dataframe

July 27, 2024

1 Pandas DataFrame

[1]:
```
import pandas as pd
import numpy as np
```

1.1 Creating DataFrame

[2]:
```
# Data
a= {"Name":[chr(i) for i in range(97,107)],
    "Age":[i for  i in range(20,30)],
    "City":[chr(i) for i in range(65,75)]
}
```

[3]:
```
a
```

[3]:
```
{'Name': ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'],
 'Age': [20, 21, 22, 23, 24, 25, 26, 27, 28, 29],
 'City': ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']}
```

[4]:
```
#Creating Data Frame
df=pd.DataFrame(a,index=[i for i in range(100,110)])      # By using Dictionary␣
 ↪keys are assigned as Column names
df
```

[4]:

	Name	Age	City
100	a	20	A
101	b	21	B
102	c	22	C
103	d	23	D
104	e	24	E
105	f	25	F
106	g	26	G
107	h	27	H
108	i	28	I
109	j	29	J

[5]:
```
# If DF is created by this column names are given using its parameters
pd.DataFrame(np.array([[1,2,3],[4,5,6]]),columns=['a','b','c'])
```

[5]:

	a	b	c
0	1	2	3
1	4	5	6

1.2 Viewing / Inpecting Data

[6]:
```
df.head(5)  # Top 5
```

[6]:

	Name	Age	City
100	a	20	A
101	b	21	B
102	c	22	C
103	d	23	D
104	e	24	E

[7]:
```
df.tail(5)  # bottom 5
```

[7]:

	Name	Age	City
105	f	25	F
106	g	26	G
107	h	27	H
108	i	28	I
109	j	29	J

[8]:
```
df.sample(5)    # Random 5
```

[8]:

	Name	Age	City
106	g	26	G
101	b	21	B
107	h	27	H
102	c	22	C
104	e	24	E

[9]:
```
df.info()   # Provides consise summary for the dataframe
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10 entries, 100 to 109
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Name    10 non-null     object
 1   Age     10 non-null     int64
 2   City    10 non-null     object
```

```
dtypes: int64(1), object(2)
memory usage: 320.0+ bytes
```

[10]:
```
df.describe()       # Generate discriptive statistics
```

[10]:
```
             Age
count  10.00000
mean   24.50000
std     3.02765
min    20.00000
25%    22.25000
50%    24.50000
75%    26.75000
max    29.00000
```

[11]:
```
print("Shape \t:",df.shape)          # Shape of the DF
print("Columns\t:",df.columns)      # Columns of the Df
print("Index \t:",df.index)          # Index of the DF
```

```
Shape   : (10, 3)
Columns : Index(['Name', 'Age', 'City'], dtype='object')
Index   : Index([100, 101, 102, 103, 104, 105, 106, 107, 108, 109],
dtype='int64')
```

1.3 Selection / Filetering

[12]:
```
# df[col]
df['Name']                        # Single Selection
```

[12]:
```
100    a
101    b
102    c
103    d
104    e
105    f
106    g
107    h
108    i
109    j
Name: Name, dtype: object
```

[13]:
```
df[['Name','City']]               # Multiple Selection
```

[13]:
```
    Name City
100    a    A
101    b    B
102    c    C
```

```
103    d    D
104    e    E
105    f    F
106    g    G
107    h    H
108    i    I
109    j    J
```

[14]:
```
# Iloc
df.iloc[1:5]        # Inter Location based indexing
```

[14]:
```
    Name  Age City
101    b   21    B
102    c   22    C
103    d   23    D
104    e   24    E
```

[15]:
```
df.iloc[1:5,1:3]        # Row , columns
```

[15]:
```
     Age City
101   21    B
102   22    C
103   23    D
104   24    E
```

[16]:
```
df.loc[102:105]     # Label Based indexing
```

[16]:
```
    Name  Age City
102    c   22    C
103    d   23    D
104    e   24    E
105    f   25    F
```

[17]:
```
df.loc[102:105,'Name':"City"]       # Label Based Indexing
```

[17]:
```
    Name  Age City
102    c   22    C
103    d   23    D
104    e   24    E
105    f   25    F
```

[18]:
```
df.query('Age>25')          # Returns Values are true in the Expression
```

[18]:
```
    Name  Age City
106    g   26    G
107    h   27    H
108    i   28    I
```

```
109    j   29    J
```

[19]:
```
df.filter(items=['Name','Age'])
```

[19]:
```
     Name  Age
100    a   20
101    b   21
102    c   22
103    d   23
104    e   24
105    f   25
106    g   26
107    h   27
108    i   28
109    j   29
```

[20]:
```
dfsub=pd.DataFrame({"Name":['a','b','c'],"Age":[20,21,22]})
dfsub
```

[20]:
```
  Name  Age
0    a   20
1    b   21
2    c   22
```

[21]:
```
df.reset_index().isin(dfsub.reset_index())      # Checks the dfsub values is in␣
↪df
```

[21]:
```
   index   Name    Age   City
0  False   True   True  False
1  False   True   True  False
2  False   True   True  False
3  False  False  False  False
4  False  False  False  False
5  False  False  False  False
6  False  False  False  False
7  False  False  False  False
8  False  False  False  False
9  False  False  False  False
```

[22]:
```
df.Age.where(df.Age >24,20)      # replaces the Age column where the condition␣
↪is false
```

[22]:
```
100    20
101    20
102    20
103    20
104    20
```

```
105    25
106    26
107    27
108    28
109    29
Name: Age, dtype: int64
```

[23]:
```
df.Age.mask(df.Age>24,30)   # Replaces where the condition is true
```

[23]:
```
100    20
101    21
102    22
103    23
104    24
105    30
106    30
107    30
108    30
109    30
Name: Age, dtype: int64
```

1.4 Data Cleaning / Manipulation

[24]:
```
df1= pd.DataFrame({'Name': ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i',␣
 ↪'a','c'],
 'Age': [20, 21, 22, 23, None, None, 26, 27, None, 20,22],
 'City': ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'A','C']})
df1
```

[24]:

	Name	Age	City
0	a	20.0	A
1	b	21.0	B
2	c	22.0	C
3	d	23.0	D
4	e	NaN	E
5	f	NaN	F
6	g	26.0	G
7	h	27.0	H
8	i	NaN	I
9	a	20.0	A
10	c	22.0	C

[25]:
```
# Drop lables
df1.drop([2,3])     # Drop Specified rows or columns
```

[25]:
```
   Name   Age City
0     a  20.0    A
1     b  21.0    B
4     e   NaN    E
5     f   NaN    F
6     g  26.0    G
7     h  27.0    H
8     i   NaN    I
9     a  20.0    A
10    c  22.0    C
```

[26]:
```
# Drop Duplicates
df1.drop_duplicates()       # To make it permanent we have to give inplace=
 ↪True in Parameter
```

[26]:
```
   Name   Age City
0     a  20.0    A
1     b  21.0    B
2     c  22.0    C
3     d  23.0    D
4     e   NaN    E
5     f   NaN    F
6     g  26.0    G
7     h  27.0    H
8     i   NaN    I
```

[27]:
```
# Fill Null Values
df1.fillna(df1.Age.mean())  # Can be filled with mean , median etc
```

[27]:
```
   Name     Age City
0     a  20.000    A
1     b  21.000    B
2     c  22.000    C
3     d  23.000    D
4     e  22.625    E
5     f  22.625    F
6     g  26.000    G
7     h  27.000    H
8     i  22.625    I
9     a  20.000    A
10    c  22.000    C
```

[28]:
```
# Replace
df1.replace('A','Changes occur Here')
```

[28]:
```
   Name   Age                 City
0     a  20.0  Changes occur Here
```

```
1     b  21.0                   B
2     c  22.0                   C
3     d  23.0                   D
4     e   NaN                   E
5     f   NaN                   F
6     g  26.0                   G
7     h  27.0                   H
8     i   NaN                   I
9     a  20.0  Changes occur Here
10    c  22.0                   C
```

[29]:
```
#Rename
df1.rename(index={0:"x",1:"y",2:"z"}, columns={"Name":"Lowercase","Age":
 ↪"Numbers","City":"Upper"})
```

[29]:
```
   Lowercase  Numbers Upper
x          a     20.0     A
y          b     21.0     B
z          c     22.0     C
3          d     23.0     D
4          e      NaN     E
5          f      NaN     F
6          g     26.0     G
7          h     27.0     H
8          i      NaN     I
9          a     20.0     A
10         c     22.0     C
```

[30]:
```
# Sort_values
data = {
    'City': ['New York', 'Los Angeles', 'Chicago', 'New York', 'Chicago',
             'Los Angeles', 'New York', 'Chicago', 'Los Angeles', 'New York'],
    'Product': ['Apple', 'Orange', 'Apple', 'Banana', 'Orange',
                'Banana', 'Apple', 'Banana', 'Orange', 'Apple'],
    'Sales': [100, 150, 200, 120, 180, 90, 300, 130, 160, 250]
}
df2=pd.DataFrame(data)
df2
```

[30]:
```
           City Product  Sales
0      New York   Apple    100
1  Los Angeles  Orange    150
2       Chicago   Apple    200
3      New York  Banana    120
4       Chicago  Orange    180
5  Los Angeles  Banana     90
6      New York   Apple    300
```

```
7      Chicago  Banana    130
8  Los Angeles  Orange    160
9     New York   Apple    250
```

[31]:
```
df2.sort_values(by="Sales",ascending=False)
```

[31]:
```
           City Product  Sales
6     New York   Apple    300
9     New York   Apple    250
2      Chicago   Apple    200
4      Chicago  Orange    180
8  Los Angeles  Orange    160
1  Los Angeles  Orange    150
7      Chicago  Banana    130
3     New York  Banana    120
0     New York   Apple    100
5  Los Angeles  Banana     90
```

[32]:
```
df2.groupby(by='City')['Sales'].aggregate({'mean','median','min','max'})
```

[32]:
```
             min  median        mean  max
City
Chicago      130   180.0  170.000000  200
Los Angeles   90   150.0  133.333333  160
New York     100   185.0  192.500000  300
```

[33]:
```
df2.groupby(by='City')['Sales'].aggregate({'mean','median','min','max'}).
↪reset_index()
```

[33]:
```
           City  min  median        mean  max
0      Chicago  130   180.0  170.000000  200
1  Los Angeles   90   150.0  133.333333  160
2     New York  100   185.0  192.500000  300
```

[34]:
```
c=df2.pivot_table(index='City',columns='Product',values='Sales',aggfunc='sum')
c
```

[34]:
```
Product      Apple  Banana  Orange
City
Chicago      200.0   130.0   180.0
Los Angeles    NaN    90.0   310.0
New York     650.0   120.0     NaN
```

[35]:
```
df2.melt(id_vars=['City'],value_vars=['Product'])
```

[35]:
```
           City variable   value
0     New York  Product   Apple
```

```
1  Los Angeles  Product  Orange
2      Chicago  Product   Apple
3     New York  Product  Banana
4      Chicago  Product  Orange
5  Los Angeles  Product  Banana
6     New York  Product   Apple
7      Chicago  Product  Banana
8  Los Angeles  Product  Orange
9     New York  Product   Apple
```

1.5 Arithmetic and Statistical Operations

[36]:
```
print(df2.Sales.mean())        # Aggregate function
print(df2.Sales.median())
print(df2.Sales.sum())
print(df2.Sales.min())
print(df2.Sales.max())
```

```
168.0
155.0
1680
90
300
```

[37]:
```
# Statistical Funtions
print(df2.Sales.std())
print(df2.Sales.var())
```

```
66.79986693266854
4462.222222222223
```

[38]:
```
df3=pd.DataFrame(data = {
    'X': [2.5, 3.1, 4.0, 5.7, 6.8, 8.2, 7.1, 9.6, 10.5, 12.2,
          13.3, 14.8, 15.0, 16.4, 17.2, 18.5, 19.1, 20.7, 21.5, 22.3],
    'Y': [3.0, 4.2, 6.5, 5.8, 12.0, 11.5, 14.0, 19.1, 18.2, 20.5,
          22.0, 24.5, 29.0, 30.5, 27.2, 34.0, 35.5, 36.2, 39.0, 42.5]
})
```

[39]:
```
print(df3.corr(method='spearman'))
print(df3.cov())
```

```
          X         Y
X  1.00000  0.98797
Y  0.98797  1.00000
            X           Y
X  40.668289   77.524211
Y  77.524211  151.074105
```

[40]:
```
df3.diff().head(7) #First discrete difference of element.
```

[40]:

	X	Y
0	NaN	NaN
1	0.6	1.2
2	0.9	2.3
3	1.7	-0.7
4	1.1	6.2
5	1.4	-0.5
6	-1.1	2.5

[41]:
```
df3.pct_change().head(7) # Fractional change between the current and a prior␣
↪element.
```

[41]:

	X	Y
0	NaN	NaN
1	0.240000	0.400000
2	0.290323	0.547619
3	0.425000	-0.107692
4	0.192982	1.068966
5	0.205882	-0.041667
6	-0.134146	0.217391

1.6 Combining DataFrame

[42]:
```
pd.concat((df2,df2),axis=1)
```

[42]:

	City	Product	Sales	City	Product	Sales
0	New York	Apple	100	New York	Apple	100
1	Los Angeles	Orange	150	Los Angeles	Orange	150
2	Chicago	Apple	200	Chicago	Apple	200
3	New York	Banana	120	New York	Banana	120
4	Chicago	Orange	180	Chicago	Orange	180
5	Los Angeles	Banana	90	Los Angeles	Banana	90
6	New York	Apple	300	New York	Apple	300
7	Chicago	Banana	130	Chicago	Banana	130
8	Los Angeles	Orange	160	Los Angeles	Orange	160
9	New York	Apple	250	New York	Apple	250

[43]:
```
j1=pd.DataFrame(data = {
    'EmployeeID': [1, 2, 3, 4, 5],
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva'],
    'DepartmentID': [101, 102, 101, 103, 102]
})
j2=pd.DataFrame(data = {
    'DepartmentID': [101, 102, 103, 104],
    'DepartmentName': ['HR', 'IT', 'Finance', 'Marketing']
```

```
})
```

[44]:
```
# Simple Merge
pd.merge(j1,j2,how='inner',on='DepartmentID')
```

[44]:

	EmployeeID	Name	DepartmentID	DepartmentName
0	1	Alice	101	HR
1	2	Bob	102	IT
2	3	Charlie	101	HR
3	4	David	103	Finance
4	5	Eva	102	IT

[45]:
```
j1.set_index('DepartmentID',inplace=True)
j2.set_index('DepartmentID',inplace=True)
```

[46]:
```
# Simple Join
j1.join(j2)
```

[46]:

DepartmentID	EmployeeID	Name	DepartmentName
101	1	Alice	HR
102	2	Bob	IT
101	3	Charlie	HR
103	4	David	Finance
102	5	Eva	IT

1.7 Reshaping Data

[47]:
```
r=pd.DataFrame(data = {
    'Date': ['2024-01-01', '2024-01-02', '2024-01-03', '2024-01-04'],
    'Sales': [200, 220, 250, 275],
    'Expenses': [150, 160, 170, 180]
})
r.set_index('Date',inplace=True)
r
```

[47]:

Date	Sales	Expenses
2024-01-01	200	150
2024-01-02	220	160
2024-01-03	250	170
2024-01-04	275	180

[48]:
```
# Basic Stacking
r.stack(level=0)
```

[48]:
```
Date
2024-01-01  Sales       200
            Expenses    150
2024-01-02  Sales       220
            Expenses    160
2024-01-03  Sales       250
            Expenses    170
2024-01-04  Sales       275
            Expenses    180
dtype: int64
```

[49]:
```
r=pd.DataFrame({
    'Product': ['A', 'A', 'B', 'B'],
    'Month': ['Jan', 'Feb', 'Jan', 'Feb'],
    'Sales': [100, 150, 200, 250]
})
r.set_index(['Product', 'Month'], inplace=True)
r
```

[49]:
```
               Sales
Product Month
A       Jan      100
        Feb      150
B       Jan      200
        Feb      250
```

[50]:
```
# Multi Index Stack
r.stack()
```

[50]:
```
Product  Month
A        Jan    Sales    100
         Feb    Sales    150
B        Jan    Sales    200
         Feb    Sales    250
dtype: int64
```

[51]:
```
# Transpose
df1.transpose()
```

[51]:

	0	1	2	3	4	5	6	7	8	9	10
Name	a	b	c	d	e	f	g	h	i	a	c
Age	20.0	21.0	22.0	23.0	NaN	NaN	26.0	27.0	NaN	20.0	22.0
City	A	B	C	D	E	F	G	H	I	A	C

[52]:
```
df1.T       # .T  also does the same
```

[52]:

	0	1	2	3	4	5	6	7	8	9	10
Name	a	b	c	d	e	f	g	h	i	a	c
Age	20.0	21.0	22.0	23.0	NaN	NaN	26.0	27.0	NaN	20.0	22.0
City	A	B	C	D	E	F	G	H	I	A	C

[53]:
```
# Resets the Index
r.reset_index(inplace=True)
r
```

[53]:

	Product	Month	Sales
0	A	Jan	100
1	A	Feb	150
2	B	Jan	200
3	B	Feb	250

[54]:
```
# Set Index
r.set_index('Product')
```

[54]:

Product	Month	Sales
A	Jan	100
A	Feb	150
B	Jan	200
B	Feb	250

I hope you found this information helpful! Feel free to save this post for future reference. Let's continue to learn and grow together!

Rajendra Prasad