Part A : Data Handling

Q1 Create a panda's series from a dictionary of values and a ndarray. Apply all the attributes of series on the above created Series.

```
import pandas as pd
import numpy as np

data = {'apple': 10, 'banana': 20, 'cherry': 30}
series = pd.Series(data, name='fruit_counts')
series.index.name = 'fruits'

data_ndarray = np.array([100, 200, 300, 400, 500])
series_from_ndarray = pd.Series(data_ndarray)

print("Series from dictionary:")
print(series)
print("\nSeries from NumPy array:")
print(series_from_ndarray)
```

```
Series from dictionary:
fruits
apple     10
banana    20
cherry    30
Name: fruit_counts, dtype: int64

Series from NumPy array:
0    100
1    200
2    300
3    400
4    500
dtype: int64
```

Attributes

```
print("Index (axis labels) of the series: ")
print(series.index, "\n")
print("Name of the index: ")
print(series.index.name, "\n")
print("Series as ndarray: ")
print(series.values, "\n")
print("Data type of the index: ")
print(series.dtype, "\n")
print("Tuple of the shape: ")
print(series.shape, "\n")
```

```
print("Number of bytes: ")
print(series.nbytes, "\n")
print("Number of dimensions: ")
print(series.ndim, "\n")
print("NUmber of elements: ")
print(series.size, "\n")
print("Size of dtype of the item: ")
print(series.values.itemsize, "\n")
print("return True if there is any NaN value: ")
print(series.hasnans, "\n")
print("Return True if the Series is empty: ")
print(series.empty, "\n")
print("Name of the Series: ")
print(series.name, "\n")
```

```
Index (axis labels) of the series:
Index(['apple', 'banana', 'cherry'], dtype='object', name='fruits')

Name of the index:
fruits

Series as ndarray:
[10 20 30]

Data type of the index:
int64

Tuple of the shape:
(3,)

Number of bytes:
24

Number of dimensions:
1

NUmber of elements:
3

Size of dtype of the item:
8

return True if there is any NaN value:
False

Return True if the Series is empty:
False

Name of the Series:
fruit_counts
```

Q2 Create a series that stores the area of some states in km2.

```
import pandas as pd

state_areas = pd.Series({'Colorado': 269601, 'Oregon': 254799,
'Michigan': 250487, 'Georgia': 153910, 'North Carolina': 139391,
'Pennsylvania': 119280, 'Virginia': 110785, 'Washington': 184827 })
```

A) Write code to find out the biggest and smallest three areas from the given series.

```
desc = state_areas.sort_values(ascending=False)
print("\n3 Biggest Areas:")
print(desc.head(3), "\n")

print("\n3 Smallest Areas:")
print(desc.tail(3), "\n")
```

```
3 Biggest Areas:
Colorado    269601
Oregon      254799
Michigan    250487
dtype: int64


3 Smallest Areas:
North Carolina    139391
Pennsylvania      119280
Virginia          110785
dtype: int64
```

B) Write code to find out the areas that are more than 50000 km2.

```
print("\nAreas Greater Than 50000 km²:")
areas = state_areas[state_areas > 50000]
print(areas)
```

```
Areas Greater Than 50000 km²:
Colorado          269601
Oregon            254799
Michigan          250487
Georgia           153910
North Carolina    139391
Pennsylvania      119280
Virginia          110785
Washington        184827
dtype: int64
```

Write a program to create a Series object with 6 random integers and having indexes as :['p', 'q', 'r', 'n', 't', 'v']. Also Write program to calculate cubes of the Series values.

```
import pandas as pd
import numpy as np
random_integers = np.random.randint(1, 101, size=6)
i=['p', 'q', 'r', 'n', 't', 'v']
random_series = pd.Series(random_integers, index=i)
print("\nRandom Series:")
print(random_series, "\n")
cubed_series = random_series ** 3
print("\nSeries with Cubed Values:")
print(cubed_series)
```

```
Random Series:
p     4
q    55
r    37
n    67
t    89
v    99
dtype: int32


Series with Cubed Values:
p        64
q    166375
r     50653
n    300763
t    704969
v    970299
dtype: int32
```

4. Consider the following dataframe: CORONA** and answer the questions given below:**

ID	State	Cases
100	Delhi	3000
110	Mumbai	4000
120	Chennai	5000
130	Surat	4500

Create the above-given dictionary with the given indexes.

```
import pandas as pd
data = {'ID': [100, 110, 120, 130],
        'State': ['Delhi', 'Mumbai', 'Chennai', 'Surat'],
        'Cases': [3000, 4000, 5000, 4500]}
```

```
df_corona = pd.DataFrame(data)
df_corona.name = "CORONA"
print("Created DataFrame:")
print(df_corona)
```

```
Created DataFrame:
    ID    State  Cases
0  100    Delhi   3000
1  110   Mumbai   4000
2  120  Chennai   5000
3  130    Surat   4500
```

(a) Write code to add a new column "Recovery" using the series method to store the number of patients recovered in every state.

```
df_corona['Recovery'] = [2500, 3500, 4000, 3800]
print(df_corona)
```

```
    ID    State  Cases  Recovery
0  100    Delhi   3000      2500
1  110   Mumbai   4000      3500
2  120  Chennai   5000      4000
3  130    Surat   4500      3800
```

(b) To add a new column "Deaths" using the assign() method to store the number of deaths in every state.

```
df_corona = df_corona.assign(Deaths=[50, 100, 120, 80])
print(df_corona)
```

```
    ID    State  Cases  Recovery  Deaths
0  100    Delhi   3000      2500      50
1  110   Mumbai   4000      3500     100
2  120  Chennai   5000      4000     120
3  130    Surat   4500      3800      80
```

(c) To add a new row to store details of another state using loc (assume values).

```
new_row_data = {'ID': 140, 'State': 'Kolkata', 'Cases': 2000,
'Recovery': 1800, 'Deaths': 30}
df_corona.loc[len(df_corona)] = new_row_data
print(df_corona)
```

```
    ID    State  Cases  Recovery  Deaths
0  100    Delhi   3000      2500      50
1  110   Mumbai   4000      3500     100
2  120  Chennai   5000      4000     120
3  130    Surat   4500      3800      80
4  140  Kolkata   2000      1800      30
```

(d) To add a new column "Percentage" using the insert() method to store the percentage of recovery in every state (assume values). The column should be added as the fourth column in the dataframe.

```
percentage_values = (df_corona['Recovery'] / df_corona['Cases']) * 100
df_corona.insert(loc=3, column='Percentage', value=percentage_values)
print(df_corona)
```

```
    ID    State  Cases  Percentage  Recovery  Deaths
0  100    Delhi   3000   83.333333      2500      50
1  110   Mumbai   4000   87.500000      3500     100
2  120  Chennai   5000   80.000000      4000     120
3  130    Surat   4500   84.444444      3800      80
4  140  Kolkata   2000   90.000000      1800      30
```

(e) To delete the column "Percentage" using del command.

```
del df_corona['Percentage']
print(df_corona)
```

```
    ID    State  Cases  Recovery  Deaths
0  100    Delhi   3000      2500      50
1  110   Mumbai   4000      3500     100
2  120  Chennai   5000      4000     120
3  130    Surat   4500      3800      80
4  140  Kolkata   2000      1800      30
```

(f) To delete the column "Deaths" using pop() method.

```
del df_corona['Deaths']
print(df_corona)
```

```
    ID    State  Cases  Recovery
0  100    Delhi   3000      2500
1  110   Mumbai   4000      3500
2  120  Chennai   5000      4000
3  130    Surat   4500      3800
4  140  Kolkata   2000      1800
```

(g) To insert a new row of values using iloc[] at the 1st position.

```
df_corona.iloc[1:] = df_corona.iloc[0:-1].values
df_corona.iloc[0] = [150, 'Bangalore', 6000, 5000]
print(df_corona)
```

```
    ID      State  Cases  Recovery
0  150  Bangalore   6000      5000
1  100      Delhi   3000      2500
2  110     Mumbai   4000      3500
```

```
3  120    Chennai   5000      4000
4  130      Surat   4500      3800
```

(h) To delete Cases and State temporarily from the dataframe.

```
print(df_corona.drop(columns=['Cases', 'State'],inplace=False), "\n")
print(df_corona)
```

```
    ID  Recovery
0  150      5000
1  100      2500
2  110      3500
3  120      4000
4  130      3800

    ID      State  Cases  Recovery
0  150  Bangalore   6000      5000
1  100      Delhi   3000      2500
2  110     Mumbai   4000      3500
3  120    Chennai   5000      4000
4  130      Surat   4500      3800
```

Create a dataframe from two series-Name and Grade, Name and Marks of five students.

```
import pandas as pd
grades = pd.Series(data=['A', 'B', 'C', 'A', 'B'],index=['Alice',
'Bob', 'Charlie', 'David', 'Eve'])
marks = pd.Series(data=[90, 85, 70, 92, 88],index=['Alice', 'Bob',
'Charlie', 'David', 'Eve'])
student_data = {'Grade': grades,'Marks': marks}
student_df = pd.DataFrame(student_data)
print(student_df)
```

```
        Grade  Marks
Alice       A     90
Bob         B     85
Charlie     C     70
David       A     92
Eve         B     88
```

(a) Display the first three records from student dataframe.

```
print(student_df.head(3))
```

```
        Grade  Marks
Alice       A     90
Bob         B     85
Charlie     C     70
```

(b) Display the last two records from student dataframe.

```
print(student_df.tail(2))
```

```
      Grade  Marks
David     A     92
Eve       B     88
```

Create a dataframe of dictionary consisting of Name, Sub1, Sub2, Sub3, Sub4, Sub5 of five students.

```
import pandas as pd
student_dict_data = {'Name': ['Alice', 'Bob', 'Charlie', 'David',
'Eve'],
                     'Sub1': [85, 78, 92, 65, 70],
                     'Sub2': [90, 80, 88, 75, 68],
                     'Sub3': [75, 82, 95, 70, 72],
                     'Sub4': [92, 70, 80, 88, 75],
                     'Sub5': [88, 75, 85, 90, 60]}
student_grades_df = pd.DataFrame(student_dict_data)
```

(a) Display the dataframe.

```
print(student_grades_df)
```

```
      Name  Sub1  Sub2  Sub3  Sub4  Sub5
0    Alice    85    90    75    92    88
1      Bob    78    80    82    70    75
2  Charlie    92    88    95    80    85
3    David    65    75    70    88    90
4      Eve    70    68    72    75    60
```

(b) Display the first 5 rows and bottom 3 rows of student dataframe.

```
print("\nFirst 5 rows of Student Grades DataFrame:")
print(student_grades_df.head(5))
print("\nBottom 3 rows of Student Grades DataFrame:")
print(student_grades_df.tail(3))
```

```

First 5 rows of Student Grades DataFrame:
      Name  Sub1  Sub2  Sub3  Sub4  Sub5
0    Alice    85    90    75    92    88
1      Bob    78    80    82    70    75
2  Charlie    92    88    95    80    85
3    David    65    75    70    88    90
4      Eve    70    68    72    75    60

Bottom 3 rows of Student Grades DataFrame:
```

```
      Name  Sub1  Sub2  Sub3  Sub4  Sub5
2  Charlie    92    88    95    80    85
3    David    65    75    70    88    90
4      Eve    70    68    72    75    60
```

Create two dataframes of salary of five employees and do the following:

```
import pandas as pd
salary_data1 = {'EmployeeID': ["Ram", "Shyam", "Mohan", "Sita",
"Gita"],
                'Salary': [50000, 60000, 55000, 70000, 62000]}
df_salary1 = pd.DataFrame(salary_data1)

salary_data2 = {'EmployeeID': ["Anil", "Ravi", "Kiran", "Pooja",
"Neha"],
                'Salary': [45000, 68000, 52000, 75000, 60000]}
df_salary2 = pd.DataFrame(salary_data2)
```

(a) Display both the dataframes.

```
print("\nFirst Salary DataFrame:")
print(df_salary1)
print("\nSecond Salary DataFrame:")
print(df_salary2)
```

```
First Salary DataFrame:
  EmployeeID  Salary
0        Ram   50000
1      Shyam   60000
2      Mohan   55000
3       Sita   70000
4       Gita   62000

Second Salary DataFrame:
  EmployeeID  Salary
0       Anil   45000
1       Ravi   68000
2      Kiran   52000
3      Pooja   75000
4       Neha   60000
```

(b) Add 5000 as bonus in both dataframes and display them.

```
df_salary1['Salary'] = df_salary1['Salary'] + 5000
df_salary2['Salary'] = df_salary2['Salary'] + 5000

print("\n(b) First Salary DataFrame after adding bonus:")
print(df_salary1)
```

```
print("\nSecond Salary DataFrame after adding bonus:")
print(df_salary2)
```

```
(b) First Salary DataFrame after adding bonus:
  EmployeeID  Salary
0        Ram   55000
1      Shyam   65000
2      Mohan   60000
3       Sita   75000
4       Gita   67000

Second Salary DataFrame after adding bonus:
  EmployeeID  Salary
0       Anil   50000
1       Ravi   73000
2      Kiran   57000
3      Pooja   80000
4       Neha   65000
```

Create a dataframe using list [10, 11, 12, 13, 14] [23,34,45,32,65] [55,60,65,70,75] and do the following:

```
import pandas as pd
data_list = [[10, 11, 12, 13, 14],
             [23, 34, 45, 32, 65],
             [55, 60, 65, 70, 75]]
df_list = pd.DataFrame(data_list)
```

(a) Display the dataframe.

```
print(df_list)
```

```
    0   1   2   3   4
0  10  11  12  13  14
1  23  34  45  32  65
2  55  60  65  70  75
```

(b) Add the list [1, 2, 3, 4, 5] to dataframe and display it.

```
df_list.loc[len(df_list)]=[1, 2, 3, 4, 5]
print(df_list)
```

```
    0   1   2   3   4
0  10  11  12  13  14
1  23  34  45  32  65
2  55  60  65  70  75
3   1   2   3   4   5
```

Create a dataframe of [23, 25], [34], [43,44,45,46] and do the following:

```
import pandas as pd
import numpy as np
data_nan = [[23, 25],[34],[43, 44, 45, 46]]
df_nan = pd.DataFrame(data_nan)
```

(a) Display the dataframe. Notice that the missing value is represented by NaN.

```
print(df_nan)
```

```
    0     1     2     3
0  23  25.0   NaN   NaN
1  34   NaN   NaN   NaN
2  43  44.0  45.0  46.0
```

(b) Replace the missing value with 0.

```
df_zero = df_nan.fillna(0)
print("\n(b) DataFrame after replacing missing values with 0:")
print(df_zero)
```

```
(b) DataFrame after replacing missing values with 0:
    0     1     2     3
0  23  25.0   0.0   0.0
1  34   0.0   0.0   0.0
2  43  44.0  45.0  46.0
```

(c) Replace the missing value with -1, -2, -3, -4 for columns 0, 1, 2, 3.

```
fill_values = {0: -1, 1: -2, 2: -3, 3: -4}
df = df_nan.fillna(value=fill_values)
print(df)
```

```
    0     1     2     3
0  23  25.0  -3.0  -4.0
1  34  -2.0  -3.0  -4.0
2  43  44.0  45.0  46.0
```

(d) Replace the missing value by copying the value from the above cell.

```
#IDK
```