RAMNIRANJAN JHUNJHUNWALA COLLEGE GHATKOPAR (W),
MUMBAI - 400 086

DEPARTMENT OF INFORMATION TECHNOLOGY

2024 - 2025

MSC (IT) PART- I SEM- I
RJSPIT103
INTRODUCTION TO DATA SCIENCE

NAME : SUDESH DINESH RAJBHAR
ROLL NO. :6623

Hindi Vidya Prachar Samiti's

Ramniranjan Jhunjhunwala College of Arts, Science & Commerce

(Empowered Autonomous College)

Affiliated to

UNIVERSITY OF MUMBAI

*This is to certify that Mr. **Rajbhar Sudesh Dinesh SushilaDevi**, Roll No. **6623** of MSc. IT Part-1 class has completed the required number of Experiments of Practical **Introduction To Data Science**, in partial fulfilment of the Requirements for the award of the degree of Bachelor of Science (**Information Technology**) during the academic year 2024- 2025.*

Prof. Bharati Bhole

College seal

Sign of Co-Ordinator

INDEX

Sr. No.	Details	Date
1.	NumPy, Pandas, Matplotlib and Seaborn Basics.	Jul 15, 2024
2.	Collecting and loading structured and unstructured data.	Aug 6, 2024
3.	Using Data Wrangling processes: Data discovery, data pre-processing, data validation etc. for various types of data.	Aug 10, 2024
4.	Basic utility design, Data auditing and Exploratory Data Analysis.	Aug 13, 2024
5.	Retrieve Superstep.	Aug 13, 2024
6.	Access Superstep.	Aug 20, 2024
7.	Processing Data.	Aug 20, 2024
8.	Data Visualization.	Aug 20, 2024
9.	Data Analysis using Excel	Sep 3, 2024

Practical 1 - NumPy, Pandas, Matplotlib and Seaborn Basics

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of non-negative integers. In NumPy dimensions are called axes.

NumPy's array class is called ndarray. It is also known by the alias array. Note that numpy.array is not the same as the Standard Python Library class array.array, which only handles one-dimensional arrays and offers less functionality. The more important attributes of an ndarray object are:

ndarray.ndim

the number of axes (dimensions) of the array.

ndarray.shape

the dimensions of the array. This is a tuple of integers indicating the size of the array in each dimension. For a matrix with n rows and m columns, shape will be (n,m). The length of the shape tuple is therefore the number of axes, ndim.

ndarray.size

the total number of elements of the array. This is equal to the product of the elements of shape.

ndarray.dtype

an object describing the type of the elements in the array. One can create or specify dtype using standard Python types. Additionally NumPy provides types of its own. numpy.int32, numpy.int16, and numpy.float64 are some examples.

ndarray.itemsize

the size in bytes of each element of the array. For example, an array of elements of type float64 has itemsize 8 (=64/8), while one of type complex32 has itemsize 4 (=32/8). It is equivalent to ndarray.dtype.itemsize.

ndarray.data

the buffer containing the actual elements of the array. Normally, we won't need to use this attribute because we will access the elements in an array using indexing facilities.

```
import numpy as np
>>> a = np.arange(15).reshape(3, 5)
>>> a
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
>>> a.shape
(3, 5)
>>> a.ndim
2
>>> a.dtype.name
'int64'
>>> a.itemsize
8
>>> a.size
15
>>> type(a)
<class 'numpy.ndarray'>
>>> b = np.array([6, 7, 8])
>>> b
array([6, 7, 8])
>>> type(b)
<class 'numpy.ndarray'>
```

NUMPY

Basic Operations

```
[1]: import numpy as np

[2]: a = np.arange(15).reshape(3, 5)

[3]: a

[3]: array([[ 0,  1,  2,  3,  4],
            [ 5,  6,  7,  8,  9],
            [10, 11, 12, 13, 14]])

[4]: a.shape

[4]: (3, 5)

[5]: a.ndim

[5]: 2

[6]: a.dtype.name

[6]: 'int32'

[7]: a.itemsize

[7]: 4

[8]: type(a)

[8]: numpy.ndarray

[10]: b = np.array([6, 7, 8])
```

Array creation

```
In [11]: b = np.array([1,2,3])

In [12]: type(b)
Out[12]: numpy.ndarray

In [13]: b.ndim
Out[13]: 1

In [14]: b
Out[14]: array([1, 2, 3])
```

Changing the array dimensions

```
In [17]: d = c.reshape(4,3)
         d
Out[17]: array([[1, 2, 5],
                [8, 4, 5],
                [6, 9, 7],
                [8, 5, 6]])

In [18]: d.shape
Out[18]: (4, 3)

In [19]: d.ndim
Out[19]: 2
```

Array using tuple

```
In [20]: #Using a tuple to cretae a NUM Array
         e = np.array((7,8,5,'w',6))
         e
Out[20]: array(['7', '8', '5', 'w', '6'], dtype='<U11')

In [22]: e
Out[22]: array(['7', '8', '5', 'w', '6'], dtype='<U11')
```

2D array

```
In [25]: f = np.array([[1,2,3],[4,5,6]])
         f
Out[25]: array([[1, 2, 3],
                [4, 5, 6]])

In [26]: f.ndim
Out[26]: 2
```

3D array

```
In [27]: g = np.array([[[1,2,3],[4,5,6]],[[7,8,9],[10,11,12]]])
         g

Out[27]: array([[[ 1,  2,  3],
                 [ 4,  5,  6]],

                [[ 7,  8,  9],
                 [10, 11, 12]]])

In [28]: g.ndim

Out[28]: 3

In [30]: h = g.reshape(2,6)
         h

Out[30]: array([[ 1,  2,  3,  4,  5,  6],
                [ 7,  8,  9, 10, 11, 12]])

In [31]: h.ndim

Out[31]: 2
```

Accessing array elements

```
In [34]: print(h[1,5])

         12

In [35]: i = g[0,1,1] + g[1,1,1]
         i

Out[35]: 16
```

Multidimensional array

```
In [36]: j = np.array([[[10,20,30]],[[40,50,60]],[[10,15,60]],[[20,25,15]]])
         j

Out[36]: array([[[10, 20, 30]],

                [[40, 50, 60]],

                [[10, 15, 60]],

                [[20, 25, 15]]])
```

Slicing an array

```
In [38]: # Slicing An Array

         k = np.array([1,2,3,4,5,6,7])
         k
Out[38]: array([1, 2, 3, 4, 5, 6, 7])

In [39]: k[1:5]
Out[39]: array([2, 3, 4, 5])

In [40]: k[0:6]
Out[40]: array([1, 2, 3, 4, 5, 6])

In [41]: k[1:]
Out[41]: array([2, 3, 4, 5, 6, 7])

In [42]: k[:5]
Out[42]: array([1, 2, 3, 4, 5])

In [43]: k[1:5:2]
Out[43]: array([2, 4])

In [44]: k[:4:2]
Out[44]: array([1, 3])

In [45]: k[2::2]
Out[45]: array([3, 5, 7])

In [46]: k[::3]
Out[46]: array([1, 4, 7])
```

Creating identity and random matrix

```
In [53]: np.ones((3,3))
Out[53]: array([[1., 1., 1.],
                [1., 1., 1.],
                [1., 1., 1.]])

In [54]: # 3 set 2 row 4 column
         np.ones((3,2,4))
Out[54]: array([[[1., 1., 1., 1.],
                 [1., 1., 1., 1.]],

                [[1., 1., 1., 1.],
                 [1., 1., 1., 1.]],

                [[1., 1., 1., 1.],
                 [1., 1., 1., 1.]]])

In [55]: np.empty((2,3))
Out[55]: array([[1.24038692e-311, 1.52768399e+301, 1.24038694e-311],
                [1.24041184e-311, 6.56971658e+047, 1.24038693e-311]])

In [58]: np.arange(10,30,5)
Out[58]: array([10, 15, 20, 25])

In [59]: # random value from 0 to 2 9 value
         np.linspace(0,2,9)
Out[59]: array([0.  , 0.25, 0.5 , 0.75, 1.  , 1.25, 1.5 , 1.75, 2.  ])
```

Array in Trigonometric

```
In [62]: from numpy import pi
         x = np.linspace(0,2*pi,100)
         y = np.sin(x)

In [63]: y

Out[63]: array([ 0.00000000e+00,  6.34239197e-02,  1.26592454e-01,  1.89251244e-01,
                 2.51147987e-01,  3.12033446e-01,  3.71662456e-01,  4.29794912e-01,
                 4.86196736e-01,  5.40640817e-01,  5.92907929e-01,  6.42787610e-01,
                 6.90079011e-01,  7.34591709e-01,  7.76146464e-01,  8.14575952e-01,
                 8.49725430e-01,  8.81453363e-01,  9.09631995e-01,  9.34147860e-01,
                 9.54902241e-01,  9.71811568e-01,  9.84807753e-01,  9.93838464e-01,
                 9.98867339e-01,  9.99874128e-01,  9.96854776e-01,  9.89821442e-01,
                 9.78802446e-01,  9.63842159e-01,  9.45000819e-01,  9.22354294e-01,
                 8.95993774e-01,  8.66025404e-01,  8.32569855e-01,  7.95761841e-01,
                 7.55749574e-01,  7.12694171e-01,  6.66769001e-01,  6.18158986e-01,
                 5.67059864e-01,  5.13677392e-01,  4.58226522e-01,  4.00930535e-01,
                 3.42020143e-01,  2.81732557e-01,  2.20310533e-01,  1.58001396e-01,
                 9.50560433e-02,  3.17279335e-02, -3.17279335e-02, -9.50560433e-02,
                -1.58001396e-01, -2.20310533e-01, -2.81732557e-01, -3.42020143e-01,
                -4.00930535e-01, -4.58226522e-01, -5.13677392e-01, -5.67059864e-01,
                -6.18158986e-01, -6.66769001e-01, -7.12694171e-01, -7.55749574e-01,
                -7.95761841e-01, -8.32569855e-01, -8.66025404e-01, -8.95993774e-01,
                -9.22354294e-01, -9.45000819e-01, -9.63842159e-01, -9.78802446e-01,
                -9.89821442e-01, -9.96854776e-01, -9.99874128e-01, -9.98867339e-01,
                -9.93838464e-01, -9.84807753e-01, -9.71811568e-01, -9.54902241e-01,
                -9.34147860e-01, -9.09631995e-01, -8.81453363e-01, -8.49725430e-01,
                -8.14575952e-01, -7.76146464e-01, -7.34591709e-01, -6.90079011e-01,
                -6.42787610e-01, -5.92907929e-01, -5.40640817e-01, -4.86196736e-01,
                -4.29794912e-01, -3.71662456e-01, -3.12033446e-01, -2.51147987e-01,
                -1.89251244e-01, -1.26592454e-01, -6.34239197e-02, -2.44929360e-16])
```

For more details :- https://numpy.org/doc/stable/user/quickstart.html

Python

Format String:

```
temp = "{0:s} got {1:.2f}% in class {2:s} last time."

temp.format("Sudesh",99,"TYBSCIT")

'Sudesh got 99.00% in class TYBSCIT last time.'
```

Write a python code to store the given values in tuples and display it in the given format by using user defined function printString.

```
tuple1 =("Amit",45,'TY')
tuple2 =("Sumit",25,'FY')
tuple3 =("Anita",65,'SY')

def printString(str, tuple):
    print(str.format(tuple[0],tuple[1],tuple[2]))

listTuple=[tuple1,tuple2,tuple3]

for item in listTuple:
    printString(temp, item)
```

```
tuple1 =("Amit",45,'TY')
tuple2 =("Sumit",25,'FY')
tuple3 =("Anita",65,'SY')

def printString(str, tuple):
    print(str.format(tuple[0],tuple[1],tuple[2]))

listTuple=[tuple1,tuple2,tuple3]

for item in listTuple:
    printString(temp, item)
```

```
Amit has 45.00% in class TY.
Sumit has 25.00% in class FY.
Anita has 65.00% in class SY.
```

```
def printString(template_str, tuple):
    print(template_str.format(tuple[0], tuple[1], tuple[2]))

listTuple = [
    ("Amit", 45, 'TY'),
    ("Sumit", 25, 'FY'),
    ("Anita", 65, 'SY')
]

temp = "My Name is  {} and i am of  Age {} studing in Grade {}"

for item in listTuple:
    printString(temp, item)
```

```
def printString(template_str, tuple):
    print(template_str.format(tuple[0], tuple[1], tuple[2]))

listTuple = [
    ("Amit", 45, 'TY'),
    ("Sumit", 25, 'FY'),
    ("Anita", 65, 'SY')
]

temp = "My Name is  {} and i am of  Age {} studing in Grade {}"

for item in listTuple:
    printString(temp, item)
```

```
My Name is  Amit and i am of  Age 45 studing in Grade TY
My Name is  Sumit and i am of  Age 25 studing in Grade FY
My Name is  Anita and i am of  Age 65 studing in Grade SY
```

Bytes and Unicode:

```
# Encoding a Unicode string to bytes (UTF-8)
encoded_bytes = "My name is Sudesh".encode('utf-8')
print(encoded_bytes)

# Decoding bytes back to Unicode string
decoded_string = encoded_bytes.decode('utf-8')
print(decoded_string)
```

```
# Encoding a Unicode string to bytes (UTF-8)
encoded_bytes = "My name is Sudesh".encode('utf-8')
print(encoded_bytes)

# Decoding bytes back to Unicode string
decoded_string = encoded_bytes.decode('utf-8')
print(decoded_string)
```

```
b'My name is Sudesh'
My name is Sudesh
```

Typecasting:

```
# Convert an integer to a float
int_num = 42
float_num = float(int_num)
print(float_num)

# Convert a string to a float
str_num = "3.14"
float_str = float(str_num)
print(float_str)
```

```
42.0
3.14
```

Range:

```
list(range(5))
```

```
[0, 1, 2, 3, 4]
```

```
list(range(5,0,-1))
```

```
[5, 4, 3, 2, 1]
```

Write a code to store and display the squares of odd numbers. Generate odd numbers using range:

```
list=[]

for i in range(1,10,2):
    list.append(i*i)

print(list)
```

```
list=[]

for i in range(1,10,2):
    list.append(i*i)

print(list)
```

```
[1, 9, 25, 49, 81]
```

Write a program to create python dictionary from sequences:

```
keyList= ['fy','sy','ty']
value =[[1,2,3],[4,5,6],[7,8,9]]


mapping = {}
for key,value in zip(keyList,value):
     mapping[key] =value
print(mapping)
```

```
{'fy': [1, 2, 3], 'sy': [4, 5, 6], 'ty': [7, 8, 9]}
```

Display the list of keys of resultant

```
print(mapping.keys())
```

```
dict_keys(['fy', 'sy', 'ty'])
```

Display the list of values of resultant

```
print(mapping.values())
```

```
dict_values([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
```

Change values of ty to 11,21,31

```
mapping['ty'] ={11,21,31}
print(mapping)
```

```
{'fy': [1, 2, 3], 'sy': [4, 5, 6], 'ty': {11, 21, 31}}
```

Get the value of sy using pop.

```
mapping.pop('sy')
print(mapping)
```

```
{'fy': [1, 2, 3], 'ty': {11, 21, 31}}
```

Sort the values pair into a dictionary to alphabetise a list of tuples for the following values.

```
words = ['apple', 'banana', 'blueberry', 'apricot', 'orange', 'pineapple']

by_letter = {}
for word in words:
    letter = word[0]
    if letter not in by_letter:
        by_letter[letter] = [word]
    else:
        by_letter[letter].append(word)

print(by_letter)
```

```python
words = ['apple', 'banana', 'blueberry', 'apricot', 'orange', 'pineapple']

by_letter = {}
for word in words:
    letter = word[0]
    if letter not in by_letter:
        by_letter[letter] = [word]
    else:
        by_letter[letter].append(word)

print(by_letter)
```

```
{'a': ['apple', 'apricot'], 'b': ['banana', 'blueberry'], 'o': ['orange'], 'p': ['pineapple']}
```

WRite code to perform add,clear,remove,pop,union,update,intertersection etc,

```
a = {1, 2, 3, 4, 5}
b = {3, 4, 5, 6, 7, 8}

a.add(6)
print("After adding 6 to set a:", a)

a.clear()
print("After clearing set a:", a)

b.remove(4)
print("removing 3 ", b)

element = b.pop()
print("Popped ", element, "after pop:", b)

union_set = a.union(b)
print("Union of sets a and b:", union_set)

a.update(b)
print("Updated set a with union of itself and b:", a)

intersection_set = a.intersection(b)
print("Intersection:", intersection_set)

a.intersection_update(b)
print("Updated  intersection ", a)
```

```
After adding 6 to set a: {1, 2, 3, 4, 5, 6}
After clearing set a: set()
removing 3  {3, 5, 6, 7, 8}
Popped  3 after pop: {5, 6, 7, 8}
Union of sets a and b: {8, 5, 6, 7}
Updated set a with union of itself and b: {8, 5, 6, 7}
Intersection: {8, 5, 6, 7}
Updated  intersection  {8, 5, 6, 7}
```

Add the numbers above 10.

```
a= [8,9,11,23,34,10]
sum=0
for i in a:
    if i>10:
        sum=sum+i
print(sum)
```

```
68
```

```
list1 = [8, 9, 11, 23, 34, 10]
s = ([i for i in list1 if i > 10])
print(sum(s))
```

```
68
```

For the given list of strings store the length of each string in a set.
‘Bharati’,’Asha’,’Easy’,’Code’

```
string= ['Bharati','Asha','Easy','Code']

set(len(item) for item in string)
```

```
{4, 7}
```

Print square and cubes of numbers using a user defined function calculate

```
def calculate (numList):
    squares = [num ** 2 for num in numList]
    cubes = [num ** 3 for num in numList]
    return squares,cubes

numList = [11,22,33]
squares,cubes = calculate(numList)
print("Sqaure",squares)
print("Cube",cubes)
```

```
Sqaure [121, 484, 1089]
Cube [1331, 10648, 35937]
```

```
import re

names = ['Anita12', 'Vishakha#', 'Ragini!', 'Venuka##', 'Rupa.', 'Rani    Yadav']

def remove_punctuation(value):
    return re.sub('[ .!#?]', '', value)

result = [remove_punctuation(name) for name in names]
print(result)
```

```
['Anita12', 'Vishakha', 'Ragini', 'Venuka', 'Rupa', 'RaniYadav']
```

```
# Define the coefficients and the value of x
a = 1
b = -3
c = 2
x = 4

# Lambda function to calculate the quadratic equation value
quadratic_value = lambda a, b, c, x: a*x**2 + b*x + c

# Calculate and print the value
result = quadratic_value(a, b, c, x)
print(f"The value of the quadratic equation for x={x} is: {result}")
```

```
The value of the quadratic equation for x=4 is: 6
```

PANDAS

Series

import pandas as pd

name = pd.Series(['Kiran', 'Nitin', 'Kunal', 'Anjani', 'Amol', 'Ahmed'])

name

```
[5]: import pandas as pd
     name = pd.Series(['Kiran', 'Nitin', 'Kunal', 'Anjani', 'Amol', 'Ahmed'])
     name

[5]: 0     Kiran
     1     Nitin
     2     Kunal
     3    Anjani
     4      Amol
     5     Ahmed
     dtype: object
```

DataFrame

import pandas as pd

```
data = {
    'roll_no': [6623, 6621, 6622, 6624, 6625, 6626, 6627, 6628],
    'name': ['sudesh', 'shivam', 'keshav', 'chris', 'thor', 'ironman', 'captain', 'spiderman'],
    'class': ['MSC-IT', 'MSC-IT', 'MSC-IT', 'MSC-IT', 'MSC-IT', 'MSC-IT', 'MSC-IT', 'MSC-IT']
}
```

dataframe = pd.DataFrame(data)

dataframe

```python
import pandas as pd

data = {
    'roll_no': [6623, 6621, 6622, 6624, 6625, 6626, 6627, 6628],
    'name': ['sudesh', 'shivam', 'keshav', 'chris', 'thor', 'ironman', 'captain', 'spiderman'],
    'class': ['MSC-IT', 'MSC-IT', 'MSC-IT', 'MSC-IT', 'MSC-IT', 'MSC-IT', 'MSC-IT', 'MSC-IT']
}

dataframe = pd.DataFrame(data)

dataframe
```

	roll_no	name	class
0	6623	sudesh	MSC-IT
1	6621	shivam	MSC-IT
2	6622	keshav	MSC-IT
3	6624	chris	MSC-IT
4	6625	thor	MSC-IT
5	6626	ironman	MSC-IT
6	6627	captain	MSC-IT
7	6628	spiderman	MSC-IT

 # Displaying the first 5 rows of the DataFrame
dataframe.head()

	roll_no	name	class
0	6623	sudesh	MSC-IT
1	6621	shivam	MSC-IT
2	6622	keshav	MSC-IT
3	6624	chris	MSC-IT
4	6625	thor	MSC-IT

 # Displaying the last 5 rows of the DataFrame
dataframe.tail()

	roll_no	name	class
3	6624	chris	MSC-IT
4	6625	thor	MSC-IT
5	6626	ironman	MSC-IT
6	6627	captain	MSC-IT
7	6628	spiderman	MSC-IT

 # Generating descriptive statistics for numerical columns
dataframe.describe()

	roll_no
count	8.00000
mean	6624.50000
std	2.44949
min	6621.00000
25%	6622.75000
50%	6624.50000
75%	6626.25000
max	6628.00000

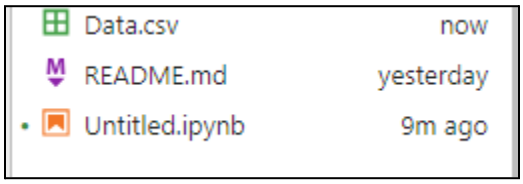
 # Displaying data types of each column
dataframe.dtypes

```
roll_no     int64
name       object
class      object
dtype: object
```

 # Displaying the index of the DataFrame
dataframe.index

```
df.index
```

```
RangeIndex(start=0, stop=8, step=1)
```



	StudID	Name	Percentage	Class	Mail
0	1	Sudesh Rajbhar	55	TYBSCIT	SUDESH@GMAIL.COM
1	2	Devesh RAJBHAR	55	FYBSCIT	DEVESH@GMAIL.COM
2	3	Vivek YADAV	75	TYBSCIT	VIVEK@HOTMAIL.COM
3	4	SWANAND SABALE	63	FYBSCIT	SABALE@TATAAIG.COM
4	5	prem Chopra	NAN	TYBSCIT	PREM@WAKODE.COM
5	6	tAnmay bHaT	45	SYBSCIT	TANMAY@GMAIL.COM

import pandas as pd

# Reading the CSV file into a DataFrame
df = pd.read_csv("data.csv")

Displaying the DataFrame
df

print(df.head()) # First 5 rows

	StudID	Name	Percentage	Class	Mail
0	1	Sudesh Rajbhar	55	TYBSCIT	SUDESH@GMAIL.COM
1	2	Devesh RAJBHAR	55	FYBSCIT	DEVESH@GMAIL.COM
2	3	Vivek YADAV	75	TYBSCIT	VIVEK@HOTMAIL.COM
3	4	SWANAND SABALE	63	FYBSCIT	SABALE@TATAAIG.COM
4	5	prem Chopra	NAN	TYBSCIT	PREM@WAKODE.COM

print(df.tail()) # Last 5 rows

	StudID	Name	Percentage	Class	Mail
1	2	Devesh RAJBHAR	55	FYBSCIT	DEVESH@GMAIL.COM
2	3	Vivek YADAV	75	TYBSCIT	VIVEK@HOTMAIL.COM
3	4	SWANAND SABALE	63	FYBSCIT	SABALE@TATAAIG.COM
4	5	prem Chopra	NAN	TYBSCIT	PREM@WAKODE.COM
5	6	tAnmay bHaT	45	SYBSCIT	TANMAY@GMAIL.COM

print(df.columns) # Column names

```
Index(['StudID', 'Name', 'Percentage', 'Class', 'Mail'], dtype='object')
RangeIndex(start=0, stop=6, step=1)
```

print(df.index) # Index

```
Index(['StudID', 'Name', 'Percentage', 'Class', 'Mail'], dtype='object')
RangeIndex(start=0, stop=6, step=1)
```

print(df.values) # Data as a numpy array

```
[[1 'Sudesh Rajbhar' '55' 'TYBSCIT' 'SUDESH@GMAIL.COM']
 [2 'Devesh RAJBHAR' '55' 'FYBSCIT' 'DEVESH@GMAIL.COM']
 [3 'Vivek YADAV' '75' 'TYBSCIT' 'VIVEK@HOTMAIL.COM']
 [4 'SWANAND SABALE' '63' 'FYBSCIT' 'SABALE@TATAAIG.COM']
 [5 'prem Chopra' 'NAN' 'TYBSCIT' 'PREM@WAKODE.COM']
 [6 'tAnmay bHaT' '45' 'SYBSCIT' 'TANMAY@GMAIL.COM']]
```

print(df.describe()) # Descriptive statistics

	StudID
count	6.000000
mean	3.500000
std	1.870829
min	1.000000
25%	2.250000
50%	3.500000
75%	4.750000
max	6.000000

```
print(df.isna())         # Check for NaN values
print(df.isnull())       # Check for NaN values (similar to isna())
print(df.dtypes)          # Data types of each column
```

```
    StudID   Name  Percentage  Class   Mail
0   False  False       False  False  False
1   False  False       False  False  False
2   False  False       False  False  False
3   False  False       False  False  False
4   False  False       False  False  False
5   False  False       False  False  False
   StudID   Name  Percentage  Class   Mail
0   False  False       False  False  False
1   False  False       False  False  False
2   False  False       False  False  False
3   False  False       False  False  False
4   False  False       False  False  False
5   False  False       False  False  False
StudID         int64
Name          object
Percentage    object
Class         object
Mail          object
dtype: object
```

```
df[['Name', 'Class']].head(3)
```

	Name	Class
0	Sudesh Rajbhar	TYBSCIT
1	Devesh RAJBHAR	FYBSCIT
2	Vivek YADAV	TYBSCIT

Q. Create the following dataframe from the given data.
Dataframe:

	c1	c2	c3
row1	10	11	12
row2	13	14	15
row3	14	17	18

Given data:
10,11,12,13,14,15,14,17,18

```
data = {
    'c1' : [10, 13, 16],
    'c2' : [11, 14, 17],
    'c3' : [12, 15, 18],
}

data_df = pd.DataFrame(data, index=['row1', 'row2', 'row3'])
data_df
```

	c1	c2	c3
row1	10	11	12
row2	13	14	15
row3	16	17	18

Display the second row of the dataframe.

```
#  Display the second row of the dataframe.
data_df.iloc[1]
```

```
c1    13
c2    14
c3    15
Name: row2, dtype: int64
```

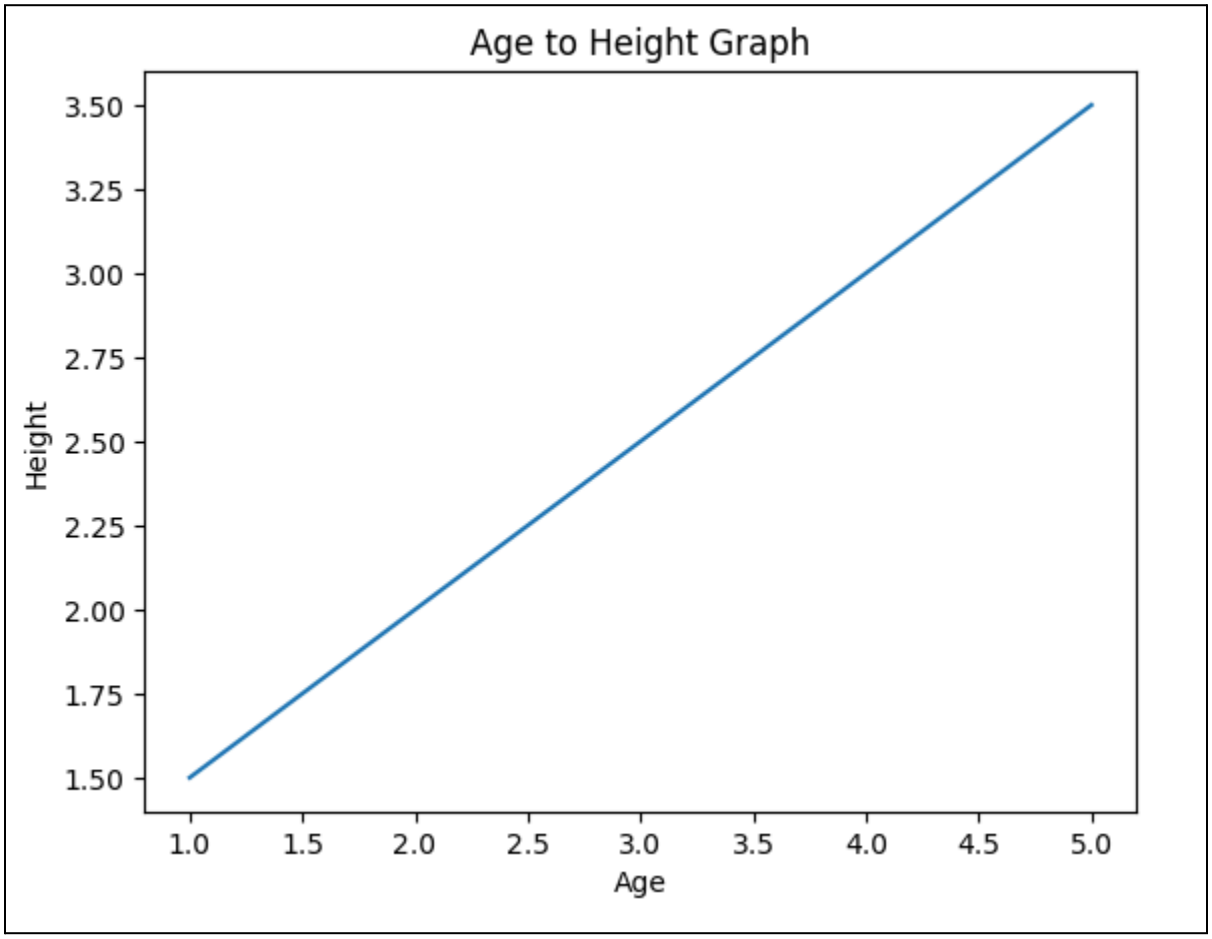
Store the first, second and third row of the dataframe in the variable r1, r2 and r3 and add each individual element of all rows.

```
r1 = data_df.loc['row1']
r2 = data_df.loc['row2']
r3 = data_df.loc['row3']

print("Row1:\n", r1)
print("Row2:\n", r2)
print("Row3:\n", r3)
```

```
Row1:
 c1    10
c2    11
c3    12
Name: row1, dtype: int64
Row2:
 c1    13
c2    14
c3    15
Name: row2, dtype: int64
Row3:
 c1    16
c2    17
c3    18
Name: row3, dtype: int64
```

MATPLOTLIB

```
import matplotlib.pyplot as plt

# Data
age = [1, 2, 3, 4, 5]
height = [1.5, 2, 2.5, 3, 3.5]

# Plotting
plt.plot(age, height)
plt.title('Age to Height Graph')
plt.xlabel('Age')
plt.ylabel('Height')
plt.show()
```



Practical 2 - Conversion of semi structured and unstructured data to csv.

CSVFile.csv

Column_1	Column_2	Column_3	Column_4
1	Sudesh Rajbhar	80	2000000
2	Iron Man	35	3000000
3	Spider Man	23	1500000
4	Captain America	45	2500000
5	Tony stark	56	3000000

Processing a csv file:

```
import pandas as pd
InputData=pd.read_csv("D:\\New folder\\MSCIT_6623\\ids\\CSVFile.csv")
print("Original Data :")
print(InputData)
print("\n")
ProcessData=InputData
ProcessData.rename(columns={'Column_1':'EmpID','Column_2':'EmpName','Column_3':'
EmpAge','Column_4':'AnnnualPackage'}, inplace=True)
print("Renaming Columns :")
print(ProcessData)
print("\n")
ProcessData.set_index('EmpID', inplace=True)
print("Setting index as EmpID :")
print(ProcessData)
print("\n")
ProcessData.sort_values('AnnnualPackage', axis=0, ascending=False, inplace=True)
print("Annual Package in descending order :")
print(ProcessData)
print("\n")
ProcessData.drop('EmpAge', axis=1, inplace=True)
print("After deleting EmpAge column :")
print(ProcessData)
print("\n")
```

```
#to save your file
ProcessData.to_csv("D:\\New folder\\MSCIT_6623\\ids\\CSVFile.csv")
#sOutputFileName='FinalEmp.csv'
#OutputData.to_csv(sOutputFilaName, a)
```

Output:

```
Original Data :
   Column_1         Column_2  Column_3  Column_4
0         1   Sudesh Rajbhar        80   2000000
1         2         Iron Man        35   3000000
2         3       Spider Man        23   1500000
3         4  Captain America        45   2500000
4         5       Tony stark        56   3000000


Renaming Columns :
   EmpID          EmpName  EmpAge  AnnnualPackage
0      1   Sudesh Rajbhar      80         2000000
1      2         Iron Man      35         3000000
2      3       Spider Man      23         1500000
3      4  Captain America      45         2500000
4      5       Tony stark      56         3000000
```

```
Setting index as EmpID :
               EmpName  EmpAge  AnnnualPackage
EmpID
1       Sudesh Rajbhar      80         2000000
2             Iron Man      35         3000000
3           Spider Man      23         1500000
4      Captain America      45         2500000
5           Tony stark      56         3000000


Annual Package in descending order :
               EmpName  EmpAge  AnnnualPackage
EmpID
2             Iron Man      35         3000000
5           Tony stark      56         3000000
4      Captain America      45         2500000
1       Sudesh Rajbhar      80         2000000
3           Spider Man      23         1500000


After deleting EmpAge column :
               EmpName  AnnnualPackage
EmpID
2             Iron Man         3000000
5           Tony stark         3000000
4      Captain America         2500000
1       Sudesh Rajbhar         2000000
3           Spider Man         1500000
```

XML to CSV:

```
import pandas as pd
import xml.etree.ElementTree as ET

def xml2df(xml_data):
    root = ET.XML(xml_data)
    all_records = []
    for i, child in enumerate(root):
        record = {}
        for subchild in child:
            record[subchild.tag] = subchild.text
        all_records.append(record)
    return pd.DataFrame(all_records)

sInputFileName="D:\\New folder\\MSCIT_6623\\ids\\csv.xml"
InputData = open(sInputFileName).read()
print(InputData)
ProcessDataXML=InputData
ProcessData=xml2df(ProcessDataXML)
print(ProcessData)
ProcessData.rename(columns={'Column_1':'EmpID','Column_2':'EmpName','Column_3':'
EmpAge','Column_4':'AnnnualPackage'}, inplace=True)
print(ProcessData)
ProcessData.set_index('EmpID', inplace=True)
print(ProcessData)
ProcessData.sort_values('AnnnualPackage', axis=0, ascending=False, inplace=True)
print(ProcessData)
ProcessData.drop('EmpAge', axis=1, inplace=True)
print(ProcessData)
OutputData=ProcessData
sOutputFileName="D:\\New folder\\MSCIT_6623\\ids\\xml.xml"
OutputData.to_csv(sOutputFileName, index = False)
```

Output:

```
<Records>
    <Record>
        <Column_1>1</Column_1>
        <Column_2>Sudesh Rajbhar</Column_2>
        <Column_3>80</Column_3>
        <Column_4>2000000</Column_4>
    </Record>
    <Record>
        <Column_1>2</Column_1>
        <Column_2>Iron Man</Column_2>
        <Column_3>35</Column_3>
        <Column_4>3000000</Column_4>
    </Record>
    <Record>
        <Column_1>3</Column_1>
        <Column_2>Spider Man</Column_2>
        <Column_3>23</Column_3>
        <Column_4>1500000</Column_4>
    </Record>
    <Record>
        <Column_1>4</Column_1>
        <Column_2>Captain America</Column_2>
        <Column_3>45</Column_3>
        <Column_4>2500000</Column_4>
    </Record>
    <Record>
        <Column_1>5</Column_1>
        <Column_2>Tony Stark</Column_2>
        <Column_3>56</Column_3>
        <Column_4>3000000</Column_4>
    </Record>
</Records>
```

```
   Column_1         Column_2 Column_3 Column_4
0         1   Sudesh Rajbhar       80  2000000
1         2         Iron Man       35  3000000
2         3       Spider Man       23  1500000
3         4  Captain America       45  2500000
4         5       Tony Stark       56  3000000
  EmpID          EmpName EmpAge AnnnualPackage
0     1   Sudesh Rajbhar     80        2000000
1     2         Iron Man     35        3000000
2     3       Spider Man     23        1500000
3     4  Captain America     45        2500000
4     5       Tony Stark     56        3000000
              EmpName EmpAge AnnnualPackage
EmpID
1       Sudesh Rajbhar     80        2000000
2             Iron Man     35        3000000
3           Spider Man     23        1500000
4      Captain America     45        2500000
5           Tony Stark     56        3000000
              EmpName EmpAge AnnnualPackage
EmpID
2             Iron Man     35        3000000
5           Tony Stark     56        3000000
4      Captain America     45        2500000
1       Sudesh Rajbhar     80        2000000
3           Spider Man     23        1500000
              EmpName AnnnualPackage
EmpID
2             Iron Man        3000000
5           Tony Stark        3000000
4      Captain America        2500000
1       Sudesh Rajbhar        2000000
3           Spider Man        1500000
```

Json to csv :

```
import pandas as pd
sInputFileName="D:\\New folder\\MSCIT_6623\\ids\\csv.json"
InputData=pd.read_json(sInputFileName, encoding="latin-1")
print('Original Data :')
print(InputData)
print('\n')
ProcessData=InputData
ProcessData.rename(columns={'Column_1':'EmpID','Column_2':'EmpName','Column_3':'
EmpAge','Column_4':'AnnnualPackage'}, inplace=True)
print(ProcessData)
ProcessData.set_index('EmpID', inplace=True)
print(ProcessData)
ProcessData.sort_values('AnnnualPackage', axis=0, ascending=False, inplace=True)
print(ProcessData)
ProcessData.drop('EmpAge', axis=1, inplace=True)
print(ProcessData)
OutputData=ProcessData
sOutputFileName="D:\\New folder\\MSCIT_6623\\ids\\json.json"
OutputData.to_csv(sOutputFileName, index = False)
```

Output:

```
Original Data :
   Column_1         Column_2  Column_3  Column_4
0         1   Sudesh Rajbhar        80   2000000
1         2         Iron Man        35   3000000
2         3       Spider Man        23   1500000
3         4  Captain America        45   2500000
4         5       Tony Stark        56   3000000


   EmpID          EmpName  EmpAge  AnnnualPackage
0      1   Sudesh Rajbhar      80         2000000
1      2         Iron Man      35         3000000
2      3       Spider Man      23         1500000
3      4  Captain America      45         2500000
4      5       Tony Stark      56         3000000
               EmpName  EmpAge  AnnnualPackage
EmpID
1       Sudesh Rajbhar      80         2000000
2             Iron Man      35         3000000
3           Spider Man      23         1500000
4      Captain America      45         2500000
5           Tony Stark      56         3000000
               EmpName  EmpAge  AnnnualPackage
```

```
               EmpName  EmpAge  AnnnualPackage
EmpID
2             Iron Man      35         3000000
5           Tony Stark      56         3000000
4      Captain America      45         2500000
1       Sudesh Rajbhar      80         2000000
3           Spider Man      23         1500000
               EmpName  AnnnualPackage
EmpID
2             Iron Man         3000000
5           Tony Stark         3000000
4      Captain America         2500000
1       Sudesh Rajbhar         2000000
3           Spider Man         1500000
```

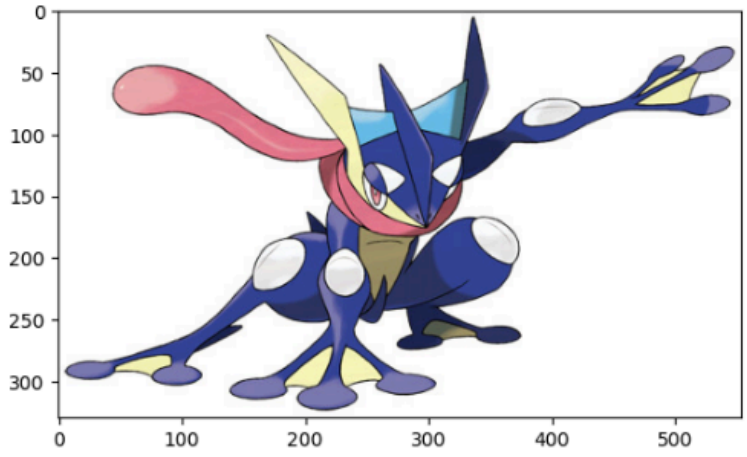
Image to csv:

```
import imageio.v3 as iio   #pip install imageio
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

sInputFileName = "D:\\New folder\\MSCIT_6623\\ids\\greninja.jpg"

# Read the image
InputData = iio.imread(sInputFileName)

# Print the dimensions of the image
print('X: ', InputData.shape[0])
print('Y: ', InputData.shape[1])

# Check if the image has 2 dimensions (grayscale) or 3 dimensions (RGB)
if len(InputData.shape) == 3:
    # RGB Image
    num_channels = InputData.shape[2]
else:
    # Grayscale Image (treat it as having 1 channel)
    num_channels = 1
    InputData = np.expand_dims(InputData, axis=-1)  # Add a channel dimension

ProcessRawData = InputData.flatten()
y = num_channels + 2
x = int(ProcessRawData.shape[0] / y)
ProcessData = pd.DataFrame(np.reshape(ProcessRawData, (x, y)))

# Adjust column names based on the number of channels
if num_channels == 3:
    sColumns = ['XAxis', 'YAxis', 'Red', 'Green', 'Blue']
elif num_channels == 4:
    sColumns = ['XAxis', 'YAxis', 'Red', 'Green', 'Blue', 'Alpha']
else:
    sColumns = ['XAxis', 'YAxis', 'Gray']

ProcessData.columns = sColumns
ProcessData.index.names = ['ID']

print('Rows: ', ProcessData.shape[0])
print('Columns :', ProcessData.shape[1])
print('\n')
print('Process Data Values :')
```

```
plt.imshow(InputData.squeeze())
plt.show()
print('\n')
OutputData = ProcessData
sOutputFileName = "D:\\New folder\\MSCIT_6623\\ids\\greninja.csv"
OutputData.to_csv(sOutputFileName, index=False)
```

Output:

```
X:  330
Y:  554
Rows:  109692
Columns : 5


Process Data Values :
```



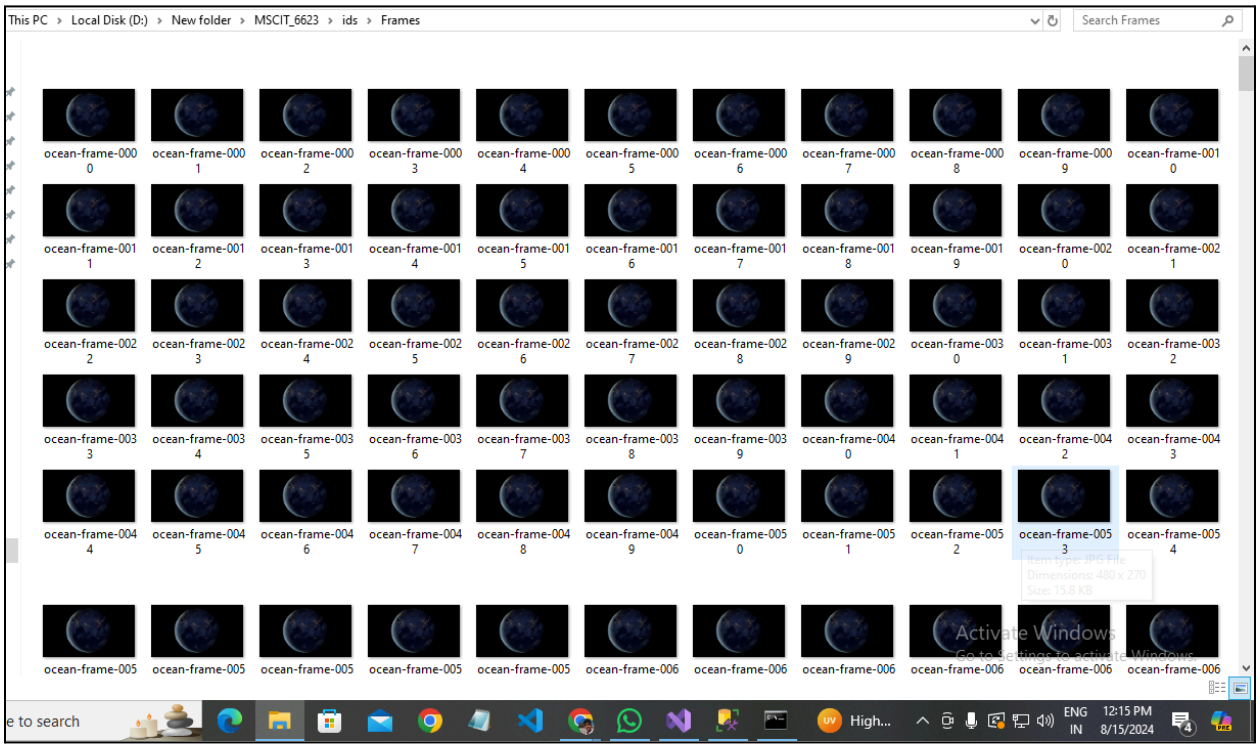
	A	B	C	D	E
1	XAxis	YAxis	Red	Green	Blue
2	255	255	255	255	255
3	255	255	255	255	255
4	255	255	255	255	255
5	255	255	255	255	255
6	255	255	255	255	255
7	255	255	255	255	255
8	255	255	255	255	255
9	255	255	255	255	255
10	255	255	255	255	255
11	255	255	255	255	255

Video to Frames:

```
import os
import shutil
import cv2    #pip install opencv-python

sInputFileName = "D:\\New folder\\MSCIT_6623\\ids\\sample.mp4"
sDataBaseDir = 'D:\\New folder\\MSCIT_6623\\ids\\Frames'
# Check if the input video file exists
if not os.path.exists(sInputFileName):
    print(f"Error: The file {sInputFileName} does not exist.")
    raise FileNotFoundError(f"{sInputFileName} not found.")

# Remove the directory if it exists and recreate it
if os.path.exists(sDataBaseDir):
    shutil.rmtree(sDataBaseDir)
os.makedirs(sDataBaseDir)
print('\nStart Movie to Frames\n')
# Open the video file
vidcap = cv2.VideoCapture(sInputFileName)
success, image = vidcap.read()
if not success:
    print("Error: Failed to read the video file.")
else:
    count = 0
    while success:
        sFrame = os.path.join(sDataBaseDir, f'ocean-frame-{count:04d}.jpg')
        print('Extracted:', sFrame)
        cv2.imwrite(sFrame, image)
        # Check if the frame was written successfully
        if os.path.getsize(sFrame) == 0:
            count -= 1
            os.remove(sFrame)
            print('Removed:', sFrame)
        # Read the next frame
        success, image = vidcap.read()
        count += 1
    print('Generated:', count, 'Frames')
print('\nMovie to Frames HORUS - Done')
```

Output:

```
Start Movie to Frames

Extracted: D:\New folder\MSCIT_6623\ids\Frames\ocean-frame-0000.jpg
Extracted: D:\New folder\MSCIT_6623\ids\Frames\ocean-frame-0001.jpg
Extracted: D:\New folder\MSCIT_6623\ids\Frames\ocean-frame-0002.jpg
Extracted: D:\New folder\MSCIT_6623\ids\Frames\ocean-frame-0003.jpg
Extracted: D:\New folder\MSCIT_6623\ids\Frames\ocean-frame-0004.jpg
Extracted: D:\New folder\MSCIT_6623\ids\Frames\ocean-frame-0005.jpg
Extracted: D:\New folder\MSCIT_6623\ids\Frames\ocean-frame-0006.jpg
Extracted: D:\New folder\MSCIT_6623\ids\Frames\ocean-frame-0007.jpg
Extracted: D:\New folder\MSCIT_6623\ids\Frames\ocean-frame-0008.jpg
Extracted: D:\New folder\MSCIT_6623\ids\Frames\ocean-frame-0009.jpg
Extracted: D:\New folder\MSCIT_6623\ids\Frames\ocean-frame-0010.jpg
Extracted: D:\New folder\MSCIT_6623\ids\Frames\ocean-frame-0011.jpg
Extracted: D:\New folder\MSCIT_6623\ids\Frames\ocean-frame-0012.jpg
Extracted: D:\New folder\MSCIT_6623\ids\Frames\ocean-frame-0013.jpg
Extracted: D:\New folder\MSCIT_6623\ids\Frames\ocean-frame-0014.jpg
```



Frame to CSV

```
import imageio.v2 as imageio
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import os

sDataBaseDir = 'D:\\New folder\\MSCIT_6623\\ids\\Frames'
f = 0
ProcessDataList = []  # To hold all DataFrames before concatenation

for file in os.listdir(sDataBaseDir):
    if file.endswith(".jpg"):
        f += 1
        sInputFileName = os.path.join(sDataBaseDir, file)
        print('Process : ', sInputFileName)
        InputData = imageio.imread(sInputFileName)
        print('Input Data Values :')
        print('X: ', InputData.shape[0])
        print('Y: ', InputData.shape[1])
        print('RGBA: ', InputData.shape[2])
        # Processing Rules ==========================================
        ProcessRawData = InputData.flatten()
        y = InputData.shape[2] + 2
        x = int(ProcessRawData.shape[0] / y)
        ProcessFrameData = pd.DataFrame(np.reshape(ProcessRawData, (x, y)))
        ProcessFrameData['Frame'] = file
        print('Number of columns in ProcessFrameData:',
len(ProcessFrameData.columns))
        print('\n')
        print('Process Data Values :')
        print('\n')
        plt.imshow(InputData)
        plt.show()

        ProcessDataList.append(ProcessFrameData)

if f > 0:
    # Check column count for the first DataFrame
    print('Columns in the first DataFrame:', ProcessDataList[0].shape[1])

    # Concatenate DataFrames
    ProcessData = pd.concat(ProcessDataList, ignore_index=True)
```

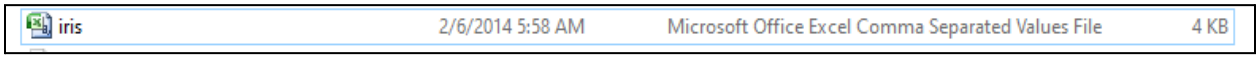
```
    # Print column count after concatenation
    print('Columns in concatenated DataFrame:', ProcessData.shape[1])

    # Ensure the column names match the number of columns
    sColumns = ['XAxis', 'YAxis', 'Red', 'Green', 'Blue', 'Alpha', 'FrameName']

    if ProcessData.shape[1] == len(sColumns):
        ProcessData.columns = sColumns
    else:
        print(f'Column count mismatch: DataFrame has {ProcessData.shape[1]}
columns but {len(sColumns)} names provided.')

    print('\n')
    ProcessData.index.names = ['ID']
    print('Rows: ', ProcessData.shape[0])
    print('Columns :', ProcessData.shape[1])
    print('\n')
    # Output Agreement ========================================
    OutputData = ProcessData
    sOutputFileName = "D:\\New folder\\MSCIT_6623\\ids\\sample.csv"
    OutputData.to_csv(sOutputFileName, index=False)

print('\n')
print('Processed ; ', f, ' frames')
```

Output:

```
Process :  D:\New folder\MSCIT_6623\ids\Frames\ocean-frame-0000.jpg
Input Data Values :
X:  270
Y:  480
RGBA:  3
Number of columns in ProcessFrameData: 6


Process Data Values :
```

Practical 3 - Data Processing for various types of data

Download iris.csv file



Importing

```
import pandas as pd
data = pd.read_csv("D:\\New folder\\MSCIT_6623\\ids\\iris.csv")
print("Original Data: ")
df=pd.DataFrame(data)
print(df)
```

```
[1]: import pandas as pd
     data = pd.read_csv("D:\\New folder\\MSCIT_6623\\ids\\iris.csv")
     print("Original Data: ")
     df=pd.DataFrame(data)
     print(df)
```

```
Original Data:
     sepal.length  sepal.width  petal.length  petal.width    variety
0             5.1          3.5           1.4          0.2     Setosa
1             4.9          3.0           1.4          0.2     Setosa
2             4.7          3.2           1.3          0.2     Setosa
3             4.6          3.1           1.5          0.2     Setosa
4             5.0          3.6           1.4          0.2     Setosa
..            ...          ...           ...          ...        ...
145           6.7          3.0           5.2          2.3  Virginica
146           6.3          2.5           5.0          1.9  Virginica
147           6.5          3.0           5.2          2.0  Virginica
148           6.2          3.4           5.4          2.3  Virginica
149           5.9          3.0           5.1          1.8  Virginica

[150 rows x 5 columns]
```

```
import numpy as np
#Information
print("Information : ")
print(df.info())
print("\n")
#Description
print("Description : ")
print(df.describe())
```

```
print("\n")
#Data types of each column
print("Data types of each column : ")
print(df.dtypes)
print("\n")
#Total number of records
print("Total number of records : ")
print(df.count())
print("\n")
#Total number of rows and columns
print("Total number of rows and columns : ")
print(df.shape)
print("\n")
#Column names
print("Column names : ")
print(df.columns)
print("\n")
#Column values
print("Column values : ")
print(df.values)
print("\n")
#Dimension of dataframe
print("Dimension of dataframe : ")
print(df.ndim)
print("\n")
#Size of dataframe
print("Size of dataframe : ")
print(df.size)
print("\n")
#Checking null/missing/blank values
print("Checking null/missing/blank values : ")
print(df.isnull())
print("\n")
#Checking null/missing/blank values
print("Checking null/missing/blank values : ")
print(df.notnull())
print("\n")
#Checking null/missing/blank values and fill with given values
print("Checking null/missing/blank values and fill with given values : ")
print(df.fillna(1))    #1 can be replaced by anything
print("\n")
#Replace the NaN values with given values
print("Replace the NaN values with given values : ")
print(df.replace(to_replace=np.nan, value=0))   #0 can be replaced by anything
```

Output:

```
Information :
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal.length  150 non-null    float64
 1   sepal.width   150 non-null    float64
 2   petal.length  150 non-null    float64
 3   petal.width   150 non-null    float64
 4   variety       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
```

```
Description :
       sepal.length  sepal.width  petal.length  petal.width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.057333      3.758000     1.199333
std        0.828066     0.435866      1.765298     0.762238
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
```

```
Data types of each column :
sepal.length    float64
sepal.width     float64
petal.length    float64
petal.width     float64
variety          object
dtype: object


Total number of records :
sepal.length    150
sepal.width     150
petal.length    150
petal.width     150
variety         150
dtype: int64
```

```
Total number of rows and columns :
(150, 5)


Column names :
Index(['sepal.length', 'sepal.width', 'petal.length', 'petal.width',
       'variety'],
      dtype='object')
```

```
Column values :
[[5.1 3.5 1.4 0.2 'Setosa']
 [4.9 3.0 1.4 0.2 'Setosa']
 [4.7 3.2 1.3 0.2 'Setosa']
 [4.6 3.1 1.5 0.2 'Setosa']
 [5.0 3.6 1.4 0.2 'Setosa']
 [5.4 3.9 1.7 0.4 'Setosa']
 [4.6 3.4 1.4 0.3 'Setosa']
 [5.0 3.4 1.5 0.2 'Setosa']
 [4.4 2.9 1.4 0.2 'Setosa']
 [4.9 3.1 1.5 0.1 'Setosa']
 [5.4 3.7 1.5 0.2 'Setosa']
 [4.8 3.4 1.6 0.2 'Setosa']
 [4.8 3.0 1.4 0.1 'Setosa']
```

```
Checking null/missing/blank values :
     sepal.length  sepal.width  petal.length  petal.width  variety
0           False        False         False        False    False
1           False        False         False        False    False
2           False        False         False        False    False
3           False        False         False        False    False
4           False        False         False        False    False
..            ...          ...           ...          ...      ...
145         False        False         False        False    False
146         False        False         False        False    False
147         False        False         False        False    False
148         False        False         False        False    False
149         False        False         False        False    False

[150 rows x 5 columns]
```

```
Checking null/missing/blank values :
     sepal.length  sepal.width  petal.length  petal.width  variety
0            True         True          True         True     True
1            True         True          True         True     True
2            True         True          True         True     True
3            True         True          True         True     True
4            True         True          True         True     True
..            ...          ...           ...          ...      ...
145          True         True          True         True     True
146          True         True          True         True     True
147          True         True          True         True     True
148          True         True          True         True     True
149          True         True          True         True     True

[150 rows x 5 columns]
```

```
Checking null/missing/blank values and fill with given values :
     sepal.length  sepal.width  petal.length  petal.width    variety
0             5.1          3.5           1.4          0.2     Setosa
1             4.9          3.0           1.4          0.2     Setosa
2             4.7          3.2           1.3          0.2     Setosa
3             4.6          3.1           1.5          0.2     Setosa
4             5.0          3.6           1.4          0.2     Setosa
..            ...          ...           ...          ...        ...
145           6.7          3.0           5.2          2.3  Virginica
146           6.3          2.5           5.0          1.9  Virginica
147           6.5          3.0           5.2          2.0  Virginica
148           6.2          3.4           5.4          2.3  Virginica
149           5.9          3.0           5.1          1.8  Virginica
```

```
[150 rows x 5 columns]


Replace the NaN values with given values :
     sepal.length  sepal.width  petal.length  petal.width    variety
0             5.1          3.5           1.4          0.2     Setosa
1             4.9          3.0           1.4          0.2     Setosa
2             4.7          3.2           1.3          0.2     Setosa
3             4.6          3.1           1.5          0.2     Setosa
4             5.0          3.6           1.4          0.2     Setosa
..            ...          ...           ...          ...        ...
145           6.7          3.0           5.2          2.3  Virginica
146           6.3          2.5           5.0          1.9  Virginica
147           6.5          3.0           5.2          2.0  Virginica
148           6.2          3.4           5.4          2.3  Virginica
149           5.9          3.0           5.1          1.8  Virginica

[150 rows x 5 columns]
```

Q. Create the data frame for the given data and process the data frame data for the null or missing values.

#create a dataframe

```
import pandas as pd

# Modified data with new names
d = {
    'EmpName' : ['Sudesh', 'Dinesh', 'Devesh', 'Ramesh', 'Suresh'],
    'DOJ' : ['01-02-2000', '03-05-2010', '01-02-2006', '10-08-2004', 'None'],
    'Exp' : ['10years', '5years', 'NaN', '15years', '30years'],
    'Pkg' : [4000000, 3000000, 4500000, 'NaN', 4000000],
    'Manager' : ['Sakpal', 'Sanap', 'Sakpal', 'Pankaj', 'Sanika'],
    'TEam' : ['Marketing', 'Digital', 'Legal', 'Marketing', 'Legal'],
}

# Create DataFrame
f = pd.DataFrame(d)
f
```

Output:

	EmpName	DOJ	Exp	Pkg	Manager	TEam
0	Sudesh	01-02-2000	10years	4000000	Sakpal	Marketing
1	Dinesh	03-05-2010	5years	3000000	Sanap	Digital
2	Devesh	01-02-2006	NaN	4500000	Sakpal	Legal
3	Ramesh	10-08-2004	15years	NaN	Pankaj	Marketing
4	Suresh	None	30years	4000000	Sanika	Legal

```
# Replace the string 'NaN' with np.nan
f.replace('NaN', np.nan, inplace=True)

# Explicitly infer data types to prevent issues
f = f.infer_objects()
```

	EmpName	DOJ	Exp	Pkg	Manager	TEam
0	Sudesh	01-02-2000	10years	4000000.0	Sakpal	Marketing
1	Dinesh	03-05-2010	5years	3000000.0	Sanap	Digital
2	Devesh	01-02-2006	NaN	4500000.0	Sakpal	Legal
3	Ramesh	10-08-2004	15years	NaN	Pankaj	Marketing
4	Suresh	None	30years	4000000.0	Sanika	Legal

```
import pandas as pd
import numpy as np

# Create a new DataFrame
d = {
    'Col_1': [2, 3, np.nan, 7, 8],
    'Col_2': [np.nan, 3, 4, 5, 'None'],
    'Col_3': [45, 4, 23, 234, 2],
    'Col_4': [56, 'None', 342, 3, 34],
    'Col_5': [67, 34, 34, np.nan, 45],
}
f = pd.DataFrame(d)

# Print initial DataFrame
print("Initial DataFrame:")
print(f)

# Replace NaN in 'Col_1' with 0
f['Col_1'] = f['Col_1'].replace(to_replace=np.nan, value=0)
print("\nAfter replacing NaN in 'Col_1' with 0:")
print(f)

# Replace 'None' in 'Col_2' with np.nan, then fill NaN with -1
f['Col_2'] = f['Col_2'].replace('None', np.nan).fillna(-1)
print("\nAfter replacing 'None' in 'Col_2' with NaN and filling NaN with -1:")
print(f)
```

```
# Set all values in 'Col_3' to -2
f['Col_3'] = -2
print("\nAfter setting all values in 'Col_3' to -2:")
print(f)

# Store the first value of 'Col_4' and replace 'None' with np.nan, then fill NaN with the stored value
v = f['Col_4'][0]
f['Col_4'] = f['Col_4'].replace('None', np.nan).fillna(v)
print("\nAfter replacing 'None' in 'Col_4' with NaN and filling NaN with the first value of 'Col_4':")
print(f)

# Calculate the mean of 'Col_5', replace NaN with this mean
av_mean = f['Col_5'].mean()
f['Col_5'] = f['Col_5'].replace(to_replace=np.nan, value=av_mean)
print("\nAfter replacing NaN in 'Col_5' with the mean of 'Col_5':")
print(f)
```

Output:

```
Initial DataFrame:
   Col_1 Col_2  Col_3 Col_4  Col_5
0    2.0   NaN     45    56   67.0
1    3.0     3      4  None   34.0
2    NaN     4     23   342   34.0
3    7.0     5    234     3    NaN
4    8.0  None      2    34   45.0

After replacing NaN in 'Col_1' with 0:
   Col_1 Col_2  Col_3 Col_4  Col_5
0    2.0   NaN     45    56   67.0
1    3.0     3      4  None   34.0
2    0.0     4     23   342   34.0
3    7.0     5    234     3    NaN
4    8.0  None      2    34   45.0
```

```
After replacing 'None' in 'Col_2' with NaN and filling NaN with -1:
   Col_1  Col_2  Col_3 Col_4  Col_5
0    2.0   -1.0     45    56   67.0
1    3.0    3.0      4  None   34.0
2    0.0    4.0     23   342   34.0
3    7.0    5.0    234     3    NaN
4    8.0   -1.0      2    34   45.0

After setting all values in 'Col_3' to -2:
   Col_1  Col_2  Col_3 Col_4  Col_5
0    2.0   -1.0     -2    56   67.0
1    3.0    3.0     -2  None   34.0
2    0.0    4.0     -2   342   34.0
3    7.0    5.0     -2     3    NaN
4    8.0   -1.0     -2    34   45.0
```

```
After replacing 'None' in 'Col_4' with NaN and filling NaN with the first value of 'Col_4':
   Col_1  Col_2  Col_3  Col_4  Col_5
0    2.0   -1.0     -2   56.0   67.0
1    3.0    3.0     -2   56.0   34.0
2    0.0    4.0     -2  342.0   34.0
3    7.0    5.0     -2    3.0    NaN
4    8.0   -1.0     -2   34.0   45.0

After replacing NaN in 'Col_5' with the mean of 'Col_5':
   Col_1  Col_2  Col_3  Col_4  Col_5
0    2.0   -1.0     -2   56.0   67.0
1    3.0    3.0     -2   56.0   34.0
2    0.0    4.0     -2  342.0   34.0
3    7.0    5.0     -2    3.0   45.0
4    8.0   -1.0     -2   34.0   45.0
```

Practical 4 - Basic Utility design, Data auditing and Exploratory Data Analysis

---

StudID	Name	Class	Elective	Msg	Test Score	Date of Birth
Student1	Sudesh	Fy	SBCM	he\x00llo	20	06-09-2003
Student2	Devesh	fy	sbcm	My\nComputer	8	02/02/24
Student3	Dinesh	FY	AI	Hello Guy\s	5	04-03-2006
Student4	Ramesh	fY	AI	Good morning	11	03/31/2003
Student5	Shivam	Fy	SBCM	he\x00llo	18	01-01-2001
Student6	Vidit	fy	sbcm	My\nComputer	9	02/02/24
Student7	Vinit	FY	AI	Hello Guy\s	50	04-03-2006
Student8	Hardik	fY	AI	Good morning	10	03/31/2003

```
# Creating the dataframe
import pandas as pd
data = {
    'StudID': ['Student1', 'Student2', 'Student3', 'Student4', 'Student5', 'Student6', 'Student7', 'Student8'],
    'Name': ['   Sudesh  ', 'Devesh', 'Dinesh     ', ' Ramesh ', '   Shivam  ', 'Vidit', 'Vinit ', ' Hardik '],
    'Class': ['Fy', 'fy', 'FY', 'fY', 'Fy', 'fy', 'FY', 'fY'],
    'Elective': ['SBCM', '   sbcm', 'AI', 'AI   ', 'SBCM', '   sbcm', 'AI', 'AI   '],
    'Msg': ['he\x00llo', 'My\nComputer', 'Hello Guy\\s', 'Good morning', 'he\x00llo', 'My\nComputer', 'Hello Guy\\s', 'Good morning'],
    'Test Score': [20, 8, 5, 11, 18, 9, 50, 10],
    'Date of Birth': ['06-09-2003', '02/02/24', '04-03-2006', '03/31/2003', '01-01-2001', '02/02/24', '04-03-2006', '03/31/2003']
}
df = pd.DataFrame(data)
print("Original Data : ")
print(df)
```

```
Original Data :
     StudID         Name Class Elective           Msg  Test Score  \
0  Student1     Sudesh      Fy     SBCM         hello          20
1  Student2       Devesh    fy     sbcm  My\nComputer           8
2  Student3  Dinesh         FY       AI   Hello Guy\s           5
3  Student4      Ramesh     fY    AI     Good morning          11
4  Student5     Shivam      Fy     SBCM         hello          18
5  Student6        Vidit    fy     sbcm  My\nComputer           9
6  Student7   Vinit         FY       AI   Hello Guy\s          50
7  Student8      Hardik     fY    AI     Good morning          10

  Date of Birth
0    06-09-2003
1      02/02/24
2    04-03-2006
3    03/31/2003
4    01-01-2001
5      02/02/24
6    04-03-2006
7    03/31/2003
```

Basic Utility Design

1. Fixer Utility

Q. Remove leading and trailing spaces from the data of all columns.

```
df = df.apply(lambda x: x.str.strip() if x.dtype == "object" else x)
print("After removing leading and trailing spaces from dataframe :")
df
```

```
After removing leading and trailing spaces from dataframe :
```

	StudID	Name	Class	Elective	Msg	Test Score	Date of Birth
0	Student1	Sudesh	Fy	SBCM	hello	20	06-09-2003
1	Student2	Devesh	fy	sbcm	My\nComputer	8	02/02/24
2	Student3	Dinesh	FY	AI	Hello Guy\s	5	04-03-2006
3	Student4	Ramesh	fY	AI	Good morning	11	03/31/2003
4	Student5	Shivam	Fy	SBCM	hello	18	01-01-2001
5	Student6	Vidit	fy	sbcm	My\nComputer	9	02/02/24
6	Student7	Vinit	FY	AI	Hello Guy\s	50	04-03-2006
7	Student8	Hardik	fY	AI	Good morning	10	03/31/2003

Q. Remove non-printable characters from the ‘Msg’ Column values.

```
# Removing non-printable characters from msg column
import re
df['Msg'] = df['Msg'].apply(lambda x: re.sub(r'[^\x20-\x7E]', '', x).replace('\\', ''))
print("After removing non-printable characters in Msg column : ")
print(df['Msg'])
```

```
After removing non-printable characters in Msg column :
0           hello
1      MyComputer
2      Hello Guys
3    Good morning
4           hello
5      MyComputer
6      Hello Guys
7    Good morning
Name: Msg, dtype: object
```

Q. Reformat the ‘Date of Birth’ column with the format ‘DD-MM-YYYY’

```
from datetime import datetime
def parse_dates(date):
    for fmt in ('%d-%m-%Y', '%d/%m/%y', '%m/%d/%Y', '%m/%d/%y'):
        try:
            return datetime.strptime(date, fmt).strftime('%d-%m-%Y')
        except ValueError:
            continue
    return pd.NaT  # If none of the formats work, return NaT

df['Date of Birth'] = df['Date of Birth'].apply(parse_dates)
print("Data after reformatting 'Date of Birth' column:")
print(df['Date of Birth'])
```

```
Data after reformatting 'Date of Birth' column:
0    06-09-2003
1    02-02-2024
2    04-03-2006
3    31-03-2003
4    01-01-2001
5    02-02-2024
6    04-03-2006
7    31-03-2003
Name: Date of Birth, dtype: object
```

2. Data Binning or Bucketing

Q. Classify the given data based on the students test score into three bins named ‘poor’, ‘average’ and ‘best’.

```
bins = [0, 10, 15, 20]
labels = ['poor', 'average', 'best']
df['Score Category'] = pd.cut(df['Test Score'], bins=bins, labels=labels,
include_lowest=True)
print("Bins:")
print(df[['StudID', 'Test Score', 'Score Category']])
```

```
Bins:
     StudID  Test Score Score Category
0  Student1          20           best
1  Student2           8           poor
2  Student3           5           poor
3  Student4          11        average
4  Student5          18           best
5  Student6           9           poor
6  Student7          50            NaN
7  Student8          10           poor
```

3. Averaging the Data

Q. Get the average test score of the class ‘fy’

```
#average of 'fy'
average_score_fy = df[df['Class'].str.lower() == 'fy']['Test Score'].mean()
print(f"Average Test Score for class 'fy': {average_score_fy}")
```

```
Average Test Score for class 'fy': 16.375
```

4. Outlier Detection

Q. Check and display the test score Outliers.

```
#Outliers
Q1 = df['Test Score'].quantile(0.25)
Q3 = df['Test Score'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers = df[(df['Test Score'] < lower_bound) | (df['Test Score'] > upper_bound)]
print("Outliers :")
print(outliers[['StudID', 'Test Score']])
```

```
Outliers :
     StudID  Test Score
6  Student7          50
```

5. Logging

Q. Check the result with the value 1, -1 etc.

```
import logging
logging.basicConfig(level=logging.DEBUG, format='%(asctime)s %(levelname)s -
%(message)s')
def perform_operation(value):
    if value < 0:
        raise ValueError("Invalid value: Value cannot be negative.")
    else:
        # Continue with normal execution
        logging.info("Operation performed successfully.")
try:
    input_value = int(input("Enter a value: "))
    perform_operation(input_value)
except ValueError as ve:
    logging.exception("Exception occurred: %s", str(ve))
```

```
Enter a value: 1
2024-08-14 15:45:09,895 INFO - Operation performed successfully.
```

```
Enter a value: 2
2024-08-14 15:51:52,762 INFO - Operation performed successfully.
```

Data auditing

Q. Check the student date of birth is in the range of 01-01-2000 to 01-01-2025.

```
start = pd.to_datetime('01-01-2000')
end = pd.to_datetime('01-01-2025')
df['Date of Birth'] = pd.to_datetime(df['Date of Birth'], errors='coerce')
dob = df[(df['Date of Birth'] >= start) & (df['Date of Birth'] <= end)]
print(dob[['Name', 'Date of Birth']])
```

```
     Name Date of Birth
0  Sudesh    2003-06-09
1  Devesh    2024-02-02
2  Dinesh    2006-04-03
4  Shivam    2001-01-01
5   Vidit    2024-02-02
6   Vinit    2006-04-03
```

Q. Check for invalid or wrong test score values. The test score range should be from 0 to 20 only.

```
d = df['Test Score'].values
v=d[(d<0) | (d>20)]
print("Invalid test score :",v)
```

```
Invalid test score : [50]
```

```
import re
s = df['Name']
def removePunc(z):
    return re.sub('[ 1-9!#.]', '' , z)
result = []
for x in map(removePunc, s):
    result.append(x)
print(result)
```

```
['Sudesh', 'Devesh', 'Dinesh', 'Ramesh', 'Shivam', 'Vidit', 'Vinit', 'Hardik']
```

Exploratory data analysis

Q. Describe students' data and check for average test scores.

```
print("Description of data :")
df.describe()
```

Description of data :

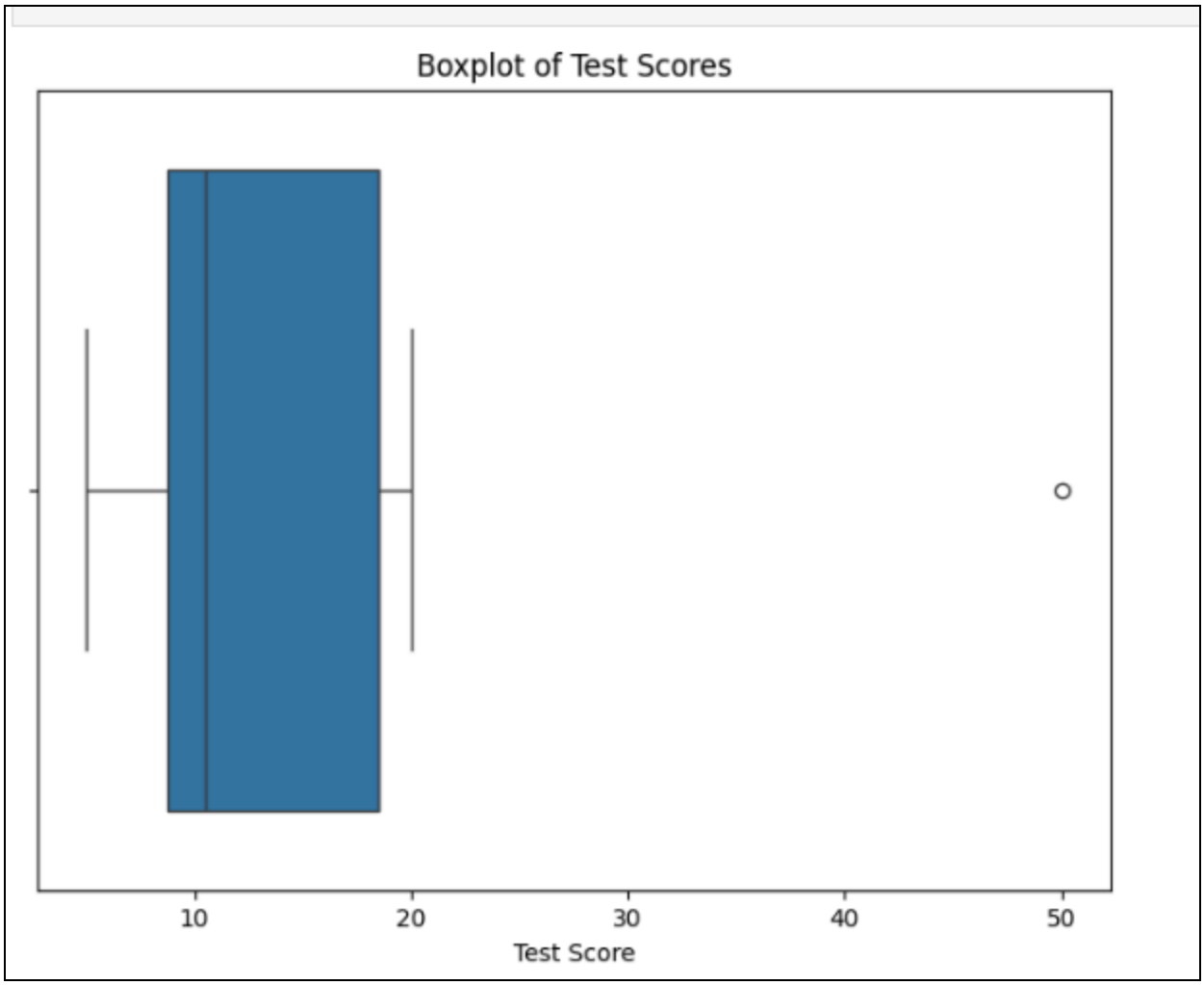
	Test Score	Date of Birth
count	8.000000	6
mean	16.375000	2010-11-08 04:00:00
min	5.000000	2001-01-01 00:00:00
25%	8.750000	2004-02-21 06:00:00
50%	10.500000	2006-04-03 00:00:00
75%	18.500000	2019-08-18 12:00:00
max	50.000000	2024-02-02 00:00:00
std	14.490761	NaN

```
print("Average of Test Score column is :",df['Test Score'].mean())
```

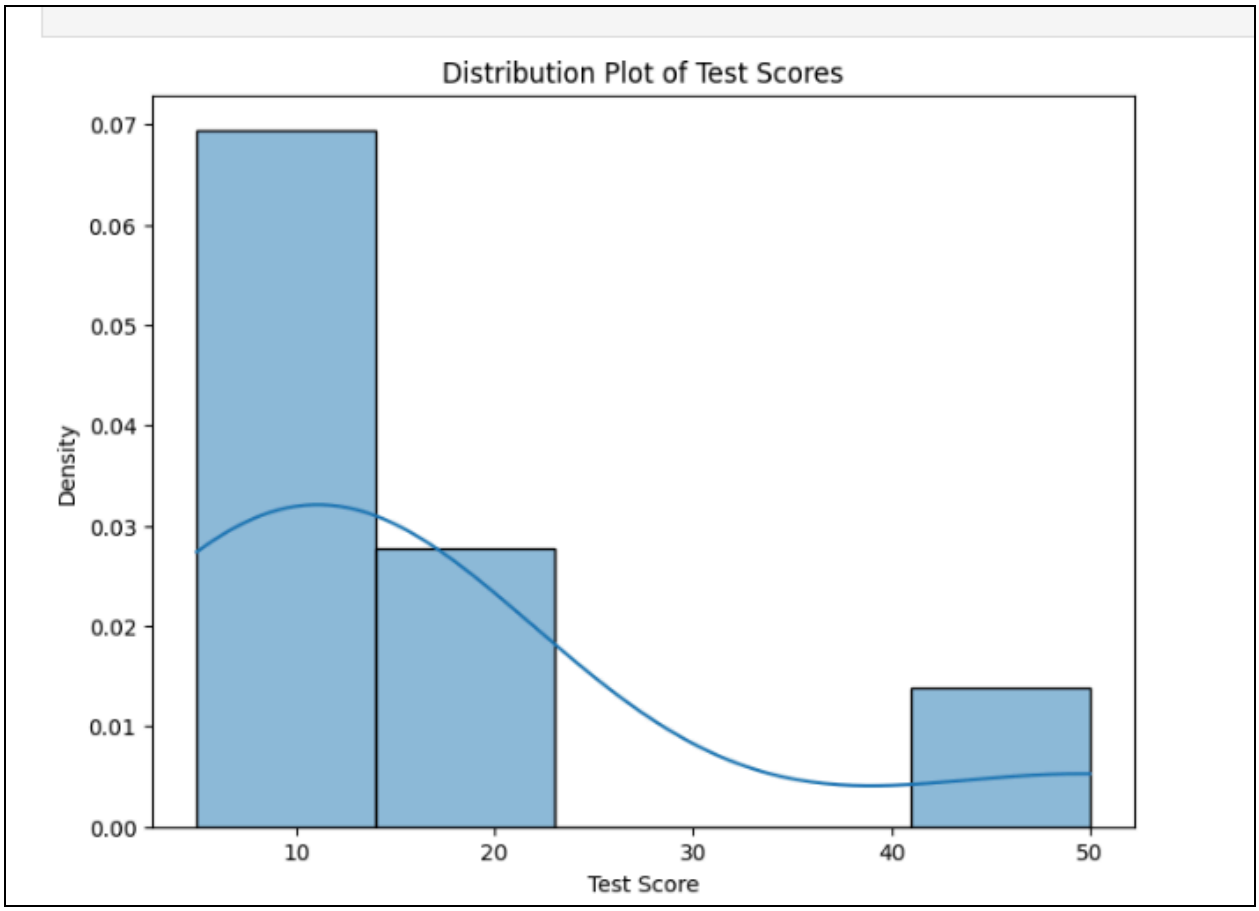
```
Average of Test Score column is : 16.375
```

Q. Check for the data value distribution of the test score column by plotting boxplot.

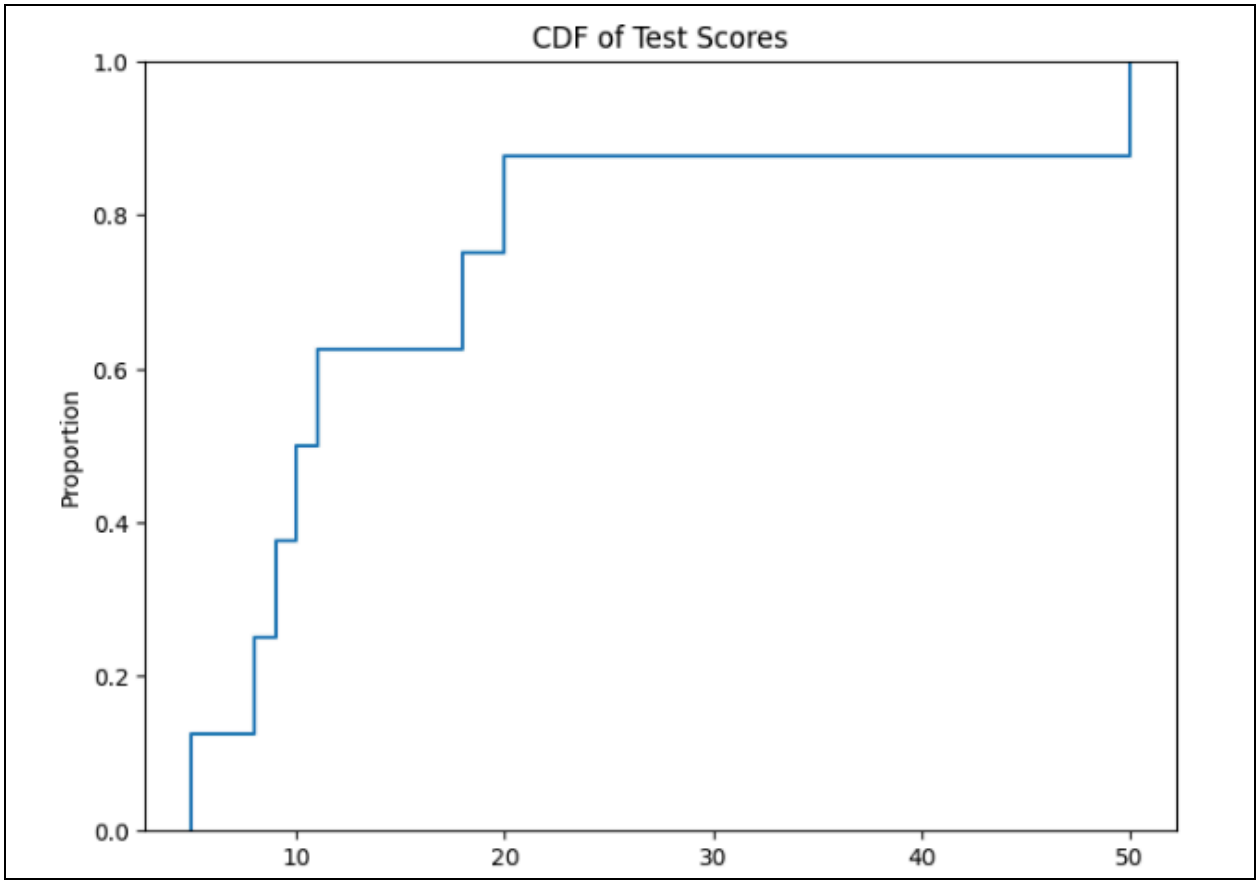
```
import matplotlib.pyplot as plt
import seaborn as sns  # pip install seaborn
plt.figure(figsize=(8, 6))
sns.boxplot(x=df['Test Score'])
plt.title('Boxplot of Test Scores')
plt.show()
```




```
#Distribution plot
plt.figure(figsize=(8, 6))
sns.histplot(df['Test Score'], kde=True, stat="density", label='Density')
plt.title('Distribution Plot of Test Scores')
plt.show()
```

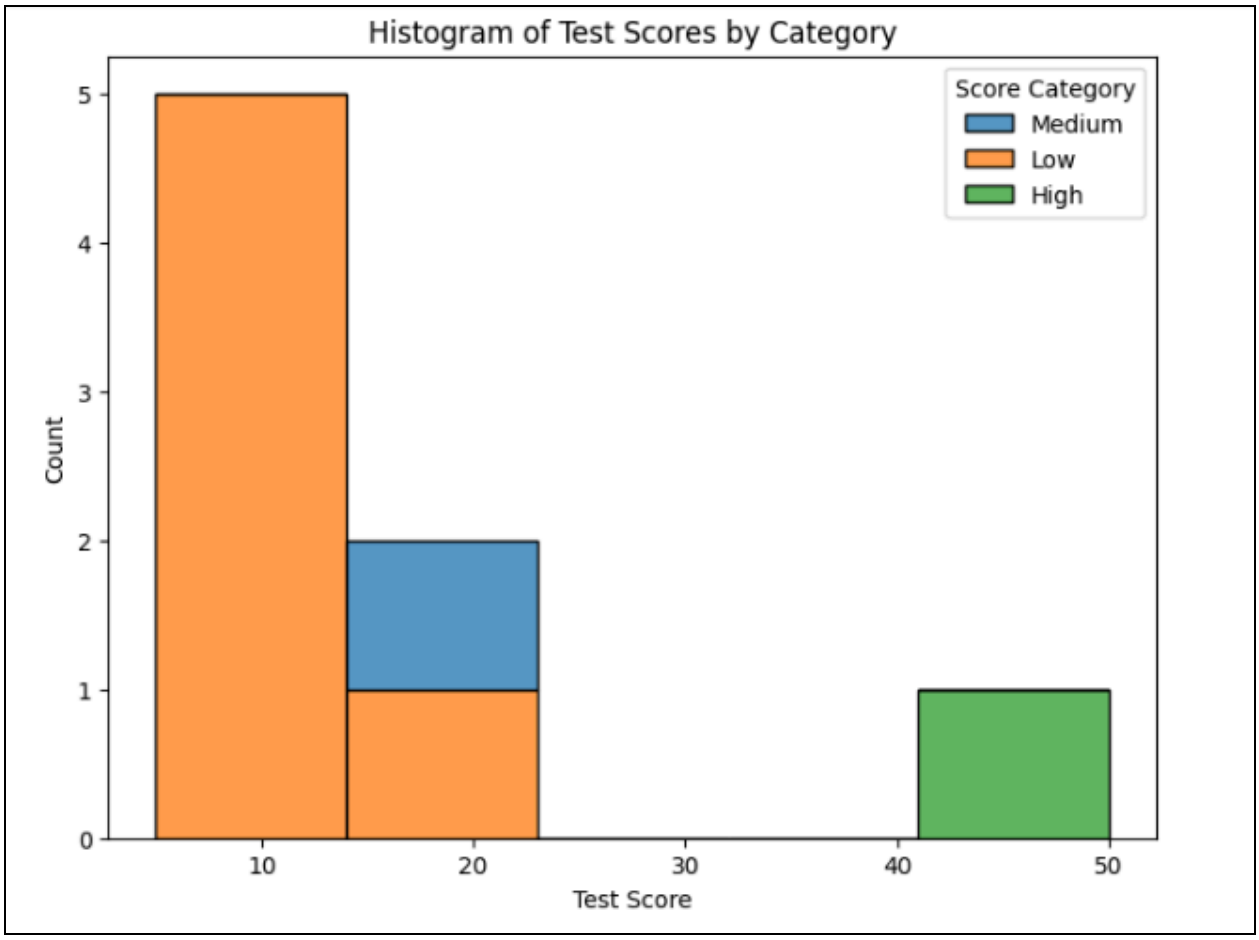



```
# CDF Plot
plt.figure(figsize=(8, 6))
sns.ecdfplot(df['Test Score'])
plt.title('CDF of Test Scores')
plt.show()
```

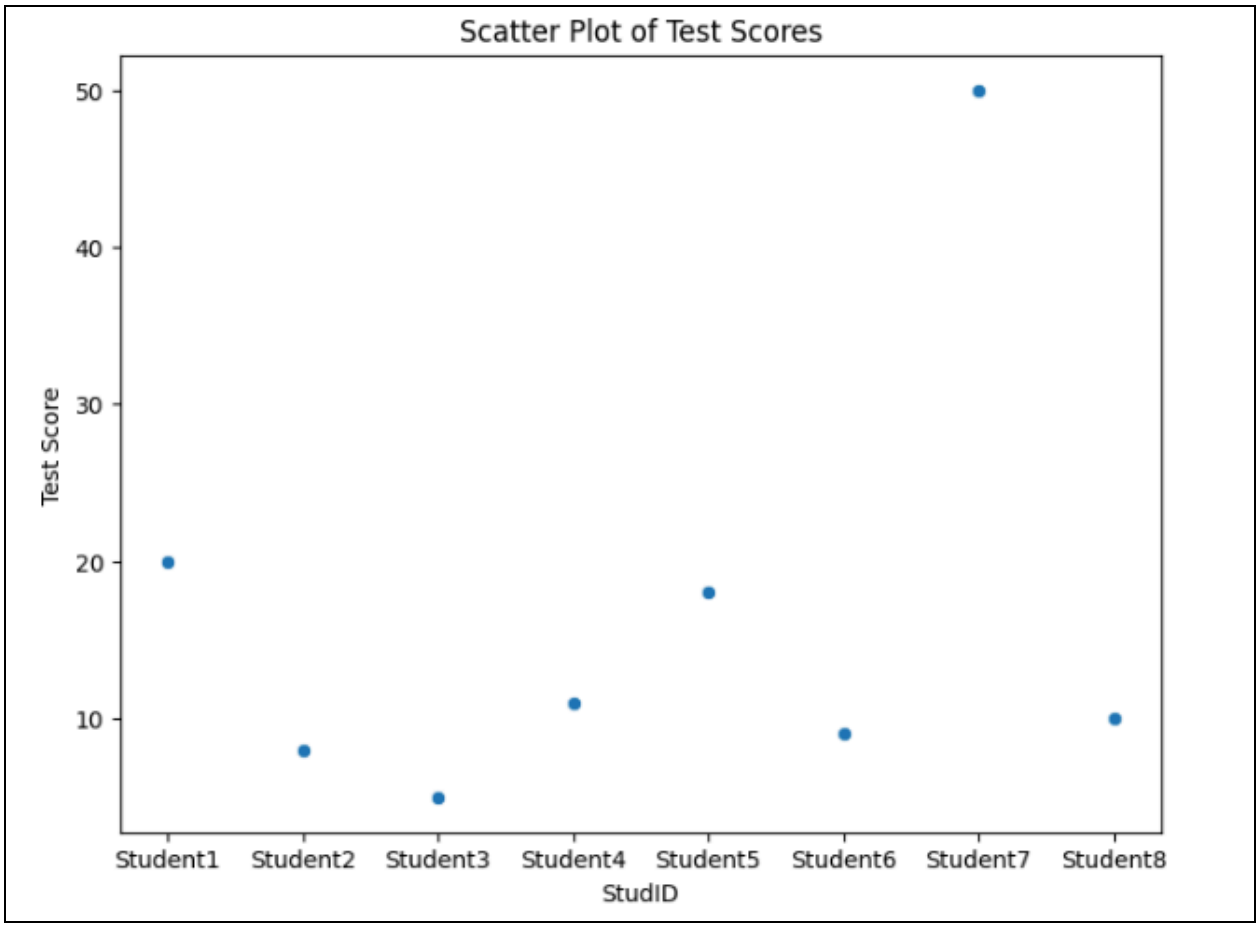


Q. Draw the histogram for the test bad, average and best test score.

```
# Define score categories
def categorize_score(score):
    if score >= 40:
        return 'High'
    elif score >= 20:
        return 'Medium'
    else:
        return 'Low'
df['Score Category'] = df['Test Score'].apply(categorize_score)
# Plot the histogram
plt.figure(figsize=(8, 6))
sns.histplot(data=df, x='Test Score', hue='Score Category', multiple='stack')
plt.title('Histogram of Test Scores by Category')
plt.show()
```




```
# Scatter plot
plt.figure(figsize=(8, 6))
sns.scatterplot(x='StudID', y='Test Score', data=df)
plt.title('Scatter Plot of Test Scores')
plt.show()
```


Scatter Plot of Test Scores
Test Score
50
40
30
20
10
Student1
Student2
Student3
Student4
Student5
Student6
Student7
Student8
StudID

Practical 5 - Retrieve Utility

Currency dataset :
https://github.com/Apress/practical-data-science/blob/master/VKHCG/01-Vermeulen/00-RawData/COUNTRY-CODES.csv

1. Load the raw data of Excel/csv file

```
import pandas as pd
import os
import sys
sFileDir='D:\\New folder\\MSCIT_6623\\ids'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
sFileName='D:\\New folder\\MSCIT_6623\\ids\\COUNTRY-CODES.csv'
print("Path :", sFileName)
```

```
Path : D:\New folder\MSCIT_6623\ids\COUNTRY-CODES.csv
```

2. Load given csv/Excel file in the pandas dataframe.

```
import pandas as pd

file_path = r"D:\\New folder\\MSCIT_6623\\ids\\COUNTRY-CODES.csv"

# List of encodings to try
encodings = ['utf-8', 'latin1', 'ISO-8859-1', 'utf-16', 'utf-32']

for encoding in encodings:
    try:
        CurrencyData = pd.read_csv(file_path, encoding=encoding)
        print(f"Successfully read file with encoding: {encoding}")
        print("Data : ")
        print(CurrencyData)
        break  # Exit the loop if reading is successful
    except UnicodeDecodeError as e:
        print(f"Failed to read file with encoding: {encoding}. Error: {e}")
    except Exception as e:
        print(f"An unexpected error occurred with encoding {encoding}: {e}")
```

```
Successfully read file with encoding: latin1
Data :
    Code          Country name    Year country code top-level domain
0     AD               Andorra  1974.0                          .ad
1     AE  United Arab Emirates  1974.0                          .ae
2     AF           Afghanistan  1974.0                          .af
3     AG   Antigua and Barbuda  1974.0                          .ag
4     AI              Anguilla  1983.0                          .ai
..   ...                   ...     ...                         ...
258  NaN                   NaN     NaN                         NaN
259   YT               Mayotte  1993.0                          .yt
260   ZA          South Africa  1974.0                          .za
261   ZM                Zambia  1974.0                          .zm
262   ZW              Zimbabwe  1980.0                          .zw

[263 rows x 4 columns]
```

3. Rename the columns of the dataframe.

#Renaming Columns

```
CurrencyData.rename({'ISO-2-CODE' : 'CountryCode1', 'ISO-3-Code' : 'CountryCode2'},
axis=1, inplace=True)
CD = CurrencyData
print("After renaming columns : ")
CD
```

After renaming columns :

	Code	Country name	Year	country code top-level domain
0	AD	Andorra	1974.0	.ad
1	AE	United Arab Emirates	1974.0	.ae
2	AF	Afghanistan	1974.0	.af
3	AG	Antigua and Barbuda	1974.0	.ag
4	AI	Anguilla	1983.0	.ai
...	...	...	...	...
258	NaN	NaN	NaN	NaN
259	YT	Mayotte	1993.0	.yt
260	ZA	South Africa	1974.0	.za
261	ZM	Zambia	1974.0	.zm
262	ZW	Zimbabwe	1980.0	.zw

263 rows × 4 columns

4. Drop not required columns from the dataframe.

```
#Drop
CD.drop('Code', axis=1, inplace=True)
CD
```

	Country name	Year	country code top-level domain
0	Andorra	1974.0	.ad
1	United Arab Emirates	1974.0	.ae
2	Afghanistan	1974.0	.af
3	Antigua and Barbuda	1974.0	.ag
4	Anguilla	1983.0	.ai
...	...	...	...
258	NaN	NaN	NaN
259	Mayotte	1993.0	.yt
260	South Africa	1974.0	.za
261	Zambia	1974.0	.zm
262	Zimbabwe	1980.0	.zw

263 rows × 3 columns

5. Save the retrieved file in the specified folder

```
CD.to_csv("D:\\New folder\\MSCIT_6623\\ids\\Datafile.csv")
print("Saved successfully!!!")
```

```
CD.to_csv("D:\\New folder\\MSCIT_6623\\ids\\Datafile.csv")
print("Saved successfully!!!")
```

```
Saved successfully!!!
```

7. Retrieve different attributes of data

Q. Retrieve country names from the dataframe CurrencyData.

```
#Retrieving Country name
print("Country names : ")
CD['Country']
```

```
Country names :
0                   Andorra
1      United Arab Emirates
2               Afghanistan
3       Antigua and Barbuda
4                  Anguilla
               ...
258                     NaN
259                 Mayotte
260            South Africa
261                  Zambia
262                Zimbabwe
Name: Country name, Length: 263, dtype: object
```

Q. Data profile the data distribution (Skew, Histogram, Min, Max).

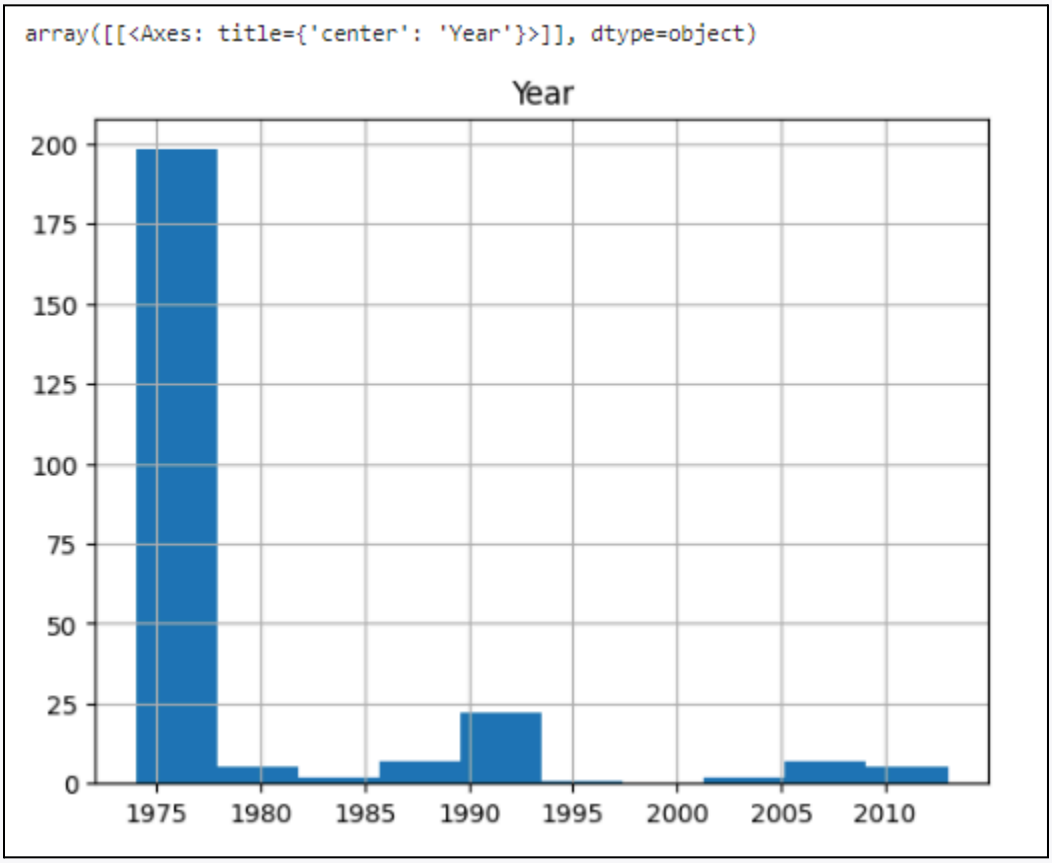
```
# skewness along the index axis
# Select only numeric columns
numeric_df = CD.select_dtypes(include=['number'])
# Calculate skewness for the numeric columns
skewness = numeric_df.skew(axis=0, skipna=True)
print("Skewness :")
print(skewness)
```

```
Skewness :
Year    2.229908
dtype: float64
```

Q. Histogram

```
CurrencyData.hist(column='Year')
```

```
array([[<Axes: title={'center': 'Year'}>]], dtype=object)
```



Q. Identify any loading characteristics (Columns Names, Data Types, Volumes).

CurrencyData.columns

```
Index(['Country', 'CountryCode2', 'ISO-M49'], dtype='object')
```

CurrencyData.dtypes

```
Country          object
CountryCode2     object
ISO-M49           int64
dtype: object
```

CurrencyData.shape

```
(247, 3)
```

CurrencyData.size

```
741
```

CurrencyData.min()

```
Country          Afghanistan
CountryCode2             ABW
ISO-M49                    4
dtype: object
```

CurrencyData.max()

```
Country          Zimbabwe
CountryCode2          ZWE
ISO-M49               894
dtype: object
```

Practical 6 - Access Superstep.

```
import pandas as pd
import numpy as np
df =
pd.DataFrame([[np.nan,2,np.nan,0],[3,4,np.nan,1],[np.nan,np.nan,np.nan,np.nan],[n
p.nan,3,np.nan,4]], columns=list("ABCD"))
df
```

	A	B	C	D
0	NaN	2.0	NaN	0.0
1	3.0	4.0	NaN	1.0
2	NaN	NaN	NaN	NaN
3	NaN	3.0	NaN	4.0

```
#Dropping columns including missing value
a = df.dropna(axis=1,how='any')
a
```

0
1
2
3

```
a = df.dropna(axis=1,how='all')
a
```

	A	B	D
0	NaN	2.0	0.0
1	3.0	4.0	1.0
2	NaN	NaN	NaN
3	NaN	3.0	4.0

```
a = df.dropna(axis=0,how='all')
a
```

	A	B	C	D
0	NaN	2.0	NaN	0.0
1	3.0	4.0	NaN	1.0
3	NaN	3.0	NaN	4.0

```
#dropping columns including a specific number of missing values
a = df[df.isnull().sum(axis=1) <=2]
a
```

	A	B	C	D
0	NaN	2.0	NaN	0.0
1	3.0	4.0	NaN	1.0
3	NaN	3.0	NaN	4.0

```
#Number of missing values in each row
a = df.isnull().sum()
a
```

```
A    3
B    1
C    4
D    1
dtype: int64
```

#replacing missing values with basic measures values like mean, median etc.

```
a = df.fillna(df.mean())
a
```

	A	B	C	D
0	3.0	2.0	NaN	0.000000
1	3.0	4.0	NaN	1.000000
2	3.0	3.0	NaN	1.666667
3	3.0	3.0	NaN	4.000000

```
a = df.fillna(df.median())
a
```

	A	B	C	D
0	3.0	2.0	NaN	0.0
1	3.0	4.0	NaN	1.0
2	3.0	3.0	NaN	1.0
3	3.0	3.0	NaN	4.0

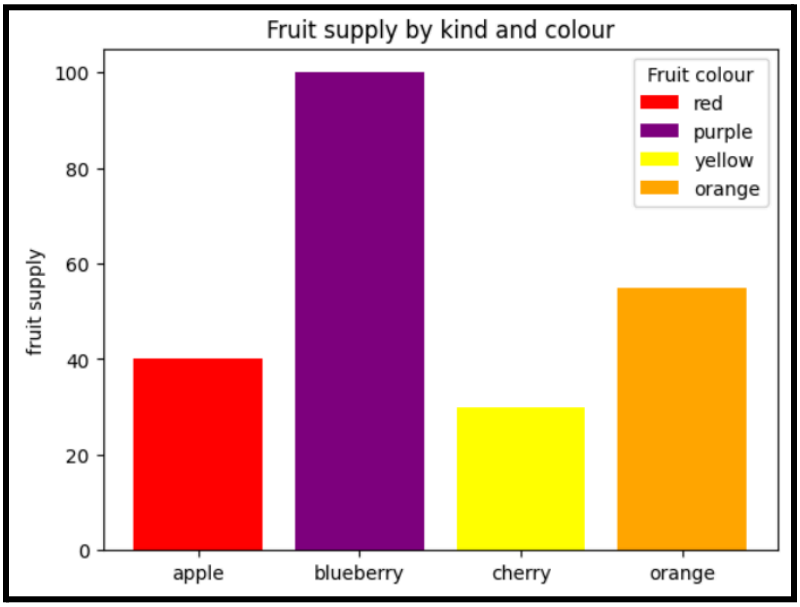
a = df.fillna(df.mode().iloc[0])

a

	A	B	C	D
0	3.0	2.0	NaN	0.0
1	3.0	4.0	NaN	1.0
2	3.0	2.0	NaN	0.0
3	3.0	3.0	NaN	4.0

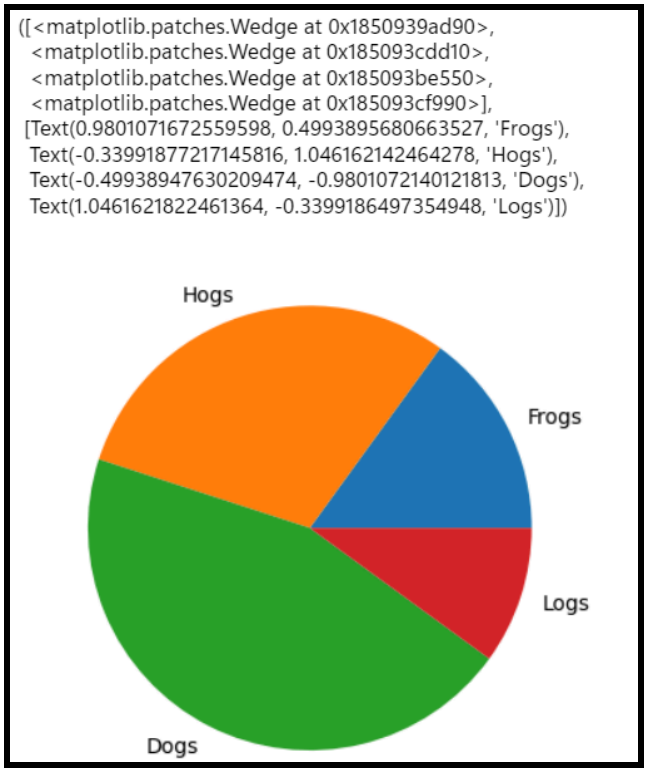
Practical 8 - Data Visualization

```
#Bar chart
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
fruits = ['apple','blueberry','cherry','orange']
counts = [ 40, 100, 30, 55]
bar_labels = ['red', 'purple', 'yellow', 'orange']
bar_colors = ['red', 'purple', 'yellow', 'orange']
ax.bar(fruits, counts, label=bar_labels, color=bar_colors)
ax.set_ylabel('fruit supply')
ax.set_title('Fruit supply by kind and colour')
ax.legend(title='Fruit colour')
plt.show()
```

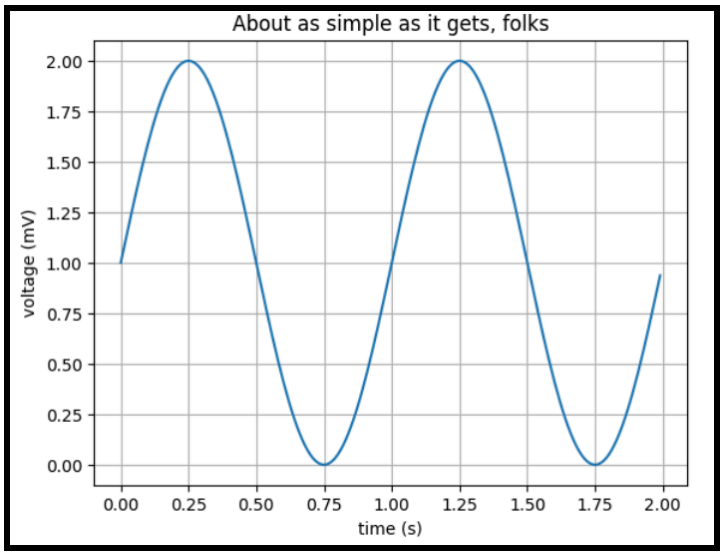



```
#Pie Chart
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15,30,45,10]
fig, ax = plt.subplots()
ax.pie(sizes, labels=labels)
```

```
([<matplotlib.patches.Wedge at 0x1850939ad90>,
  <matplotlib.patches.Wedge at 0x185093cdd10>,
  <matplotlib.patches.Wedge at 0x185093be550>,
  <matplotlib.patches.Wedge at 0x185093cf990>],
 [Text(0.9801071672559598, 0.4993895680663527, 'Frogs'),
  Text(-0.33991877217145816, 1.046162142464278, 'Hogs'),
  Text(-0.49938947630209474, -0.9801072140121813, 'Dogs'),
  Text(1.0461621822461364, -0.3399186497354948, 'Logs')])
```




```
#Line Graph
t = np.arange(0.0, 2.0, 0.01)
s = 1 + np.sin(2 * np.pi * t)
fig, ax = plt.subplots()
ax.plot(t, s)
ax.set(xlabel='time (s)', ylabel='voltage (mV)', title='About as simple as it gets, folks')
ax.grid()
fig.savefig("D:\\6626_Ariba\\test.png")
plt.show()
```

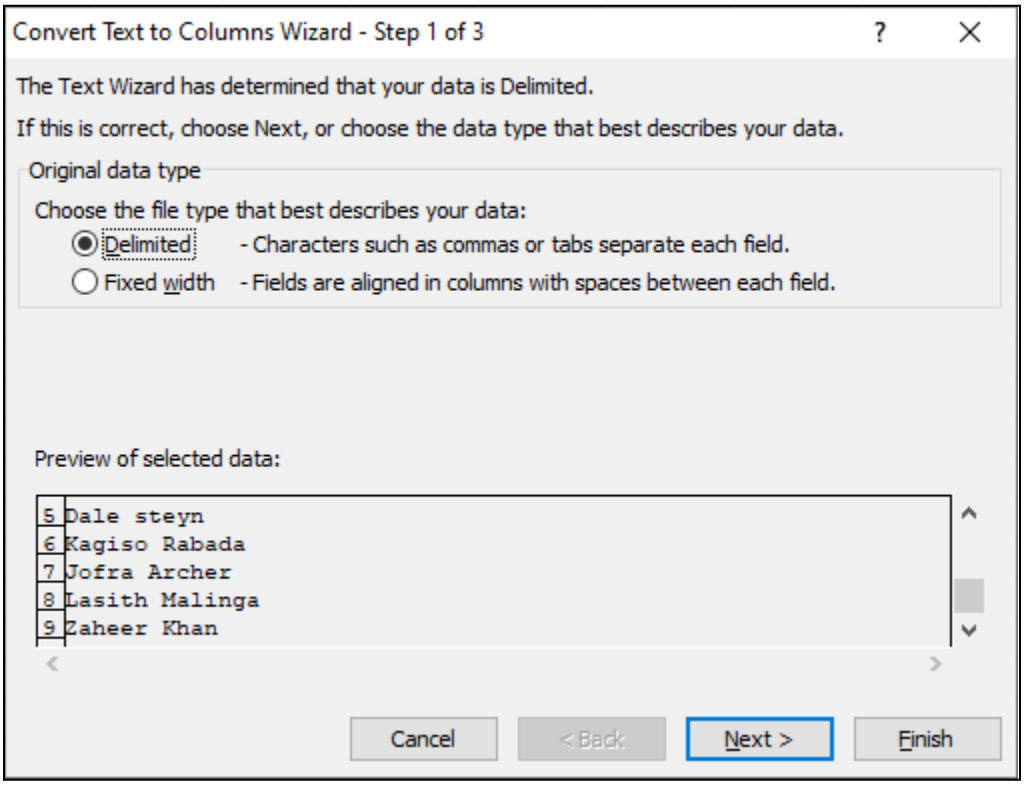

About as simple as it gets, folks
voltage (mV)
time (s)
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

Practical 9 : Data Analysis using Excel

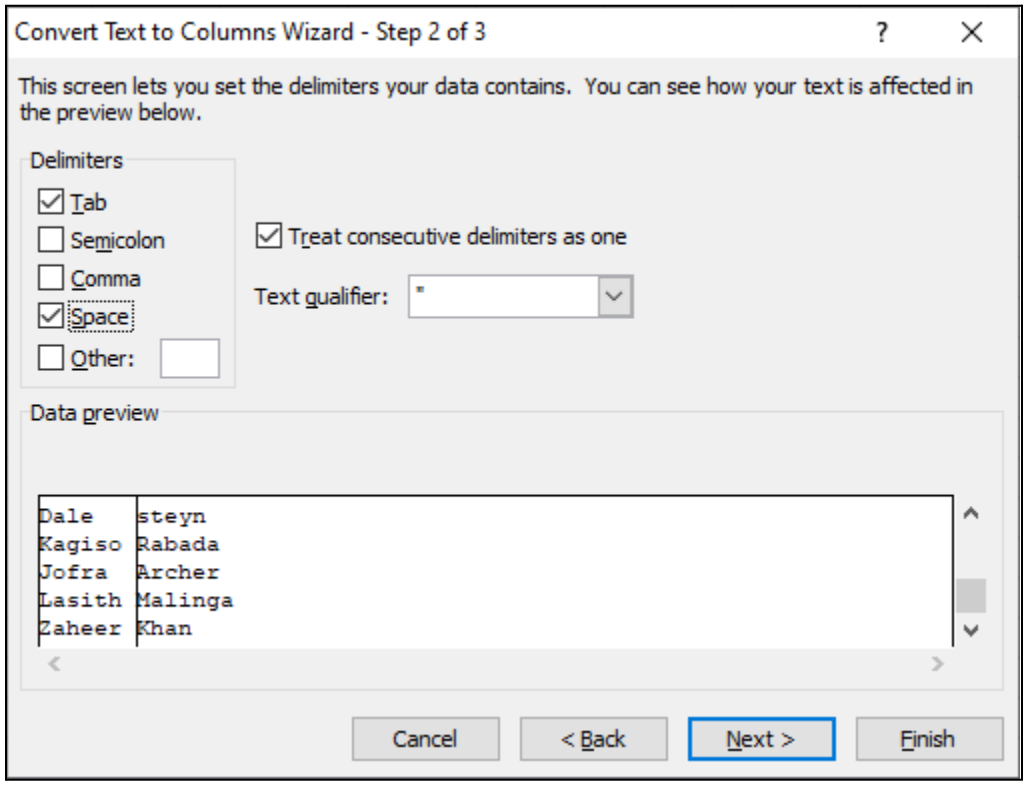
Create an excel sheet

	A	B	C
1	Student Name	F_Name	L_Name
2	Sudesh Rajbhar		
3	Shivam Vishwakarma		
4	Keshav Maharaj		
5	Dale steyn		
6	Kagiso Rabada		
7	Jofra Archer		
8	Lasith Malinga		
9	Zaheer Khan		

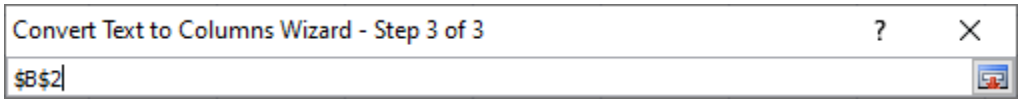
Select the values from Student Name column -> Data tab -> Text to Columns -> Click Next



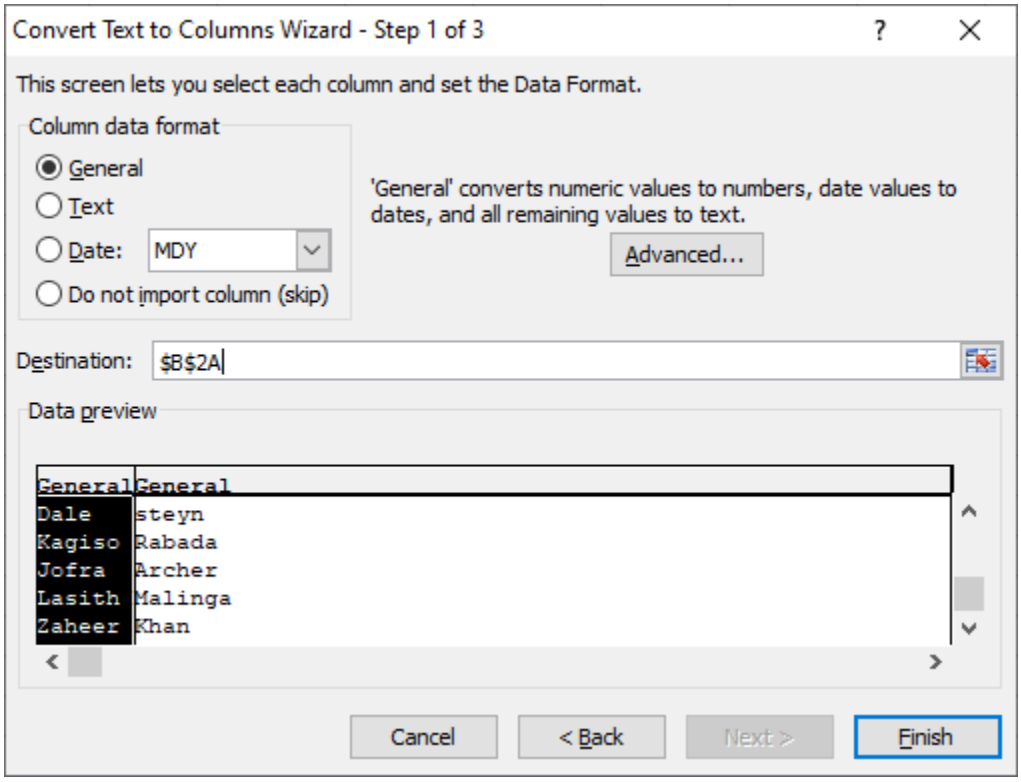
Click Next



Select B2 -> Click down arrow



Select Text -> Click Finish



Output:

	A	B	C
1	Student Name	F_Name	L_Name
2	Sudesh Rajbhar	Sudesh	Rajbhar
3	Shivam Vishwakarma	Shivam	Vishwakarma
4	Keshav Maharaj	Keshav	Maharaj
5	Dale steyn	Dale	steyn
6	Kagiso Rabada	Kagiso	Rabada
7	Jofra Archer	Jofra	Archer
8	Lasith Malinga	Lasith	Malinga
9	Zaheer Khan	Zaheer	Khan

Create U1 columns having A, B and C as sub columns -> CreateTotal Q1 column for sum value

H3 | =SUM(E3:G3)

A	B	C	D	E	F	G	H
					U1		
Student Name	F_Name	L_Name	Roll no	S1	S2	S3	TOTAL
Sudesh Rajbhar	Sudesh	Rajbhar	101	15	15	20	50
Shivam Vishwakarma	Shivam	Vishwakarma	102	18	17	16	51
Keshav Maharaj	Keshav	Maharaj	103	20	20	20	60
Dale steyn	Dale	steyn	104	20	15	15	50
Kagiso Rabada	Kagiso	Rabada	105	20	16	15	51
Jofra Archer	Jofra	Archer	106	17	17	16	50
Lasith Malinga	Lasith	Malinga	107	17	18	16	51
Zaheer Khan	Zaheer	Khan	108	15	15	16	46

Similarly create a U2 and Total U2 columns

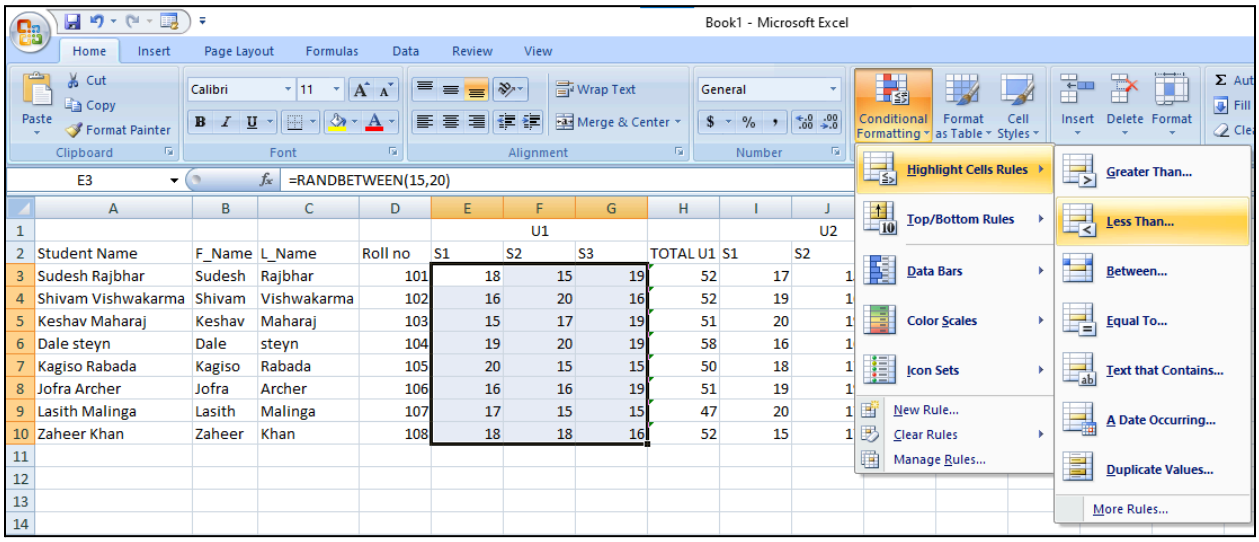
A	B	C	D	E	F	G	H	I	J	K	L
				U1				U2			
Student Name	F_Name	L_Name	Roll no	S1	S2	S3	TOTAL U1	S1	S2	S3	TOTAL U2
Sudesh Rajbhar	Sudesh	Rajbhar	101	16	20	19	55	20	17	15	52
Shivam Vishwakarma	Shivam	Vishwakarma	102	15	18	16	49	18	17	17	52
Keshav Maharaj	Keshav	Maharaj	103	16	16	19	51	20	18	15	53
Dale steyn	Dale	steyn	104	17	20	20	57	17	20	15	52
Kagiso Rabada	Kagiso	Rabada	105	16	18	19	53	20	15	20	55
Jofra Archer	Jofra	Archer	106	19	18	18	55	15	17	17	49
Lasith Malinga	Lasith	Malinga	107	19	16	17	52	15	19	16	50
Zaheer Khan	Zaheer	Khan	108	15	19	15	49	15	15	18	48

Create a column named Average for calculating average value of U1 and U2

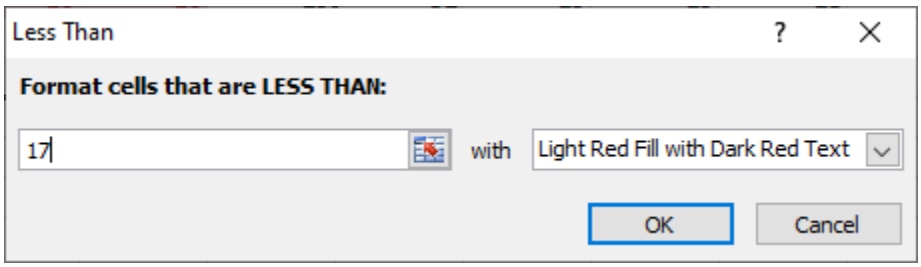
A	B	C	D	E	F	G	H	I	J	K	L	M	N
				U1				U2					
Student Name	F_Name	L_Name	Roll no	S1	S2	S3	TOTAL U1	S1	S2	S3	TOTAL U2	AVERAGE	
Sudesh Rajbhar	Sudesh	Rajbhar	101	20	16	19	55	18	16	17	51	=AVERAGE(H3,L3)	
Shivam Vishwakarma	Shivam	Vishwakarma	102	19	16	19	54	15	15	18	48		
Keshav Maharaj	Keshav	Maharaj	103	18	18	16	52	15	15	16	46		
Dale steyn	Dale	steyn	104	18	19	20	57	20	19	20	59		
Kagiso Rabada	Kagiso	Rabada	105	15	17	16	48	19	17	16	52		
Jofra Archer	Jofra	Archer	106	18	18	19	55	15	17	20	52		
Lasith Malinga	Lasith	Malinga	107	16	18	20	54	19	16	19	54		
Zaheer Khan	Zaheer	Khan	108	16	19	17	52	19	19	19	57		

Student Name	F_Name	L_Name	Roll no	S1	S2	S3	TOTAL U1	S1	S2	S3	TOTAL U2	AVERAGE
Sudesh Rajbhar	Sudesh	Rajbhar	101	18	15	19	52	17	15	18	50	51
Shivam Vishwakarma	Shivam	Vishwakarma	102	16	20	16	52	19	16	16	51	51.5
Keshav Maharaj	Keshav	Maharaj	103	15	17	19	51	20	19	18	57	54
Dale steyn	Dale	steyn	104	19	20	19	58	16	16	15	47	52.5
Kagiso Rabada	Kagiso	Rabada	105	20	15	15	50	18	17	18	53	51.5
Jofra Archer	Jofra	Archer	106	16	16	19	51	19	19	18	56	53.5
Lasith Malinga	Lasith	Malinga	107	17	15	15	47	20	17	17	54	50.5
Zaheer Khan	Zaheer	Khan	108	18	18	16	52	15	17	16	48	50

Apply Conditional Formatting (Less than) on values of Q1



Enter a value -> Click OK



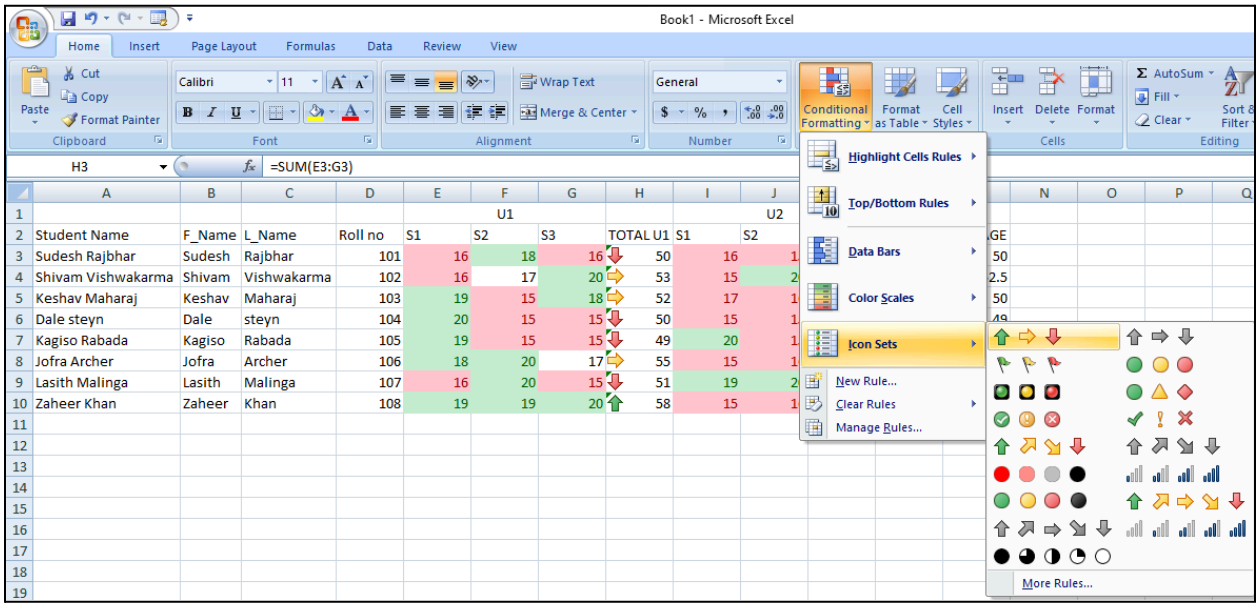
Similarly apply Greater than on same values

Student Name	F_Name	L_Name	Roll no	S1	S2	S3	TOTAL U1
Sudesh Rajbhar	Sudesh	Rajbhar	101	16	18	16	50
Shivam Vishwakarma	Shivam	Vishwakarma	102	16	17	20	53
Keshav Maharaj	Keshav	Maharaj	103	19	15	18	52
Dale steyn	Dale	steyn	104	20	15	15	50
Kagiso Rabada	Kagiso	Rabada	105	19	15	15	49
Jofra Archer	Jofra	Archer	106	18	20	17	55
Lasith Malinga	Lasith	Malinga	107	16	20	15	51
Zaheer Khan	Zaheer	Khan	108	19	19	20	58

Do the same thing on values of Q2

				U1				U2				
Student Name	F_Name	L_Name	Roll no	S1	S2	S3	TOTAL U1	S1	S2	S3	TOTAL U2	AVERAGE
Sudesh Rajbhar	Sudesh	Rajbhar	101	16	18	16	50	16	15	19	50	50
Shivam Vishwakarma	Shivam	Vishwakarma	102	16	17	20	53	15	20	17	52	52.5
Keshav Maharaj	Keshav	Maharaj	103	19	15	18	52	17	16	15	48	50
Dale steyn	Dale	steyn	104	20	15	15	50	15	15	18	48	49
Kagiso Rabada	Kagiso	Rabada	105	19	15	15	49	20	15	19	54	51.5
Jofra Archer	Jofra	Archer	106	18	20	17	55	15	16	18	49	52
Lasith Malinga	Lasith	Malinga	107	16	20	15	51	19	20	18	57	54
Zaheer Khan	Zaheer	Khan	108	19	19	20	58	15	16	16	47	52.5

Select values of Total Q1 -> Home tab -> Conditional Formatting -> Icon Sets (any desirable icon set)



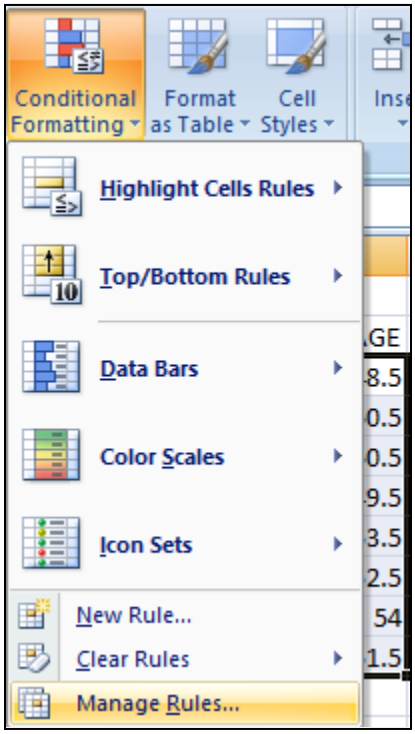
Similarly on Total Q2

				U1				U2				
Student Name	F_Name	L_Name	Roll no	S1	S2	S3	TOTAL U1	S1	S2	S3	TOTAL U2	AVERAGE
Sudesh Rajbhar	Sudesh	Rajbhar	101	16	18	16	50	16	15	19	50	50
Shivam Vishwakarma	Shivam	Vishwakarma	102	16	17	20	53	15	20	17	52	52.5
Keshav Maharaj	Keshav	Maharaj	103	19	15	18	52	17	16	15	48	50
Dale steyn	Dale	steyn	104	20	15	15	50	15	15	18	48	49
Kagiso Rabada	Kagiso	Rabada	105	19	15	15	49	20	15	19	54	51.5
Jofra Archer	Jofra	Archer	106	18	20	17	55	15	16	18	49	52
Lasith Malinga	Lasith	Malinga	107	16	20	15	51	19	20	18	57	54
Zaheer Khan	Zaheer	Khan	108	19	19	20	58	15	16	16	47	52.5

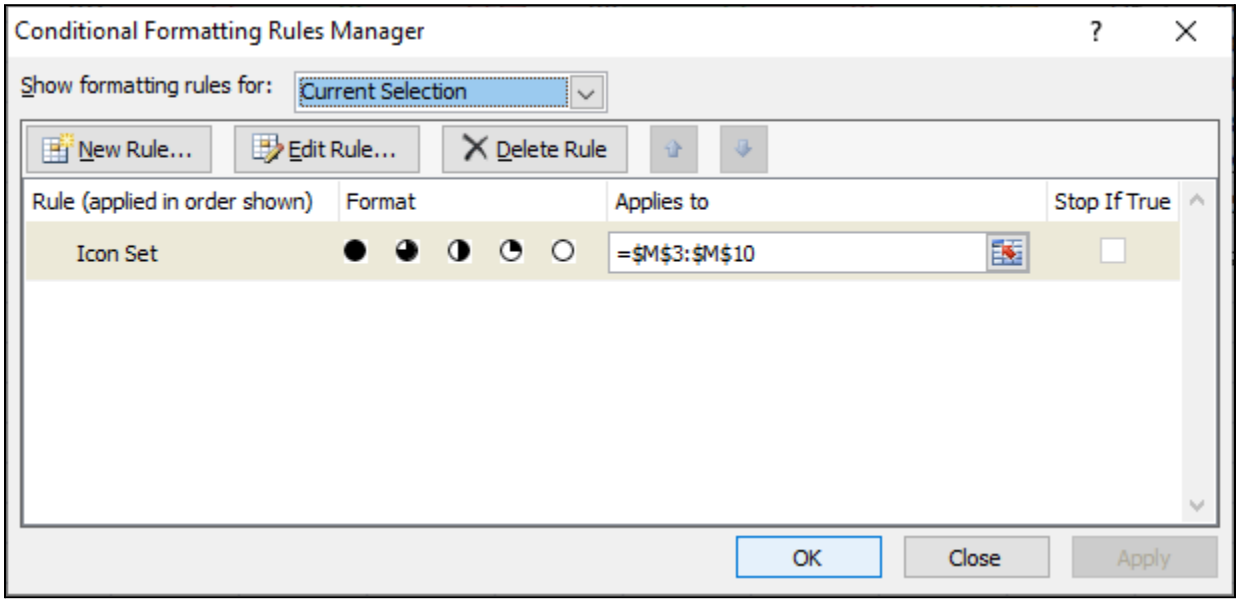
Do the same thing for Average column

Student Name	F_Name	L_Name	Roll no	S1	S2	S3	TOTAL U1	S1	S2	S3	TOTAL U2	AVERAGE
Sudesh Rajbhar	Sudesh	Rajbhar	101	18	15	17	50	15	15	17	47	48.5
Shivam Vishwakarma	Shivam	Vishwakarma	102	15	18	15	48	19	16	18	53	50.5
Keshav Maharaj	Keshav	Maharaj	103	19	15	19	53	15	17	16	48	50.5
Dale steyn	Dale	steyn	104	20	15	15	50	17	15	17	49	49.5
Kagiso Rabada	Kagiso	Rabada	105	16	19	20	55	17	19	16	52	53.5
Jofra Archer	Jofra	Archer	106	18	15	19	52	19	15	19	53	52.5
Lasith Malinga	Lasith	Malinga	107	15	18	18	51	19	20	18	57	54
Zaheer Khan	Zaheer	Khan	108	15	18	16	49	17	18	19	54	51.5

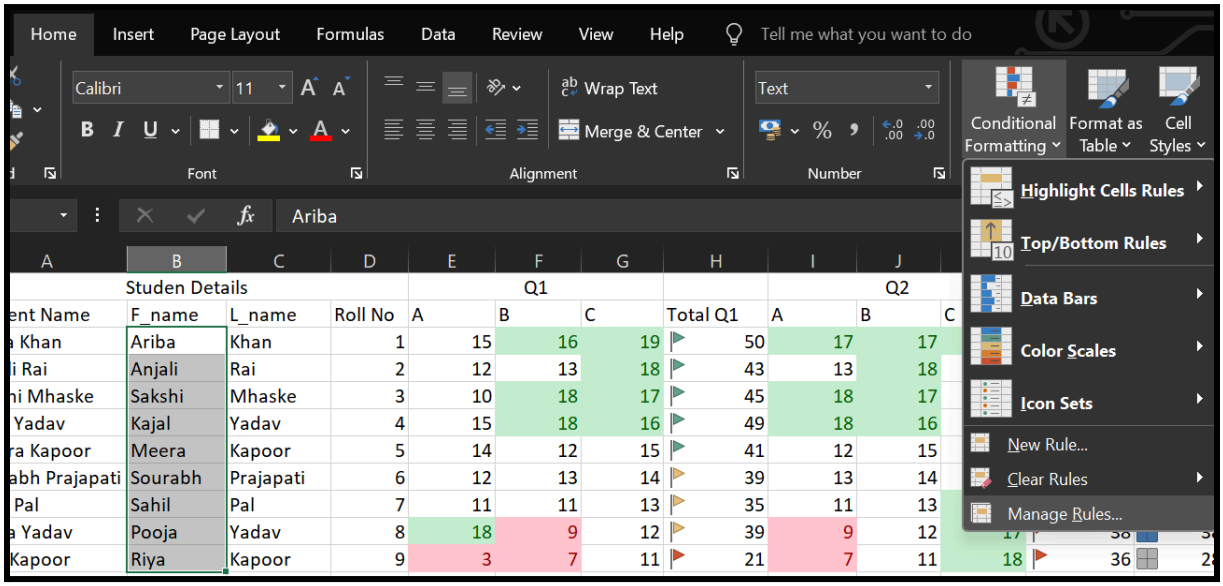
Select values of the Average column -> Conditional Formatting (Home tab) -> Click Manage Rules



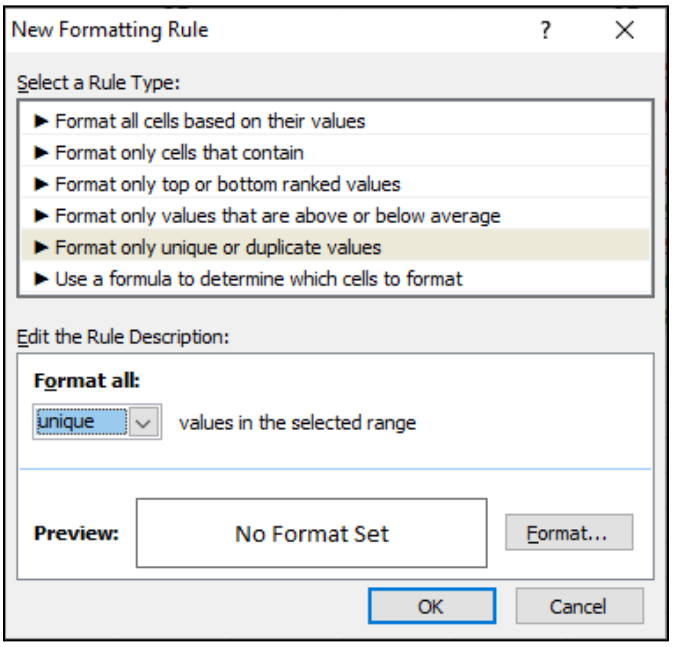
You can also edit from here



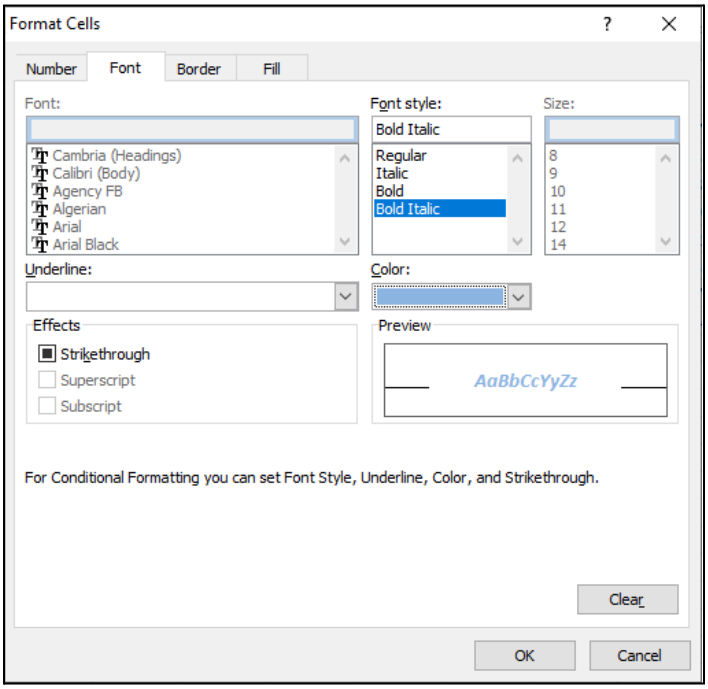
Select values of F_name -> Conditional Formatting (Home tab) -> Manage rules -> New Rule



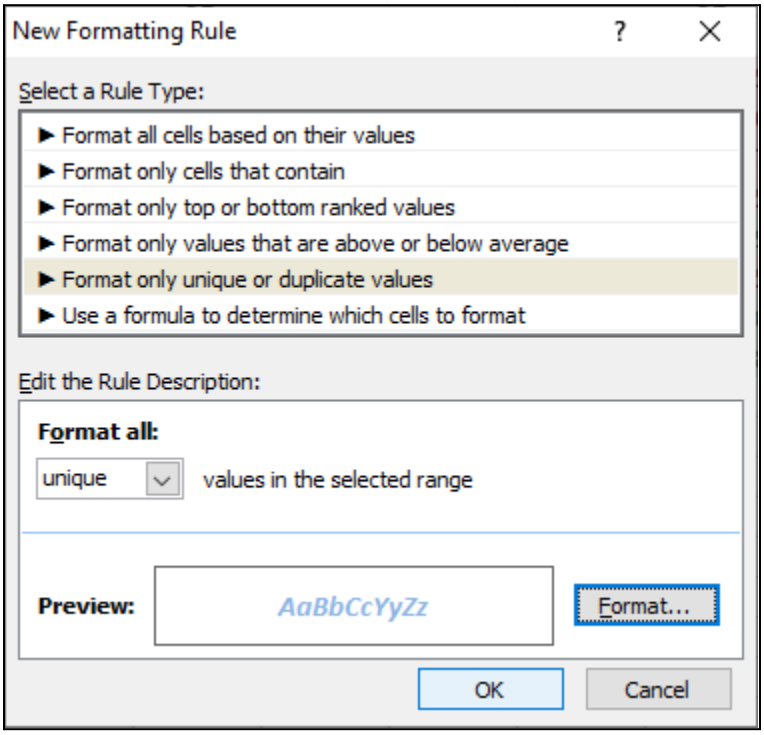
Select the following and click Format



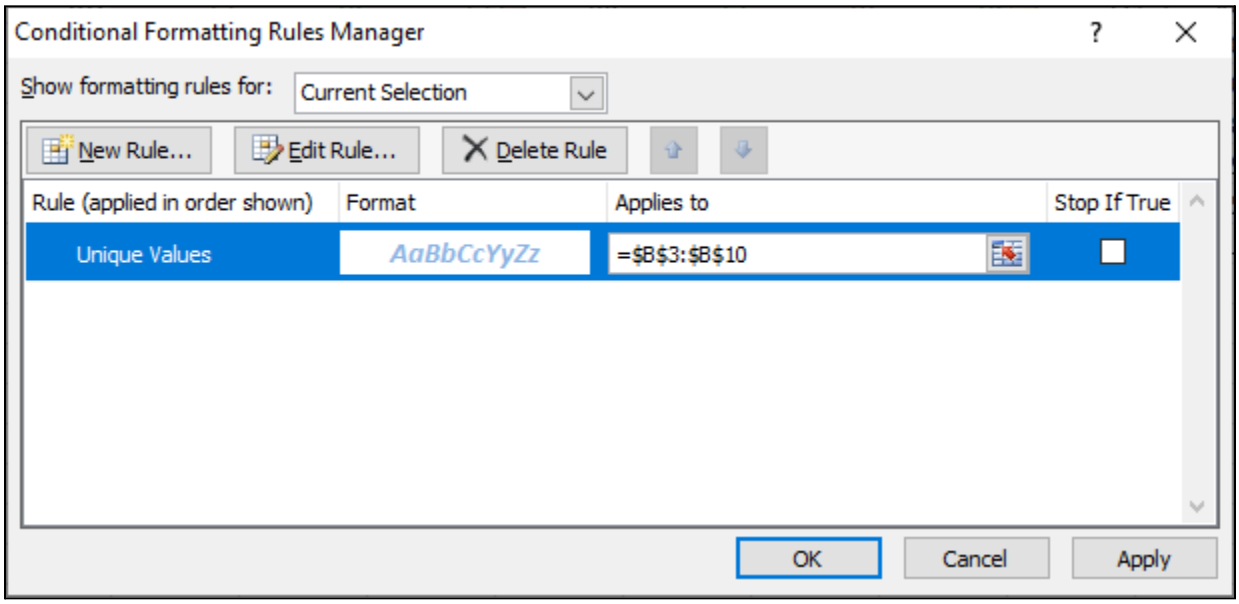
Make changes as you want -> Click OK



Click Ok



Click Apply -> Click OK

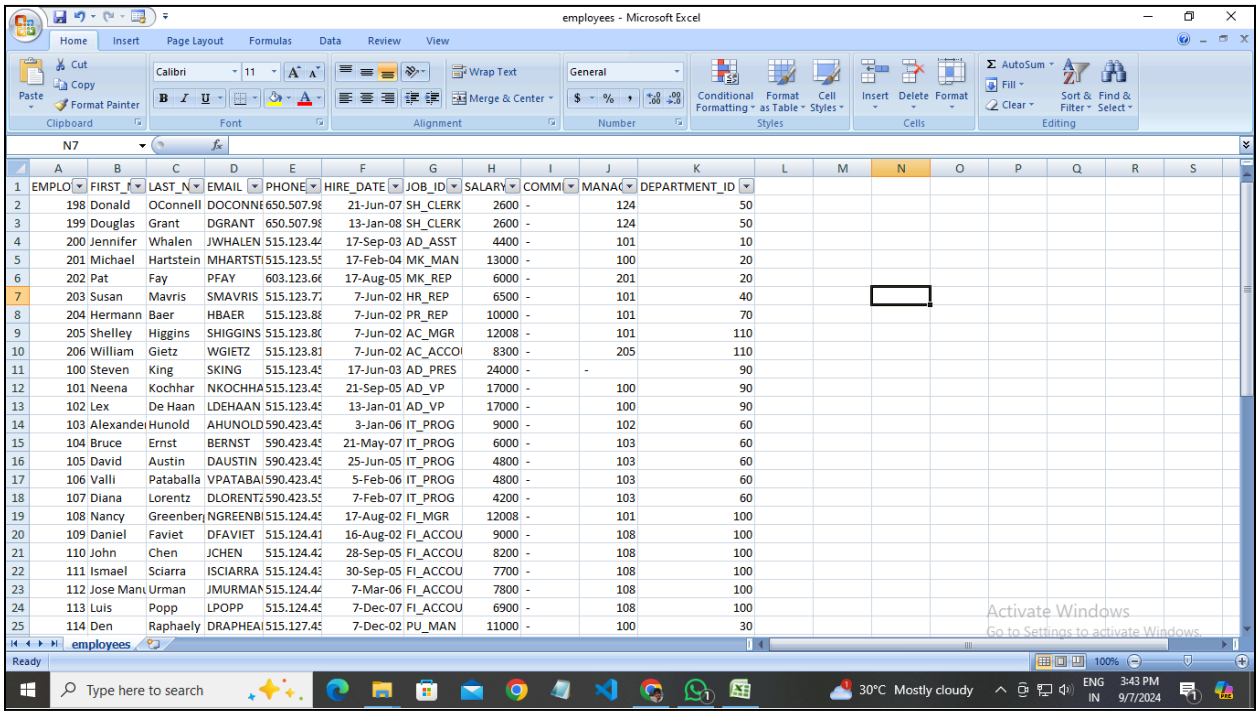


Result:

				U1				U2				
Student Name	F_Name	L_Name	Roll no	S1	S2	S3	TOTAL U1	S1	S2	S3	TOTAL U2	AVERAGE
Sudesh Rajbhar	*Sudesh*	Rajbhar	101	18	15	17	50	15	15	17	47	48.5
Shivam Vishwakarma	*Shivam*	Vishwakarma	102	15	18	15	48	19	16	18	53	50.5
Keshav Maharaj	*Keshav*	Maharaj	103	19	15	19	53	15	17	16	48	50.5
Dale steyn	*Dale*	steyn	104	20	15	15	50	17	15	17	49	49.5
Kagiso Rabada	*Kagiso*	Rabada	105	16	19	20	55	17	19	16	52	53.5
Jofra Archer	*Jofra*	Archer	106	18	15	19	52	19	15	19	53	52.5
Lasith Malinga	*Lasith*	Malinga	107	15	18	18	51	19	20	18	57	54
Zaheer Khan	*Zaheer*	Khan	108	15	18	16	49	17	18	19	54	51.5

Employee data csv file :
https://gist.github.com/kevin336/acbb2271e66c10a5b73aacf82ca82784

Open the csv file



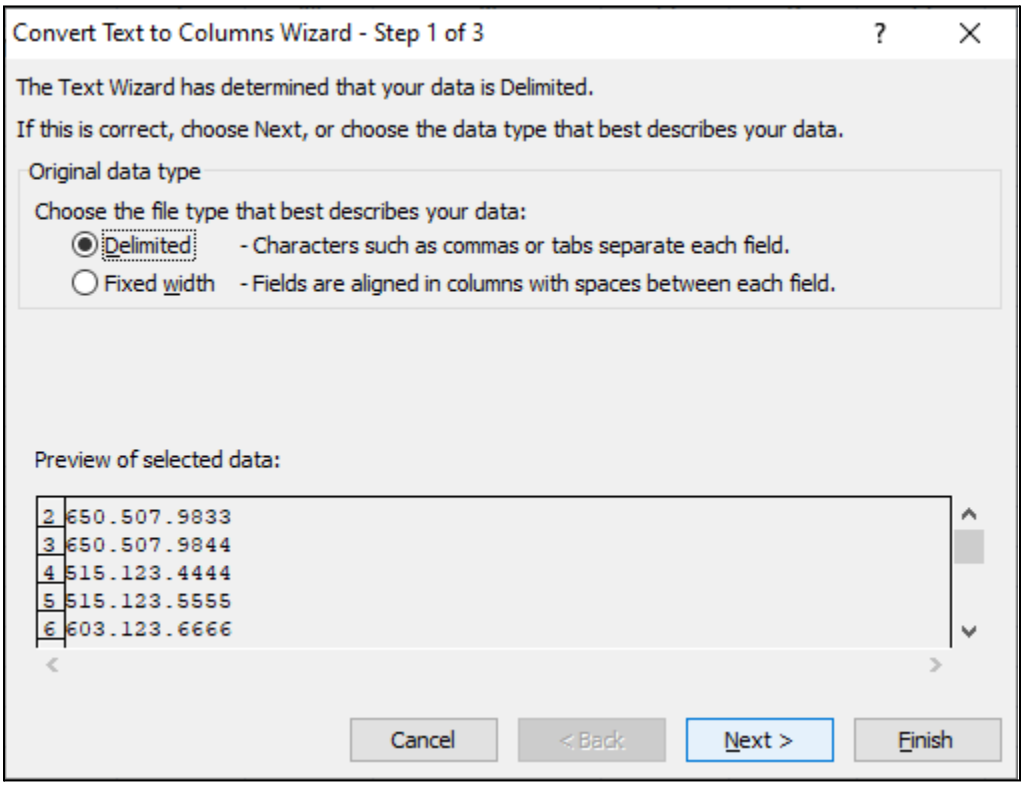
1. Total employee

TOTAL EMPLOYEE	50

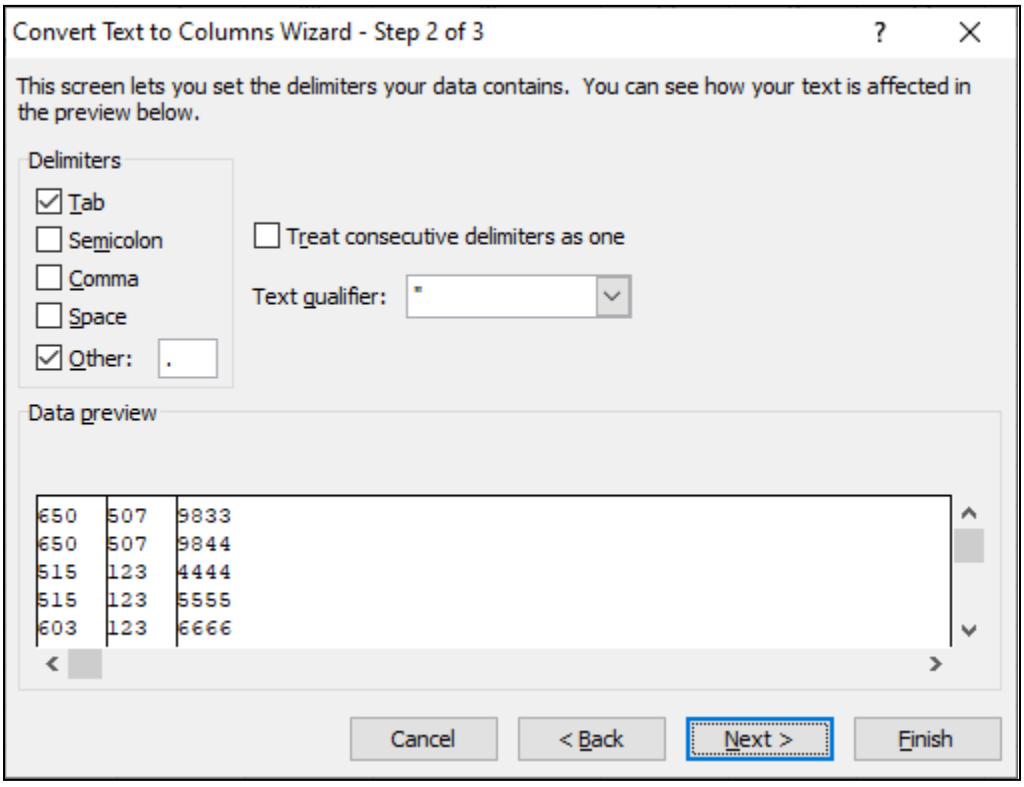
```
=COUNT(A2:A51)
```

2. List the Emp with region code 650

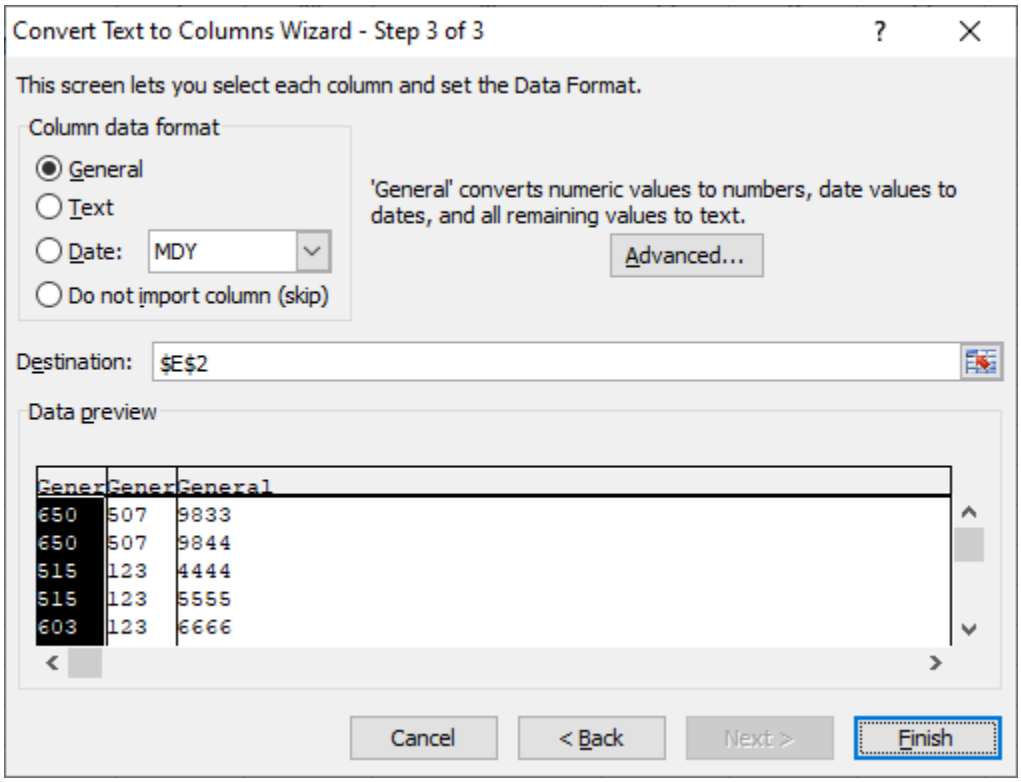
Do the following steps



Enter dot in Other



Enter destination

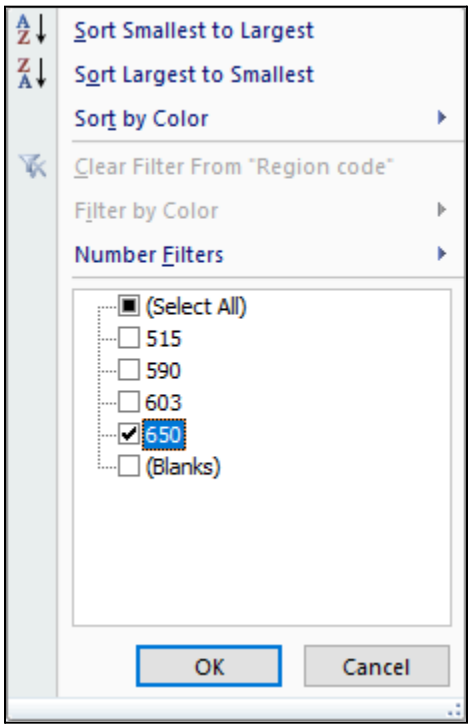


Give the column name as Region code

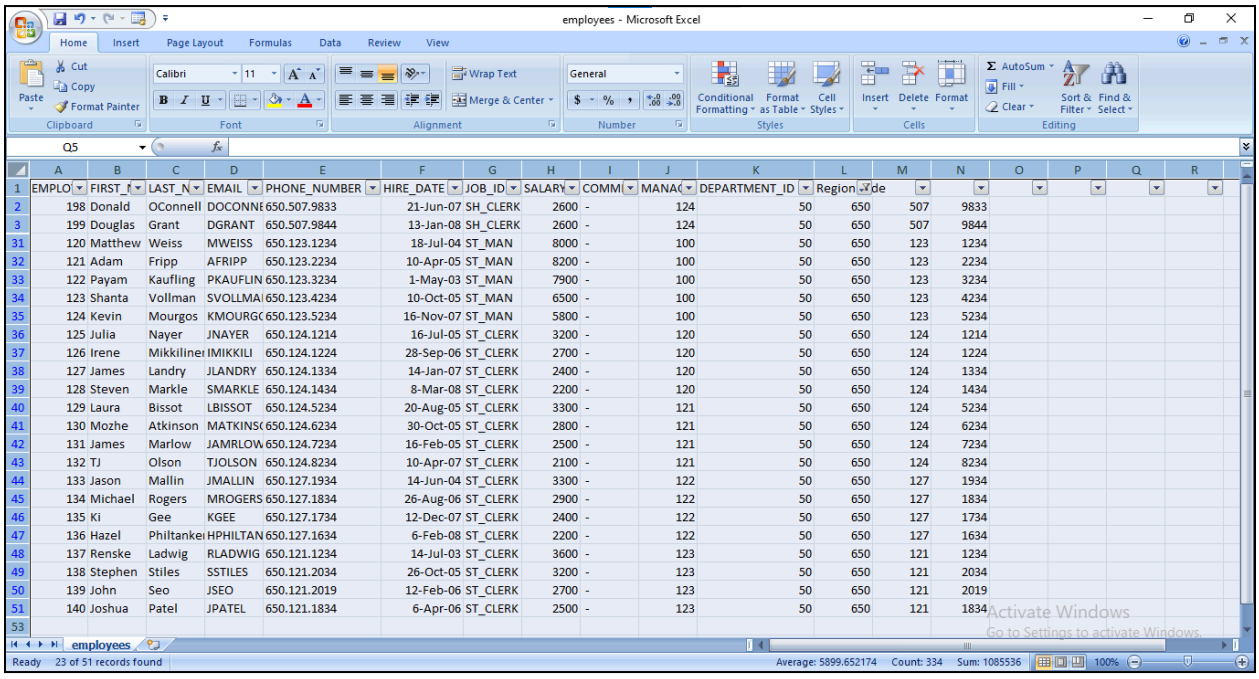
P	Q	R
650	507	9833
650	507	9844
515	123	4444
515	123	5555
603	123	6666
515	123	7777
515	123	8888
515	123	8080
515	123	8181
515	123	4567
515	123	4568
515	123	4569
590	423	4567
590	423	4568
590	423	4569
590	423	4560

Select values of Region code column -> Right click -> Filter -> Filter by Selected Cell's Value

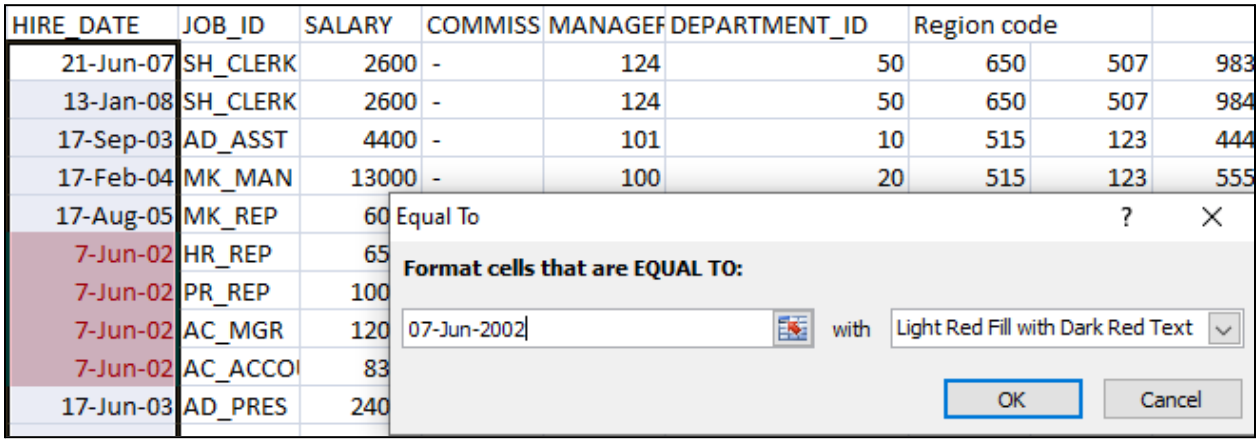
Click on Filter icon -> Select 650 -> Click OK



Results:



3. Highlight the emp with the joining date as 7 june 2002
Do the following steps



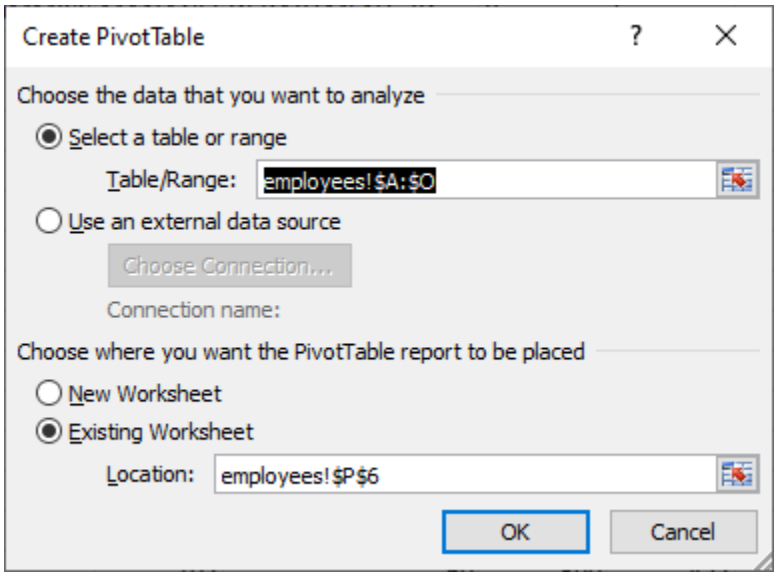
HIRE_DATE	JOB_ID	SALARY	COMMISS	MANAGEF	DEPARTMENT_ID	Region code		
21-Jun-07	SH_CLERK	2600	-	124	50	650	507	983
13-Jan-08	SH_CLERK	2600	-	124	50	650	507	984
17-Sep-03	AD_ASST	4400	-	101	10	515	123	444
17-Feb-04	MK_MAN	13000	-	100	20	515	123	555
17-Aug-05	MK_REP	60						
7-Jun-02	HR_REP	65						
7-Jun-02	PR_REP	100						
7-Jun-02	AC_MGR	120						
7-Jun-02	AC_ACCOI	83						
17-Jun-03	AD_PRES	240						

4. Get year wise salary trend
Create a new column Year

YEAR
2010
2004
2005
2010
2008
2009
2004
2007
2006
2003
2005
2009

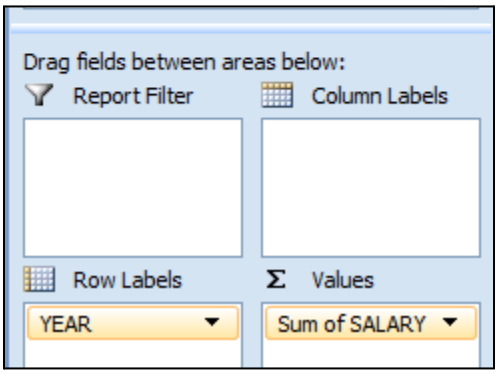
Select entire sheet -> Insert tab -> Pivot Table -> From Table/Range

Click OK



Drag Year column to rows and Salary column to Values

Row Labels	Sum of SALARY
2003	15208
2004	67800
2005	33900
2006	67500
2007	10400
2008	20500
2009	33608
2010	60200
Grand Total	**309116**



5. List Designations with Highest and Lowest Salary

Highest Salary

24000
MAX(H2:H51)

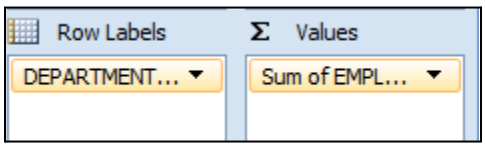
Highest Job Id
AD_PRES
INDEX(G2:G51, MATCH(MAX(H2:H51), H2:H51, 0))

6. List emp who are working under manager with manager id 114
Do the following steps (similar to steps of filter)

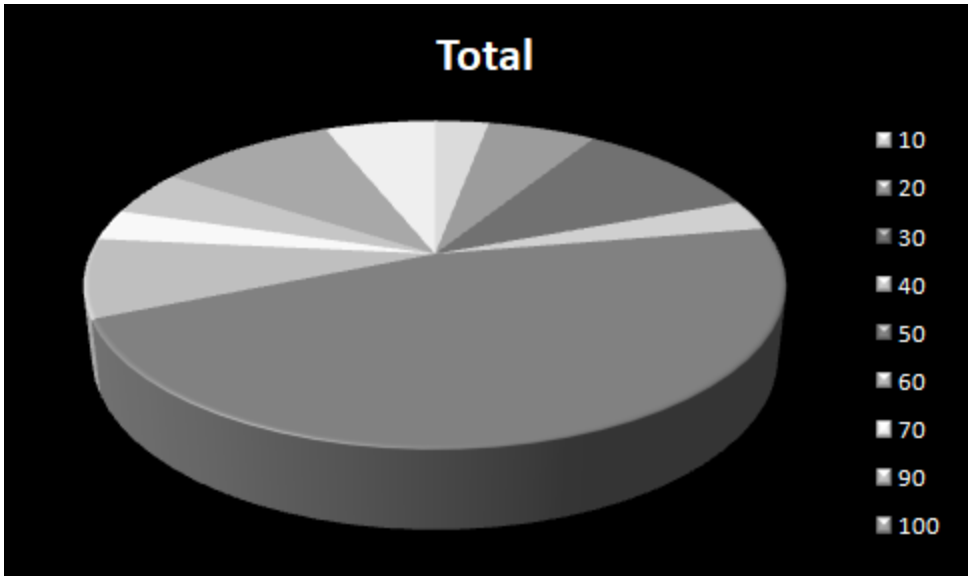
EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID
198	Donald	OConnell	DOCONNEL	650.507.9833	21-Jun-07	SH_CLERK	2600	-	1
115	Alexander	Khoo	AKHOO	515.127.4562	18-May-03	PU_CLERK	3100	-	114
116	Shelli	Baida	SBAIDA	515.127.4563	24-Dec-05	PU_CLERK	2900	-	114
117	Sigal	Tobias	STOBIAS	515.127.4564	24-Jul-05	PU_CLERK	2800	-	114
118	Guy	Himuro	GHIMURO	515.127.4565	15-Nov-06	PU_CLERK	2600	-	114
119	Karen	Colmenares	KCOLMENA	515.127.4566	10-Aug-07	PU_CLERK	2500	-	114

7. Count Employees in Each Department
Drag Department_ID in rows and Count of Employee_ID in values

Row Labels	Count of EMPLOYEE_ID
10	1
20	2
30	6
40	1
50	23
60	5
70	1
90	3
100	6
110	2
Grand Total	**50**

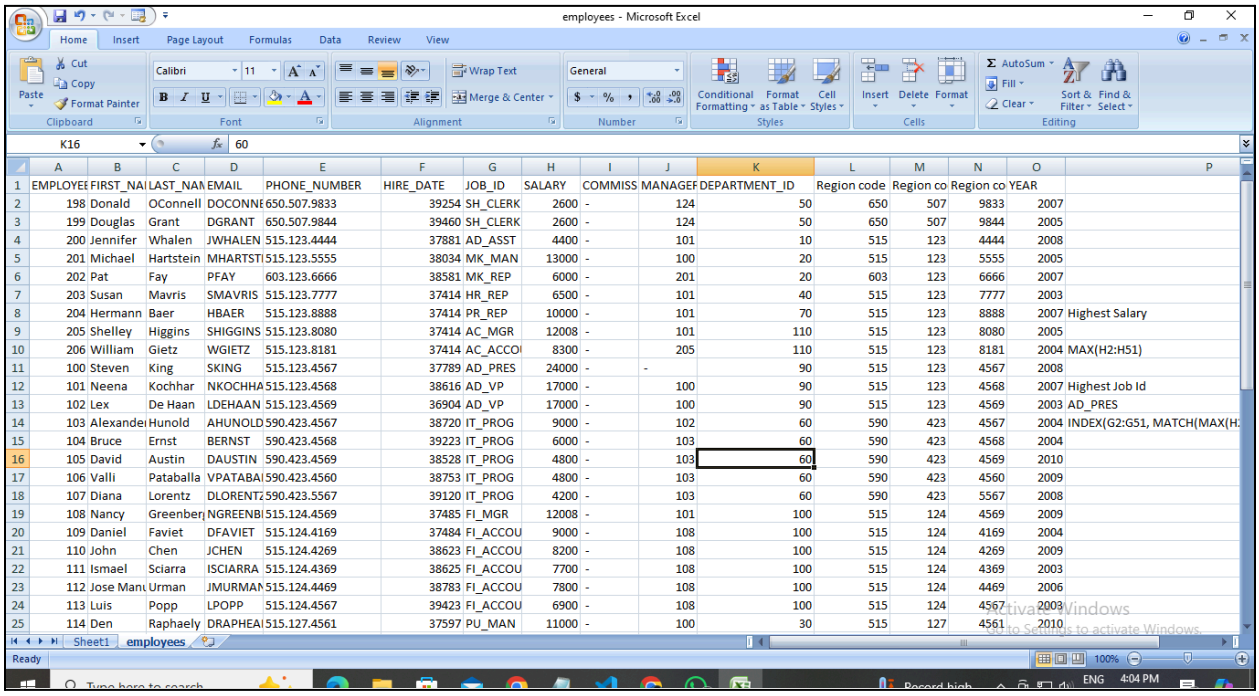


8. Plot Employees of Each Department in a Pie Chart
Insert tab -> Pie chart



9. Display all salaries with the data.

Displaying whole data without any filter or formatting



EMPLOYEE	FIRST_NAI	LAST_NAN	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISS	MANAGEF	DEPARTMENT_ID	Region code	Region co	Region co	YEAR	
198	Donald	OConnell	DOCONNE	650.507.9833	39254	SH_CLERK	2600	-	124	50	650	507	9833	2007	
199	Douglas	Grant	DGRANT	650.507.9844	39460	SH_CLERK	2600	-	124	50	650	507	9844	2005	
200	Jennifer	Whalen	JWHALEN	515.123.4444	37881	AD_ASST	4400	-	101	10	515	123	4444	2008	
201	Michael	Hartstein	MHARTST	515.123.5555	38034	MK_MAN	13000	-	100	20	515	123	5555	2005	
202	Pat	Fay	PFAY	603.123.6666	38581	MK_REP	6000	-	201	20	603	123	6666	2007	
203	Susan	Mavris	SMAVRIS	515.123.7777	37414	HR_REP	6500	-	101	40	515	123	7777	2003	
204	Hermann	Baer	HBAER	515.123.8888	37414	PR_REP	10000	-	101	70	515	123	8888	2007	Highest Salary
205	Shelley	Higgins	SHIGGINS	515.123.8080	37414	AC_MGR	12008	-	101	110	515	123	8080	2005	
206	William	Gietz	WGIETZ	515.123.8181	37414	AC_ACCO	8300	-	205	110	515	123	8181	2004	MAX(H2:H51)
100	Steven	King	SKING	515.123.4567	37789	AD_PRES	24000	-	-	90	515	123	4567	2008	
101	Neena	Kochhar	NKOCHHA	515.123.4568	38616	AD_VP	17000	-	100	90	515	123	4568	2007	Highest Job Id
102	Lex	De Haan	LDEHAAN	515.123.4569	36904	AD_VP	17000	-	100	90	515	123	4569	2003	AD_PRES
103	Alexande	Hunold	AHUNOLD	590.423.4567	38720	IT_PROG	9000	-	102	60	590	423	4567	2004	INDEX(G2:G51, MATCH(MAX(H:
104	Bruce	Ernst	BERNST	590.423.4568	39223	IT_PROG	6000	-	103	60	590	423	4568	2004	
105	David	Austin	DAUSTIN	590.423.4569	38528	IT_PROG	4800	-	103	60	590	423	4569	2010	
106	Valli	Pataballa	VPATABA	590.423.4560	38753	IT_PROG	4800	-	103	60	590	423	4560	2009	
107	Diana	Lorentz	DLORENT	590.423.5567	39120	IT_PROG	4200	-	103	60	590	423	5567	2008	
108	Nancy	Greenber	NGREENB	515.124.4569	37485	FI_MGR	12008	-	101	100	515	124	4569	2009	
109	Daniel	Faviet	DFAVIET	515.124.4169	37484	FI_ACCOU	9000	-	108	100	515	124	4169	2004	
110	John	Chen	JCHEN	515.124.4269	38623	FI_ACCOU	8200	-	108	100	515	124	4269	2009	
111	Ismael	Sciarra	ISCIARRA	515.124.4369	38625	FI_ACCOU	7700	-	108	100	515	124	4369	2003	
112	Jose Man	Urman	JMURMAN	515.124.4469	38783	FI_ACCOU	7800	-	108	100	515	124	4469	2006	
113	Luis	Popp	LPOPP	515.124.4567	39423	FI_ACCOU	6900	-	108	100	515	124	4567	2003	
114	Den	Raphaely	DRAPHEA	515.127.4561	37597	PU_MAN	11000	-	100	30	515	127	4561	2010	