PANDAS.

Use case of Pandas :-

1. Data cleaning
2. Data Analysis
3. Data transformation
4. Data visualization (Basic level)
5. Data Aggregation
6. File handling
7. Data filtering & selection
8. Time series analysis.

① Series

① Stacking a data into vertical way.

② 1D labeled array capable of holding any data type.

③ Axis labels are collectively called the index.

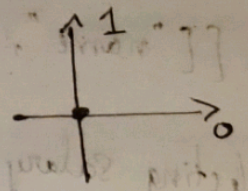
→ Creating Series :-

```
import numpy as np
import pandas as pd

labels = ['a', 'b', 'c']
my_lists = [10, 20, 30]
arr = np.array([10, 20, 30])
d = {'a':10, 'b':20, 'c':30}

pd.Series(labels) # Same with my_list, arr & d.
```

→
```
0  a
1  b
2  c
```

② Data Frames

axis = 1 [col]
axis = 0 [row]



① Dataframe using Dict

```
data = {
    'Name' : ['John', 'Anna', 'peter', 'Linda'],
    'Age'  : [28, 34, 29, 42],
    'City' : ['Newyork', 'paris', 'Berlin', 'London'],
    'Salary' : [65000, 70000, 62000, 85000]
}
df = pd.DataFrame(data)
```

② Dataframe using list :-

```
data_list = [
    ['John', 28, 'Newyork', 65000],
    ['Anna', 34, 'paris', 70000],
    ['peter', 29, 'Berlin', 62000],
    ['Linda', 42, 'London', 85000]
]
df2 = pd.DataFrame(data_list)
```

→ giving col names to list :-

```
cols = ["name", "age", "city", "salary"]
df2 = pd.Dataframe(data_list, columns = columns)
df2
```

② Selecting & indexing of cols

```
# Selecting Name & city
df2[["name", "city"]]

# selecting salary.
df2["salary"]

# Creating a new col :
df2["Designation"] = ["Doctor", "Scientist", "Lawyer",
                      "Designer"]

# Removing col :-
df2.drop(["designation"], axis = 1, inplace = True)

# Removing rows :-
df2.drop(3, axis = 0)

# Selecting rows using iloc function
df2.loc[[0, 1]]

df2.iloc[[2, 3]]
```

Selecting subsets of Rows & cols

① df2.loc[[0,1]][["city", "salary"]]

② df2.loc[[2,3]][["name", "age"]]

---

③ Conditional Selection

Selecting People whose age is greater than 30

df2[df2["age"]>30]

selecting people with age > 30 & live in paris

df2[(df2["age"]>30) & (df2["city"]=="paris")]

Finding Missing Data

```
import numpy as np
import pandas as pd

data = {
    'A' : [1, 2, np.nan, 4, 5],
    'B' : [1, 2, 3, 4, 5],
    'C' : [1, 2, 3, np.nan, np.nan],
    'D' : [1, np.nan, np.nan, np.nan, 5]
}

df = pd.DataFrame(data)
```

① df.isna() # returns Bool. True means NaN

② df.isna().sum()

③ df.isna().any() # whether the column contains at least one missing value.

② Removing a missing data

① df.dropna()

② df.dropna(thresh=1) # It will keep all rows in a dataframe that have atleast one non-missing value.

---

③ Filling the missing data

① df.fillna(0)

② values = {"A": 0,
'B': 100,
'C': 300,
'D': 400
}

df.fillna(value = values)

③ df.fillna(df.mean())

Merging joining & concatination :-

```
employees = pd.DataFrame({
    'employee_id' = [1, 2, 3, 4, 5],
    'name' = ['John', 'Anna', 'Peter', 'Linda', 'Bob'],
    'department' = ['HR', 'IT', 'Finance', 'IT', 'HR']
})

Salaries = pd.Dataframe({
    'employee_id' : [1, 2, 3, 6, 7],
    'Salary' : [60000, 80000, 65000, 70000, 90000],
    'bonus' : [5000, 10000, 7000, 8000, 12000]
})
```

---

① Merging two dataframes :-

```
pd.merge(employees, Salaries, on = "employee_id",
         how = 'inner')

# outer
# Left
# right
```

② Concatination of two dataframes :-

① pd.concat ([df1, df2]) #col

② pd.concat ([df1, df2], axis = 1) #row

③ Joining two dataframes :-

① df1.join (df2, how = 'outer')

② df2.join (df1)

Groupby & aggregation

```
import numpy as np
import pandas as pd

data = {
    'Category' : ['A', 'B', 'A', 'B', 'A', 'B', 'A', 'B']
    'store' : ['S1', 'S1', 'S2', 'S2', 'S1', 'S2', 'S2',
                                            'S1' ],
    'Sales' : [100, 200, 150, 250, 120, 180, 200, 300],
    'Quantity' : [10, 15, 12, 18, 8, 20, 15, 25],
    'Date' : pd.date_range('2023-01-01', periods=8)
}
df = pd.DataFrame(date)
```

---

① Groupby Category & Calculate the Sum of Sales

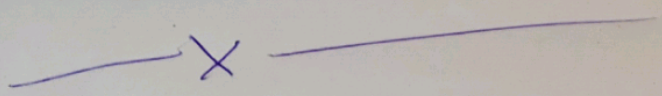
```
Cat = df.groupby('category')['Sales'].sum()
Cat.
```

② Groupby by Multiple cols & Groupby Category and ~~Sales~~ Sales :

```
Cat = df.groupby(['category', 'Sales'])['Sales'].sum
```

→ df['sales'].mean()

→ df['sales'].agg(['sum', 'mean', 'min', 'max',
'count', 'std', 'median'])



Source code on Github