ADVANCED PANDAS

1. What is Pandas?

Pandas is an open-source Python library that provides high-performance data structures and data analysis tools. It is widely used for data cleaning, transformation, analysis, and visualization. The core idea behind Pandas is to provide easy-to-use data structures: Series (1D) and DataFrame (2D), similar to Excel tables or SQL data.

Pandas is built on top of NumPy and is especially useful for structured data.

2. Installing and Importing

To install pandas:

```
pip install pandas
```

To import pandas in Python:

```
import pandas as pd
```

3. Core Data Structures

Pandas provides two primary data structures:

Series: A one-dimensional labeled array.

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

DataFrame: A two-dimensional labeled data structure with columns of potentially different types.

```
data = {
  'Name': ['Alice', 'Bob'],
  'Age': [25, 30]
}
df = pd.DataFrame(data)
```

4. Reading and Writing Data

Reading data from various sources:

```
pd.read_csv('file.csv')
pd.read_excel('file.xlsx')
pd.read_json('file.json')
```

Writing data to files:

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

df.to_excel('output.xlsx')

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

5. Data Exploration

Inspecting the structure and content of a DataFrame:

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

df.tail()          # Last 5 rows

df.info()          # Summary info

df.describe()      # Descriptive statistics

df.columns         # Column names

df.shape           # Dimensions

df.dtypes          # Data types

df.isnull().sum()  # Null value count
```

6. Selecting and Filtering Data

Selecting columns:

```
df['Name']

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

Selecting rows:

```
df.loc[0]    # By index label

df.iloc[0]   # By position
```

Conditional filtering:

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

df[(df['Age'] > 25) & (df['Gender'] == 'Male')]
```

7. Data Cleaning and Preparation

Handling missing data:

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

df.dropna()
```

df.fillna(0)

df.fillna(method='ffill')

Removing duplicates:

df.duplicated()

df.drop_duplicates()

Changing data types:

df['Age'] = df['Age'].astype(float)

Replacing values:

df['Gender'].replace({'M': 'Male', 'F': 'Female'})

8. Data Transformation

Applying functions:

df['Taxed_Salary'] = df['Salary'].apply(lambda x: x * 0.7)

Using map and replace:

df['Category'].map({'A': 1, 'B': 2})

df['Status'].replace(['Single', 'Married'], [0, 1])

Renaming columns:

df.rename(columns={'Name': 'FullName'})

9. Merging and Joining

Merging (SQL-style joins):

pd.merge(df1, df2, on='ID', how='inner') # how: inner, outer, left, right

Concatenation:

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

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

Joining on index:

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

10. GroupBy and Aggregation

Single column aggregation:

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

Multiple column aggregation:
df.groupby(['Department', 'Gender']).agg({'Salary': ['sum', 'mean'], 'Age': 'max'})

11. Pivot Tables and Crosstabs

Creating pivot tables:

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

Creating crosstab (frequency table):

pd.crosstab(df['Gender'], df['Department'])

12. Time Series

Converting to datetime:

df['Date'] = pd.to_datetime(df['Date'])

Extracting date components:

df['Year'] = df['Date'].dt.year

df['Month'] = df['Date'].dt.month

df['Day'] = df['Date'].dt.day

Resampling:

df.set_index('Date', inplace=True)

df.resample('M').mean() # Monthly average

13. Sorting and Indexing

Sorting data:

df.sort_values('Age')

df.sort_values('Salary', ascending=False)

Changing and resetting index:

df.set_index('ID', inplace=True)

df.reset_index(inplace=True)

Sorting index:

df.sort_index()

14. Advanced Operations

Rolling window:

```
df['Rolling_Mean'] = df['Sales'].rolling(window=3).mean()
```

Binning:

```
pd.cut(df['Age'], bins=[0, 18, 60, 100], labels=['Child', 'Adult', 'Senior'])
```

MultiIndexing:

```
df.set_index(['City', 'Year'])
```

```
df.loc[('Delhi', 2022)]
```

Custom aggregation:

```
df.groupby('Gender').agg({
   'Age': 'mean',
   'Salary': ['min', 'max']
})
```

15. Visualization with Pandas

Basic plotting (requires matplotlib):

```
df['Salary'].plot()                # Line plot
df['Age'].plot(kind='hist')        # Histogram
df['Gender'].value_counts().plot(kind='bar')  # Bar plot
df.boxplot(column='Salary')        # Box plot
```

You can also use seaborn or matplotlib for more advanced plots.