**Test 2**

**Q.1 What is Pandas ?**

**Ans.** Pandas is a python library . Pandas is used to analyze data. Pandas is a python library used for working with datasets.

It has functions for analyzing , cleaning , exploring , and manipulating data.

The name “Pandas” has a reference to both “Panel Data” , and “Python Data Analysis “ and was created by Wes Mckinney in 2008.

**Features :**

* High performance Data analysis tool
* Working with large data set
* Supports more load files with with different formats
* More flexible
* Represents in Tabular way(Rows and columns)
* Working on missing data
* Indexing – slicing – subsetting the large data set
* Merge and join two different datasets easily
* Reshape datasets

**Why Use Pandas ?**

Pandas allows us to analyze bid data and make conclusions based on statistical theories.

Pandas can clean messy data sets , and make them readable and relevant.

Relevant data is very important in data science.

**Q.2 Difference between series and dataframe ?**

**Ans.**

**Series :**

* A pandas series is like a column in a table.
* It is a one – dimensional array holding data of any type. We will get only values not lable (column). We can check series using type(variable).
* Ex. import pandas as pd

a = [1,7,2]

myvar =pd.Series(a)

print(myvar)

* A series can store more simple and homogeneous data .
* A series can only have one data type for the whole array. Series can only handle one data type at a time .
* Series are more suited for representing and manipulating single columns of data.
* Series are designed for element-wise operations , descriptive statistics , and indexing within a single variable

**DataFrame :**

* Dataframe is multi-dimensional data.We will get column name with values.we can check dataframe using type(variable).

Ex : a = { “Emp\_id” : [1, 2 , 3],

“Emp\_Name “ : [‘sam’ , ‘kirti’ , ‘pallavi’],

“Salary” : [20000 , 35000 , 40000] }

df= pd.DataFrame(a)

df

Output :

Emp\_id Emp\_Name Salary

0 1 sam 20000

1 2 kirti 35000

1. 3 pallavi 40000

* A dataframe can store more complex and heterogeneous data.
* A dataframe can have different data type for each column.This means that a dataframe can handle mixed data types , such as numbers ,string , Booleans , or dates etc.
* Dataframes are designed for handling structured , tabular data with multiple variables.
* Dataframes are versatile for operations like data selection , aggregation , merging , reshaping , and visualization on structured ,multi-variable data.

**Q.3 Difference between loc and iloc ?**

**Ans.** These are used in slicing data from the pandas dataframe .

**loc( )** : Selects subsets of rows and columns by label (column name) only.

* The loc( ) function is label based data selecting method which means that we have to pass the name of the row or column which we want to select.
* loc( ) can accept the Boolean data.
* Many operation can be performed using the loc( ).

Ex. import pandas as pd

df=pd.read\_csv("C:\\Users\\DIPALI\\Desktop\\Desktop\\Data Analyst class 2023\\covid\_toy.csv")

df.head()

df.loc[2:5,['age','city']]

Output :

Age city

2 42 Delhi

3 31 kolkata

4 65 Mumbai

5 84 Bengalore

**iloc( ) :** Selects subsets of rows and columns by integer location only.

* The iloc() function is an indexed-based selecting method which means that we have to pass an integer index in the method to select a specific row/column.
* This method does not includes the last element of the range passed in it .
* iloc( ) does not accept Boolean data.

Ex . df. Iloc[2:5,[0, 4]]

Output : Age city

2 42 Delhi

3 31 kolkata

4 65 Mumbai

**Q.4 What does value\_counts() ?**

**Ans.** Pandas value counts will count the frequency of the unique values in your series, or simply , count how many each value occurs.

**Syntax :** Series.value\_counts(normalize=False , sort=True , asending=False ,bins=None , dropn=True )

**Q.5 How do you handle missing values using pandas ?**

**Ans.** A dataframe , the null/missing values are replaced with NaN( not a number) in dataframe. To handle missing data , pandas uses the followingfunctions :

**Dropna( ) –** removes missing values( rows/ columns)

* Pandas dropna() method allows the user to analyze and drop rows/columns with Null/missing values.

Syntax : df.dropna( )

Ex. p= df.dropna( )

p

**Fillna( ) –** When the dataset is very large , handling NaN or Null values becomes very critical. Usually larger datasets hold more number of Nan values in different forms, so standardizing these Nan’s to a single value or to a value which is needed is a critical process while handling larger datasets.

* The fillna() function comes handy to perform such function.
* It verifies all the Nan values and replaces them with the assigned replacementvalue.

**Syntax -** Dataframe.fillna( )

**Q.6 How do you merge 2 tables using pandas ?**

**Ans.** There are two functions to merge the table : 1.merge( ) 2.concat( )

**1.merge( )**

**Ex.** import pandas as pd

var1=pd.DataFrame({"A":[1,2,3,4],"B":[11,12,13,14]})

var2=pd.DataFrame({"A":[1,2,3,4],"c":[21,22,23,24]})

pd.merge(var1,var2,on="A")

Output : A B C

0 1 11 21

1 2 12 22

2 3 13 23

3 4 14 24

**2. Concat( ) :** Thsese function is used to combine two or multiple dataframe or series in pandas

**Ex**. 1) import pandas as pd

s1=pd.Series([1,2,3,4])

s2=pd.Series([11,21,31,41])

pd.concat([s1,s2])

**Output :** 0 1

1 2

2 3

3 4

0 11

1. 21
2. 31

2) import pandas as pd

d1 =pd.DataFrame({"A":[1,2,3,4],"B":[11,12,13,14]})

d2=pd.DataFrame({"A":[1,2,3,4],"B":[21,22,23,24]})

pd.concat([d1,d2])

Output : A B

0 1 11

1 2 12

2 3 13

3 4 14

0 1 21

1 2 22

2 3 23

3 4 24

**Q.7 How do you connect jupyter notebook to mysql ?**

**Ans.**

**step 1=** import numpy as np

import pandas as pd

**step2**= !pip install mysql.connector

**step3**= import mysql.connector

**step4=** conn=mysql.connector.connect(host='localhost',user='root',password='',database='school’')-database name

**step5=** pd.read\_sql\_query("select \* from sudent",conn)-table name

**Q.8 What does group by in pandas ?**

**Ans.** Pandas Groupby function is used to split the data into groups based on some criteria. So we can perform analytics groupwise.

**Ex**. import pandas as pd

var=pd.DataFrame({"Name":['a','b','c','d','b','c','a','d','a','c'],

"s\_1":[24,24,21,58,63,54,57,78,59,85],

"s\_2":[58,45,78,86,95,56,52,42,63,51] })

var

var\_new=var.groupby("Name")

var\_new

for x,y in var\_new:

print(x)

print(y)

print()

Output : a **Name s\_1 s\_2**

0 a 24 58

6 a 57 52

8 a 59 63

b

**Name s\_1 s\_2**

1 b 24 45

4 b 63 95

c

**Name s\_1 s\_2**

2 c 21 78

5 c 54 56

9 c 85 51

d

**Name s\_1 s\_2**

3 d 58 86

7 d 78 42

var\_new.get\_group("a")

>> a **Name s\_1 s\_2**

0 a 24 58

6 a 57 52

8 a 59 63

**Q.9 Difference between CrossTab vs Pivot table ?**

**Ans.** Pivot tables and crosstabs are ways to display & analyze sets of data. Both are similar to each other, with pivot tables having just a few added features.

* Pivot tables and crosstabs present data in tabular format, with rows and columns displayig certain data.This data can be aggregated as a sum , count , max, min, or average if desired.
* These tool allow the user to easily recognize trends, see relationships between their data , and access information quickly and efficient.

**Pivot Table –**

* Pivot table allow the user to create additional reports on the sport by easily rearranging , adding , counting , and deleting certain data entries.
* Pivot tables work well with hierarchal organization where data sets can be drilled into to reveal more information.
* Pivot tables let the user filter through their data , add or remove custom fields, and change the appearance of their report.
* Pivot table can be used for both categorical and numerical data.
* Pivot table can analyze the relationship between multiple variable , both categorical and numerical.
* Pivot table allow you to aggregate data based on different criteria, such as sum ,count, average and more.
* Pivot table allow you to perform calculations such as sums,averages,and standard deviations.

**CrossTab –**

* Crosstabs, also known as crosstabulation tables, are a type of table used to summarize the relationship between two categorical variables.
* They display the frequency of observations that fall into each combination of categories for the two variable.
* Crosstabs can be used to indentify patterns or relationship between variables and to test hypotheses about those relationships.
* Crosstab are used for categorical data.
* Crosstab are used to analyze the relationship between two categorical variables.
* Cross tab display raw frequencies of observations.
* Crosstab do not perform calculations on the data.

**Q.10 In which scenerio what kind of plot you will use ? Numerical ==? , Categorical ==?**

**Ans.**

**Categorical ==** Both categorical data then used these plots ,

* Stacked bar chart
* Stacked column chart
* Clustered bar chart
* Clustered column chart
* 100% Stacked bar chart
* 100% Stacked column chart

**Numerical ==** Both numerical data then used these plots ,

* Waterfall chart
* Funnel
* Scatter chart
* Pie chart
* Donut chart
* Treemap

If one categorical data and one numerical data then used thses plots ,

* Line chart
* Area chart
* Stacked area chart
* Line and Stacked column chart
* Line and Clustered column chart
* Ribbon chart

**Q.11 What does TimeDelta in Pandas ?**

**Ans.** TimeDelta( ) function is also available in datetime library.

* Timedeltas are different in times, expressed in difference units, e.g. days, hours, minites, seconds.They can be both positive and negative.
* Pandas represents Timedelta in nanoseconds resolution using 64 bit integer.As such, the 64 bit integer limits determine the Timedelta limits.

Syntax: pandas.Timedelta(value=<object object>,unit =None, \*\*kwargs)

* Represents a duration , the difference between two dates or times.
* Timedelta is the pandas equivalent of pythons datetime.timedelta and is interchangeable with it in most cases.

Ex. td=pd.Timedelta(1,”d”)

td

**Output :** Timedelta(‘1 days 00:00:00)

**Q.12 How can you extract day , month , year in a date column using pandas ?**

**Ans.**

import pandas as pd

df=pd.read\_csv("C:\\Users\\DIPALI\\Desktop\\Desktop\\Data Analyst class 2023\\click.csv")

df.head()

df.columns

**>>** Index['Daily Time Spent on Site', 'Age', 'Area Income',

'Daily Internet Usage', 'Ad Topic Line', 'City', 'Gender', 'Country','Timestamp', 'Clicked on Ad']

**# Date and time**

df["Timestamp"]=pd.to\_datetime(df["Timestamp"])

df["Timestamp"]

>>0 2016-06-09 21:43:05

1 2016-01-16 17:56:05

2 2016-06-29 10:50:45

3 2016-06-21 14:32:32

4 2016-07-21 10:54:35

...

9995 2016-01-03 03:22:15

9996 2016-05-28 12:20:15

9997 2016-01-05 11:53:17

9998 2016-04-04 07:07:46

9999 2016-04-03 21:13:46

**#year**

df['Timestamp'].dt.year

>>0 2016

1 2016

2 2016

3 2016

4 2016

...

9995 2016

9996 2016

9997 2016

9998 2016

9999 2016

**#month**

df['Timestamp'].dt.month

>>0 6

1 1

2 6

3 6

4 7

..

9995 1

9996 5

9997 1

9998 4

9999 4

**#day**

df['Timestamp'].dt.day

>>0 9

1 16

2 29

3 21

4 21

..

9995 3

9996 28

9997 5

9998 4

9999 3

**# To add the sepretly year ,month, day columns in a dataframe :**

df['year']=df['Timestamp'].dt.year

df['month']=df['Timestamp'].dt.month

df['day']=df['Timestamp'].dt.day

df

**Q.13 How can you extract hours , minutes , seconds in a datetime column ?**

**Ans.**

import pandas as pd

df=pd.read\_csv("C:\\Users\\DIPALI\\Desktop\\Desktop\\Data Analyst class 2023\\click.csv")

df.head()

df['Timestamp']

>>0 2016-06-09 21:43:05

1 2016-01-16 17:56:05

2 2016-06-29 10:50:45

3 2016-06-21 14:32:32

4 2016-07-21 10:54:35

...

9995 2016-01-03 03:22:15

9996 2016-05-28 12:20:15

9997 2016-01-05 11:53:17

9998 2016-04-04 07:07:46

9999 2016-04-03 21:13:46

Name: Timestamp, Length: 10000, dtype: object

**# hour**

df['Timestamp'].dt.hour

>>0 21

1 17

2 10

3 14

4 10

..

9995 3

9996 12

9997 11

9998 7

9999 21

Name: Timestamp, Length: 10000, dtype: int64

**# minute**

df['Timestamp'].dt.minute

>>0 43

1 56

2 50

3 32

4 54

..

9995 22

9996 20

9997 53

9998 7

9999 13

Name: Timestamp, Length: 10000, dtype: int64

**# second**

df['Timestamp'].dt.second

>>0 5

1 5

2 45

3 32

4 35

..

9995 15

9996 15

9997 17

9998 46

9999 46

Name: Timestamp, Length: 10000, dtype: int64

**Q.14 How do you read a excel file , csv file and json file in jupyter notebook using pandas ?**

**Ans.**

**1.Read CSV file using pandas :**

import pandas as pd

df=pd.read\_csv("C:\\Users\\DIPALI\\Desktop\\Desktop\\Data Analyst class 2023\\Attrition.csv")

df.head()

**2.Read EXCEL file using pandas :**

import pandas as pd

df=pd.read\_excel("C:\\Users\\DIPALI\\Desktop\\Desktop\\Data Analyst class 2023\\click.xlsx")

df

**3.Read JSON file using pandas :**

import pandas as pd

df=pd.read\_json("C:\\Users\\DIPALI\\Desktop\\Desktop\\Data Analyst class 2023\\ABC.json", lines=True)

df.head()

**Q.15 How can you create excel data from a dataframe using pandas ?**

**Ans.**

import pandas as pd

my\_dict={'Name':['dipa','sham','pooja'],

'ID':[11,22,33],

'Marks':[54,85,65]}

df=pd.DataFrame(data=my\_dict)

#df

df.to\_excel("C:\\Users\\DIPALI\\Desktop\\Desktop\\Data Analyst class 2023\\my\_file.xlsx")

df

Output :

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Name** | **ID** | **Marks** |
| **0** | dipa | 11 | 54 |
| **1** | sham | 22 | 85 |
| **2** | pooja | 33 | 65 |