Day 6 – Pandas in Python

📅 – 12/06/2025

---

1. Introduction to Pandas

Pandas is a high-level data manipulation and analysis tool built on the Python programming language. It provides fast, flexible, and expressive data structures.

Why Pandas?

- Clean, analyze, and transform structured data
- Built on NumPy (great performance)
- Integrates well with other tools (Matplotlib, Seaborn, SQLAlchemy, etc.)

Key Features:

- Easy CSV/Excel/SQL/JSON handling
- Grouping, filtering, aggregation
- Time series support
- Missing data handling
- Built-in visualization support

---

2. Pandas Data Structures

🔹 Series (1D)

```
import pandas as pd

s = pd.Series([10, 20, 30], index=["a", "b", "c"])

print(s)
```

🔹 DataFrame (2D)

```
data = {"Name": ["Alice", "Bob"], "Age": [25, 30]}

df = pd.DataFrame(data)
```

🔹 Panel (3D, deprecated)

Use MultiIndex DataFrames or external libraries like xarray.

---

3. Reading and Writing Data

Read

```
df_csv = pd.read_csv("data.csv")

df_excel = pd.read_excel("data.xlsx", sheet_name="Sheet1")

df_json = pd.read_json("data.json")
```

Write

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

df.to_json("output.json")
```

---

4. Basic DataFrame Operations

```
df.head()        # First 5 rows

df.tail()        # Last 5 rows

df.shape         # Rows, Columns

df.columns       # Column names

df.info()        # Data types & non-null counts

df.describe()    # Summary statistics
```

Data Selection

```
df['column']            # Single column

df[['col1', 'col2']]    # Multiple columns

df.iloc[0:3]            # Index-based selection

df.loc[0:3]             # Label-based selection
```

---

5. Data Cleaning

Handling Nulls

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

df.dropna(inplace=True)
df.fillna("Unknown", inplace=True)
```

Duplicates

```
df.duplicated().sum()

df.drop_duplicates(inplace=True)
```

Rename / Replace

```
df.rename(columns={'old': 'new'}, inplace=True)

df.replace({"old_val": "new_val"}, inplace=True)
```

---

6. Data Transformation

Filtering

```
df[df['Age'] > 30]

df.query("Age > 30 and Country == 'India'")
```

Sorting

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

Creating Columns

```
df['Total'] = df['Quantity'] * df['Price']
```

---

7. Grouping and Aggregation

groupby() Usage

```
# Total sales by category

df.groupby("Category")['Sales'].sum()


# Multiple aggregations

df.groupby("Region").agg({

    'Sales': 'sum',

    'Profit': 'mean'

})
```

---

8. Joining and Merging

Merge (SQL-style joins)

```
pd.merge(customers, orders, on="customer_id", how="inner")  # left, right, outer
```

Concatenate

```
pd.concat([df1, df2], axis=0)  # Row-wise
pd.concat([df1, df2], axis=1)  # Column-wise
```

---

9. Date and Time Operations

```
df['order_date'] = pd.to_datetime(df['order_date'])
df['year'] = df['order_date'].dt.year
df['month'] = df['order_date'].dt.month
```

---

10. File Input/Output Operations

Read from different formats

```
pd.read_csv("file.csv")
pd.read_excel("file.xlsx")
pd.read_sql("SELECT * FROM users", conn)
```

Write to files

```
df.to_csv("output.csv")
df.to_json("output.json")
```

---

11. Hands-on Data Processing with NumPy & Pandas

NumPy integration

```
import numpy as np
df['discounted_price'] = np.where(df['price'] > 1000, df['price'] * 0.9, df['price'])
```

Error Handling

```
try:
   df = pd.read_csv("data.csv")
```

```
except FileNotFoundError:
  print("File not found!")
```

---

12. Useful Functions

Function	Description
mean()	Mean value
count()	Count non-null values
value_counts()	Count frequency of values
apply()	Apply custom function to column
map()	Apply mapping to series
pivot_table()	Create pivot tables
duplicated()	Detect duplicate rows

---

13. Real-World Use Cases

Use Case	Method
Salary analytics	groupby(['dept'])['salary'].mean()
Monthly revenue trend	Extract year_month and groupby()
Finding top customers	groupby('customer_id')['total'].sum()
Exporting dashboards	to_excel(), to_json()
