**📘 Pandas Library – Detailed Quick Notes with Explanations (2–3 lines each)**

**📦 Import**

import pandas as pd

Imports the pandas library with alias pd, a standard convention for ease of use.

**📄 Data Structures**

**Series**

pd.Series(data, index=None, dtype=None)

Creates a 1D labeled array.

* data: list, array, scalar, etc.
* index: optional labels for elements.
* dtype: optional data type (e.g., int, float).

**DataFrame**

pd.DataFrame(data, index=None, columns=None, dtype=None)

Creates a 2D table with labeled rows and columns.

* data: dict, list of dicts, arrays.
* index: row labels.
* columns: column names.

**📥 Input/Output**

**pd.read\_csv()**

pd.read\_csv(filepath, sep=',', header='infer')

Reads a CSV file into a DataFrame.

* sep: delimiter (default ',').
* header: row to use as column names.

**df.to\_csv()**

df.to\_csv(filepath, index=True)

Writes DataFrame to CSV.

* index: whether to write row indices.

**pd.read\_excel() / df.to\_excel()**

Read and write Excel files.

* sheet\_name: name/index of sheet to read/write.

**pd.read\_json() / df.to\_json()**

Import/export JSON files.

* orient: structure of JSON like 'records', 'columns', etc.

**📊 Basic Exploration**

df.head(n), df.tail(n)

Return first or last n rows (default is 5).

df.info(), df.describe()

info() shows summary of columns and data types.  
describe() provides statistical summary of numeric columns.

df.shape, df.columns, df.index, df.dtypes

Return shape, column names, row labels, and data types.

**🧭 Selection**

df['col'], df[['col1', 'col2']]

Select one or multiple columns.

df.loc[row\_label, col\_label]

df.iloc[row\_index, col\_index]

Access rows/columns using label or integer position.

df.at[row, col], df.iat[row, col]

Fast access to single scalar values.

**🔍 Filtering**

df[df['col'] > value]

Returns rows where condition is True.

df[(df['A'] > 10) & (df['B'] == 'x')]

Combines multiple conditions using &, |.

**✍️ Editing / Updating**

df['new\_col'] = value

Adds or modifies a column.

df.loc[2, 'col'] = 100

Updates a specific cell.

df.replace({old: new}), df.rename(columns={})

Replaces values or renames columns.

df.drop(columns=['col1'], index=[0], inplace=True)

Drops rows or columns.

* inplace=True: applies changes directly.

**🔁 Iteration**

df.iterrows(), df.itertuples()

Iterate over rows as Series or namedtuples (faster).

**📐 Sorting**

df.sort\_values(by='col', ascending=True)

Sorts rows based on a column.

df.sort\_index()

Sorts DataFrame by row index.

**🧮 Aggregation / Statistics**

df.sum(), df.mean(), df.median(), df.mode()

Returns aggregate statistics per column.

df.min(), df.max(), df.std(), df.var()

Minimum, maximum, standard deviation, variance.

df.count(), df['col'].value\_counts()

Counts non-null values or unique occurrences.

**🔁 Grouping**

df.groupby('col')

Groups rows by unique values in a column.

df.groupby(['A']).agg({'B': 'sum'})

Performs aggregation on grouped data.

**🔗 Merging / Joining**

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

Merge two DataFrames on a key.

* how: 'left', 'right', 'outer', 'inner'.

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

Join using index or key.

pd.concat([df1, df2], axis=0/1)

Concatenate DataFrames vertically or horizontally.

**🎛️ Reshaping**

df.pivot(index, columns, values)

Reshapes data (wide format).

df.pivot\_table(values, index, columns, aggfunc)

Pivot with aggregation like mean/sum.

df.melt(id\_vars=[], value\_vars=[])

Unpivots wide to long format.

df.stack(), df.unstack(), df.transpose()

Convert between columns and rows.

**🧹 Missing Data**

df.isnull(), df.notnull()

Detect missing values.

df.dropna(), df.fillna(value)

Drop or fill missing values.

df.ffill(), df.bfill()

Forward-fill or backward-fill missing values.

**🧠 Apply Functions**

df.apply(func, axis=0/1)

Apply a function to rows or columns.

df['col'].map(func or dict)

Map each value to a new value using a function or dictionary.

df.applymap(func)

Apply function to all DataFrame elements.

**📏 Rolling / Expanding / EWM**

df.rolling(window=n).mean()

Moving average over rolling window.

df.expanding().sum()

Cumulative operation expanding from start.

df.ewm(span=n).mean()

Exponentially weighted moving average.

**🔁 Type Conversion**

df.astype({'col': 'int'})

Convert column to new data type.

pd.to\_numeric(), pd.to\_datetime()

Convert strings to numeric or datetime format.

**📅 Date/Time**

df['date'].dt.year, .month, .day

Extract year, month, or day from datetime.

df['date'].dt.strftime('%Y-%m-%d')

Convert datetime to formatted string.

**🔤 String Operations**

df['col'].str.lower(), .contains(), .replace()

Perform string manipulations like lowercase, search, replace.

**✅ Utilities**

df.duplicated(), df.drop\_duplicates()

Identify or remove duplicate rows.

df.sample(n)

Returns a random sample of rows.

df.nunique(), df.clip(lower, upper)

Unique value count and bounding of values.

**📈 Cumulative Functions**

df.cumsum(), df.cumprod(), df.cummin(), df.cummax()

Cumulative sum, product, min, and max.

**📌 Categoricals**

df['col'].astype('category')

Convert to category type to save memory.

df['col'].cat.codes

Access category codes (integers).

**🧾 Exporting Data**

df.to\_csv(), df.to\_excel(), df.to\_json()

Export to CSV, Excel, or JSON formats.

**⚠️ Error Handling**

try:

...

except pd.errors.EmptyDataError:

...

Handle file reading or format errors.

Would you like me to convert this Markdown into an actual downloadable **PDF** and send you the content so you can paste it into Typora or another tool?