Name:- KARTHIK. M.

Assignment:-

Question 1: pandas assignment:

1) We can use the utility function pd.
show_versions() to check the version of the dependencies. and we can also use pd._version - to check the version of the pandas running on any system.

```
#importing pandas as pd.
import pandas as pd.
# check the version.
Print (Pd.-- version --).
```

2) pandas is a powerful tool which is used for data analysis and is built on top of the python library. The pandas library enables users to create. and manipulate dataframes (Tables of data) & time series effectively and efficiently.

Code:-

```
# importing the pandas library as
Import pandas as pd.
# Creating the dataframe df
df = pd. Dataframe ({'Roll Number' : ['20CSE29',
'20CSE49', '20CSE36', '20CSE44'],
                    'Name' : ['Amelia', 'Sam', 'Dean',
'Jessica'],
                    'Marks In percentage' : [97, 90, 70, 82],
                    'Grade' : ['A', 'A', 'C', 'B'],
                    'Subject' : ['Physics', 'Physics',
'Physics', 'Physics']}).
```

3). Using Series.append().

This method is a shortcut to concat. This method Concatenate along axis=0 i.e. rows. series.append() can take multiple objects to concatenate.

Code:-

```
# Import pandas Library.
import pandas as pd.
# Create a series.
a = pd.Series(["ABC", "DEF", "GHI"])
# Create a series.
b = pd.Series(["JKL", "MNO", "PQR"])
# Combine two series then
# Create a dataframe.
df = pd.DataFrame(a.append(b, ignore_index
                           = True))
# Show the dataframe.
df.
```

4) rename() function is used to alter Series index labels (or) name for the given series object. inplace: whether to return a new series, If true then value of copy is ignored. level: In Case of a multiindex, only rename labels in the specified level.

Syntax:- Series.rename(index = None, **kwargs).

Code:-

```
# Importing pandas as pd.
import pandas as pd.
# Creating the series.
Sr = pd.Series([10, 25, 3, 11, 24, 6])
```

```
# Create the Index.
Index_ = ['Coca Cola', 'Sprite', 'Coke', 'Fanta',
          'Dew', 'Thumbsup']
# get the Index
sr. Index = Index_
# print the series
Print (sr)
```

5) Soln:

Code: 1) import pandas as pd.
2) import numpy as np
3) num_state = np. random. Random State (100)
4) num - series = pd . series (num - state . normal (10, 4, 20))
5) Print ("Original series:")
6) print (num - series)
7) result = np. percentile (num - series, q = [0, 25, 50, 75, 100])
8) print ("\n minimum, 25 th percentile, median, 75 th, and maximum of a given series:")
9) print (result).

6) qcut() - "Quantile - based discretization function"

This is nothing but qcut tries to divide up the underlying data into equal sized bins. .

* qcut is to define the number of quantiles and let pandas figure out how to divide up the data.

7) A simple solution is to iterate over all number from 1 to n and increment count from 1 to n and increment count wherever a number is a multiple of 3.

1. import pandas as pd.
2. import numpy as np.
3. num_series = pd.Series(np.random.~~rand~~randint(1,10,9))
4. print("Original Series:")
5. print(num_series)
6. result = np.argwhere(num_series % 5==0)
7. print("positions of numbers that are multiples of 5:")
8. print(result)

8) Horizontally:-

Code:-

```
# Importing the module.
import pandas as pd.
# Creating the series.
Series1 = pd.Series(['g','e','e','k','s'])
Print("series 1:")
Print(series1)
Series2 = pd.Series([9, 8,7,6,5])
Print("Series 2:")
Print(Series2)
# Stacking the series horizontally.
df = pd.concat([series1, series2], axis=1)
Print("\n Stack two series horizontally:")
display(df)
```

Vertically:-

code:-

```
#importing the module
import pandas as pd.
# creating the series
series 1 = pd. series([ 'g', 'e' , 'e', 'k', 's'])
Print ( " series 1: ")
Print (series 1)
Series 2 = pd. series ([9, 8, 7, 6, 5])
Print ("series 2: ")
Print (Series 2)
# stacking the series vertically.
df = pd. Contact ([ series 1, series 2], axis = 0)
Print ("\n Stack two series vertically: ")
display (df)
```

9)

```
1) import pandas as pd.
2) from dateutil. parser import parse
3) date - series = pd. series ([ '01 jan 2015', '10-2-2016',
   '20180307', '2014/05/06', '2016-04-12', '2019-04-
   06T11:20' ])
4) print ("Original series:")
5) print (data - series)
6) data - series = date - series. map (lamba x : parse(x))
7) print ("Day of month:")
8) print (date - series. dt. day. tolist())
9) print ("Day of year:")
10) print (date - series. dt. dayofyear. tolist())
11) Print ("week number:")
12) print (date - series. dt. week of year. tolist())
13) print ("Day of week")
14) print (date - series. dt. weekday - name. tolist())
```

10) The formula $d(x,y) = \sqrt{\sum_{i=0}^{n}(x_i - y_i)^2}$

Code:-

```
import pandas as pd.
import numpy as np
x = pd.series([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
y = pd.series([12, 8, 7, 5, 6, 5, 3, 9, 7, 1])
dist = np.sqrt(np.sum([(a-b)*(a-b) for a,
        b in zip(x, y)]))
print ("series 1: ")
Print(x)
 Print ("series 2: ")
 Print(y)
Print("Euclidean distance between two series:",
                                        dist)
```

11). There are different ways to apply a function to each row or column in Dataframe. we will learn about various ways in this post. let's create a small dataframe first and see that.

Code:-

```
# import pandas and numpy library
   import pandas as pd.
   import numpy as np
# list of tuples.
matrix = [(1,2,3,4),
          (5,6,7,8,),
          (9,10,11,12),
          (13,14,15,16)
         ]
# Create a Dataframe object
df = pd.DataFrame(matrix, columns = list('abcd'))
# output
df
```