```
Start coding or generate with AI.
```

What is Pandas?

Pandas is a Python library used for working with data sets.

It has functions for analyzing, cleaning, exploring, and manipulating data.

The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

Pandas (stands for Python Data Analysis) is an open-source software library designed for data manipulation and analysis.

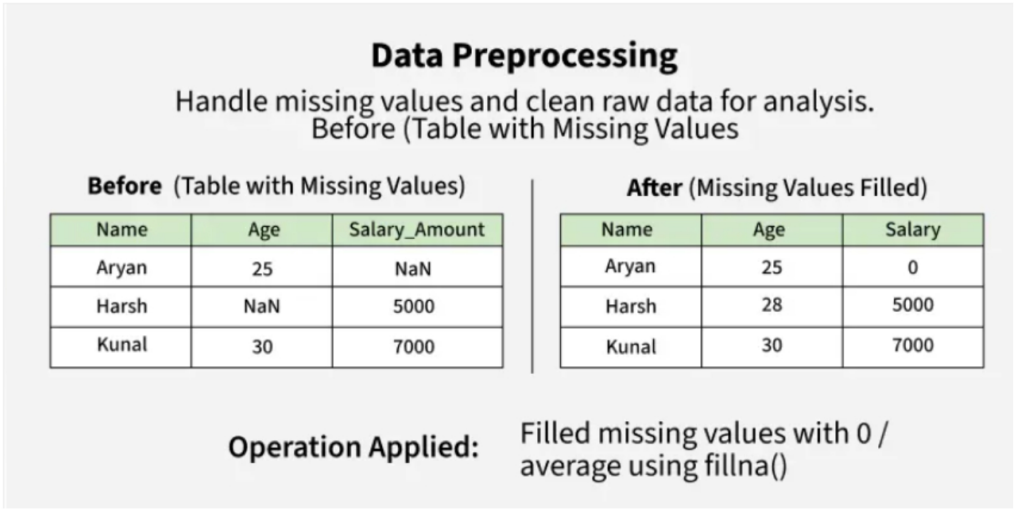
Revolves around two primary Data structures: Series (1D) and DataFrame (2D)

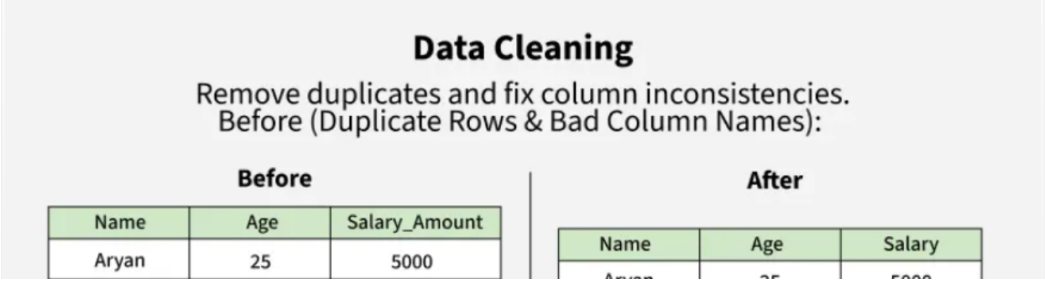
Built on top of NumPy, efficiently manages large datasets, offering tools for data cleaning, transformation, and analysis.

Tools for working with time series data, including date range generation and frequency conversion. For example, we can convert date or time columns into pandas' datetime type using pd.to_datetime(), or specify parse_dates=True during CSV loading.

Seamlessly integrates with other Python libraries like NumPy, Matplotlib, and scikit-learn.

Provides methods like .dropna() and .fillna() to handle missing values seamlessly





Harsh	30	7000
Aryan	25	5000

Aryan	25	5000
Harsh	30	7000

Operation Applied: Removed duplicates using drop_duplicates(), renamed columns using rename()

Data Transformation

Filter and sort data for better insights.

Before (unfiltered Data)

Product	Category	Price
Laptop	Electronics	800
Hoodie	Clothing	20
Smartphone	Electronics	500
Jeans	Clothing	40

After (Filtered & Sorted Data)

Product	Category	Price
Smartphone	Electronics	500
Laptop	Electronics	800

Operation Applied: Filtered Electronics using query(), sorted by price using sort_values()

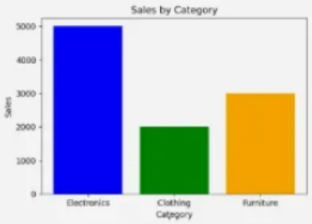
Data Visualization

Represent data graphically for insights

Before (Raw Data Table)

Category	Price
Electronics	5000
Clothing	2000
Furniture	3000

After (Bar Chart Visual)



Operation Applied: Plotted sales data using matplotlib

Double-click (or enter) to edit

What is Pandas Used for?

Reading and writing data from various file formats like CSV, Excel and SQL databases.

Cleaning and preparing data (handling missing values, filtering, removing duplicates).

Merging, joining, and reshaping datasets.

Performing statistical analysis and descriptive statistics.

Visualizing data quickly.

Relevant data is very important in data science.

Double-click (or enter) to edit

Installation of Pandas

If you have Python and PIP already installed on a system, then installation of Pandas is very easy.

Install it using this command:

```
!pip install pandas
```

Now Pandas is imported and ready to use.

```
import pandas

mydataset = {
  'cars': ["BMW", "Volvo", "Ford"],
  'passings': [3, 7, 2]
}
```

```
myvar = pandas.DataFrame(mydataset)

print(myvar)
```

```
    cars  passings
0    BMW         3
1  Volvo         7
2   Ford         2
```

Pandas as pd

Pandas is usually imported under the pd alias.

alias: In Python alias are an alternate name for referring to the same thing.

Create an alias with the as keyword while importing:

```
import pandas as pd

mydataset = {
  'cars': ["BMW", "Volvo", "Ford"],
  'passings': [3, 7, 2]
}

myvar = pd.DataFrame(mydataset)

print(myvar)
```

```
    cars  passings
0    BMW         3
1  Volvo         7
2   Ford         2
```

Pandas Series

What is a Series?

A Pandas Series is like a column in a table.

It is a one-dimensional array holding data of any type.

Create a simple Pandas Series from a list:

```
[ ]
import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a)

print(myvar)
```

```
0    1
1    7
2    2
dtype: int64
```

1 Create DataFrame using List

Example: Student Marks

```
[1]
✓ 0s
import pandas as pd

data = [
    ["Rahul", 85],
    ["Anita", 90],
    ["Suresh", 78]
]

df = pd.DataFrame(data, columns=["Name", "Marks"])
print(df)
```

```
     Name  Marks
0   Rahul     85
1   Anita     90
2  Suresh     78
```

2 Create DataFrame using Tuple

Example: Employee Details

```
[3]
✓ 0s
import pandas as pd

data = (
    ("Amit", 25),
    ("Neha", 28),
    ("Ravi", 30)
)

df = pd.DataFrame(data, columns=["Name", "Age"])
print(df)
```

```
   Name  Age
0  Amit   25
```

```
1  Neha   28
2  Ravi   30
```

Create DataFrame using Dictionary (Most Common)

Example: Student Information

[4] ✓ 0s

```python
import pandas as pd

data = {
    "Name": ["Pooja", "Karan", "Meena"],
    "Marks": [88, 76, 92]
}

df = pd.DataFrame(data)
print(df)
```

```
    Name  Marks
0  Pooja     88
1  Karan     76
2  Meena     92
```

Writing data to CSV Files

Step 1: Create a DataFrame

[5] ✓ 0s

```python
import pandas as pd

data = {
    "Name": ["Rahul", "Anita", "Suresh"],
    "Marks": [85, 90, 78]
}

df = pd.DataFrame(data)
print(df)
```

```
     Name  Marks
0   Rahul     85
1   Anita     90
2  Suresh     78
```

Step 2: Save DataFrame into CSV

[6] ✓ 0s

```python
df.to_csv("students.csv", index=False)
```

Step 3: Read CSV File into DataFrame

[7] ✓ 0s

```python
import pandas as pd

df = pd.read_csv("students.csv")
print(df)
```

```
    Name  Marks
0   Rahul     85
1   Anita     90
2  Suresh     78
```

1 df.head() – View first rows

[8] ✓ 0s

```
df.head()
```

	Name	Marks
0	Rahul	85
1	Anita	90
2	Suresh	78

Next steps: Generate code with df | New interactive sheet

2 df.tail() – View last rows

[9] ✓ 0s

```
df.tail()
```

	Name	Marks
0	Rahul	85
1	Anita	90
2	Suresh	78

3 df.shape – Size of DataFrame

[10] ✓ 0s

```
df.shape
```

```
(3, 2)
```

4 df.columns – Column names

[11] ✓ 0s

```
df.columns
```

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

5 df.index – Row index

[12] ✓ 0s

```
df.index
```

```
RangeIndex(start=0, stop=3, step=1)
```

6 df.info() – Complete DataFrame summary

Shows:

Column names

Data types

Non-null values

[13] ✓ 0s

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
```

```
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Name    3 non-null      object
 1   Marks   3 non-null      int64
dtypes: int64(1), object(1)
memory usage: 180.0+ bytes
```

7 df.describe() – Statistical summary

Works for numerical columns

Mean

Min / Max

Std

[14] ✓ 0s

```
df.describe()
```

	Marks
count	3.000000
mean	84.333333
std	6.027714
min	78.000000
25%	81.500000
50%	85.000000
75%	87.500000
max	90.000000

8 df.dtypes – Data types of columns

[15] ✓ 0s

```
df.dtypes
```

	0
Name	object
Marks	int64

dtype: object

9 df.isnull() – Check missing values

Returns True / False

[16] ✓ 0s

```
df.isnull()
```

	Name	Marks
0	False	False
1	False	False
2	False	False

10 df.isnull().sum() – Count missing values

[17] ✓ 0s

```
df.isnull().sum()
```

	0
Name	0
Marks	0

dtype: int64

11 df.nunique() – Unique values

[18] ✓ 0s

```
df.nunique()
```

	0
Name	3
Marks	3

dtype: int64

12 df.value_counts() – Frequency count (for one column)

[19] ✓ 0s

```
df["Marks"].value_counts()
```

Marks	count
85	1
90	1
78	1

dtype: int64

13 Indexing & Selecting Data

Single Column selection

[20] ✓ 0s

```
df["Marks"]
```

	Marks
0	85
1	90
2	78

dtype: int64

Multiple columns

[21] ✓ 0s

```
df[["Name", "Marks"]]
```

	Name	Marks
0	Rahul	85
1	Anita	90
2	Suresh	78

row selection using loc[] and iloc[].

[22] ✓ 0s

```
import pandas as pd

data = {
    "Name": ["Rahul", "Anita", "Suresh", "Meena"],
    "Marks": [85, 90, 78, 92],
    "Age": [20, 21, 22, 20]
}
```

```
df = pd.DataFrame(data)
print(df)
```

```
    Name  Marks  Age
0   Rahul     85   20
1   Anita     90   21
2  Suresh     78   22
3   Meena     92   20
```

loc[] → Label-based selection

Select row with index label 1

[23] ✓ 0s

```
df.loc[1]
```

	1
Name	Anita
Marks	90
Age	21

dtype: object

Select multiple rows (labels 1 to 3)

[24] ✓ 0s

```
df.loc[1:3]
```

	Name	Marks	Age
1	Anita	90	21
2	Suresh	78	22
3	Meena	92	20

loc includes the last index

Select specific row & column

[25] ✓ 0s

```
df.loc[2, "Marks"]
```

```
np.int64(78)
```

Select rows using condition

[27] ✓ 0s

```
df.loc[df["Marks"] > 85]
```

	Name	Marks	Age
1	Anita	90	21
3	Meena	92	20

Double-click (or enter) to edit

iloc[] → Index-based selection (position)

Select row at position 0

Select row at position 0

[28] ✓ 0s

```
df.iloc[0]
```

	0
Name	Rahul
Marks	85
Age	20

dtype: object

Select multiple rows (positions 1 to 2)

[29] ✓ 0s

```
df.iloc[1:3]
```

	Name	Marks	Age
1	Anita	90	21
2	Suresh	78	22

iloc excludes the last index

Select specific row & column by position

[30] ✓ 0s

```
df.iloc[3, 1]
```

```
np.int64(92)
```

Select first two rows and first two columns

[31] ✓ 0s

```
df.iloc[0:2, 0:2]
```

	Name	Marks
0	Rahul	85
1	Anita	90

Labels

If nothing else is specified, the values are labeled with their index number. First value has index 0, second value has index 1 etc.

This label can be used to access a specified value.

Example

Return the first value of the Series:

Create Labels

With the index argument, you can name your own labels.

Example

Create your own labels:

```
import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a, index = ["x", "y", "z"])

print(myvar)
```

```
x    1
y    7
z    2
dtype: int64
```

When you have created labels, you can access an item by referring to the label.

Example

Return the value of "y":

```
print(myvar["y"])
```

```
7
```

Key/Value Objects as Series

we can also use a key/value object, like a dictionary, when creating a Series.

Example

Create a simple Pandas Series from a dictionary:

```
import pandas as pd

calories = {"day1": 420, "day2": 380, "day3": 390}

myvar = pd.Series(calories)

print(myvar)
```

```
day1    420
day2    380
day3    390
dtype: int64
```

The keys of the dictionary become the labels.

To select only some of the items in the dictionary, use the index argument and specify only the items you want to include in the Series.

Example

Create a Series using only data from "day1" and "day2":

```python
import pandas as pd

calories = {"day1": 420, "day2": 380, "day3": 390}

myvar = pd.Series(calories, index = ["day1", "day2"])

print(myvar)
```

```
day1    420
day2    380
dtype: int64
```

DataFrames

Data sets in Pandas are usually multi-dimensional tables, called DataFrames.

Series is like a column, a DataFrame is the whole table.

Example

Create a DataFrame from two Series:

```python
import pandas as pd

data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}

myvar = pd.DataFrame(data)

print(myvar)
```

```
   calories  duration
0       420        50
1       380        40
2       390        45
```

Pandas DataFrames

What is a DataFrame?

A Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.

Example

Create a simple Pandas DataFrame:

```python
import pandas as pd

data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}
```

```
#load data into a DataFrame object:
df = pd.DataFrame(data)

print(df)
```

```
   calories  duration
0       420        50
1       380        40
2       390        45
```

Locate Row

As you can see from the result above, the DataFrame is like a table with rows and columns.

Pandas use the loc attribute to return one or more specified row(s)

Example

Return row 0:

```
#refer to the row index:
print(df.loc[0])
```

```
calories    420
duration     50
Name: 0, dtype: int64
```

Example

Return row 0 and 1:

Double-click (or enter) to edit

Double-click (or enter) to edit

```
#use a list of indexes:
print(df.loc[[0, 1]])
```

```
   calories  duration
0       420        50
1       380        40
```

Note: When using [], the result is a Pandas DataFrame.

Named Indexes

With the index argument, you can name your own indexes.

Example

Add a list of names to give each row a name:

```
import pandas as pd

data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}
```

```
df = pd.DataFrame(data, index = ["day1", "day2", "day3"])

print(df)
```

```
      calories  duration
day1       420        50
day2       380        40
day3       390        45
```

Locate Named Indexes

Use the named index in the loc attribute to return the specified row(s).

Example

Return "day2":

```
Start coding or generate with AI.
```

```
print(myvar[0])
```

```
Start coding or generate with AI.
```

Question For Students

Name	Marks	Age	City	Result
Rahul	85	20	Indore	Pass
Anita	90	21	Bhopal	Pass
Suresh	78	22	Indore	Fail
Meena	92	20	Ujjain	Pass
Karan	88	23	Bhopal	Pass

Double-click (or enter) to edit

Section A: Short Questions

1.Write Python code to display the first 3 rows of the DataFrame.

2.Find the number of rows and columns in the DataFrame.

3. Display only the Name and Marks columns.

4.Write a command to show column names of the DataFrame.

5. Which function is used to get statistical summary of Marks?

6.Select the row where Name = "Meena" using loc[].

7. Select the second and third rows using iloc[].

8. Display students whose Marks > 85.

9. Find the average marks of all students.

10. Count how many students belong to each City.

Close

Double-click (or enter) to edit

[ ]

```
Start coding or generate with AI.
```