# Pandas Notes (Zero ➜ Advanced)

These notes are written for **absolute beginners** and gradually move to **advanced, industry-level Pandas usage**. They are **note-friendly**, **simple**, and **exam/interview oriented**.

---

## 1. What is Pandas?

**Pandas** is a Python library used for: - Data Analysis - Data Cleaning - Data Manipulation - Working with structured data (tables)

### Why Pandas?

- Easy handling of large datasets
- Fast operations
- Works well with NumPy, Matplotlib, Seaborn, ML

### Install Pandas

```
pip install pandas
```

### Import Pandas

```
import pandas as pd
```

---

## 2. Core Data Structures in Pandas

### 2.1 Series

A **Series** is a **1D labeled array**.

```
s = pd.Series([10, 20, 30, 40])
print(s)
```

With custom index:

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

### Important Series Attributes

- `s.values`
- `s.index`
- `s.dtype`

---

### 2.2 DataFrame

A **DataFrame** is a **2D table (rows + columns)**.

```
data = {
    'Name': ['Ali', 'Ahmed', 'Sara'],
    'Age': [20, 22, 21]
}

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

---

## 3. Creating DataFrames

### From Dictionary

```
pd.DataFrame({'A':[1,2], 'B':[3,4]})
```

### From List of Lists

```
pd.DataFrame([[1,2],[3,4]], columns=['A','B'])
```

### From CSV File

```
df = pd.read_csv('data.csv')
```

### From Excel

```
pd.read_excel('data.xlsx')
```

---

## 4. Exploring Data

### Basic Functions

```
df.head()      # first 5 rows
df.tail()      # last 5 rows
df.shape       # rows, columns
df.info()      # structure
df.describe()  # statistics
```

### Column & Index

```
df.columns
df.index
```

---

## 5. Selecting Data

### Select Column

```
df['Age']
df.Age
```

### Select Multiple Columns

```
df[['Name','Age']]
```

### Select Rows (iloc & loc)

#### iloc (index-based)

```
df.iloc[0]
df.iloc[0:3]
```

#### loc (label-based)

```
df.loc[0]
df.loc[:, 'Age']
```

---

## 6. Adding & Removing Columns

### Add Column

```
df['Salary'] = [20000, 25000, 30000]
```

### Insert Column

```
df.insert(0, 'ID', [1,2,3])
```

### Delete Column

```
df.drop('Salary', axis=1, inplace=True)
```

---

## 7. Adding & Removing Rows

### Add Row

```
df.loc[len(df)] = ['Usman', 23]
```

### Drop Row

```
df.drop(0, inplace=True)
```

---

## 8. Handling Missing Values (Very Important)

### Detect Missing Values

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

### Remove Missing Values

```
df.dropna()
```

### Fill Missing Values

```
df.fillna(0)
df['Age'].fillna(df['Age'].mean())
```

---

## 9. Data Type Handling

### Check Data Types

```
df.dtypes
```

### Change Data Type

```
df['Age'] = df['Age'].astype(int)
```

---

## 10. Sorting Data

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

---

## 11. Filtering Data

### Simple Condition

```
df[df['Age'] > 20]
```

### Multiple Conditions

```
df[(df['Age'] > 20) & (df['Salary'] > 25000)]
```

---

## 12. String Operations

```
df['Name'].str.upper()
df['Name'].str.lower()
df['Name'].str.contains('a')
```

---

## 13. Apply & Lambda Functions

```
df['Age'] = df['Age'].apply(lambda x: x + 1)
```

---

## 14. GroupBy (Very Important)

Used for **aggregation**.

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

Common Aggregations: - mean() - sum() - count() - max() - min()

---

## 15. Merging & Joining DataFrames

### Merge

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

### Types of Join

- inner
- left
- right
- outer

---

## 16. Concatenation

```
pd.concat([df1, df2])
```

## 17. Pivot Tables

```
pd.pivot_table(df, values='Salary', index='Department', aggfunc='mean')
```

---

## 18. Time Series Data

```
df['Date'] = pd.to_datetime(df['Date'])
df['Year'] = df['Date'].dt.year
```

---

## 19. Data Visualization (Basic)

```
df['Salary'].plot(kind='hist')
df.plot(x='Age', y='Salary', kind='scatter')
```

---

## 20. Performance & Optimization (Advanced)

- Use vectorized operations
- Avoid loops
- Use `category` dtype

```
df['Dept'] = df['Dept'].astype('category')
```

---

## 21. Real-World Data Analysis Workflow

1. Load data
2. Explore data
3. Clean data
4. Handle missing values
5. Feature engineering
6. Analysis
7. Visualization
8. Export results

---

## 22. Exporting Data

```
df.to_csv('output.csv', index=False)
df.to_excel('output.xlsx', index=False)
```

---

## 23. Pandas for Data Analyst (Must-Know)

- read_csv / read_excel
- isnull / fillna
- groupby
- merge
- sort_values
- apply
- pivot_table

---

## 24. Common Interview Questions

- Difference between Series & DataFrame
- loc vs iloc
- groupby use case
- handling missing values
- merge vs concat

---

## 25. Summary

Pandas is the **heart of Data Analysis** in Python. Mastering Pandas means you can: - Clean real-world messy data - Perform business analysis - Prepare data for Machine Learning

---

✅**You are now Pandas-ready from Zero to Advanced**