Dataframe

Pandas DataFrame: It provides a powerful, flexible data structure called the DataFrame (similar to tables in databases or Excel). This allows you to handle data in rows and columns efficiently.

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".

1. Creation of Dataframes

You can create a DataFrame from various inputs such as lists, dictionaries, NumPy arrays, etc.

pandas.DataFrame(data, index, columns)

Parameters:

- **data** : It is a dataset from which a DataFrame is to be created. It can be a list, dictionary, scalar value, series, and arrays, etc.
- **index** : It is optional, by default the index of the DataFrame starts from 0 and ends at the last data value(n-1). It defines the row label explicitly.
- **columns** : This parameter is used to provide column names in the DataFrame. If the column name is not defined by default, it will take a value from 0 to n-1.

`pd.DataFrame()` ➔ Creating an empty dataframe

`pd.DataFrame(data)` ➔ Creating a dataframe from list or dictionary

2. Viewing Data

We use viewing operations in pandas to quickly understand the structure and content of a DataFrame.

`df.head(n)` ➔ View the first n rows (default is 5).
`df.tail(n)` ➔ View the last n rows.
`df.info()` ➔ Summary of the DataFrame (including data types).
`df.describe()` ➔ Generate descriptive statistics for numeric columns.

3. Dealing with Rows and Columns

We can perform basic operations on rows/columns like selecting, deleting, adding, and renaming.

- **Column Selection:** In Order to select a column in Pandas DataFrame, we can either access the columns by calling them by their columns name.
  `df['col_name']` ➔ Select a column
  `df[['col1', 'col2']]` ➔ Select multiple columns
- **Row Selection:** Pandas provide a unique method to retrieve rows from a Data frame. DataFrame.loc[] method is used to retrieve rows from Pandas DataFrame. Rows can also be selected by passing integer location to an iloc[] function.
  `df.loc['row_name']` ➔ Selecting a row by name
  `df.iloc[1,]` ➔ Selecting a row by index
- **Cell Selection:**
  `df.loc['row_name, col_name']` ➔ selecting a cell by names
  `df.loc[0, 1]` ➔ selecting a cell by indices

4. Filtering and Conditional Selection

It is essential for extracting specific subsets of data based on conditions.

`df[df['column'] > value]` ➔ Filter rows based on condition
`df.query(' column > value ')` ➔ Filter using query string syntax

5. Sorting Data

`df.sort_values(by='column')` ➔ Sort by a specific column
`df.sort_index()` ➔ Sort by row index

6. Working with Missing Data

Missing Data is a very big problem in real life scenario.

- **Checking for missing values:** In order to check missing values in Pandas DataFrame, we use a function `isnull()` and `notnull()`.
- **Filling missing values:** In order to fill null values in a datasets, we use `fillna()` and `replace()` functions.
- **Dropping missing values using dropna() :** In order to drop a null values from a dataframe, we used `dropna()` function this fuction drop Rows/Columns of datasets with Null values in different ways.

7. Adding and Removing Data

`df['new_column'] = value` ➔ Add a new column
`df.drop(columns=['col']):` ➔ Remove a column
`df.append()` ➔ Add new rows
`df.drop(index)` ➔ Drop rows by index

8. Iterating over rows and columns

Pandas DataFrame consists of rows and columns so, in order to iterate over dataframe, we have to iterate a dataframe like a **dictionary**.

- **Iterating over rows:** In order to iterate over rows, we can use three function `iteritems()`, `iterrows()`, `itertuples()`.
- **Iterating over Columns:** In order to iterate over columns, we need to create a **list of dataframe columns** and then iterate through that list to pull out the dataframe columns.

9. Grouping and Aggregation

`df.groupby('column')` ➔ Group data based on column values

`df.agg()` ➔ Aggregate multiple columns using different functions

10. Exporting and Importing Data

`df.to_csv('file.csv')` ➔ Export a DataFrame to a CSV file

`pd.read_csv('file.csv')` ➔ Read a CSV file into a DataFrame

`df.to_excel('file.xlsx')` ➔ Export to Excel