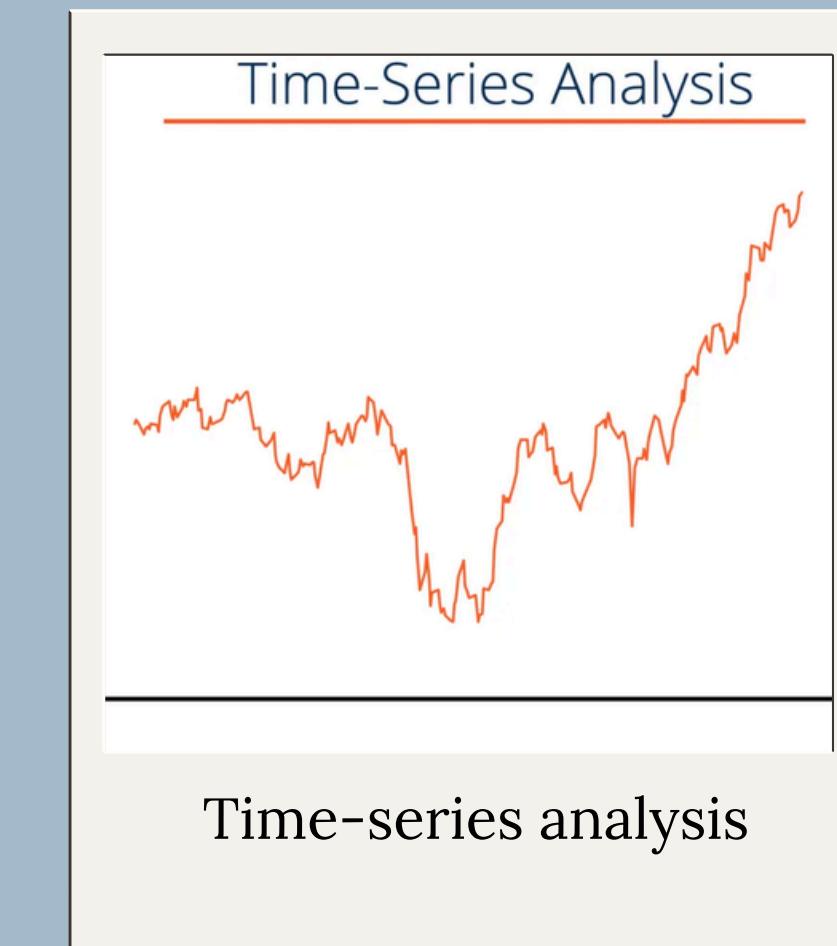
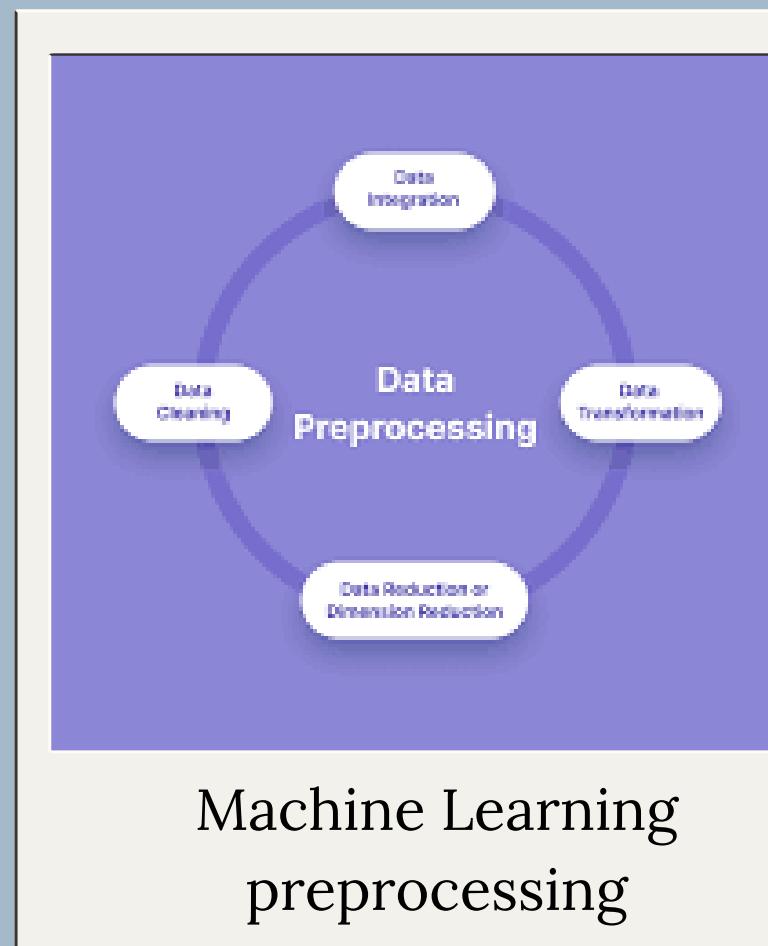
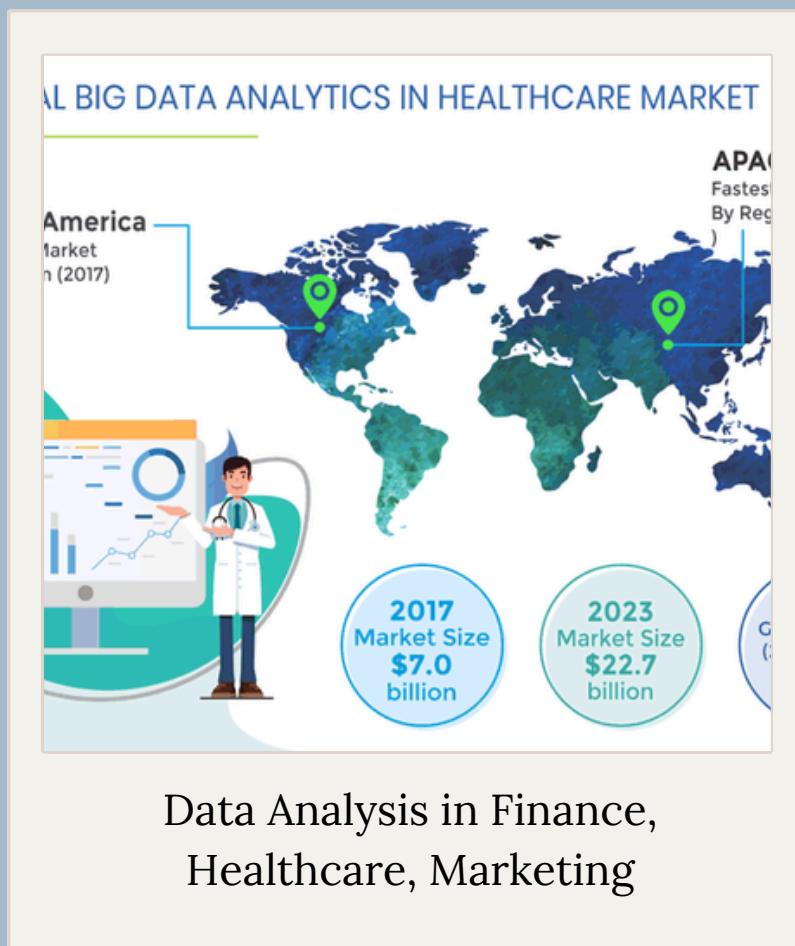
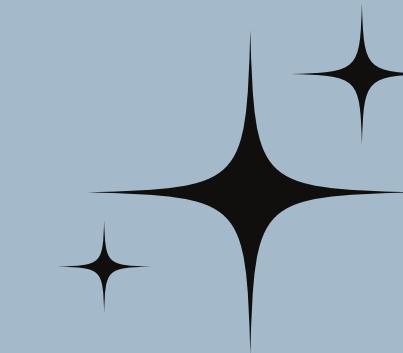
```
df1.join(df2, how='left')
```

## ***concat()***

```
pd.concat([df1, df2]) # Row-wise
```

```
pd.concat([df1, df2], axis=1) # Column-wise
```

# Real-World Use Cases





THANK YOU

