1. Array Computations using NumPy

AIM: Perform arithmetic operations using array.

```
Import numpy as np

a=np.arange(1,6)

b=np.arange(6,11)

print('Multiplication of a & b is',np.multiply(a,b))

print('Subtraction of a & b is',np.subtract(a,b))

print('Addition of a & b is',np.add(a,b))
```

output

```
Multiplication of a & b is [ 6 14 24 36 50]
Subtraction of a & b [-5 -5 -5 -5 -5]
Addition of a & b [ 7  9 11 13 15]
```

AIM: Perform slicing and indexing on multi-dimensional arrays.

```
import numpy as np
x=np.arange(1,9).reshape(2,2,2)
print(x)
print(x[0,0,0])
print(x[1,0:3,0])
```

output

```
[[[1 2]
  [3 4]]

 [[5 6]
  [7 8]]]
1
[5 7]
```

AIM: Perform computations on multi-dimensional array using universal functions (ufunc).

```
import numpy as np

array1=np.arange(1,19).reshape(2,3,3)

print('Apply floor division',np.floor(array1))

print('Apply sign function',np.sign(array1))

print('Apply rint',np.rint(array1))
```

output:

```
Apply floor division [[[ 1.  2.  3.]
  [ 4.  5.  6.]
  [ 7.  8.  9.]]

 [[10. 11. 12.]
  [13. 14. 15.]
  [16. 17. 18.]]]
Apply sign function [[[1 1 1]
  [1 1 1]
  [1 1 1]]

 [[1 1 1]
  [1 1 1]
  [1 1 1]]]
Apply rint [[[ 1.  2.  3.]
  [ 4.  5.  6.]
  [ 7.  8.  9.]]

 [[10. 11. 12.]
  [13. 14. 15.]
  [16. 17. 18.]]]
```

AIM: Compute arithmetic mean, standard deviation, variance, percentile, minimum and maximum, cumulative sum and product using statistical functions in NumPy.

```
import numpy as np
y=np.arange(11,20).reshape(3,3)
print("Arithmetic Mean is",np.mean(y))
print("Standard Deviation is ",np.std(y))
print("Variance is ",np.var(y))
print("Percentile is ",np.percentile(y,50))
print("Minimum number is ",np.min(y))
print("Maximum number is ",np.max(y))
print("Cumulative Sum is ",np.cumsum(y))
print("Cumulative Product is",np.cumprod(y))
```

output:

```
Arithmetic Mean is 15.0
Standard Deviation is  2.581988897471611
Variance is  6.666666666666667
Percentile is  15.0
Minimum number is  11
Maximum number is  19
Cumulative Sum is  [ 11  23  36  50  65  81  98 116 135]
Cumulative Product is [        11        132       1716      24024     360360
5765760  98017920   1764322560 -837609728]
```

AIM: Perform set theory operations such as union, intersection, symmetric difference and fetching unique values.

```
import numpy as np
S1={2,6,-5,8,0,10}
S2={1,5,-6,7,11,12}
print("Perform set theory operations")
print("Union of S1,S2 ",S1|S2)
print("Intersection of S1,S2 ",S1&S2)
print("Symmetric of S1,S2 ",S1^S2)
print("Unique Values of S1,S2 are",np.unique(S1,S2))
```

OUTPUT:

```
Perform set theory operations
Union of S1,S2  {0, 1, 2, 5, 6, 7, 8, 10, 11, 12, -6, -5}
Intersection of S1,S2  set()
Symmetric of S1,S2  {0, 1, 2, 5, 6, 7, 8, 10, 11, 12, -6, -5}
Unique Values of S1,S2 are (array([{0, 2, 6, 8, 10, -5}], dtype=object),
array([0], dtype=int64))
```

2. Linear Algebra and Random Number generation using linalg and random module in NumPy

AIM : Compute dot product, vector product and inner product of two arrays.

```
import numpy as np
arr1=np.arange(11,15).reshape(2,2)
arr2=np.arange(16,20).reshape(2,2)
print("Dot Product of arr1,arr2 is",np.dot(arr1,arr2))
print("Vector Product of arr1,arr2 is",np.cross(arr1,arr2))
print("Inner Product of arr1,arr2 is",np.inner(arr1,arr2))
```

output:

```
Dot Product of arr1,arr2 is [[392 415]
 [460 487]]
Vector Product of arr1,arr2 is [-5 -5]
Inner Product of arr1,arr2 is [[380 426]
 [446 500]]
```

AIM:. Perform matrix operations such as multiplication, determinant, sum of diagonal elements and inverse.

```
import numpy as np
matrix1=np.array([5,7,9,2,5,7,12,46,0]).reshape(3,3)
matrix2=np.array([1,3,5,7,3,2,8,5,1]).reshape(3,3)
print("Multiplication of Matrix1,Matrix2  is ",np.dot(matrix1,matrix2))
print("Determinant of Matrix1 is",np.linalg.det(matrix1))
print("Sum of Diagonal Elemnts of Matrix1 is",np.diagonal(matrix1))
print("Inverse of Matrix1 is",np.linalg.inv(matrix1))
```

output:

```
Multiplication of Matrix1,Matrix2  is  [[126  81  48]
 [ 93  56  27]
 [334 174 152]]
Determinant of Matrix1 is -733.9999999999997
Sum of Diagonal Elemnts of Matrix1 is [5 5 0]
Inverse of Matrix1 is [[ 0.4386921  -0.5640327  -0.00544959]
 [-0.11444142  0.14713896  0.02316076]
 [-0.04359673  0.19891008 -0.01498638]]
```

AIM: Compute eigenvalues, eigenvectors and singular value decomposition for a square matrix.

```
import numpy as np
matrix=np.array([12,25,36,4,55,0,12,14,9]).reshape(3,3)
print("Eigen Values and Eigen Vectors of Matrix is ")
print(np.linalg.eig(matrix))
```

output:

```
Eigen Values and Eigen Vectors of Matrix is
(array([-10.30731481,  27.60883566,  58.69847915]), array([[ 0.864898  , -
0.87452555, -0.63654958],[-0.05297404,  0.12770915, -0.68844468],
 [-0.49914447, -0.46786262, -0.34763278]]))
```

AIM: Generate random samples from uniform, normal, binomial, chi-square and Gaussian distributions using numpy. random functions.

```
import numpy as np
import random
z=np.array([1,2,3,4])
print(np.random.chisquare(z))
print(np.random.uniform(z))
print(np.random.normal(z))
print(np.random.standard_gamma(z))
```

output:

```
[ 1.62902114  4.31964105 10.24425652  4.42377299]
[1.         1.58726154 1.14344986 1.25718037]
[2.27260057 2.23203252 2.33503519 3.86542733]
[0.38112322 3.38247612 1.25329174 2.90818472]
```

AIM: Implement a single random walk with 1000 steps using random module and extract the statistics like minimum and maximum value along the walk‘s trajectory.

```
import numpy as np
import random
p=0
walk=[]
for i in range (10):
    step=1 if random.randint(0,1) else -1
    p+=step
walk.append(p)
w=np.array(walk)
print(w)
```

output:

```
[ 1  0 -1  0  1  0 -1  0 -1 -2]
```

3. Data Manipulation using pandas

AIM: Create DataFrame from List, Dict, List of Dicts, Dicts of Series and perform operations such as column selection, addition, deletion and row selection, addition and deletion.

```
import pandas as pd
import numpy as np
l=pd.Series([1,2,3,4])
print(l)
x=pd.DataFrame([1,2,3,4,5],[11,13,43,15,37])
print(x)
data={'Name':['Eswar','Pavan','Sai'],'Age':[19,19,18]}
d1=pd.DataFrame(data)
print(d1)
d2=pd.Series(['Eswar','Pavan','Sai','Kumar'])
d3=pd.Series([19,19,18,22])
d4=pd.Series(['BVRM','BVRM','VIZAG','KKD'])
d5={'Name':d2,'Age':d3,'City':d4}
d6=pd.DataFrame(d5)
print(d6)
print(d6.Name)
d6['Age']=[20,19,18,21]
print(d6)
```

<u>output:</u>

```
0    1
1    2
2    3
3    4
dtype: int64
    0
11  1
13  2
43  3
15  4
37  5
 Name  Age
0  Eswar   19
1  Pavan   19
2    Sai   18
```

```
    Name  Age   City
0  Eswar   19   BVRM
1  Pavan   19   BVRM
2    Sai   18  VIZAG
3  Kumar   22    KKD

0    Eswar
1    Pavan
2      Sai
3    Kumar

Name: Name, dtype: object

    Name  Age   City
0  Eswar   20   BVRM
1  Pavan   19   BVRM
2    Sai   18  VIZAG
3  Kumar   21    KKD
```

AIM: Create a Data Frame and perform descriptive statistics functions such as sum, mean, median, mode, standard deviation, skewness, kurtosis, cumulative sum, cumulative product and percent changes.

```
import pandas as pd
import numpy as np
d=[12,13,14,15]
data=pd.DataFrame({'Data':np.array(d)})
print(data)
print("Mean",data.mean())
print("Median",data.median())
print("Mode",data.mode())
print(data.skew())
print(data.kurtosis())
print(np.cumsum(d))
print(np.cumprod(d))
print(data.pct_change())
```

output:

```
Data
0    12
1    13
2    14
3    15

Mean Data    13.5
dtype: float64

Median Data    13.5
dtype: float64

Mode   Data

0    12
1    13
2    14
3    15
Data    0.0
dtype: float64
Data   -1.2

dtype: float64
```

AIM: Implement the computation of correlation and covariance by considering the Data Frames of stock prices and volumes obtained from Yahoo Finance! Using pandas-data reader package.

```
import pandas as pd
import numpy as np
import Pandas_datareader as pdr
data=pdr.DataReader(name='TSLA',data_source="yahoo")
x=data['High']
p=data['Volume']
cov=np.cov(x,p)
corr=y.corr(p)
print("Covariance of High,Volume is ")
print(cov)
print("coorelation of volume is ")
print(corr)
```

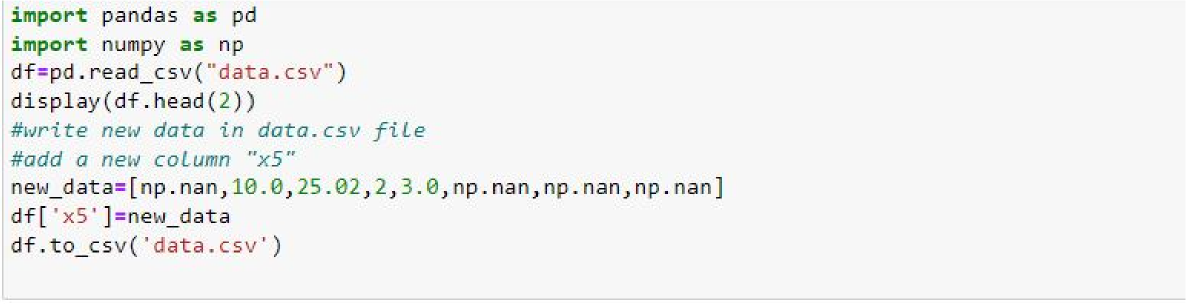
Output:

```
Covariance of High,Volume is
[[ 1.48808936e+04 -3.68193388e+09]
 [-3.68193388e+09  8.16362634e+15]]
coorelation of volume is
-0.3340564954021143
```

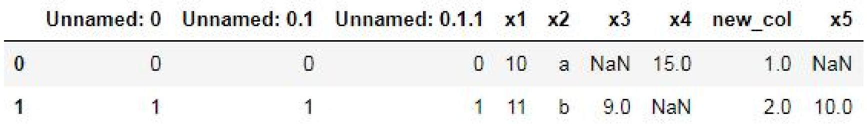
4. Working with different data formats using pandas

AIM: Perform reading and writing data in text format using read_csv and read_table considering any online dataset in delimited format (CSV).

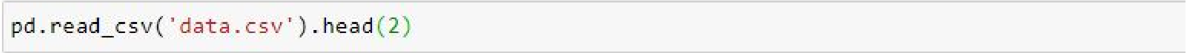
<u>Code:</u> reading and writing data in text format using read_csv



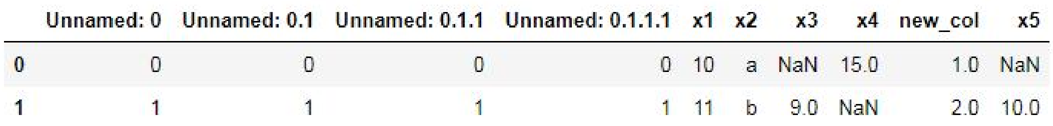
```
import pandas as pd
import numpy as np
df=pd.read_csv("data.csv")
display(df.head(2))
#write new data in data.csv file
#add a new column "x5"
new_data=[np.nan,10.0,25.02,2,3.0,np.nan,np.nan,np.nan]
df['x5']=new_data
df.to_csv('data.csv')
```



	Unnamed: 0	Unnamed: 0.1	Unnamed: 0.1.1	x1	x2	x3	x4	new_col	x5
0	0	0	0	10	a	NaN	15.0	1.0	NaN
1	1	1	1	11	b	9.0	NaN	2.0	10.0



```
pd.read_csv('data.csv').head(2)
```

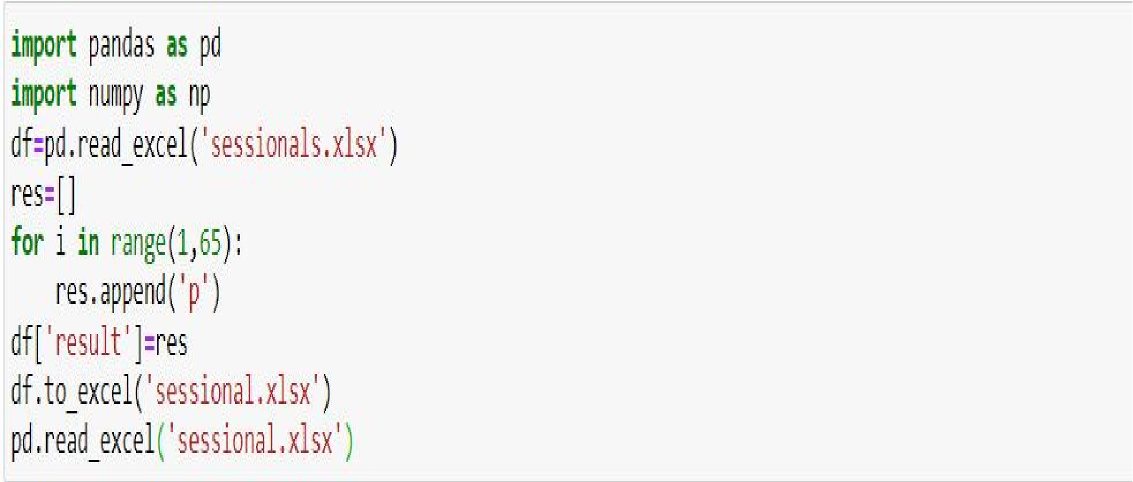


	Unnamed: 0	Unnamed: 0.1	Unnamed: 0.1.1	Unnamed: 0.1.1.1	x1	x2	x3	x4	new_col	x5
0	0	0	0	0	10	a	NaN	15.0	1.0	NaN
1	1	1	1	1	11	b	9.0	NaN	2.0	10.0

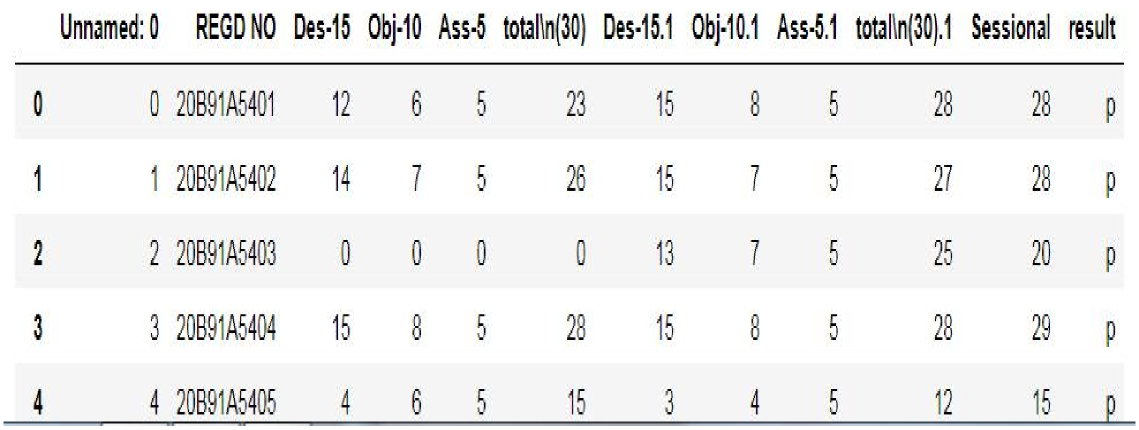
#Accessing data from online data set

```
import pandas as pd
import numpy as np

url="https://www.stats.govt.nz/assets/Uploads/Annual-enterprise-survey/Annual-enterprise-survey-2021-financial-year-provisional/Download-data/annual-enterprise-survey-2021-financial-year-provisional-csv.csv"

data=pd.read_csv(url)
data.head()
```

AIM: Perform reading and writing of Microsoft Excel Files (xslx) using read_excel.



```
import pandas as pd
import numpy as np
df=pd.read_excel('sessionals.xlsx')
res=[]
for i in range(1,65):
    res.append('p')
df['result']=res
df.to_excel('sessional.xlsx')
pd.read_excel('sessional.xlsx')
```



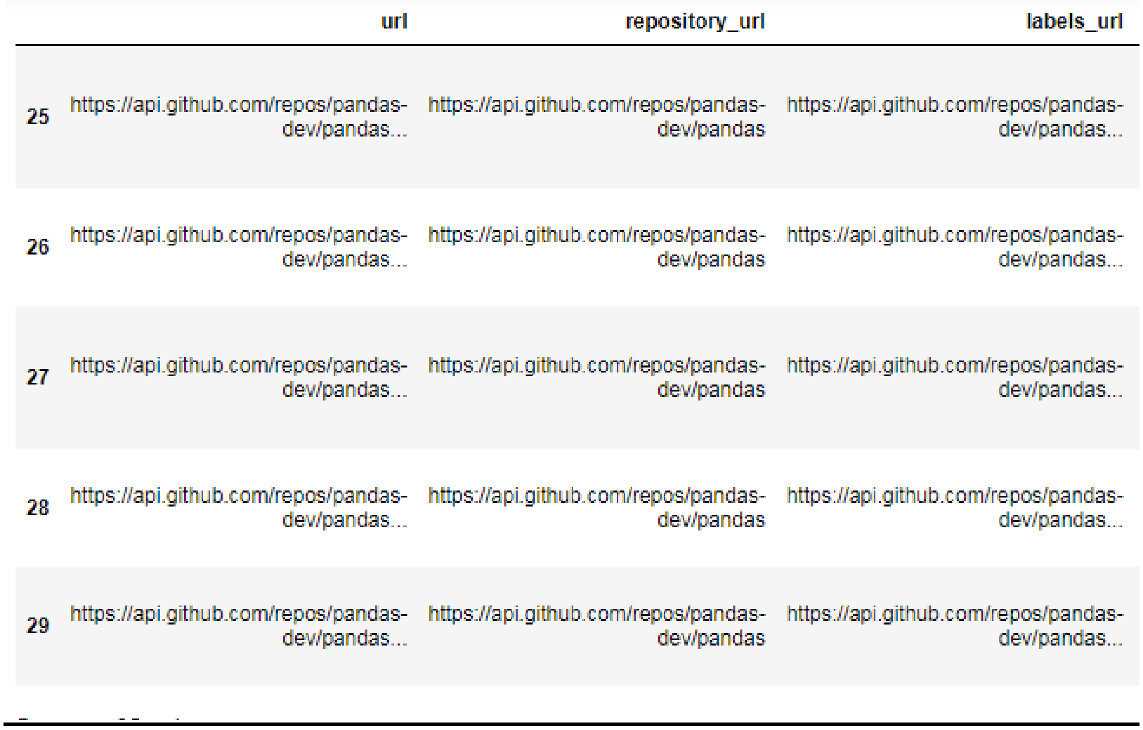
	Unnamed: 0	REGD NO	Des-15	Obj-10	Ass-5	total\n(30)	Des-15.1	Obj-10.1	Ass-5.1	total\n(30).1	Sessional	result
0	0	20B91A5401	12	6	5	23	15	8	5	28	28	p
1	1	20B91A5402	14	7	5	26	15	7	5	27	28	p
2	2	20B91A5403	0	0	0	0	13	7	5	25	20	p
3	3	20B91A5404	15	8	5	28	15	8	5	28	29	p
4	4	20B91A5405	4	6	5	15	3	4	5	12	15	p

5. Interacting with Web APIs and Databases

AIM: Predict the last 30 GitHub issues for pandas using request and response object‘s json method. Move the extracted data to DataFrame and extract fields of interest. (Use url: 'https://api.github.com/repos/pandas-dev/pandas/issues')

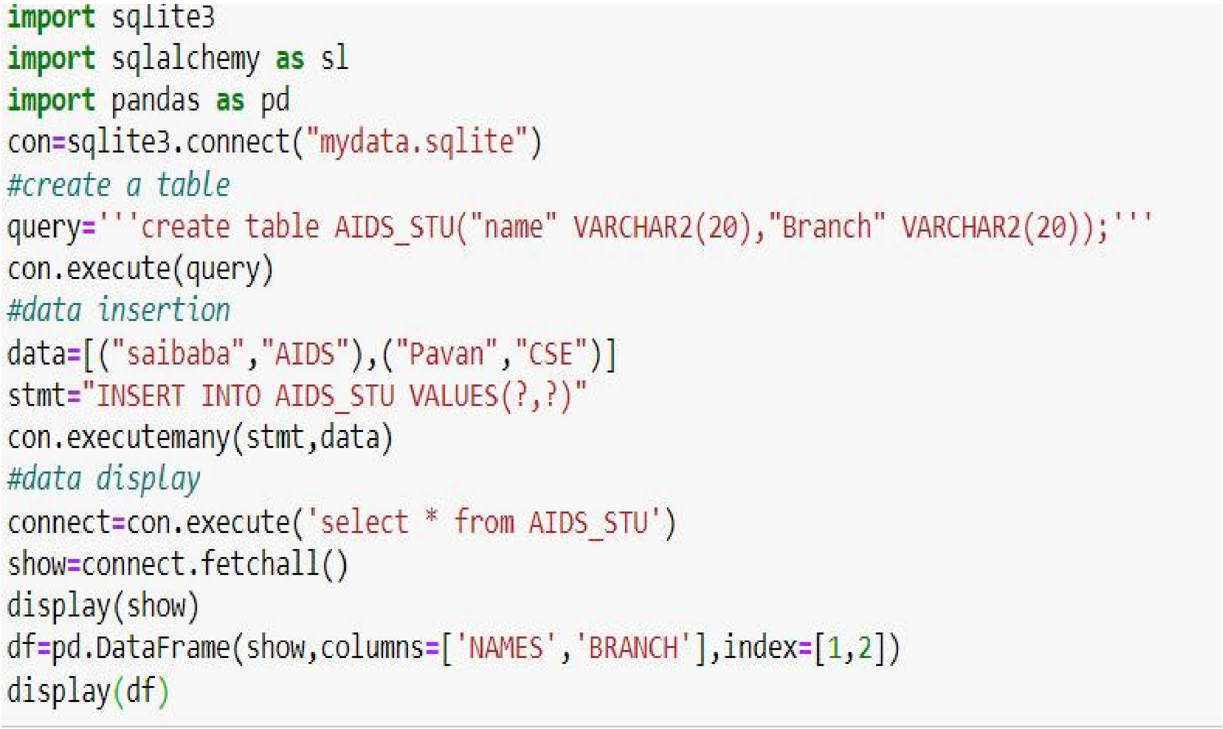
```
import requests as rq
import pandas as pd
d=rq.get('https://api.github.com/repos/pandas-dev/pandas/issues')
data=d.json()
data1=pd.DataFrame(data)
print("The last 30 Github issues")
data1.tail()
```

output:

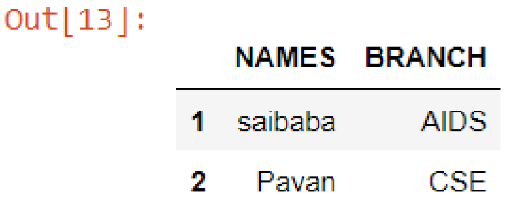


	url	repository_url	labels_url
25	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pandas-dev/pandas...
26	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pandas-dev/pandas...
27	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pandas-dev/pandas...
28	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pandas-dev/pandas...
29	https://api.github.com/repos/pandas-dev/pandas...	https://api.github.com/repos/pandas-dev/pandas	https://api.github.com/repos/pandas-dev/pandas...

AIM: Connect to any relational database using corresponding SQL drivers and perform operations such as table creation, populating the table, selecting data from table, moving data from table to DataFrame, updating records and deleting records in a table



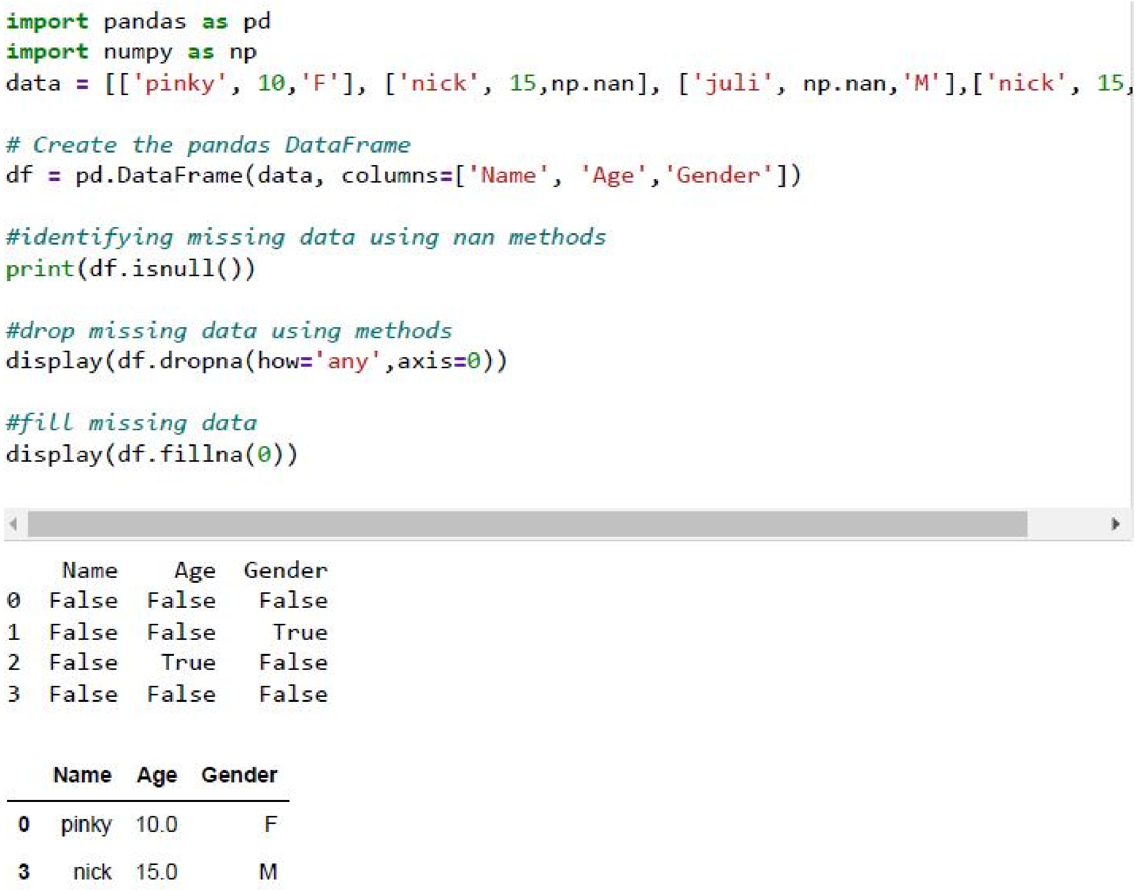
```
import sqlite3
import sqlalchemy as sl
import pandas as pd
con=sqlite3.connect("mydata.sqlite")
#create a table
query='''create table AIDS_STU("name" VARCHAR2(20),"Branch" VARCHAR2(20));'''
con.execute(query)
#data insertion
data=[("saibaba","AIDS"),("Pavan","CSE")]
stmt="INSERT INTO AIDS_STU VALUES(?,?)"
con.executemany(stmt,data)
#data display
connect=con.execute('select * from AIDS_STU')
show=connect.fetchall()
display(show)
df=pd.DataFrame(show,columns=['NAMES','BRANCH'],index=[1,2])
display(df)
```



```
Out[13]:
        NAMES   BRANCH
    1   saibaba   AIDS
    2     Pavan    CSE
```

6. Data Cleaning and Preparation

AIM: Perform data cleaning by creating a DataFrame and identifying missing data using NA(Not Available) handling methods, filter out missing data using dropna function, fill the missing data using fillna function and remove duplicates using duplicated and drop_duplicates functions.

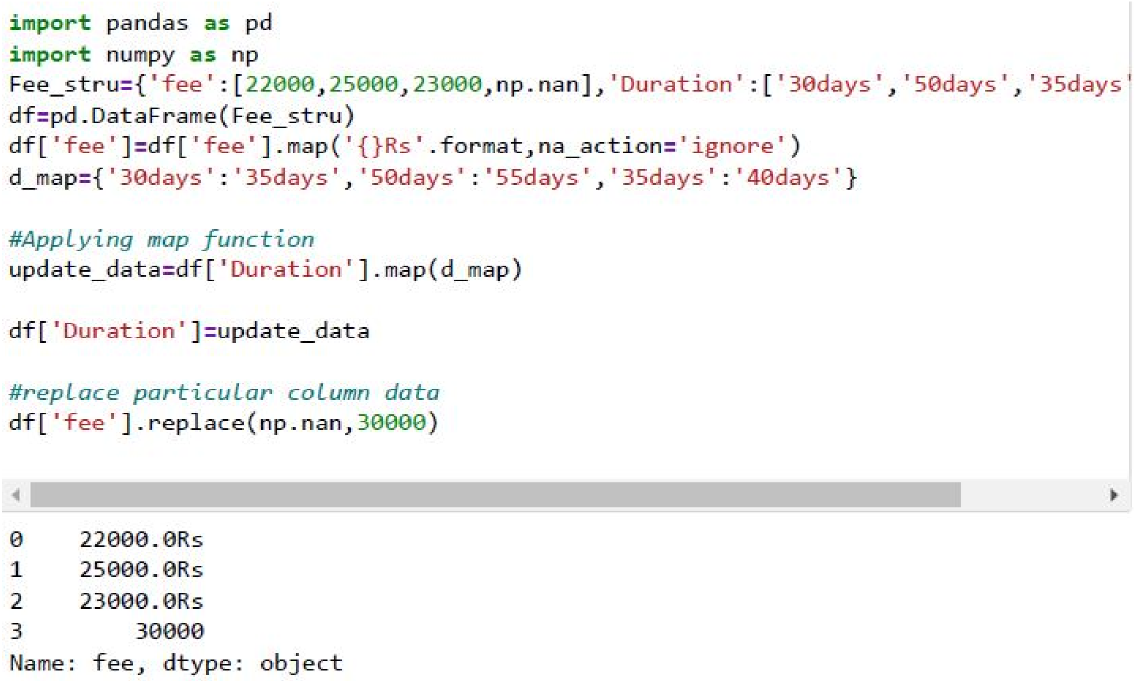


```python
import pandas as pd
import numpy as np
data = [['pinky', 10,'F'], ['nick', 15,np.nan], ['juli', np.nan,'M'],['nick', 15,

# Create the pandas DataFrame
df = pd.DataFrame(data, columns=['Name', 'Age','Gender'])

#identifying missing data using nan methods
print(df.isnull())

#drop missing data using methods
display(df.dropna(how='any',axis=0))

#fill missing data
display(df.fillna(0))
```

```
    Name    Age  Gender
0  False  False   False
1  False  False    True
2  False   True   False
3  False  False   False
```

	Name	Age	Gender
0	pinky	10.0	F
3	nick	15.0	M

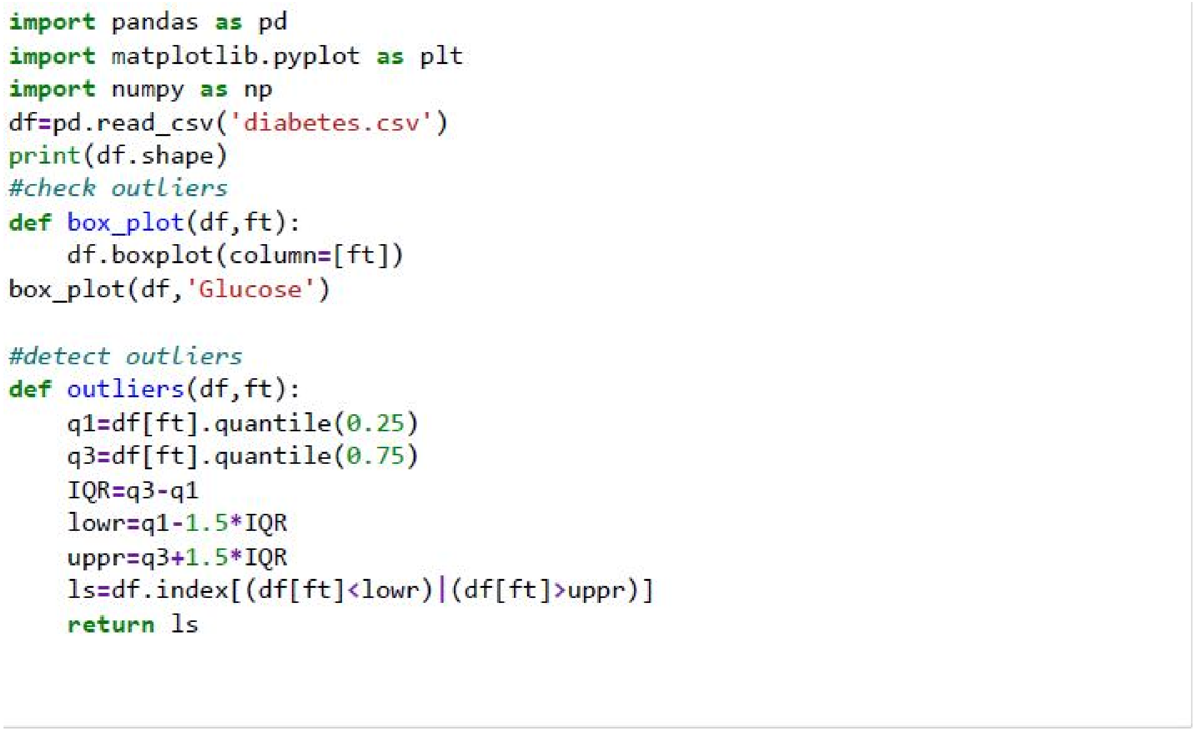
AIM: Perform data transformation by modifying set of values using map and replace method and create transformed version of original dataset without modification using rename method.



```
import pandas as pd
import numpy as np
Fee_stru={'fee':[22000,25000,23000,np.nan],'Duration':['30days','50days','35days'
df=pd.DataFrame(Fee_stru)
df['fee']=df['fee'].map('{}Rs'.format,na_action='ignore')
d_map={'30days':'35days','50days':'55days','35days':'40days'}

#Applying map function
update_data=df['Duration'].map(d_map)

df['Duration']=update_data

#replace particular column data
df['fee'].replace(np.nan,30000)
```

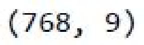
```
0    22000.0Rs
1    25000.0Rs
2    23000.0Rs
3        30000
Name: fee, dtype: object
```

AIM: Create a DataFrame with normally distributed data using random sampling and detect possible outliers

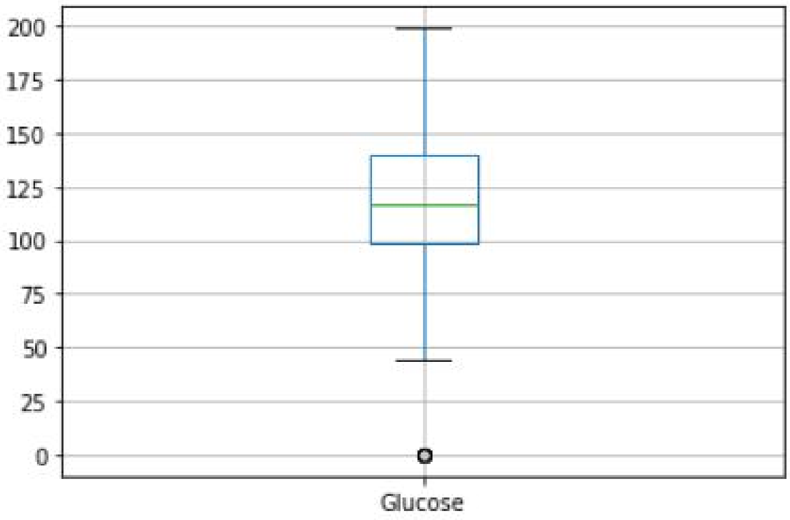


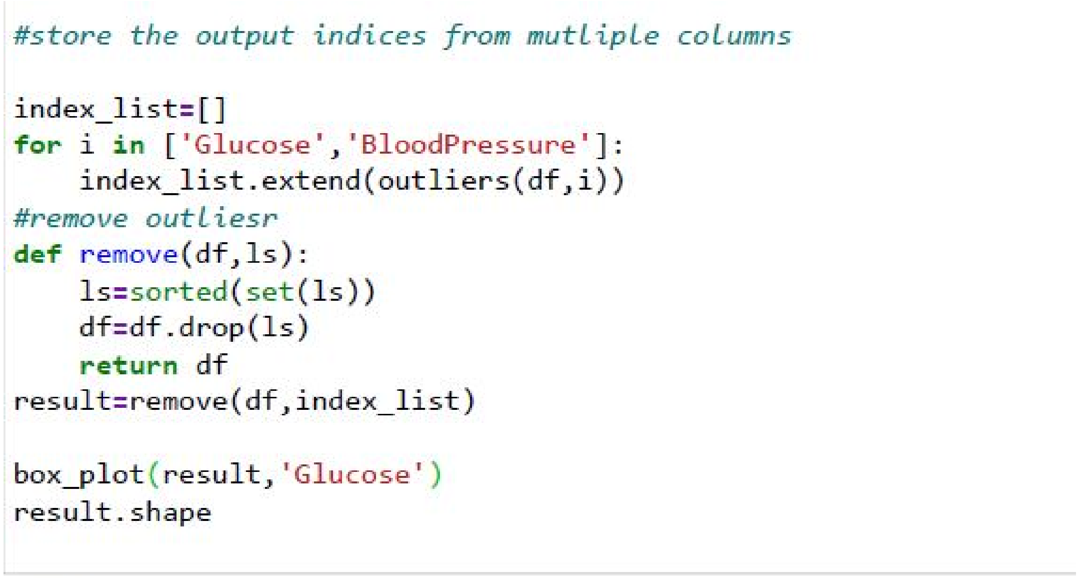
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
df=pd.read_csv('diabetes.csv')
print(df.shape)
#check outliers
def box_plot(df,ft):
    df.boxplot(column=[ft])
box_plot(df,'Glucose')

#detect outliers
def outliers(df,ft):
    q1=df[ft].quantile(0.25)
    q3=df[ft].quantile(0.75)
    IQR=q3-q1
    lowr=q1-1.5*IQR
    uppr=q3+1.5*IQR
    ls=df.index[(df[ft]<lowr)|(df[ft]>uppr)]
    return ls
```

Output:



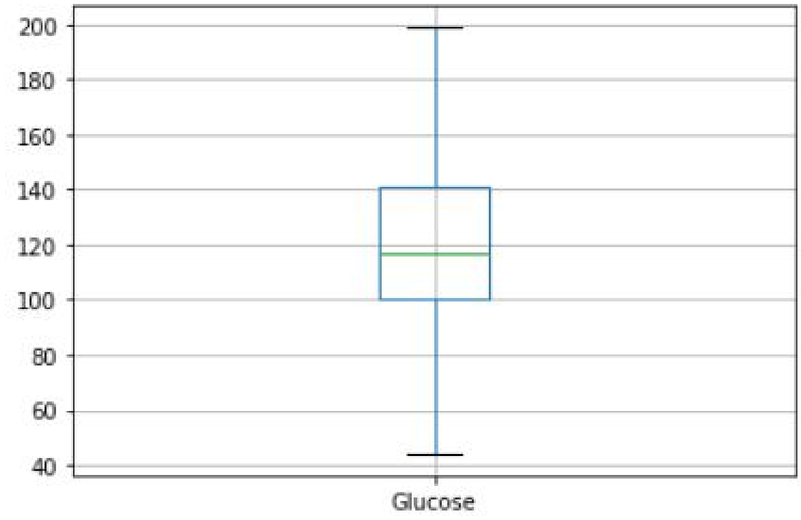
```
(768, 9)
```



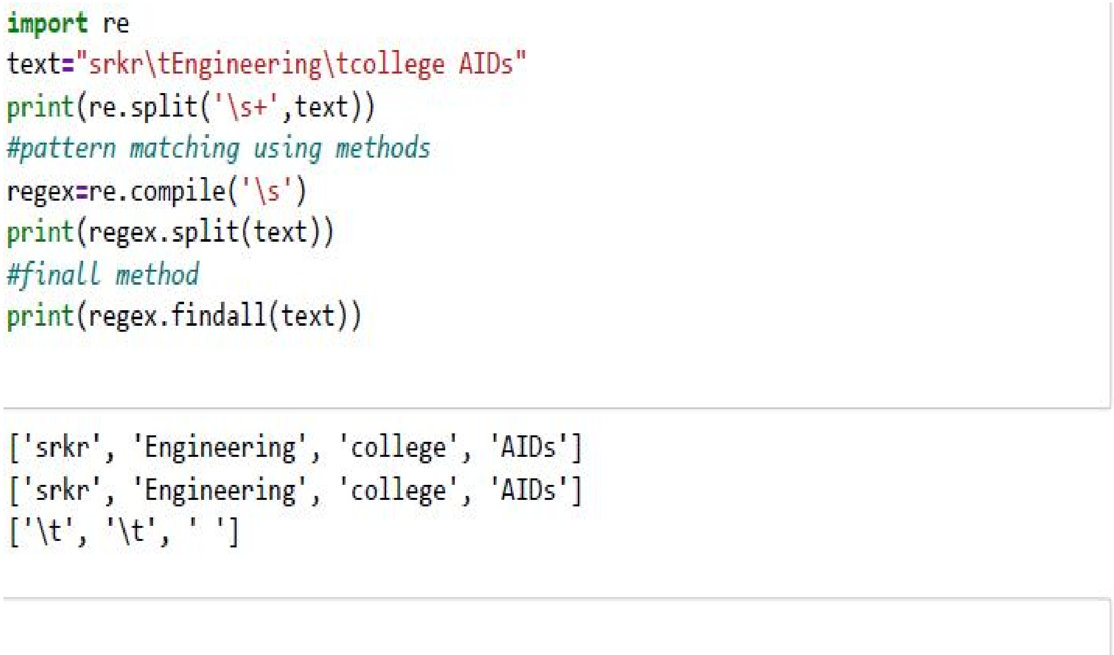


```
#store the output indices from mutliple columns

index_list=[]
for i in ['Glucose','BloodPressure']:
    index_list.extend(outliers(df,i))
#remove outliesr
def remove(df,ls):
    ls=sorted(set(ls))
    df=df.drop(ls)
    return df
result=remove(df,index_list)

box_plot(result,'Glucose')
result.shape
```

```
(718, 9)
```



AIM : Perform text manipulation with regular expression by applying relevant regular expression methods to split a string with a variable number of whitespace characters (tabs, spaces, and newlines) and get a list of all patterns matching**.**

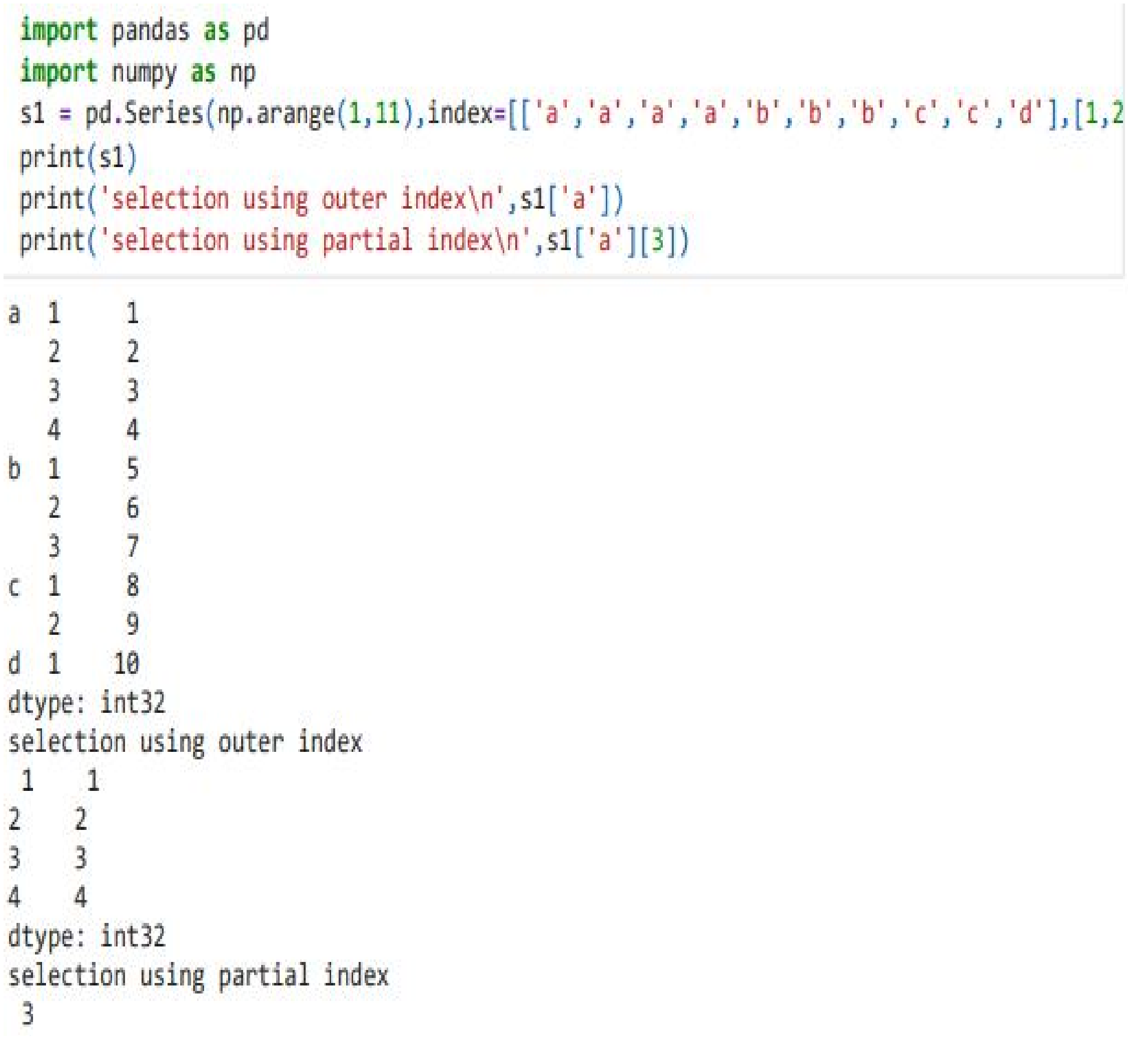


```
import re
text="srkr\tEngineering\tcollege AIDs"
print(re.split('\s+',text))
#pattern matching using methods
regex=re.compile('\s')
print(regex.split(text))
#finall method
print(regex.findall(text))
```

```
['srkr', 'Engineering', 'college', 'AIDs']
['srkr', 'Engineering', 'college', 'AIDs']
['\t', '\t', ' ']
```

7. Data Wrangling

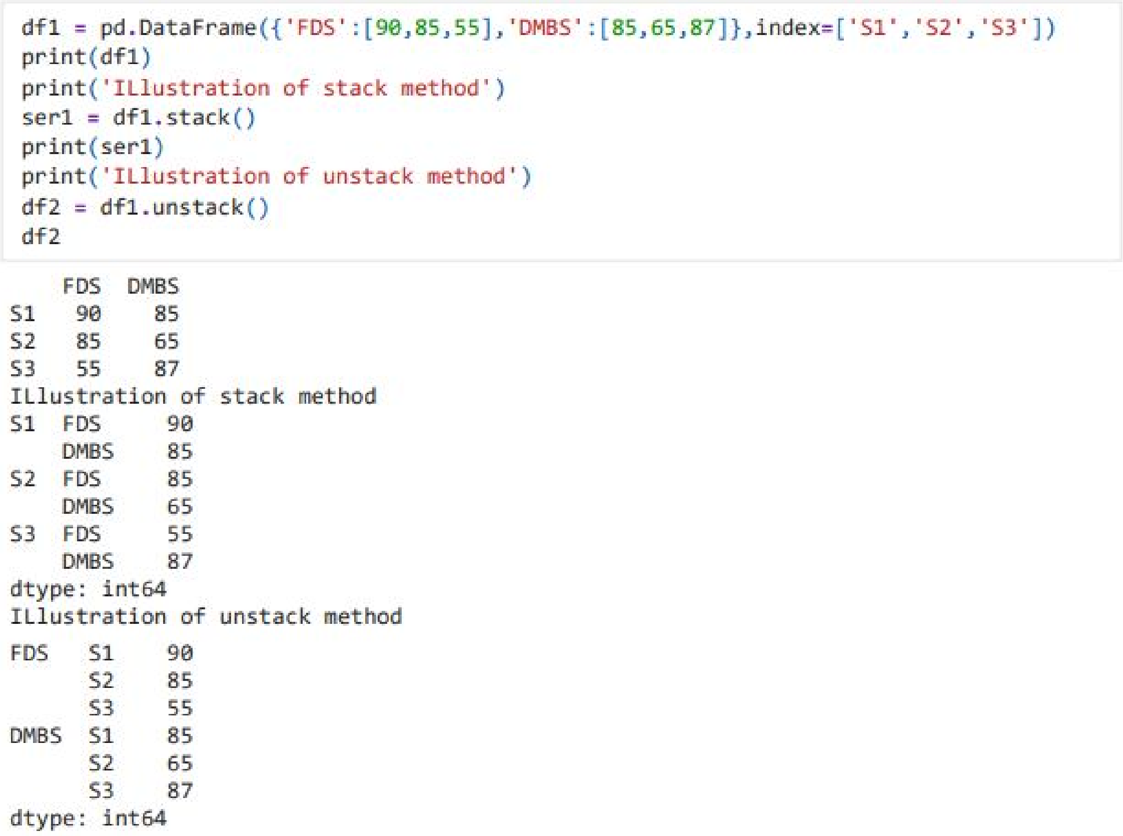
AIM: Perform hierarchical indexing by creating a series with a list of lists (or arrays) as the index, select subsets of data at outer and inner levels using partial indexing.



```
import pandas as pd
import numpy as np
s1 = pd.Series(np.arange(1,11),index=[['a','a','a','a','b','b','b','c','c','d'],[1,2
print(s1)
print('selection using outer index\n',s1['a'])
print('selection using partial index\n',s1['a'][3])
```

```
a  1     1
   2     2
   3     3
   4     4
b  1     5
   2     6
   3     7
c  1     8
   2     9
d  1    10
dtype: int32
selection using outer index
 1    1
2    2
3    3
4    4
dtype: int32
selection using partial index
 3
```

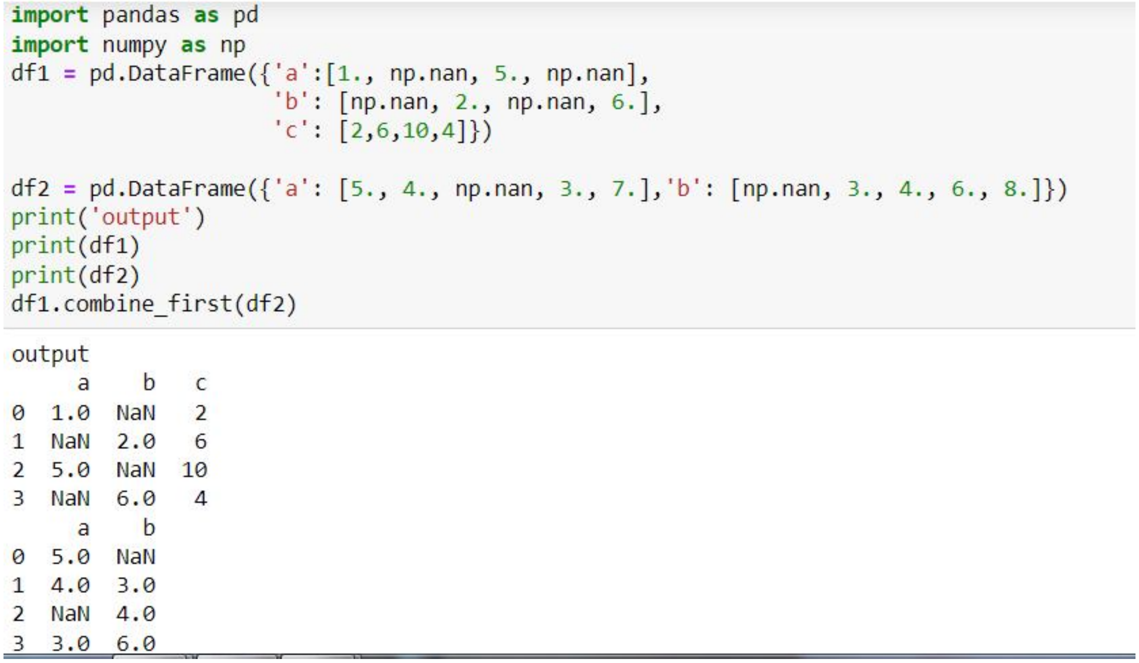
AIM: Rearrange the tabular data with hierarchical indexing using unstack and stack method.



```
df1 = pd.DataFrame({'FDS':[90,85,55],'DMBS':[85,65,87]},index=['S1','S2','S3'])
print(df1)
print('ILlustration of stack method')
ser1 = df1.stack()
print(ser1)
print('ILlustration of unstack method')
df2 = df1.unstack()
df2
```

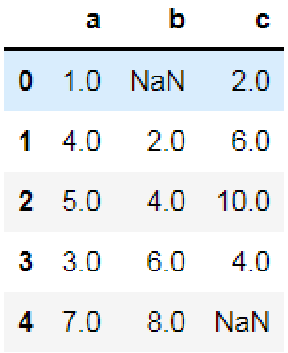
```
    FDS  DMBS
S1   90    85
S2   85    65
S3   55    87
ILlustration of stack method
S1  FDS     90
    DMBS    85
S2  FDS     85
    DMBS    65
S3  FDS     55
    DMBS    87
dtype: int64
ILlustration of unstack method
FDS   S1    90
      S2    85
      S3    55
DMBS  S1    85
      S2    65
      S3    87
dtype: int64
```

AIM: Create two different DataFrames and merge them using index as merge key and combine data with overlap using combine_first method



```
import pandas as pd
import numpy as np
df1 = pd.DataFrame({'a':[1., np.nan, 5., np.nan],
                    'b': [np.nan, 2., np.nan, 6.],
                    'c': [2,6,10,4]})

df2 = pd.DataFrame({'a': [5., 4., np.nan, 3., 7.],'b': [np.nan, 3., 4., 6., 8.]})
print('output')
print(df1)
print(df2)
df1.combine_first(df2)
```

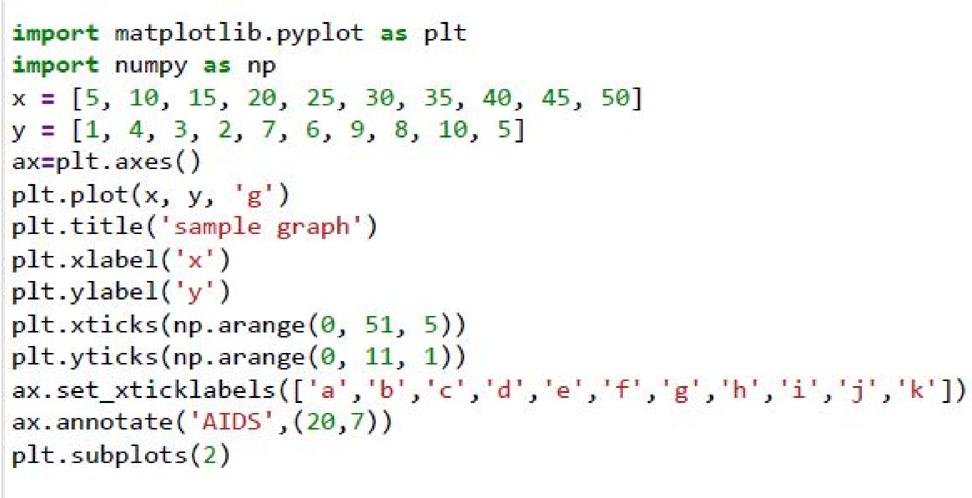
```
output
     a    b   c
0  1.0  NaN   2
1  NaN  2.0   6
2  5.0  NaN  10
3  NaN  6.0   4
     a    b
0  5.0  NaN
1  4.0  3.0
2  NaN  4.0
3  3.0  6.0
```



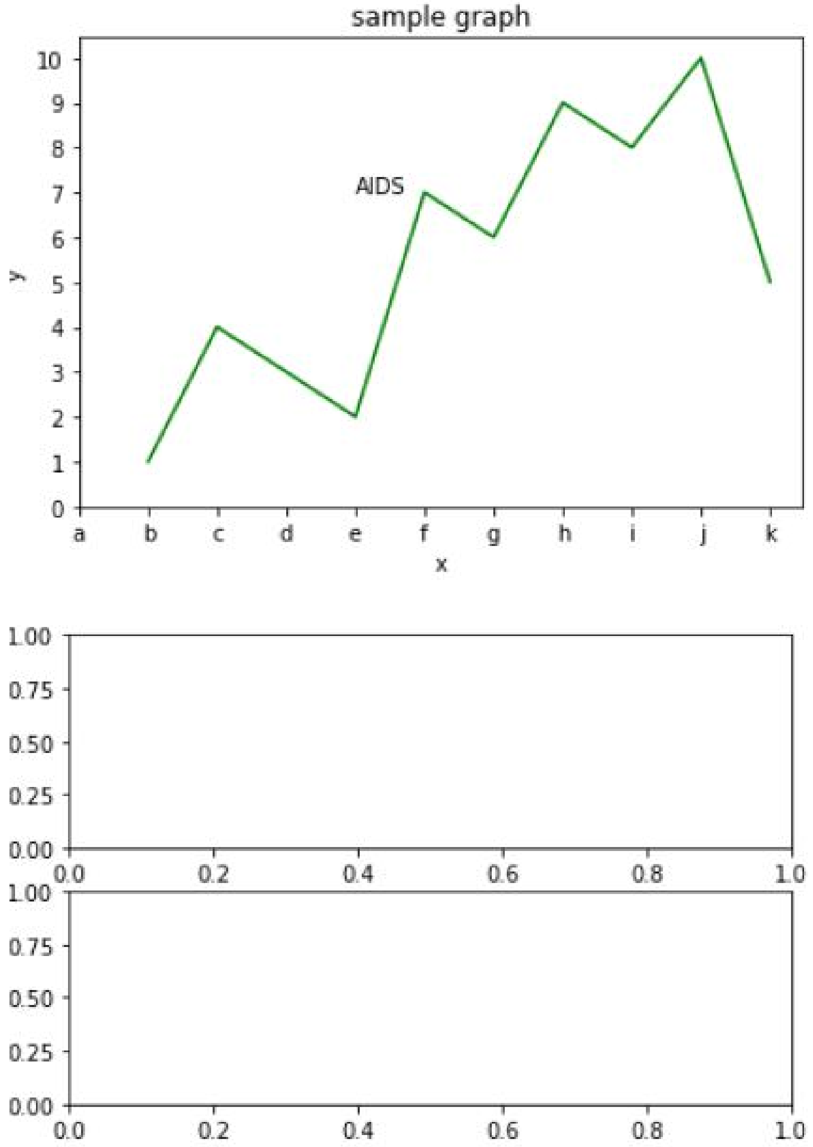
	a	b	c
0	1.0	NaN	2.0
1	4.0	2.0	6.0
2	5.0	4.0	10.0
3	3.0	6.0	4.0
4	7.0	8.0	NaN

8. Perform Data Visualization with Matplotlib and SeaBorn considering online dataset for processing

AIM : Create a Line Plot by setting the title, axis labels, ticks, ticklabels , annotations on subplots and save to a file.

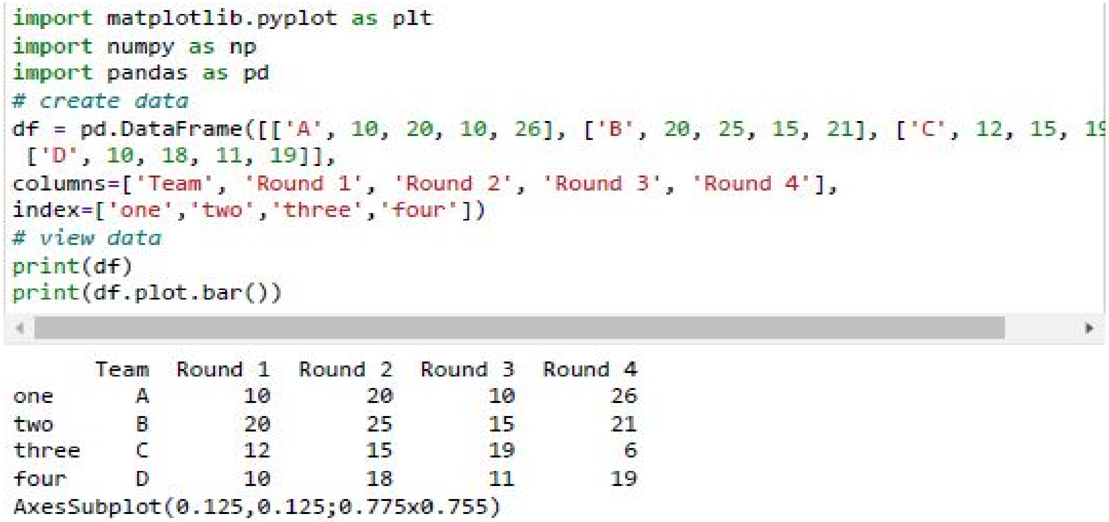


```
import matplotlib.pyplot as plt
import numpy as np
x = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]
y = [1, 4, 3, 2, 7, 6, 9, 8, 10, 5]
ax=plt.axes()
plt.plot(x, y, 'g')
plt.title('sample graph')
plt.xlabel('x')
plt.ylabel('y')
plt.xticks(np.arange(0, 51, 5))
plt.yticks(np.arange(0, 11, 1))
ax.set_xticklabels(['a','b','c','d','e','f','g','h','i','j','k'])
ax.annotate('AIDS',(20,7))
plt.subplots(2)
```


sample graph
AIDS
y
0
1
2
3
4
5
6
7
8
9
10
a
b
c
d
e
f
g
h
i
j
k
x
1.00
0.75
0.50
0.25
0.00
0.0
0.2
0.4
0.6
0.8
1.0
1.00
0.75
0.50
0.25
0.00
0.0
0.2
0.4
0.6
0.8
1.0

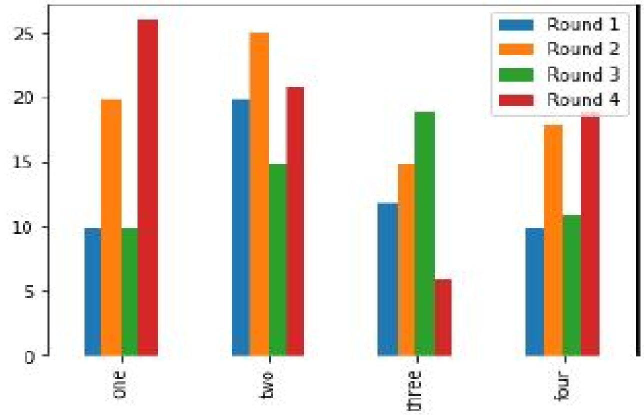
AIM: Create Bar Plots using Series and DataFrame index.

i. Create bar plots with a DataFrame to group the values in each row together in a group in bars side by side for each value.

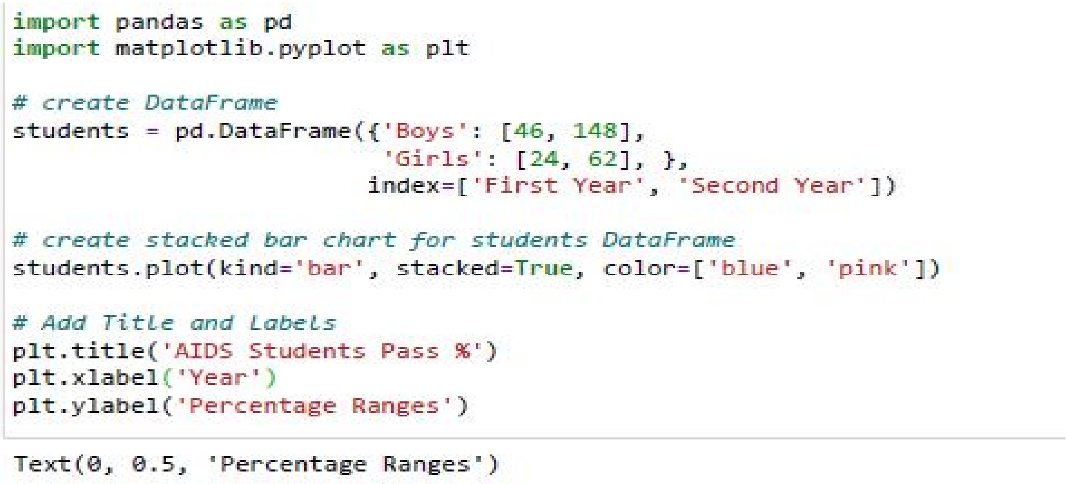


```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
# create data
df = pd.DataFrame([['A', 10, 20, 10, 26], ['B', 20, 25, 15, 21], ['C', 12, 15, 19
 ['D', 10, 18, 11, 19]],
columns=['Team', 'Round 1', 'Round 2', 'Round 3', 'Round 4'],
index=['one','two','three','four'])
# view data
print(df)
print(df.plot.bar())
```

```
      Team  Round 1  Round 2  Round 3  Round 4
one      A       10       20       10       26
two      B       20       25       15       21
three    C       12       15       19        6
four     D       10       18       11       19
AxesSubplot(0.125,0.125;0.775x0.755)
```

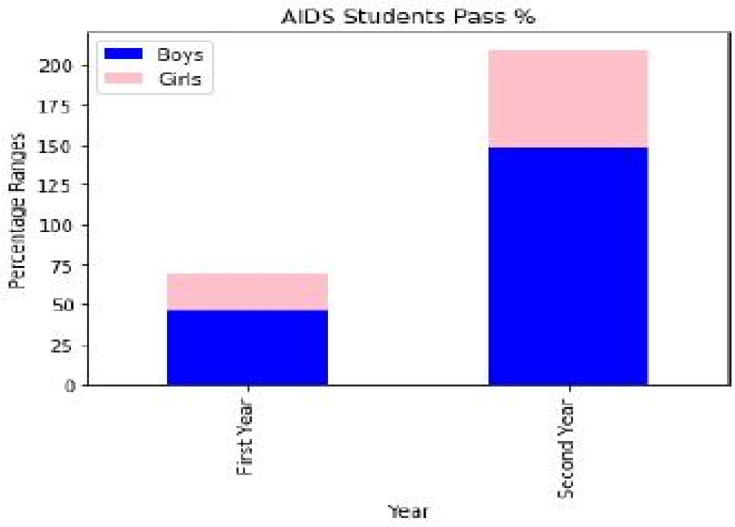


ii) Create stacked bar plots from a DataFrame.

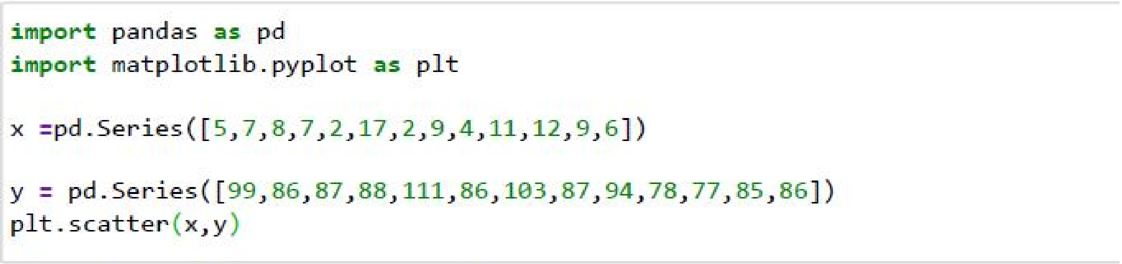


```
import pandas as pd
import matplotlib.pyplot as plt

# create DataFrame
students = pd.DataFrame({'Boys': [46, 148],
                         'Girls': [24, 62], },
                        index=['First Year', 'Second Year'])

# create stacked bar chart for students DataFrame
students.plot(kind='bar', stacked=True, color=['blue', 'pink'])

# Add Title and Labels
plt.title('AIDS Students Pass %')
plt.xlabel('Year')
plt.ylabel('Percentage Ranges')
```

```
Text(0, 0.5, 'Percentage Ranges')
```

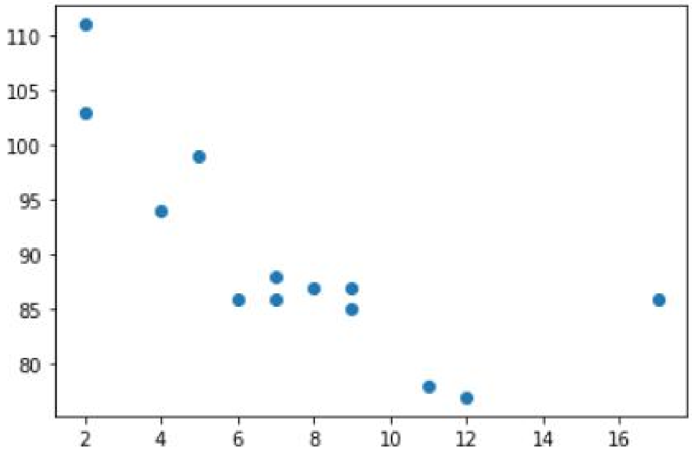


AIM : Create Histogram to display the value frequency and Density Plot to generate continuous probability distribution function for observed data.

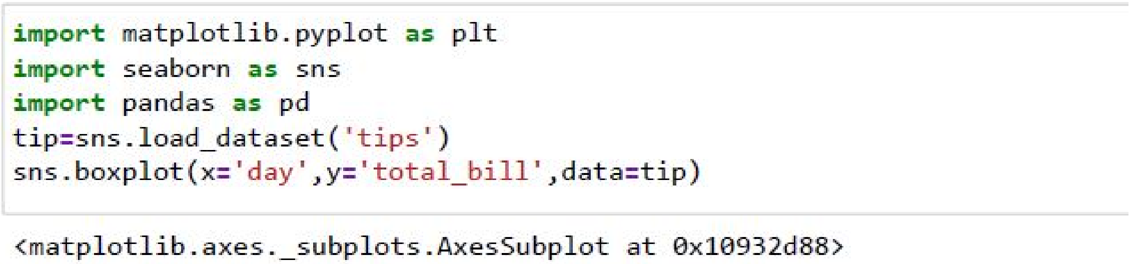


```
import pandas as pd
import matplotlib.pyplot as plt

x =pd.Series([5,7,8,7,2,17,2,9,4,11,12,9,6])

y = pd.Series([99,86,87,88,111,86,103,87,94,78,77,85,86])
plt.scatter(x,y)
```

```
<matplotlib.collections.PathCollection at 0xe78f988>
```

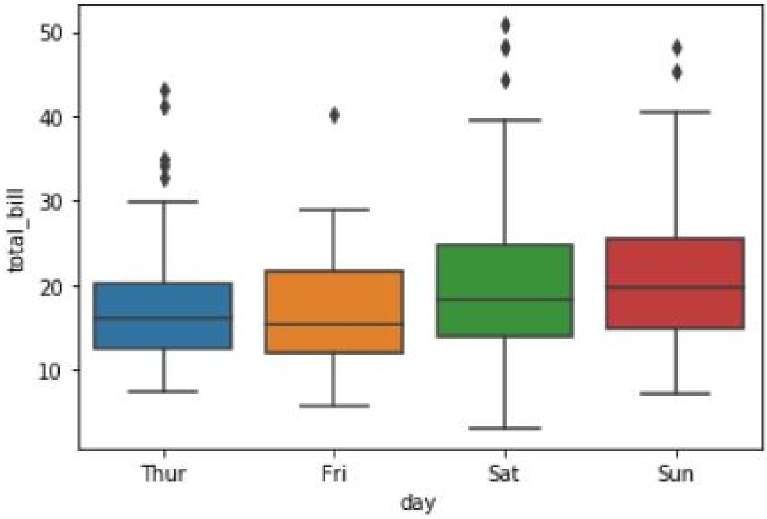


AIM: Create Scatter Plot and examine the relationship between two one-dimensional data series.

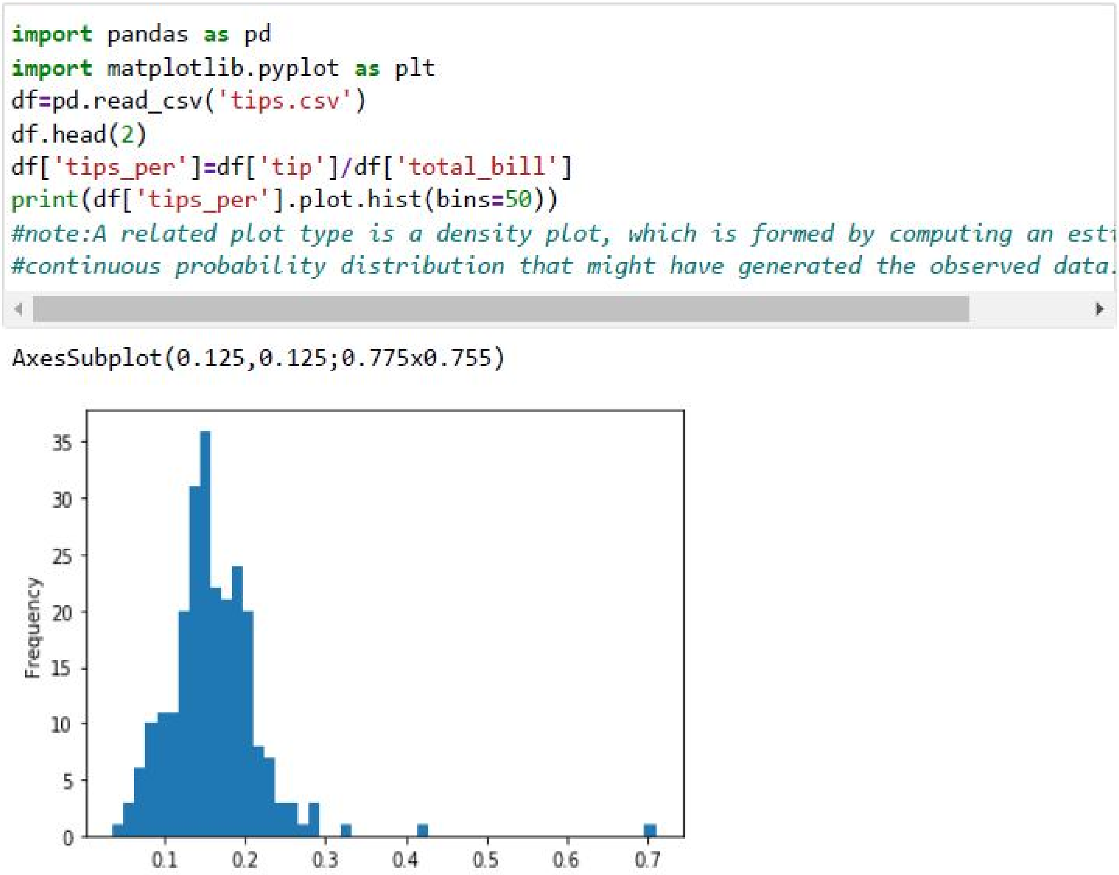


```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
tip=sns.load_dataset('tips')
sns.boxplot(x='day',y='total_bill',data=tip)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x10932d88>
```



AIM : Create Box plots to visualize data with many categorical variables.

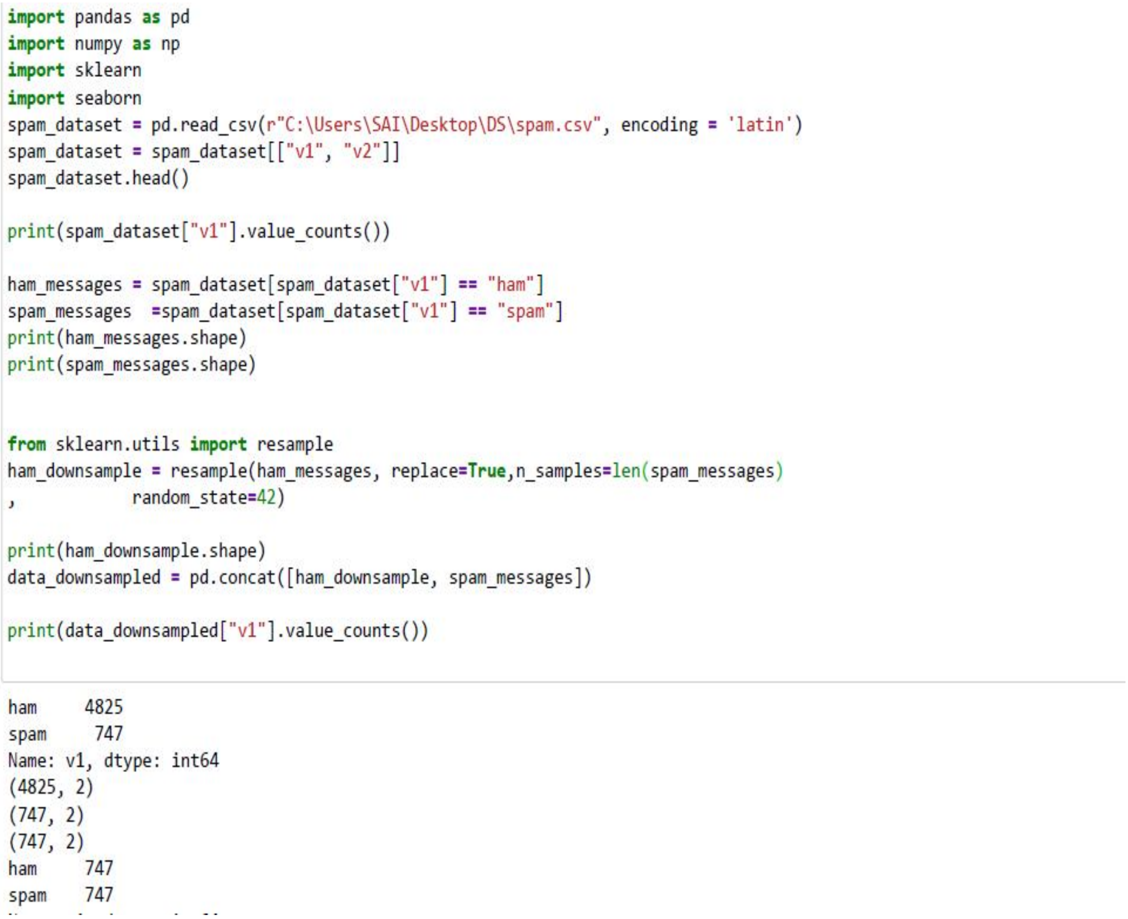


```
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv('tips.csv')
df.head(2)
df['tips_per']=df['tip']/df['total_bill']
print(df['tips_per'].plot.hist(bins=50))
#note:A related plot type is a density plot, which is formed by computing an esti
#continuous probability distribution that might have generated the observed data.
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```

9. Time Series Analysis

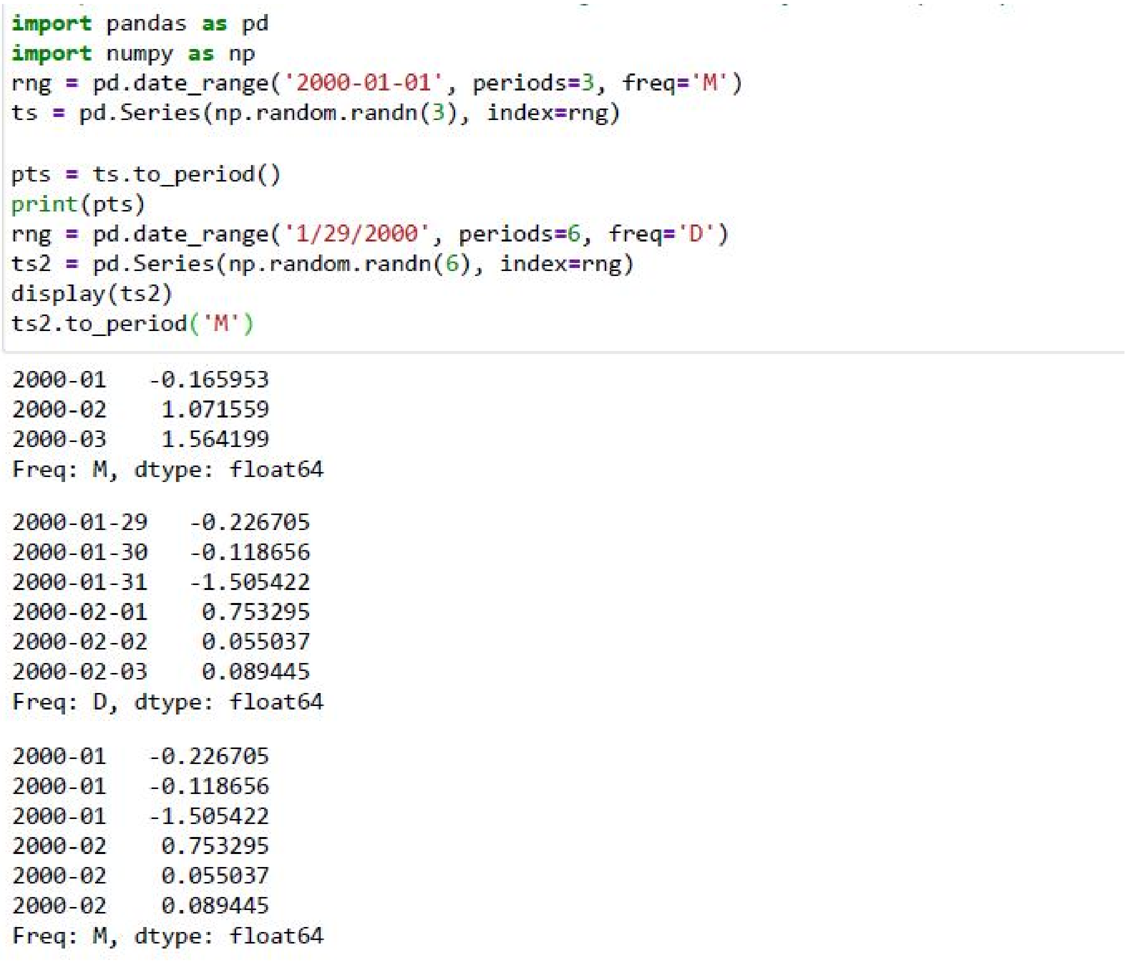
AIM : Perform resampling, downsampling and upsampling for the time series**.**



```
import pandas as pd
import numpy as np
import sklearn
import seaborn
spam_dataset = pd.read_csv(r"C:\Users\SAI\Desktop\DS\spam.csv", encoding = 'latin')
spam_dataset = spam_dataset[["v1", "v2"]]
spam_dataset.head()

print(spam_dataset["v1"].value_counts())

ham_messages = spam_dataset[spam_dataset["v1"] == "ham"]
spam_messages  =spam_dataset[spam_dataset["v1"] == "spam"]
print(ham_messages.shape)
print(spam_messages.shape)


from sklearn.utils import resample
ham_downsample = resample(ham_messages, replace=True,n_samples=len(spam_messages)
,            random_state=42)

print(ham_downsample.shape)
data_downsampled = pd.concat([ham_downsample, spam_messages])

print(data_downsampled["v1"].value_counts())
```

```
ham     4825
spam     747
Name: v1, dtype: int64
(4825, 2)
(747, 2)
(747, 2)
ham     747
spam    747
```

AIM : Convert Series and DataFrame objects indexed by timestamps to periods with the to_period method.



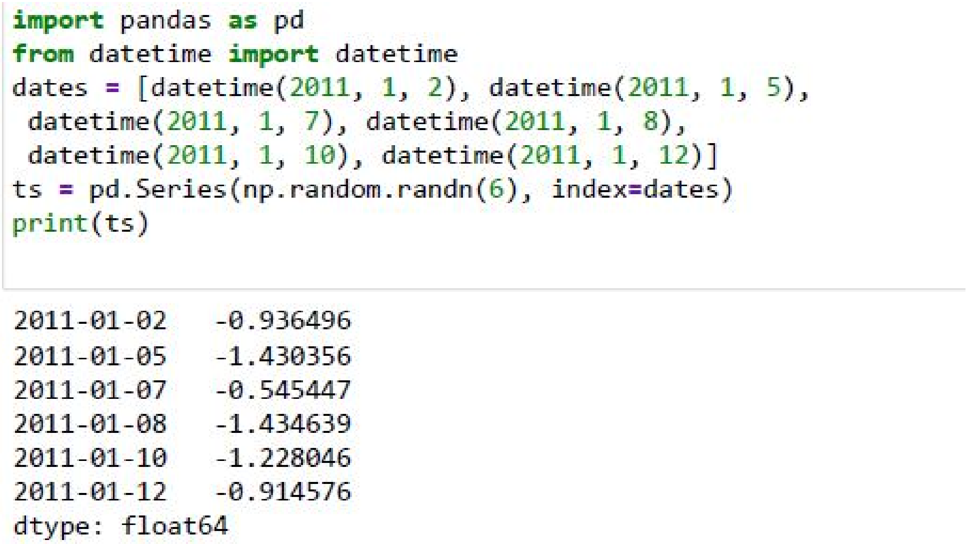
```
import pandas as pd
import numpy as np
rng = pd.date_range('2000-01-01', periods=3, freq='M')
ts = pd.Series(np.random.randn(3), index=rng)

pts = ts.to_period()
print(pts)
rng = pd.date_range('1/29/2000', periods=6, freq='D')
ts2 = pd.Series(np.random.randn(6), index=rng)
display(ts2)
ts2.to_period('M')
```

```
2000-01   -0.165953
2000-02    1.071559
2000-03    1.564199
Freq: M, dtype: float64
```

```
2000-01-29   -0.226705
2000-01-30   -0.118656
2000-01-31   -1.505422
2000-02-01    0.753295
2000-02-02    0.055037
2000-02-03    0.089445
Freq: D, dtype: float64
```

```
2000-01   -0.226705
2000-01   -0.118656
2000-01   -1.505422
2000-02    0.753295
2000-02    0.055037
2000-02    0.089445
Freq: M, dtype: float64
```

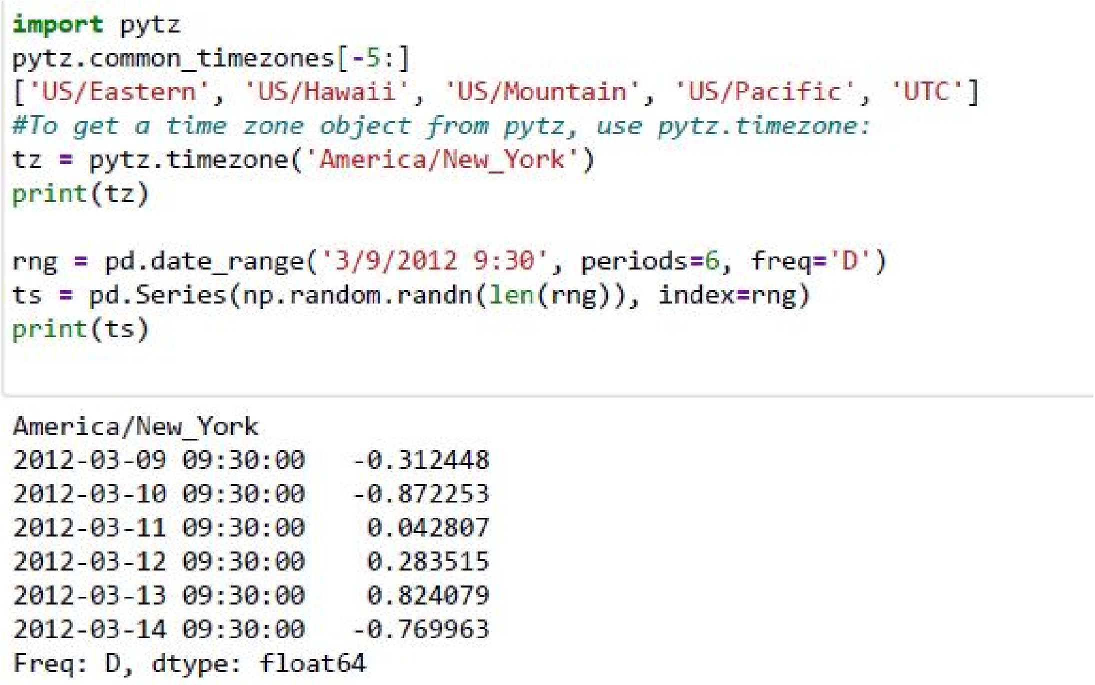
AIM : Create time series using datetime object in pandas indexed by timestamps.



```
import pandas as pd
from datetime import datetime
dates = [datetime(2011, 1, 2), datetime(2011, 1, 5),
 datetime(2011, 1, 7), datetime(2011, 1, 8),
 datetime(2011, 1, 10), datetime(2011, 1, 12)]
ts = pd.Series(np.random.randn(6), index=dates)
print(ts)
```

```
2011-01-02   -0.936496
2011-01-05   -1.430356
2011-01-07   -0.545447
2011-01-08   -1.434639
2011-01-10   -1.228046
2011-01-12   -0.914576
dtype: float64
```

AIM : Generate data ranges by setting time zone, localize time zone and convert to particular time zone using tz_convert and combine two different time zones.

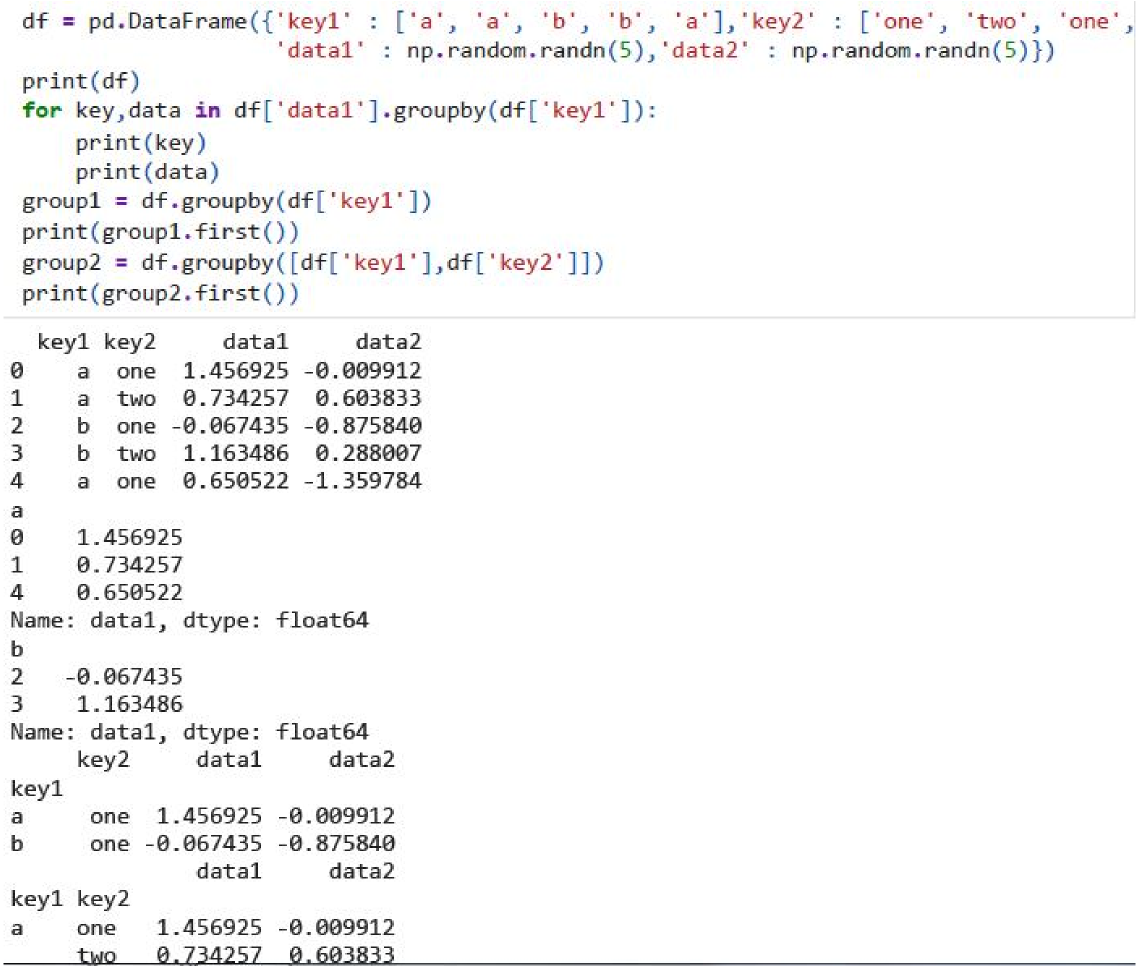


```
import pytz
pytz.common_timezones[-5:]
['US/Eastern', 'US/Hawaii', 'US/Mountain', 'US/Pacific', 'UTC']
#To get a time zone object from pytz, use pytz.timezone:
tz = pytz.timezone('America/New_York')
print(tz)

rng = pd.date_range('3/9/2012 9:30', periods=6, freq='D')
ts = pd.Series(np.random.randn(len(rng)), index=rng)
print(ts)
```

```
America/New_York
2012-03-09 09:30:00   -0.312448
2012-03-10 09:30:00   -0.872253
2012-03-11 09:30:00    0.042807
2012-03-12 09:30:00    0.283515
2012-03-13 09:30:00    0.824079
2012-03-14 09:30:00   -0.769963
Freq: D, dtype: float64
```

10. Data Aggregation

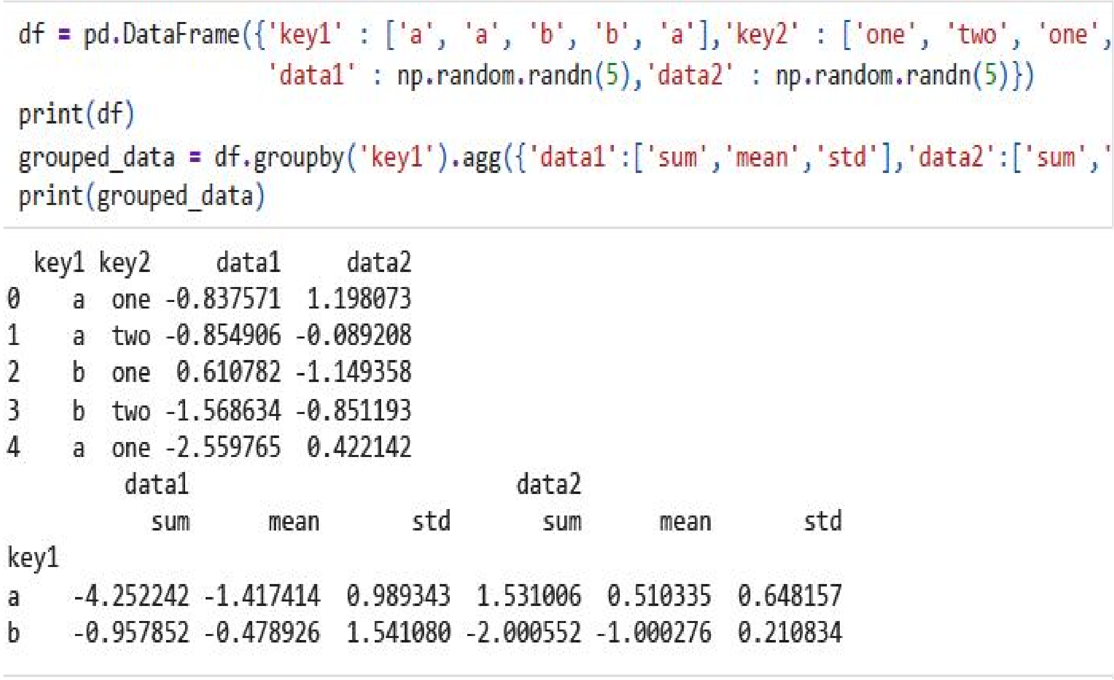
AIM : Create a tabular dataset as a DataFrame and split data into groups using group by method including single key and multiple key values. Select group by considering single and multiple columns.



```
df = pd.DataFrame({'key1' : ['a', 'a', 'b', 'b', 'a'],'key2' : ['one', 'two', 'one',
                    'data1' : np.random.randn(5),'data2' : np.random.randn(5)})
print(df)
for key,data in df['data1'].groupby(df['key1']):
    print(key)
    print(data)
group1 = df.groupby(df['key1'])
print(group1.first())
group2 = df.groupby([df['key1'],df['key2']])
print(group2.first())
```

```
  key1 key2     data1     data2
0    a  one  1.456925 -0.009912
1    a  two  0.734257  0.603833
2    b  one -0.067435 -0.875840
3    b  two  1.163486  0.288007
4    a  one  0.650522 -1.359784
a
0    1.456925
1    0.734257
4    0.650522
Name: data1, dtype: float64
b
2   -0.067435
3    1.163486
Name: data1, dtype: float64
     key2     data1     data2
key1
a     one  1.456925 -0.009912
b     one -0.067435 -0.875840
               data1     data2
key1 key2
a    one    1.456925 -0.009912
     two    0.734257  0.603833
```

AIM : Compute summary statistics such as sum, mean and standard deviation for the grouped data using aggregate method.



```
df = pd.DataFrame({'key1' : ['a', 'a', 'b', 'b', 'a'],'key2' : ['one', 'two', 'one',
                   'data1' : np.random.randn(5),'data2' : np.random.randn(5)})
print(df)
grouped_data = df.groupby('key1').agg({'data1':['sum','mean','std'],'data2':['sum','
print(grouped_data)
```

```
  key1 key2     data1     data2
0    a  one -0.837571  1.198073
1    a  two -0.854906 -0.089208
2    b  one  0.610782 -1.149358
3    b  two -1.568634 -0.851193
4    a  one -2.559765  0.422142
         data1                          data2
           sum      mean       std       sum      mean       std
key1
a    -4.252242 -1.417414  0.989343  1.531006  0.510335  0.648157
b    -0.957852 -0.478926  1.541080 -2.000552 -1.000276  0.210834
```