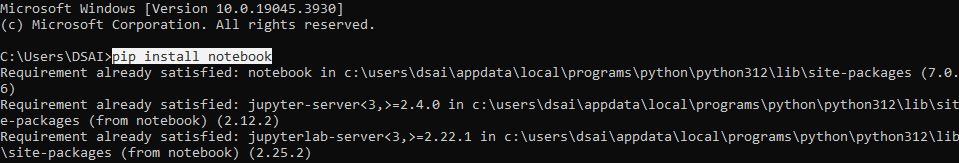
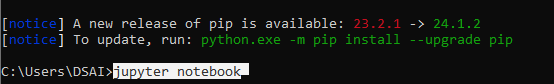
# Practical No : 1 [15/07/24]

Basics of numpy, pandas and matplotlib

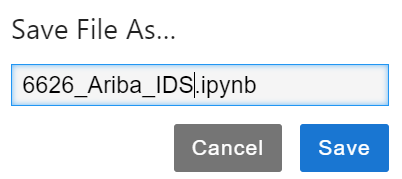
Run the following commands

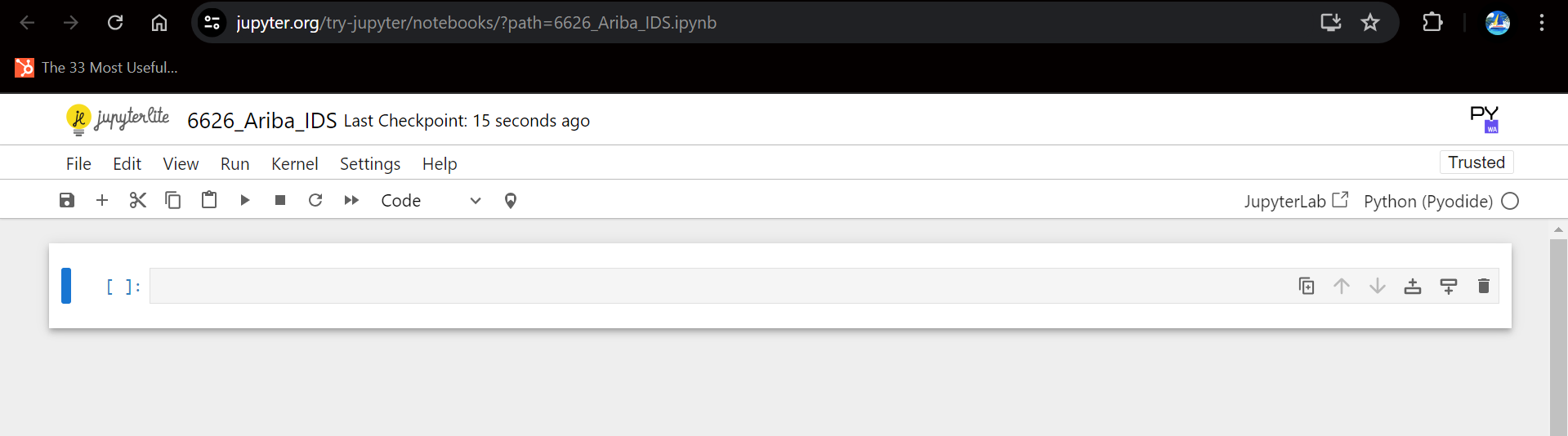






Give it a name and save it.



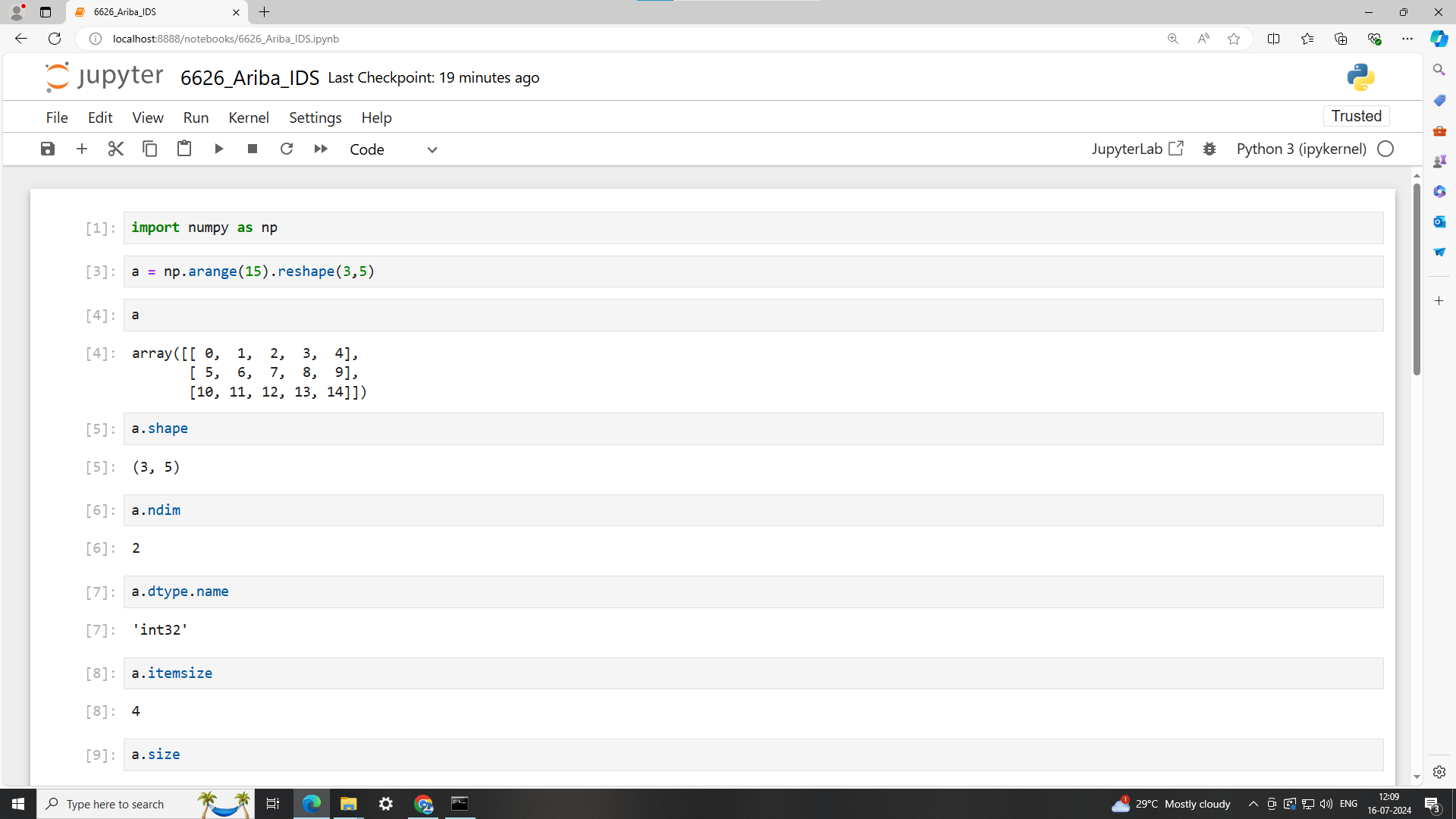


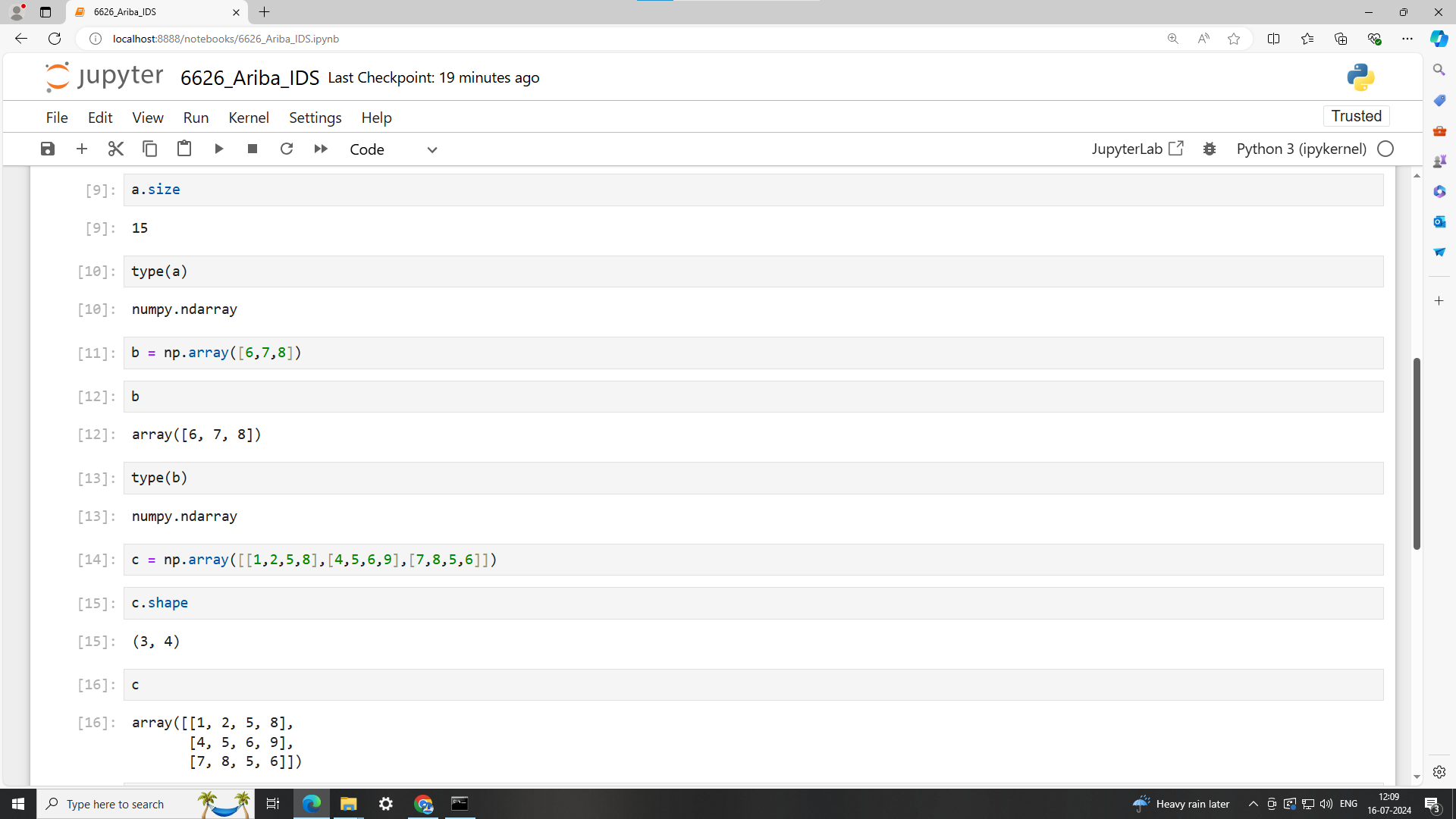
Numpy module : used for scientific computing in Python. Easy for storing metrics and arrays.

There are various array functions available in numpy as follows:

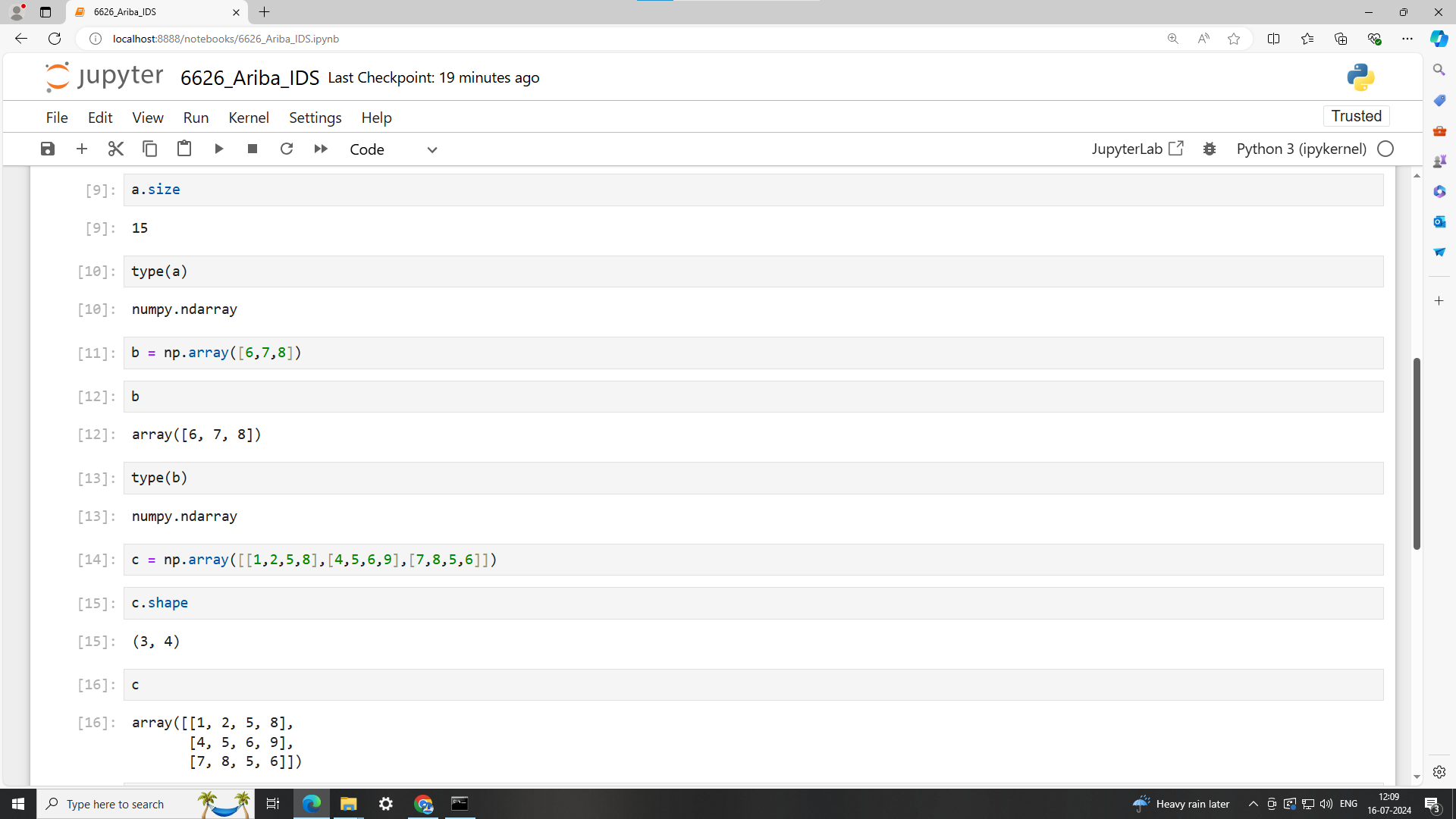
A. Function for array creation

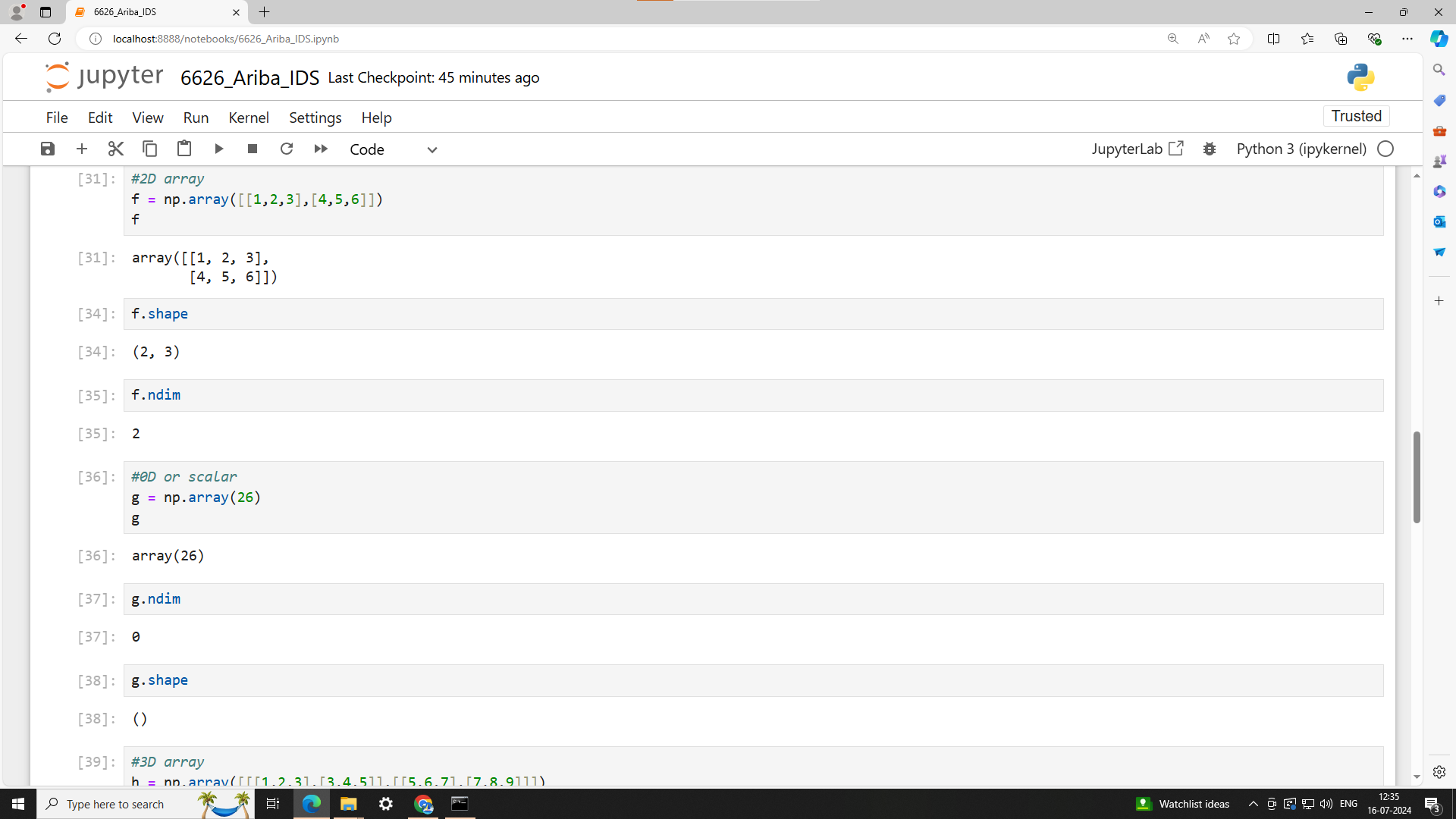
1. arange() : to create an array with a range of elements.

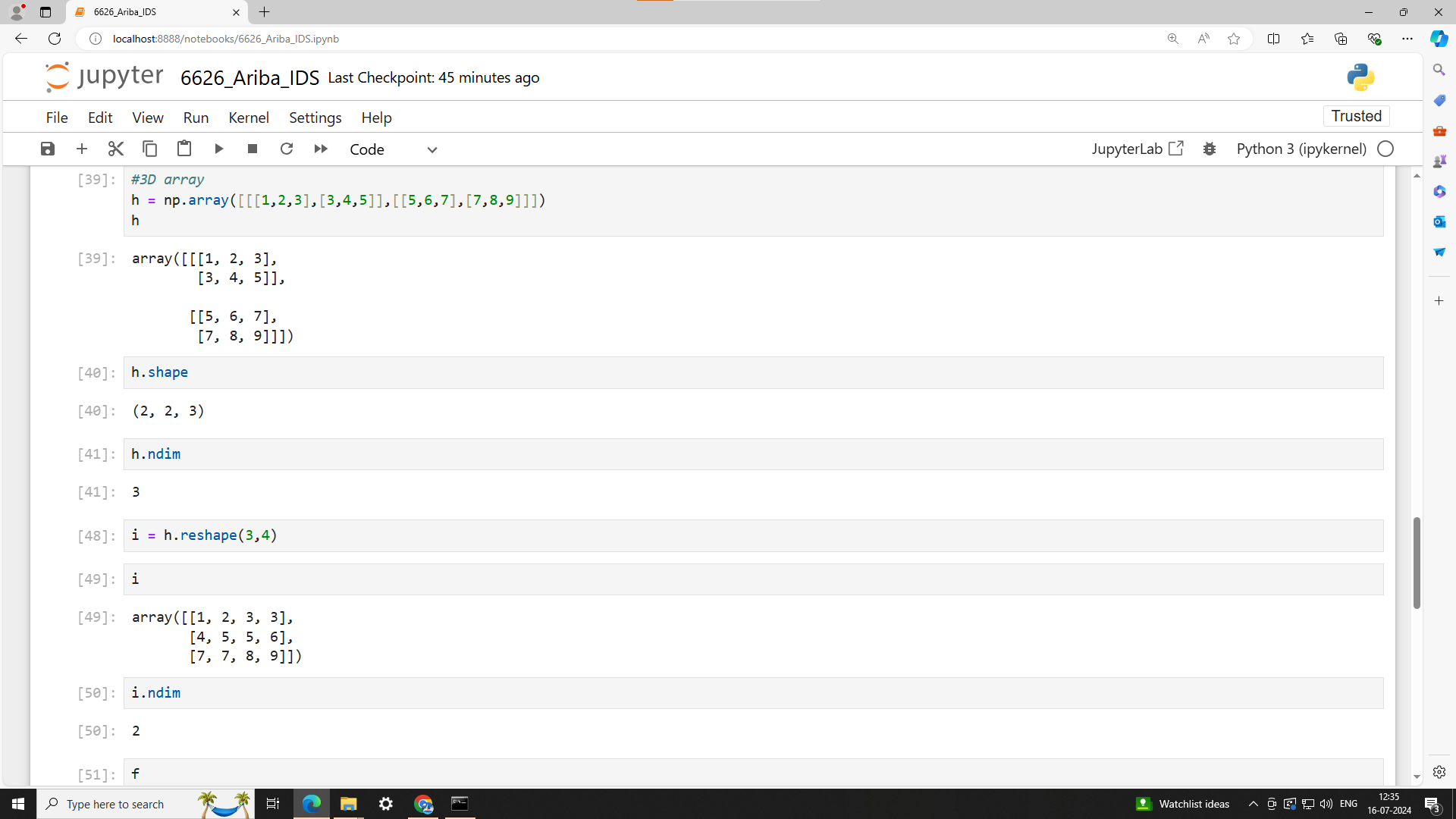


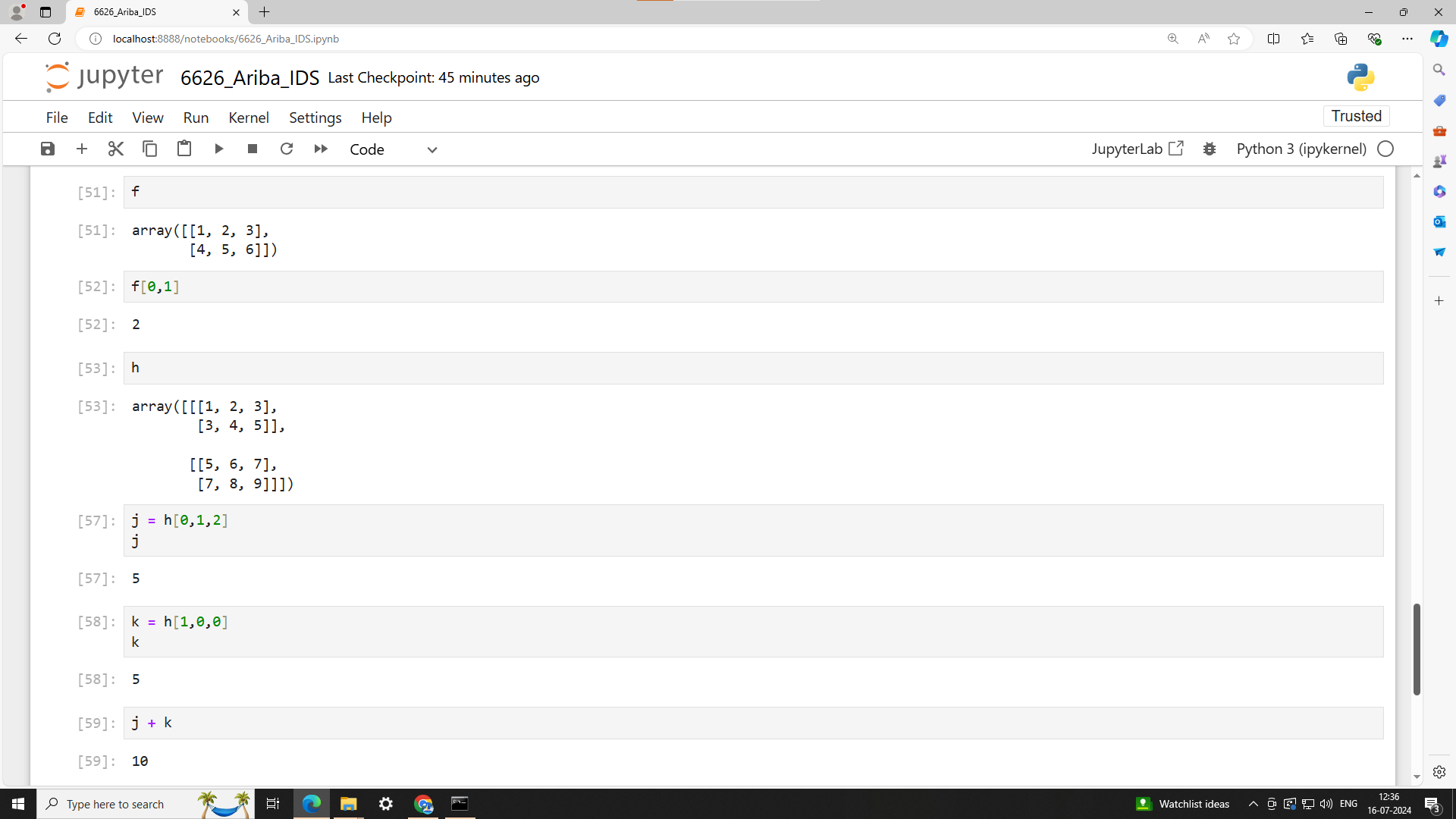


2. array() : A way to initialize an array is using a Python sequence, such as a list.

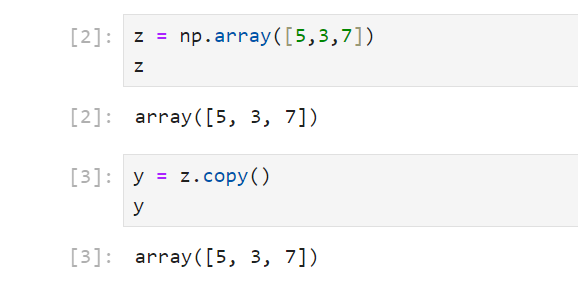




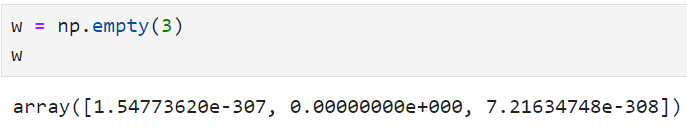




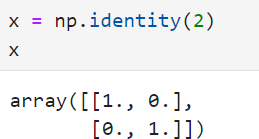
3. copy() : Using the copy method will make a complete copy of the array and its data. It is entirely independent of the original array.



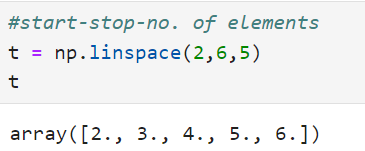
4. empty() : creates an array whose initial content is random and depends on the state of the memory. The reason to use empty over zeros (or something similar) is speed.

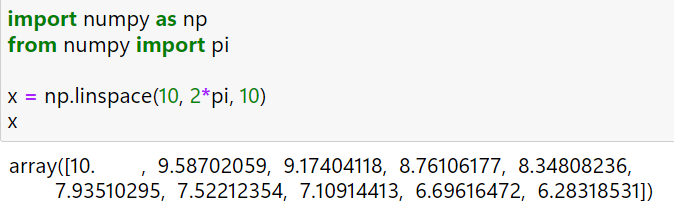


5. identity() : A square array with ones on the main diagonal.

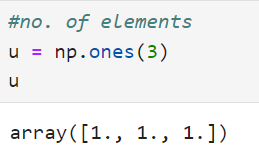


6. linspace() : to create an array with values that are spaced linearly in a specified interval.

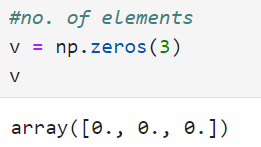




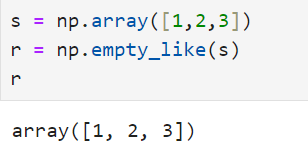
7. ones() : an array filled with 1’s



8. zeros() : to create an array filled with 0’s

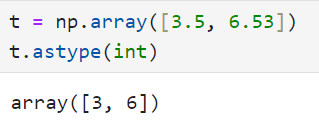


9. empty-like() : creates a new array with the same shape and data type as a given array.

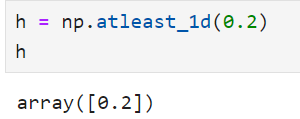


B. Conversions :

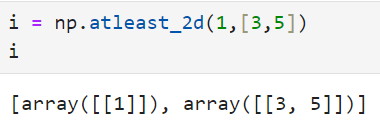
1. astype() : Copy of the array, cast to a specified type.



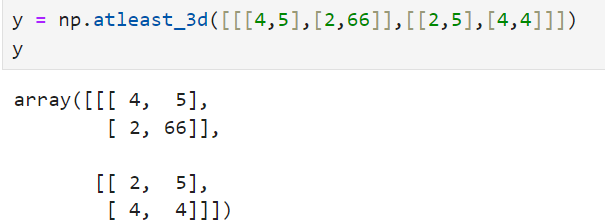
2. atleast\_1d() : Convert inputs to arrays with at least one dimension.



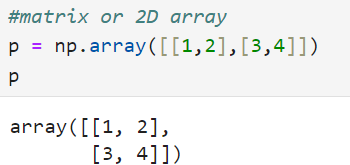
3. atleast\_2d() : View inputs as arrays with at least two dimensions.



4. atleast\_3d() : View inputs as arrays with at least three dimensions.



5. matrix

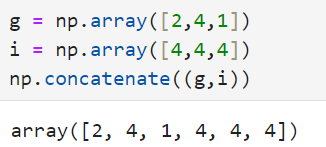


C.

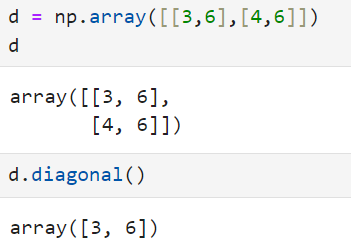
1. array\_split() : to split an array into multiple sub-arrays.



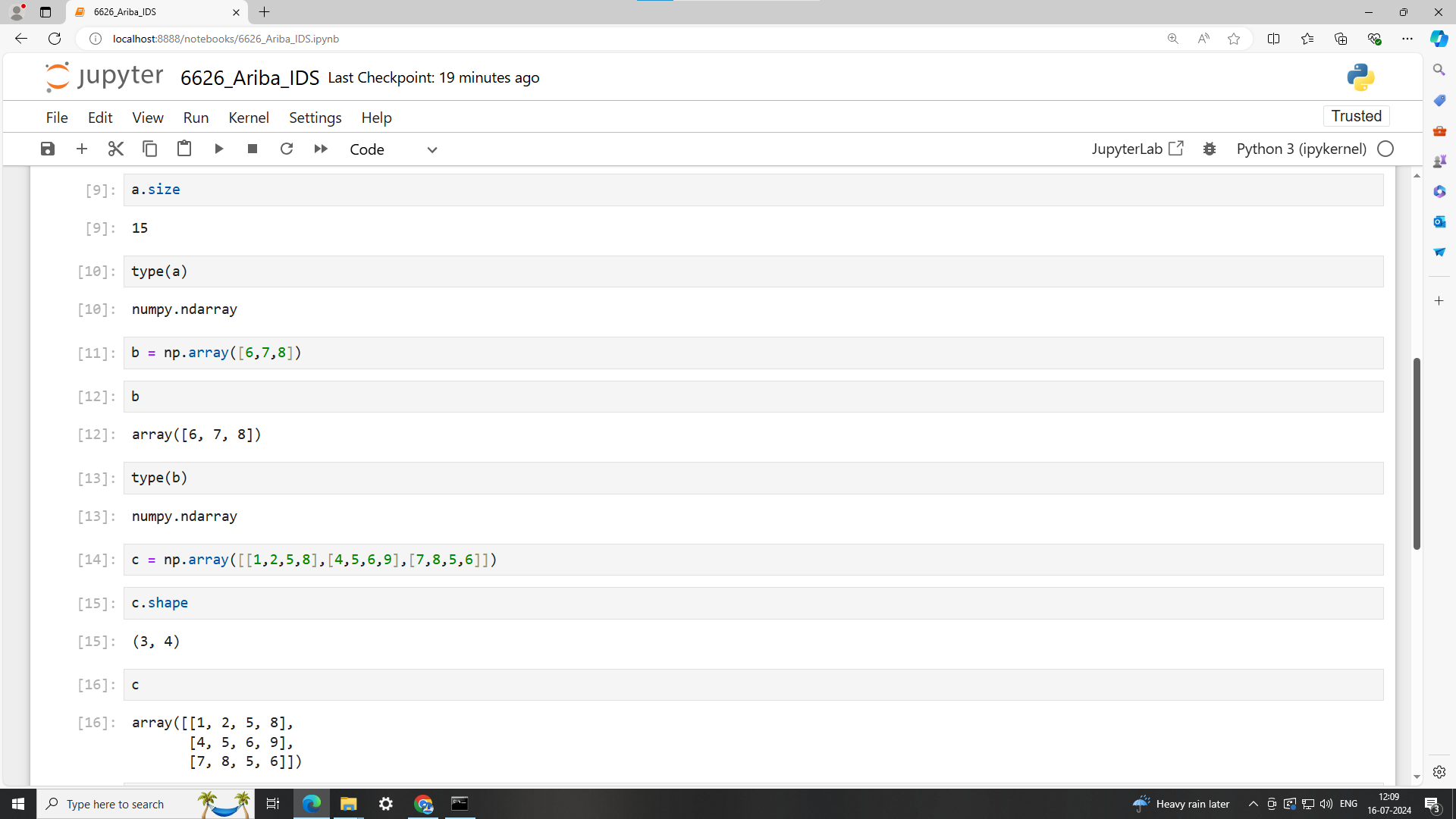
2. concatenate()

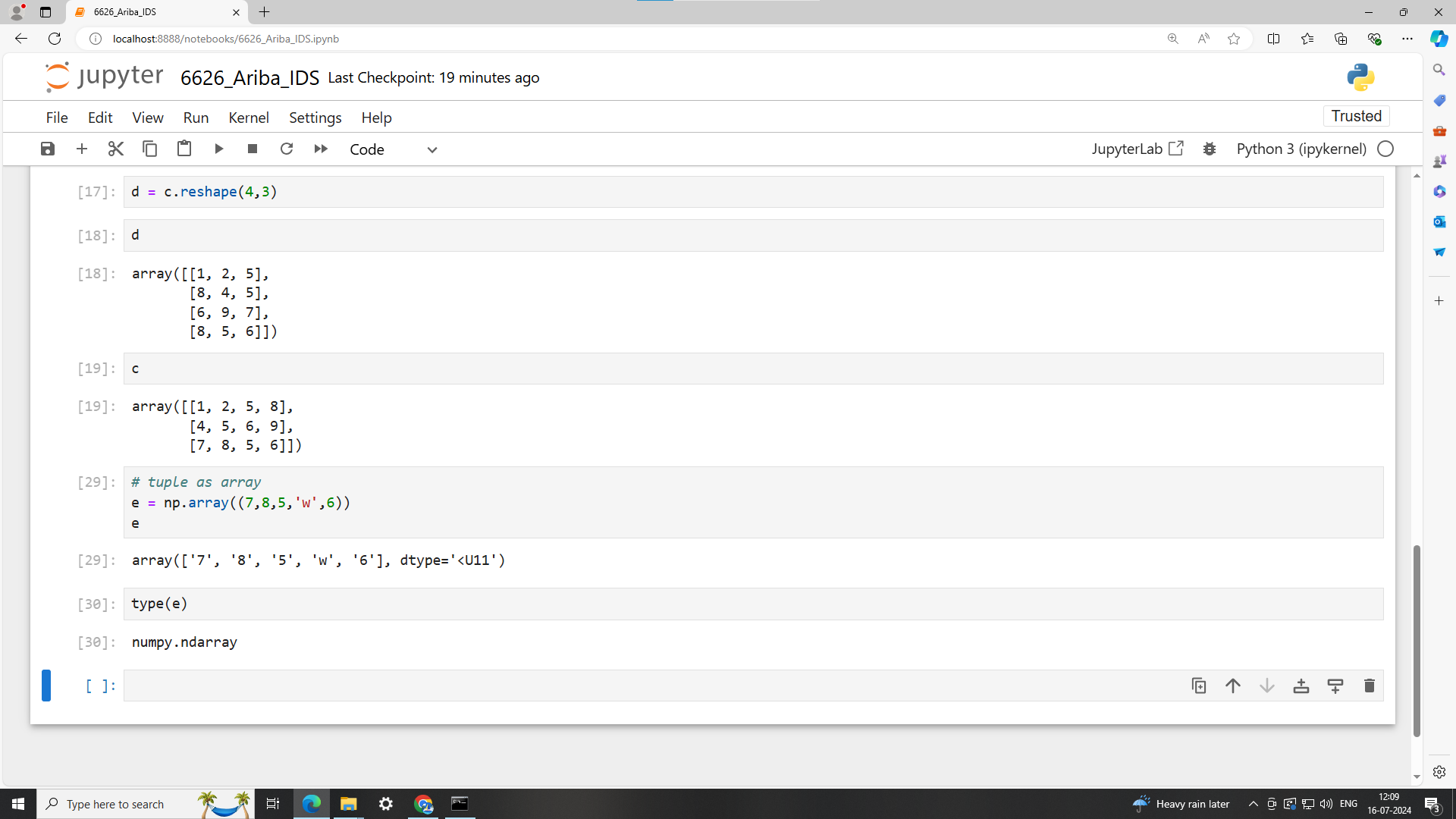


3. diagonal() : Return specified diagonals.

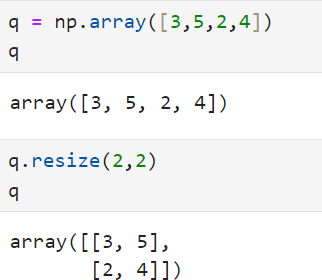


4. reshape() : to give a new shape to an array without changing the data.

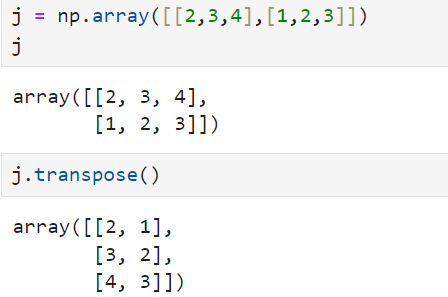




5. resize() : Return a new array with the specified shape.

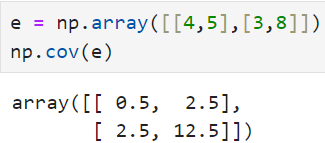


6. transpose() : To reverse or change the axes of an array

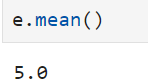


D. Basic statistics :

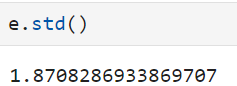
1. cov() : Estimate a covariance matrix, given data and weights.



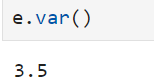
2. mean() : To get the average.



3. std() : To get the standard deviation.

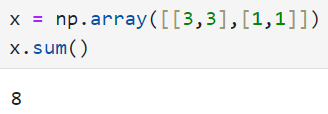


4. var()



E. Operations :

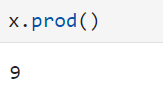
1. sum() : To find the sum of the elements in an array.



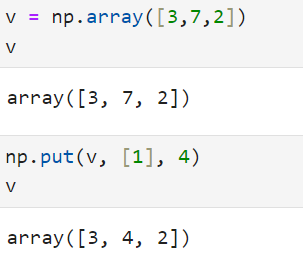
2. cumsum() : To find the cumulative of the elements in an array.



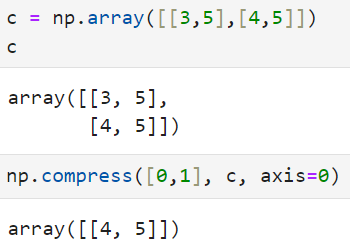
3. prod() : To get the result of multiplying the elements together.



4. put() : Replaces specified elements of an array with given values.

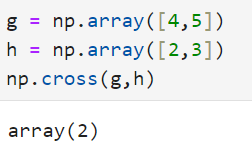


5. compress() : Return selected slices of an array along a given axis.

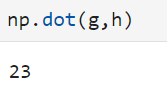


F. Linear Algebra :

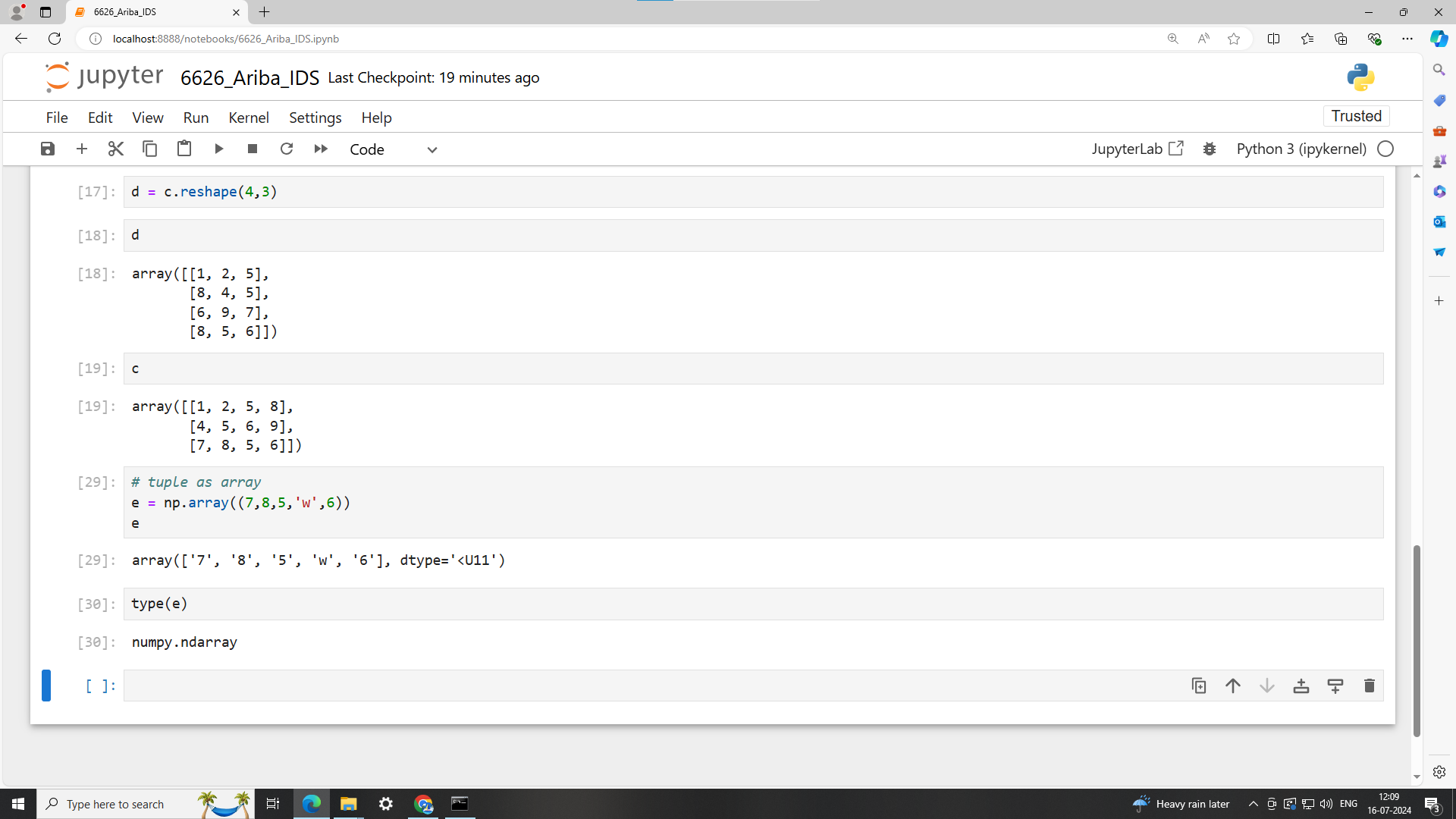
1. cross() : Return the cross product of two (arrays of) vectors.



2. dot() : Dot product of two arrays.



Tuple as array :



Q, Write a python code to store the given values in tuples and display it in the given format by using the user defined function named 'printValues'.

Values:

('Amit',45,'TY')

('Sumit',25,'SY')

('Anita',24,'FY')

Format:

Amit has 45.00% marks in the Class TY.

Code:

a = '{0:s} has {1:.2f}% marks in class {2:s}'

l1 = [('Amit', 45, 'TY'), ('Sumit', 25, 'SY'), ('Anita', 24, 'FY')]

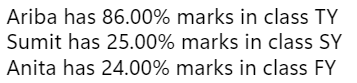
def printValue():

for x in l1:

z = a.format(x[0],x[1],x[2])

print(z)

printValue()



Mutable (list, dict, array) and Immutable (string and tuple)

Scalar types (str, bytes, float, bool and int)

Numeric types (int, float)

rand()

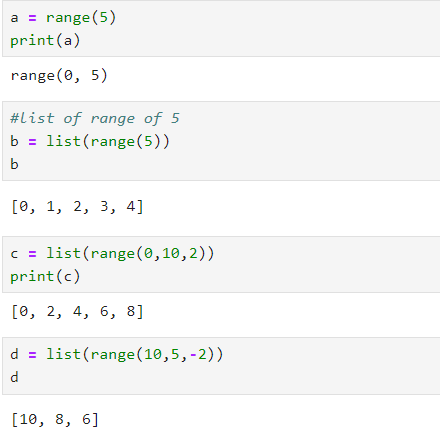
import numpy as np

a = {i : np.random.randn() for i in range(3)}

a



range() :



Q. Write a python code to store and display the square of odd numbers upto 10. Generate the odd numbers using the range() function.

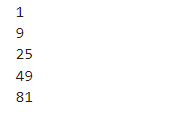
Code:

s = list(range(1,10,2))

for a in s:

v = a\*a

print(v)



#Create a python dictionary from the given sequence

kl = ['fy','sy','ty']

vl = [[1,2,3],[4,5,6],[7,8,9]]

mapping = {}

for key, value in zip(kl,vl):

mapping[key] = value

print(mapping)

#Display the list of keys of the resultant dictionary

print(mapping.keys())

#Display the list of values of the resultant dictionary

print(mapping.values())

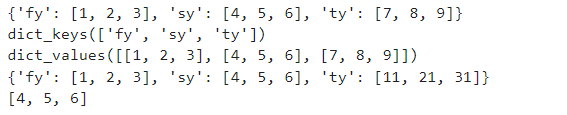
#Change the values of 'ty' to 11,21,31 using update() function

mapping.update({'ty':[11,21,31]})

print(mapping)

#Get the value of 'sy' key by using pop() function

print(mapping.pop('sy'))



#for the following data, create the dictionary of the alphabitwise list of fruit.

l = ['apple', 'banana', 'blue berry', 'apricot', 'orange', 'pine apple']

d = {}

for x in l:

i = x[0]

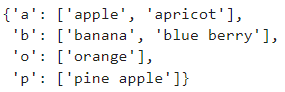
if i not in d:

d[i] = [x]

else:

d[i].append(x)

d



Bytes:

1. Represent raw data, like images, audio, or text encoded in a specific way.
2. Sequences of 8-bit values (0-255).
3. Eg : b'hello'.

Unicode:

1. Represent text in a universal way, supporting characters from all languages.
2. Sequences of Unicode code points.
3. Created using the str type: 'hello'.

Encoding and Decoding:

1. To convert Unicode strings to bytes, use the encode() method with a specific encoding like UTF-8: 'hello'.encode('utf-8').
2. To convert bytes to Unicode strings, use the decode() method with the appropriate encoding: b'hello'.decode('utf-8').

Code:

a = "Ariba"

print("Data type of a is : ",type(a))



b = a.encode('utf-8')

print("The value of b is : ",b)

print("Data type of b is : ",type(b))



c = b.decode('utf-8')

print("The value of c : ",c)

print("Data type of c is : ",type(c))



zip() : zip “pairs” up the elements of a number of lists, tuples, or other sequences to create a list of tuples

Code:

d = [6626,6433,3007]

e = ["Ariba","Shizuka","Mogli"]

print("Using zip() : ", list(zip(d,e)))



Set : unordered collection of values. Uses {} braces.

Dictionary : key and values

Set : only values

# define sets a and b

a = {1,2,3,4,5}

b = {3,4,5,6,7,8,9}

print('a : ',a)

print('b : ',b)

#adds an element

a.add(8)

print('add() : ',a)

#removing specific element

a.remove(4)

print('remove() : ',a)

#removes and returns an arbitrary element from the set. You cannot predict which element will be removed.

a.pop()

print('pop() : ',a)

#new set containing all elements

print('union() : ',a.union(b))

#adds all elements from set b to set a

a.update(b)

print('update() : ',a)

#new set containing elements that are common to both sets

a.intersection(b)

print('intersection() : ',a)

#updates set a to contain only elements that are common to both sets a and b

a.intersection\_update(b)

print('intersection\_update() : ',a)

#new set containing elements that are in set a but not in set b

a.difference(b)

print('difference() : ',a)

#updates set a to remove elements that are also in set b

a = {1, 2, 3, 4, 5}

a.difference\_update(b)

print("difference\_update() : ",a)

#new set containing elements that are in either set a or set b but not in both

a.symmetric\_difference(b)

print("symmetric\_difference() : ",a)

#updates set a to contain elements that are in either set a or set b but not in both

a.symmetric\_difference\_update(b)

print("symmetric\_difference\_update() : ",a)

#checks if all elements of set a are contained in set b

print("issubset() : ",a.issubset(b))

#checks if set a contains all elements of set b

print("issuperset() : ",a.issuperset(b))

#checks if sets a and b have no elements in common

print("isdisjoint() : ",a.isdisjoint(b))



#sum the numbers from the list 8, 9, 11, 23, 34, 10 that are greater than 10

l = [8, 9, 11, 23, 34, 10]

sum([x for x in l if x > 10])



#For the given list of strings store the length of each string into a set variable.

l = ['Ariba', 'writes']

s = set([])

v = []

for x in l:

v.append(len(x))

s.update(v)

print(s)



#Return square and cube of the following numbers:

def cal():

a = 34

b = 45

c = 56

x = ("square : ",{'a':a\*a, 'b':b\*b, 'c':c\*c})

y = ("cube : ",{'a':a\*a\*a, 'b':b\*b\*b, 'c':c\*c\*c})

return x ,y

cal()



#Get the clean list of students

Code 1:

import re

s = ['Ariba396','Priyanka8','Jagruti!','Sunio#','Gogo.']

def cleanString(x):

result = []

for z in x:

z = z.strip()

z = re.sub('[1-9!#.]', '' , z)

z = z.title()

result.append(z)

return result

cleanString(s)

Code 2:

import re

s = ['Ariba396','Priyanka8 ','Jagruti! ','Sunio#','Gogo.']

def removePunc(z):

return re.sub('[ 1-9!#.]', '' , z)

result = []

for x in map(removePunc, s):

result.append(x)

print(result)



#Lambda function with single value

a = lambda y: 5 + y

print("Using lambda :",a(3))



#Lambda function with two value

a = lambda x,y: x + y

print("Using lambda :",a(6,3))



#write a program to calculate the value of quadratic equation using the lambda function

#ax^2 + bx + c

z = lambda a,b,c,x: (((a\*(x\*\*2))+b\*x)+c)

print("Answer of quadratic equation having a = 6, b = 3, c = 2 and x = 1 :",z(6,3,2,1))



Pandas : A Python library used for working with data sets. It has functions for analysing, cleaning, exploring, and manipulating data.

Series : A one-dimensional array-like object containing a sequence of values and an associated array of data labels, called its index.

index parameter: to specify your own index

Code:

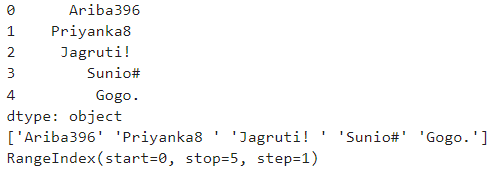
import pandas as pd

a = pd.Series(['Ariba396','Priyanka8 ','Jagruti! ','Sunio#','Gogo.'])

print(a)

print(a.values)

print(a.index)

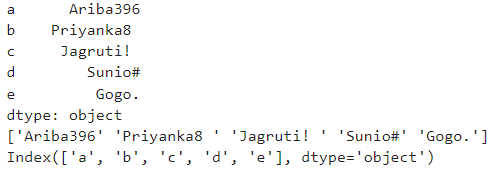


a = pd.Series(['Ariba396','Priyanka8 ','Jagruti! ','Sunio#','Gogo.'], index=['a','b','c','d','e'])

print(a)

print(a.values)

print(a.index)



Dataframe : store data into rows and columns

#create a dataframe to store 5 employees data : empid, empname, salary

d = { 'EmpId' : [101,102,103,104,105,106],

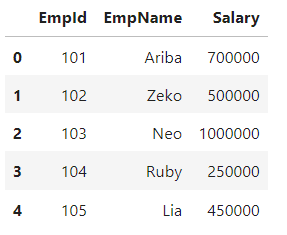
'EmpName' : ['Ariba', 'Zeko', 'Neo', 'Ruby', 'Lia', 'Zahra'],

'Salary' : [700000, 500000, 1000000, 250000, 450000, 440000]

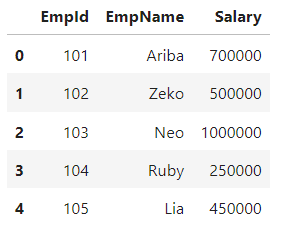
}

f = pd.DataFrame(d)

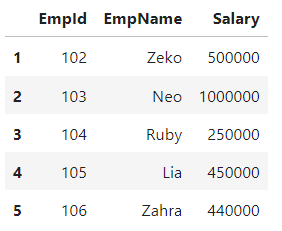
f



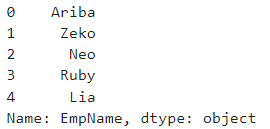
f.head()



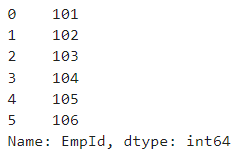
f.tail()



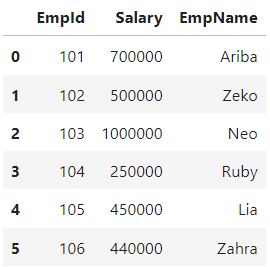
f['EmpName']



f.EmpId



pd.DataFrame(d, columns=['EmpId', 'Salary', 'EmpName'])

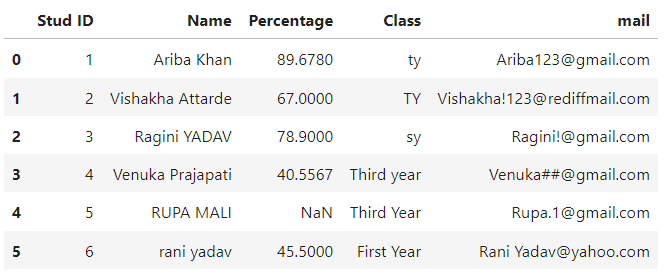


read\_csv():

std = pd.read\_csv("D:\\6626\_Ariba\\StudInfo.csv")

frame = pd.DataFrame(std)

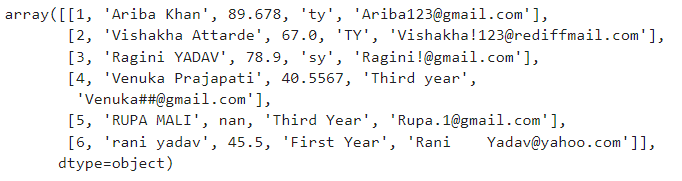
frame



std.columns



std.values

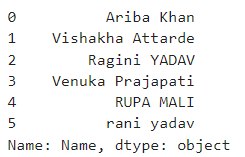


std.index



x = std['Name']

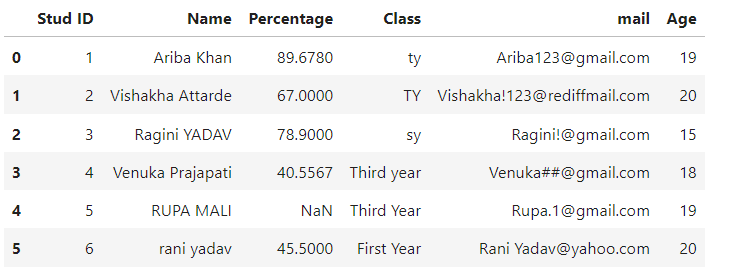
x



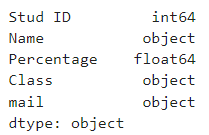
val = pd.Series([19,20,15,18,19,20])

frame['Age']=val

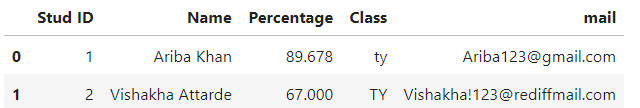
frame



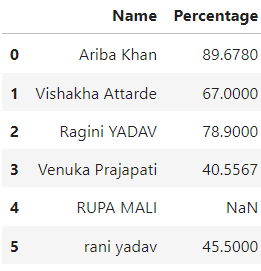
std.dtypes



std[0:2]

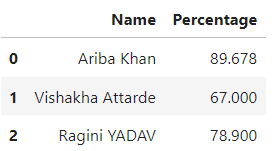


std[['Name', 'Percentage']]

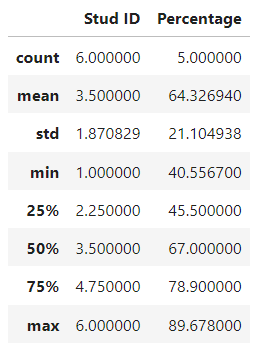


#to get name and percentage value of first 3 rows

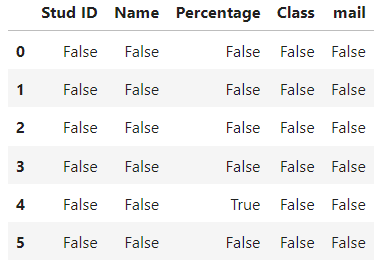
std.iloc[0:3, [1,2]]



std.describe()



pd.isna(std) : missing value -> True



[30/07/24]

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

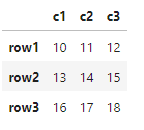
Code:

import pandas as pd

import numpy as np

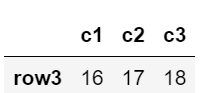
f = pd.DataFrame(np.arange(10, 19). reshape((3,3)), columns=['c1','c2','c3'], index=['row1', 'row2', 'row3'])

f



#Display the second row of the dataframe.

f[2:]



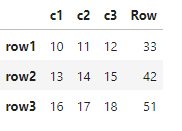
#Store the first, second and third row of the dataframe in the variable r1, r2 and r3 and add each individual element of all rows.

def df(row):

return row.sum()

f['Row'] = f.apply(df, axis=1)

f



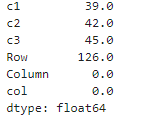
#add each individual element of all column.

def add(col):

return col.sum()

f = f.apply(add, axis=0)

f

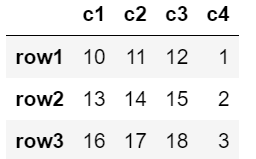


#Store the values 1,2,3 in a series s1. Add the series s1 into the above dataframe.

s = pd.Series([1, 2, 3], index=['row1', 'row2', 'row3'], name='c4')

f['c4'] = s

f



#Draw a plot for given data

age = [1,2,3,4,5]

height = [1.5,2,2.5,3,3.5]

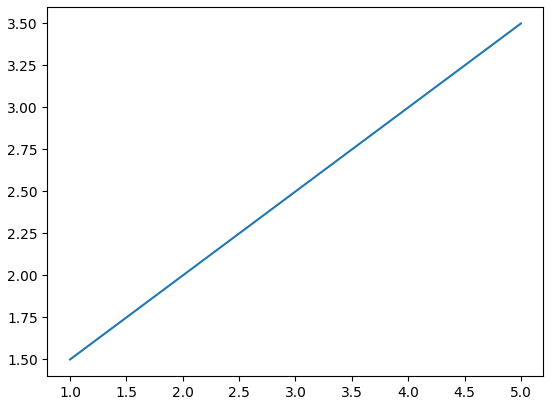
import matplotlib.pyplot as plt

import numpy as np

fig, ax = plt.subplots()

ax.plot(age, height)

plt.show()



# Practical No : 2 [06/08/24]

Conversion of semi structured and unstructured data to csv

CSVFile.csv

| Column\_1 | Column\_2 | Column\_3 | Column\_4 |
| --- | --- | --- | --- |
| 1 | Ariba Khan | 80 | 2000000 |
| 2 | Rahul Deshpande | 35 | 3000000 |
| 3 | Surya Yadav | 23 | 1500000 |
| 4 | Rohit Sharma | 45 | 2500000 |
| 5 | Shaymal Kulkarni | 56 | 3000000 |

Processing a csv file

Code:

import pandas as pd

InputData=pd.read\_csv("D:\\6626\_Ariba\\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)

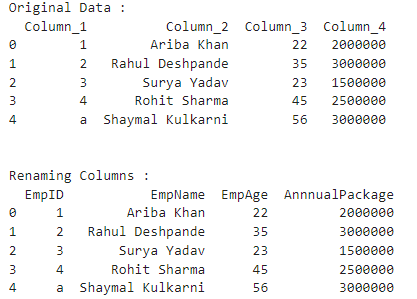
print("\n")

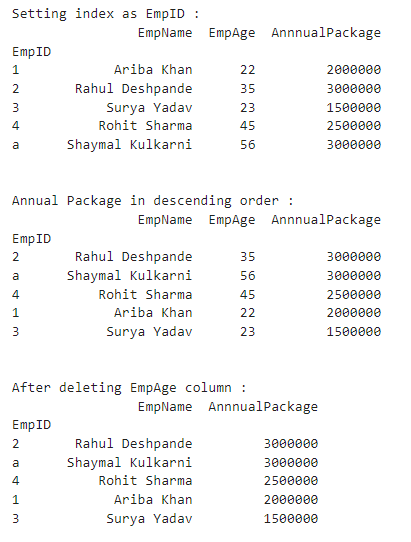
#to save your file

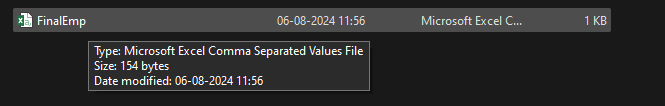
ProcessData.to\_csv("D:\\6626\_Ariba\\FinalEmp.csv")

#sOutputFileName=’FinalEmp.csv’

#OutputData.to\_csv(sOutputFilaName, a)







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:\\6626\_Ariba\\convertcsv.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)

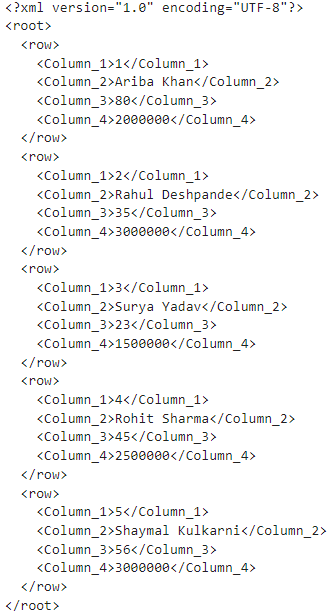
ProcessData.drop('EmpAge', axis=1, inplace=True)

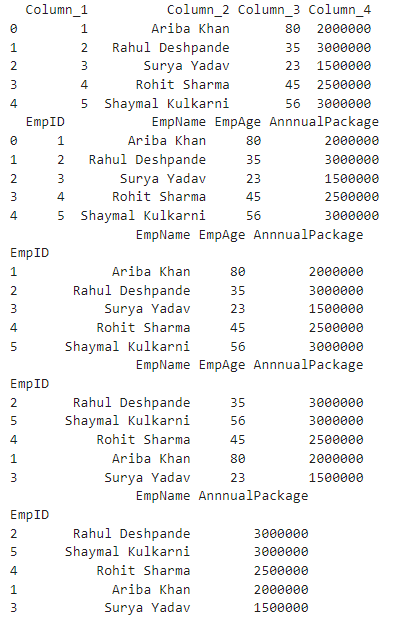
print(ProcessData)

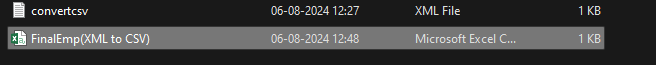
OutputData=ProcessData

sOutputFileName="D:\\6626\_Ariba\\FinalEmp(XML to CSV).csv"

OutputData.to\_csv(sOutputFileName, index = False)







Json to csv :

import pandas as pd

sInputFileName="D:\\6626\_Ariba\\csvjson.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)

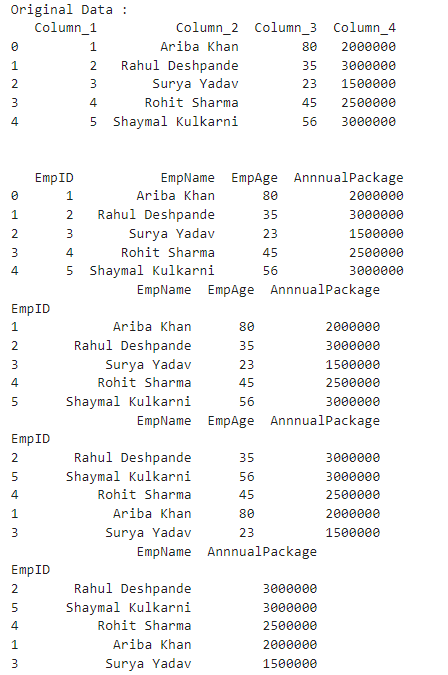
ProcessData.drop('EmpAge', axis=1, inplace=True)

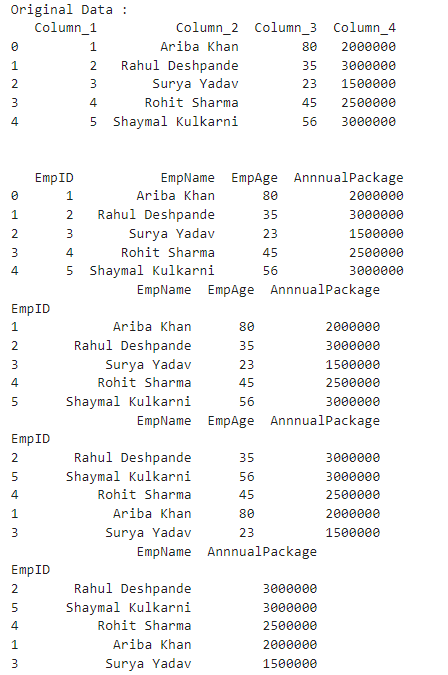
print(ProcessData)

OutputData=ProcessData

sOutputFileName="D:\\6626\_Ariba\\FinalEmp(JSON to CSV).json"

OutputData.to\_csv(sOutputFileName, index = False)





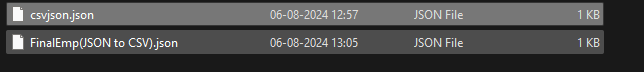


Image to csv:

import imageio.v3 as iio #pip install imageio

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

sInputFileName = "D:\\6626\_Ariba\\ocean.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())

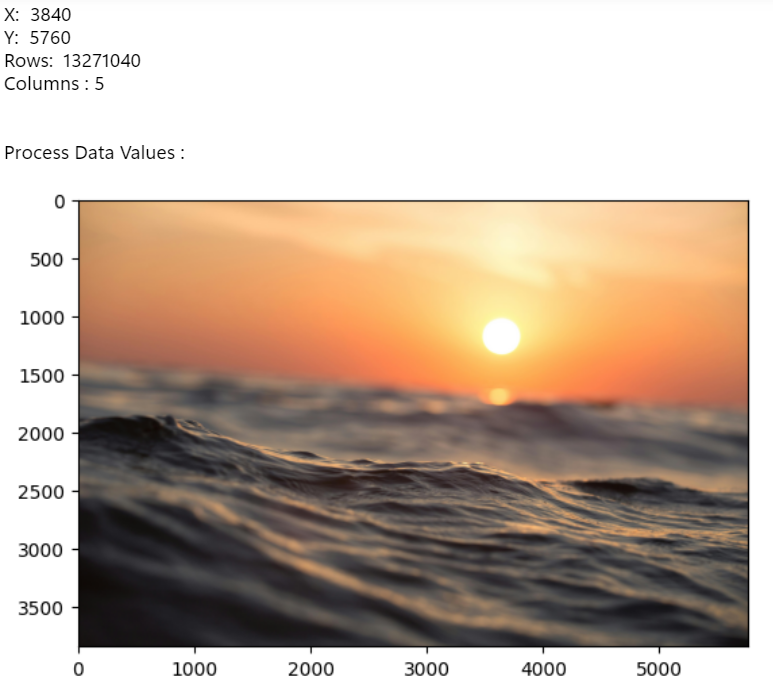
plt.show()

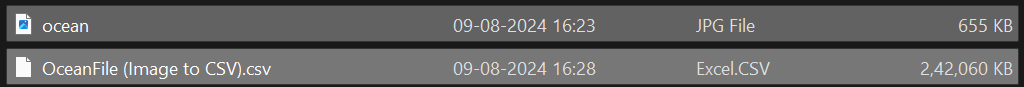
print('\n')

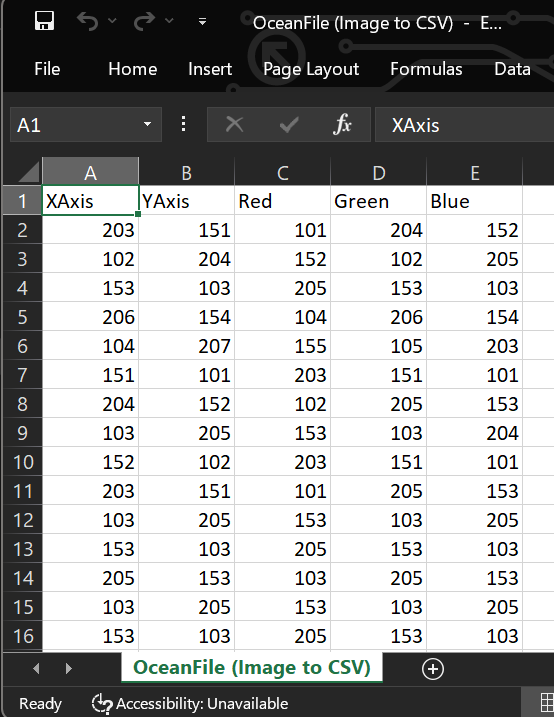
OutputData = ProcessData

sOutputFileName = "D:\\6626\_Ariba\\OceanFile (Image to CSV).csv"

OutputData.to\_csv(sOutputFileName, index=False)







Video to Frames:

import os

import shutil

import cv2 #pip install opencv-python

sInputFileName = "D:\\6626\_Ariba\\ocean.mp4"

sDataBaseDir = 'D:\\6626\_Ariba\\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

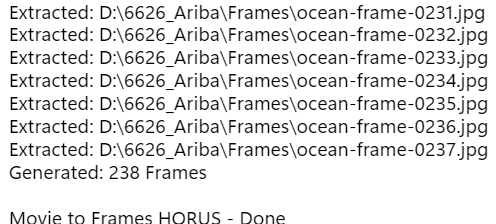
success, image = vidcap.read()

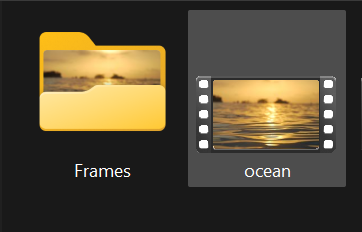
count += 1

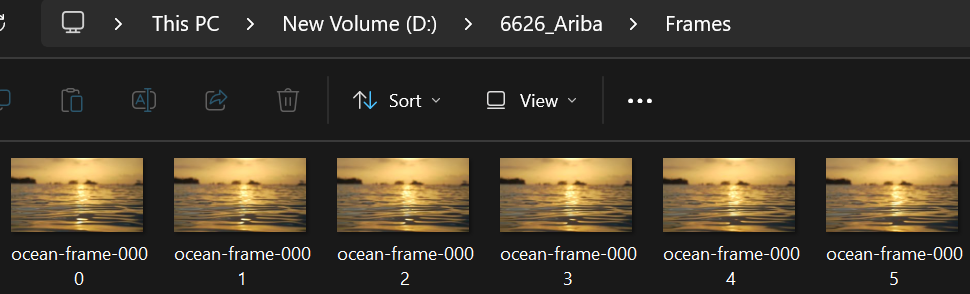
print('Generated:', count, 'Frames')

print('\nMovie to Frames HORUS - Done')









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:\\6626\_Ariba\\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 ===========================================

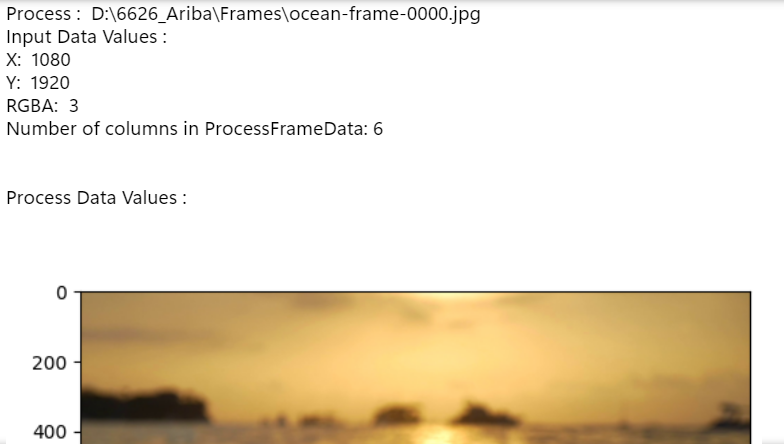
OutputData = ProcessData

sOutputFileName = "D:\\6626\_Ariba\\ocean(Frame to csv)\_new.csv"

OutputData.to\_csv(sOutputFileName, index=False)

print('\n')

print('Processed ; ', f, ' frames')

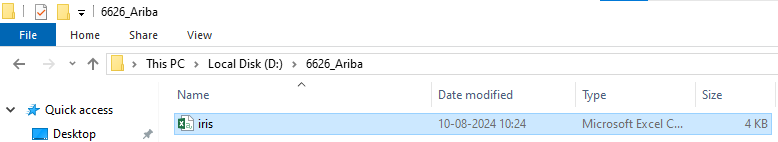




# Practical No : 3 [10/08/24]

Data Processing for various types of data

Download iris.csv file



Importing

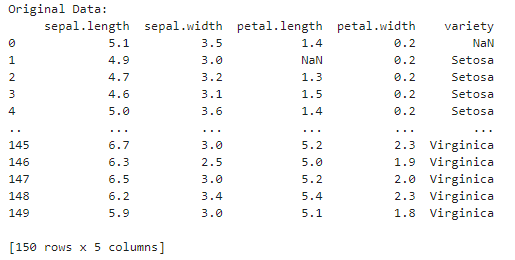
import pandas as pd

data = pd.read\_csv("D:\\6626\_Ariba\\iris.csv")

print("Original Data: ")

df=pd.DataFrame(data)

print(df)



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 : ")

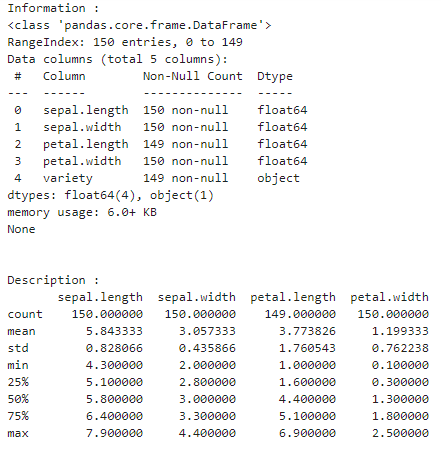
print(df.fillna(1)) #1 can be replaced by anything

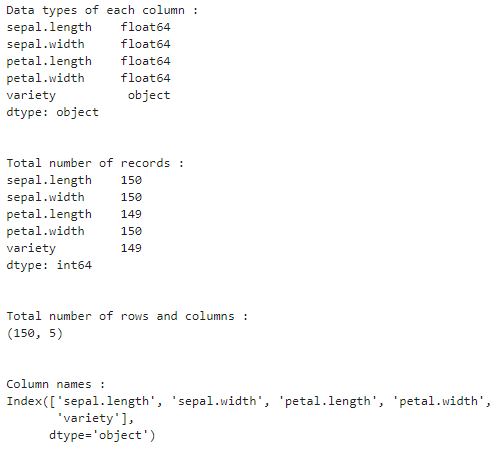
print("\n")

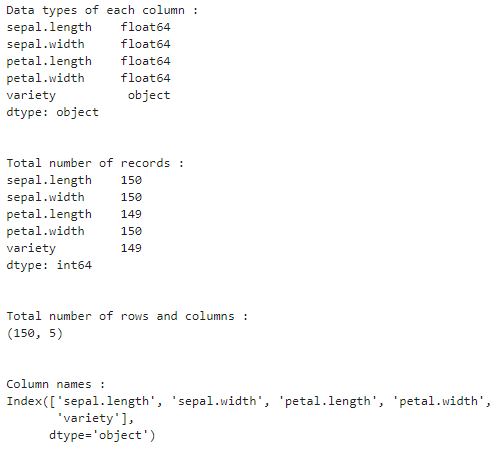
#Replace the NaN values with given values

print("Replace the NaN values with given values : ")

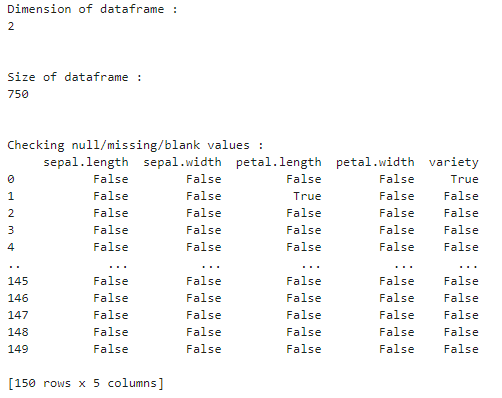
print(df.replace(to\_replace=np.nan, value=0)) #0 can be replaced by anything

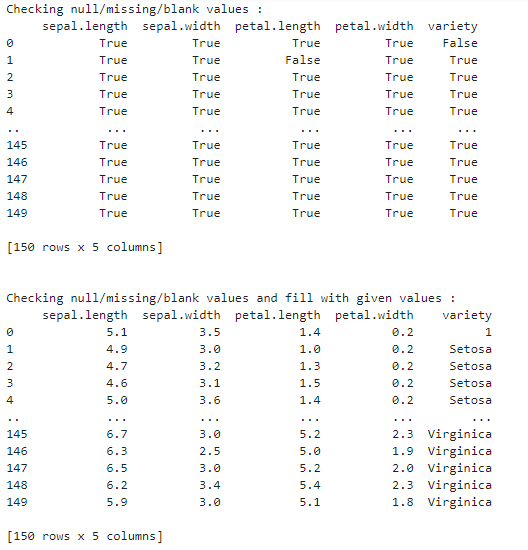


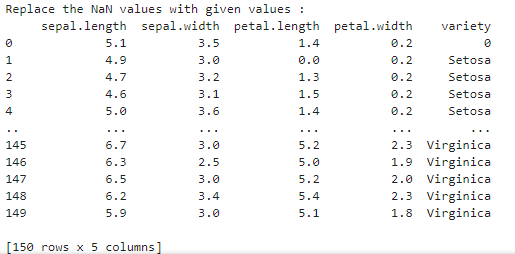












Q. Create the data frame for the given data and process the data frame data for the null or missing values.

#create a dataframe

d = { 'EmpName' : ['Ariba','Vishakha','Aniket','NaN', 'Shailesh'],

'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],

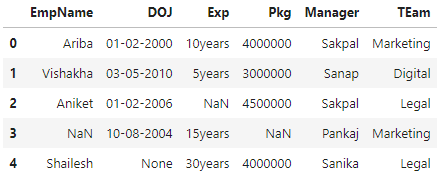
'Manager' : ['Sakpal', 'Sanap', 'Sakpal', 'Pankaj', 'Sanika'],

'TEam' : ['Marketing', 'Digital', 'Legal', 'Marketing', 'Legal'],

}

f = pd.DataFrame(d)

f



# 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()

f



f\_new = f.dropna(subset=['EmpName'])

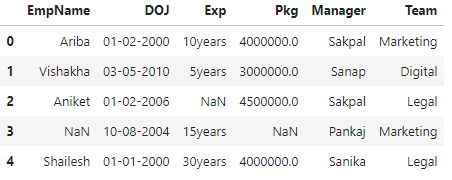
f\_new



f.replace({'DOJ':'None'}, np.nan, inplace=True)

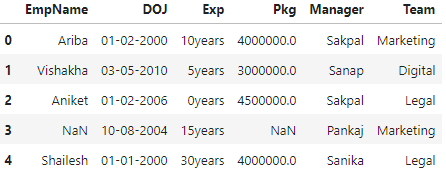
f['DOJ'] = f['DOJ'].fillna('01-01-2000')

f



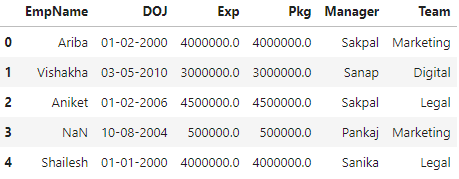
f['Exp'].replace(to\_replace=np.nan, value='0years', inplace=True)

f



f['Pkg'].replace(to\_replace=np.nan, value=500000, inplace=True)

f



#create a 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],

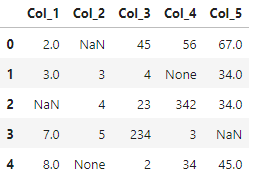
'Col\_4' : [56,'None',342,3,34],

'Col\_5' : [67, 34, 34, np.nan, 45],

}

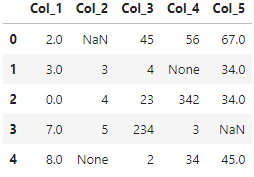
f = pd.DataFrame(d)

f



f['Col\_1'].replace(to\_replace=np.nan, value=0, inplace=True)

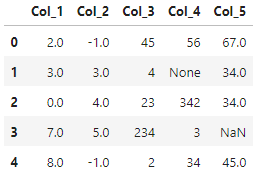
f



f.replace({'Col\_2':'None'}, np.nan, inplace=True)

f['Col\_2'] = f['Col\_2'].fillna(-1)

f



f['Col\_3']=-2

f

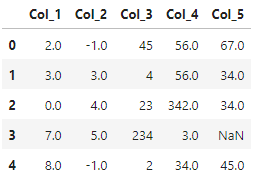


v=f.Col\_4[0]

f.replace({'Col\_4':'None'}, np.nan, inplace=True)

f['Col\_4'] = f['Col\_4'].fillna(v)

f

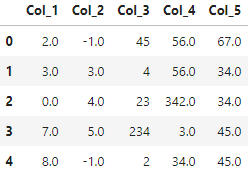


av=f.Col\_5

av\_mean=av.mean()

f['Col\_5'].replace(to\_replace=np.nan, value=av\_mean, inplace=True)

f



# 

# 

# 

# 

# 

# Practical No : 4 [13/08/24]

Basic Utility design, Data auditing and Exploratory Data Analysis

| StudID | Name | Class | Elective | Msg | Test Score | Date of Birth |
| --- | --- | --- | --- | --- | --- | --- |
| Student1 | ‘  Ariba ‘ | Fy | ‘SBCM’ | ‘he\x00llo’ | 20 | ‘06-09-2003’ |
| Student2 | ‘Vilas’ | fy | ‘  sbcm’ | ‘My\nComputer’ | 8 | ‘02/02/24’ |
| Student3 | ‘Vinay  ‘ | FY | ‘AI’ | ‘Hello Guy\\s’ | 5 | ‘04-03-2006’ |
| Student4 | “ Pranay ‘ | fY | ‘AI  ‘ | Good morning | 11 | ‘03/31/2003’ |
| Student5 | ‘  AnIL ‘ | Fy | ‘SBCM’ | ‘he\x00llo’ | 18 | ‘01-01-2001’ |
| Student6 | ‘ViKas’ | fy | ‘  sbcm’ | ‘My\nComputer’ | 9 | ‘02/02/24’ |
| Student7 | ‘Vinit  ‘ | FY | ‘AI’ | ‘Hello Guy\\s’ | 50 | ‘04-03-2006’ |
| Student8 | “ Pranali ‘ | fY | ‘AI  ‘ | Good morning | 10 | ‘03/31/2003’ |

# Creating the dataframe

# Creating the dataframe

import pandas as pd

data = {

'StudID': ['Student1', 'Student2', 'Student3', 'Student4', 'Student5', 'Student6', 'Student7', 'Student8'],

'Name': [' Ariba ', 'Vilas', 'Vinay ', ' Pranay ', ' AnIL ', 'ViKas', 'Vinit ', ' Pranali '],

'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],

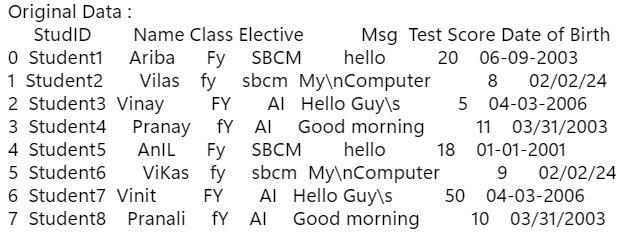
'Date of Birth': ['06-09-2003', '02/02/24', '04-03-2006', '03/31/2003', '01-01-2001', '02/02/24', '04-03-2006', '03/31/2003']

}

df = pd.DataFrame(data)

print("Original Data : ")

print(df)





Basic Utility Design

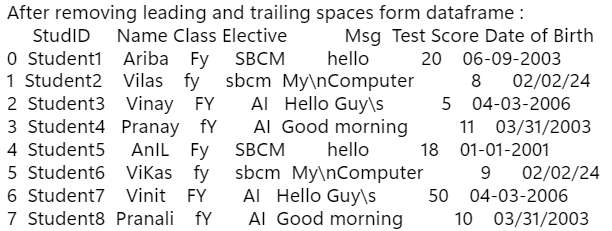
1. Fixer Utility

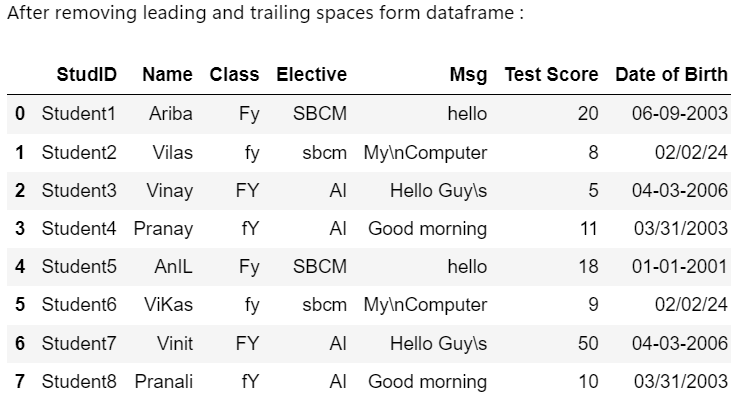
Q. Remove leading and trailing spaces from the data of all columns.

df = df.apply(lambda x: x.str.strip() if x.dtype == "object" else x)

print("After removing leading and trailing spaces from dataframe :")

df





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

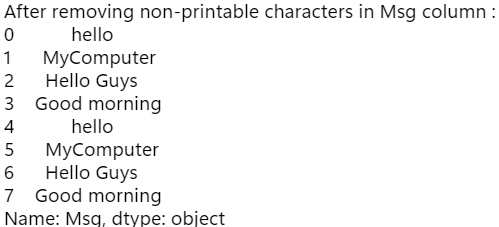
# Removing non-printable characters from msg column

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'])



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:

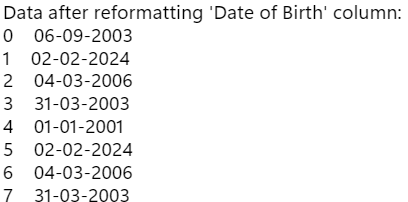
continue

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'])



2. Data Binning or Bucketing

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

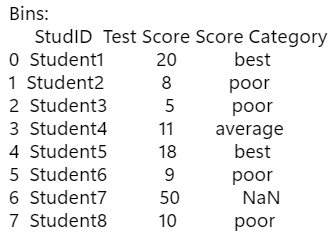
bins = [0, 10, 15, 20]

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']])



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}")



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

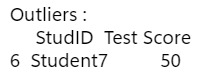
lower\_bound = Q1 - 1.5 \* IQR

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']])



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.")

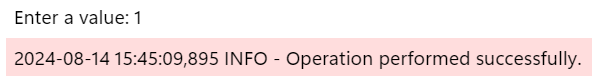
try:

input\_value = int(input("Enter a value: "))

perform\_operation(input\_value)

except ValueError as ve:

logging.exception("Exception occurred: %s", str(ve))





Data auditing

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

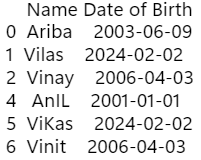
start = pd.to\_datetime('01-01-2000')

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']])



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)



Q. The student's name should have only alphabetic characters.

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)

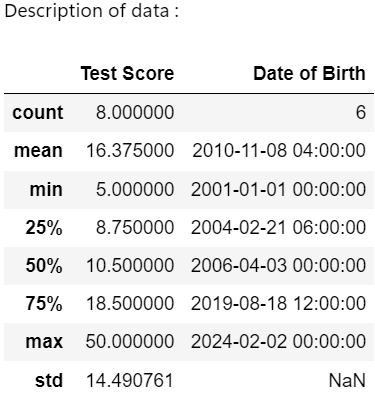


Exploratory data analysis

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

print("Description of data :")

df.describe()



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



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

import matplotlib.pyplot as plt

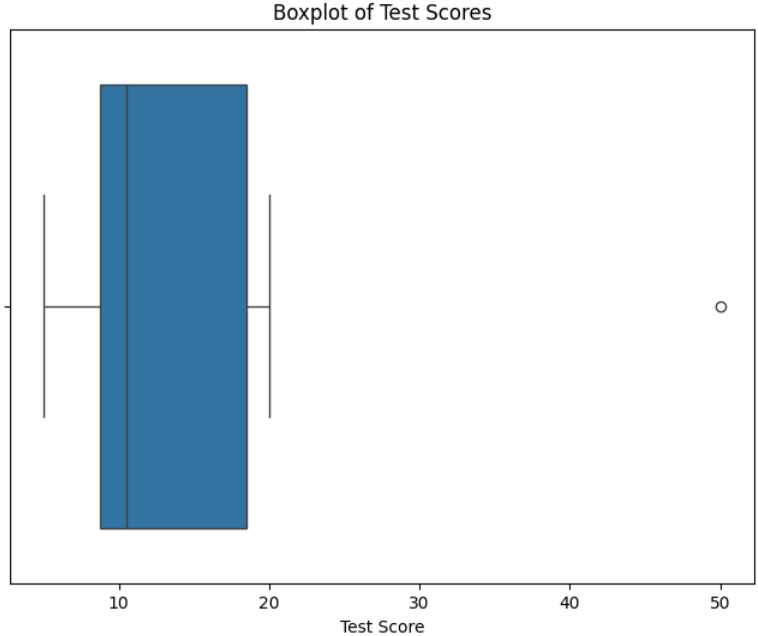
import seaborn as sns # pip install seaborn

plt.figure(figsize=(8, 6))

sns.boxplot(x=df['Test Score'])

plt.title('Boxplot of Test Scores')

plt.show()



Q. Draw the Distribution plot and cdf curve for the test score column.

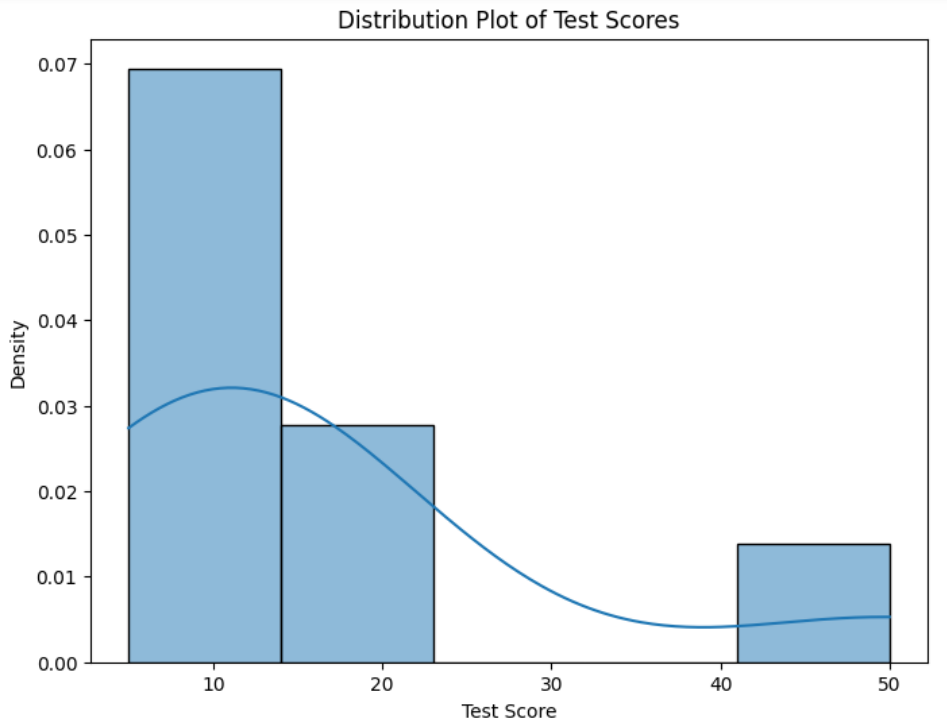
#Distribution plot

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()



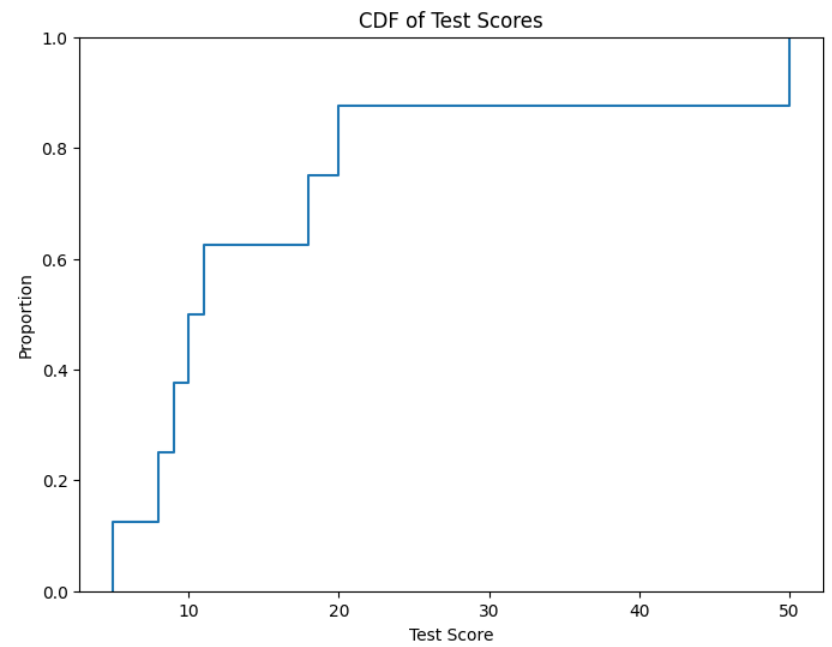
# CDF Plot

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.

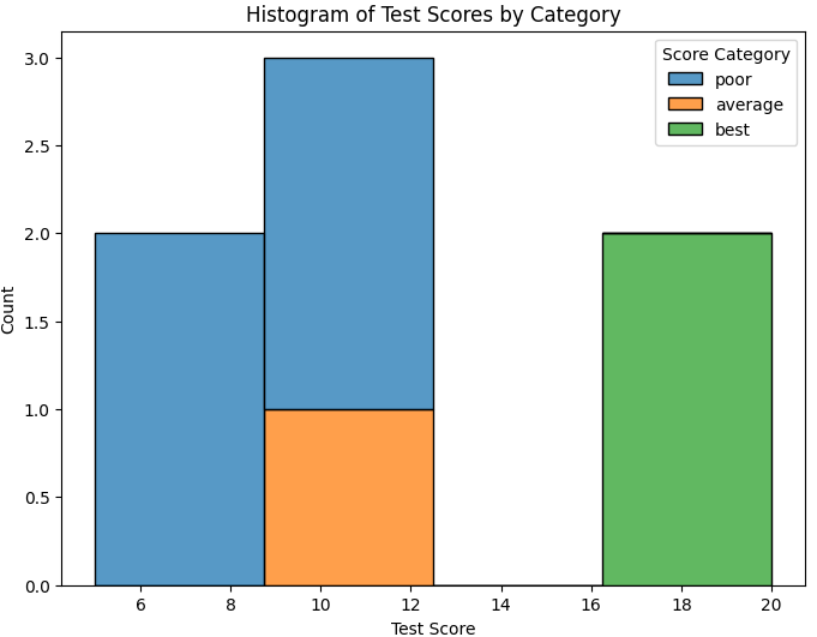
#histograms

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()



Q. Draw the scatter plot of test scores.

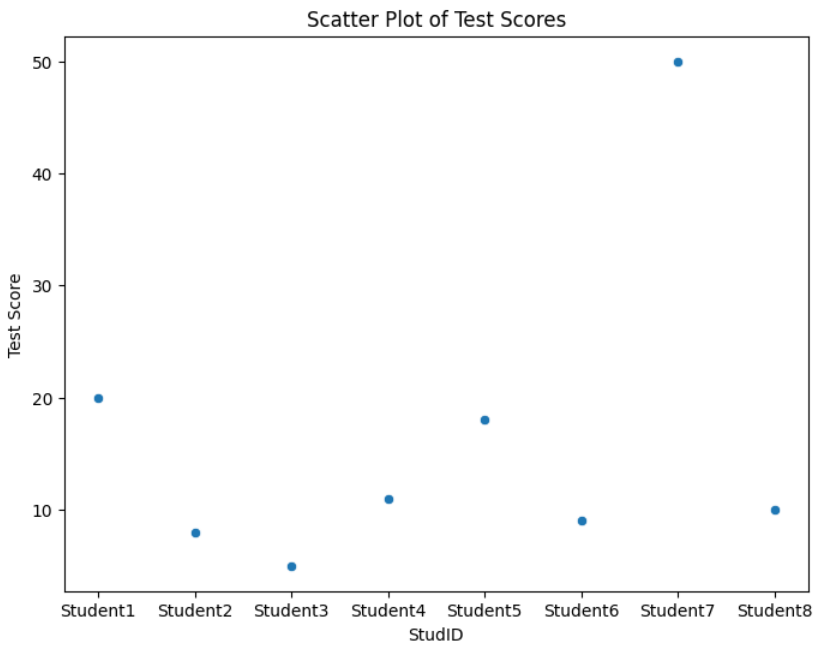
# Scatter plot

plt.figure(figsize=(8, 6))

sns.scatterplot(x='StudID', y='Test Score', data=df)

plt.title('Scatter Plot of Test Scores')

plt.show()



# Practical No : 5 [13/08/24]

Retrieve Utility

Currency dataset : <https://github.com/Apress/practical-data-science/tree/master>

1. Load the raw data of Excel/csv file

import pandas as pd

import os

import sys

sFileDir='D:\\6626\_Ariba'

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

sFileName='D:\\6626\_Ariba\\Country\_Code.csv'

print("Path :", sFileName)

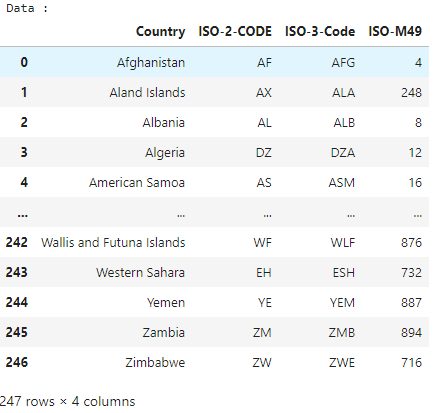


2. Load given csv/Excel file in the pandas dataframe.

CurrencyData = pd.read\_csv("D:\\6626\_Ariba\\Country\_Code.csv", encoding='cp1252')

print("Data : ")

CurrencyData



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

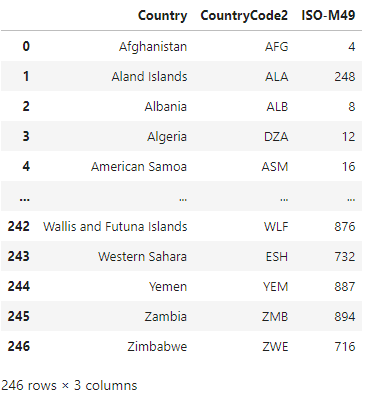


4. Drop not required columns from the dataframe.

#Drop

CD.drop('CountryCode1', axis=1, inplace=True)

CD



5. Remove leading and trailing spaces of the column values.

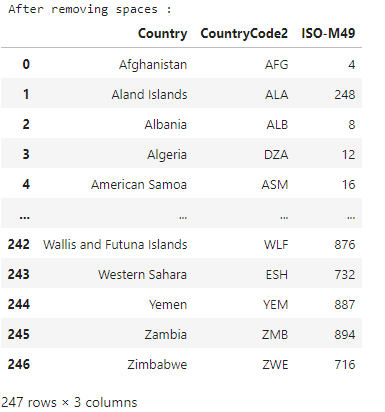
#Removing

CD['Country']=CD['Country'].map(lambda x:x.strip())

CD['CountryCode2']=CD['CountryCode2'].map(lambda x:x.strip())

print("After removing spaces : ")

CD

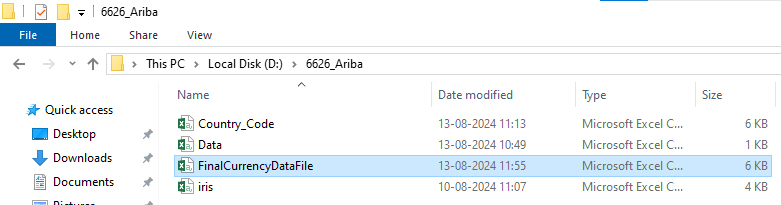


6. Save the retrieved file in the specified folder

CD.to\_csv("D:\\6626\_Ariba\\FinalCurrencyDataFile.csv")

print("Saved successfully!!!")





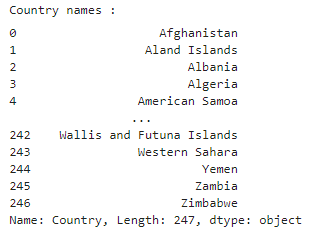
7. Retrieve different attributes of data

Q. Retrieve country names from the dataframe CurrencyData.

#Retrieving Country name

print("Country names : ")

CD['Country']



Q. Load country names and country codes of first five countries in the variables cCode and cName from the dataframe CurrencyData.

five=CD.head(5)

cName=five['Country']

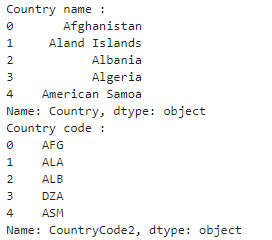
cCode=five['CountryCode2']

print("Country name :")

print(cName)

print("Country code :")

print(cCode)



Q. Display the country code of country Canada.

ctr = CD[CD['Country']=='Canada']

print("Country code of Canada :",ctr.iloc[:,[1]].values)



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)

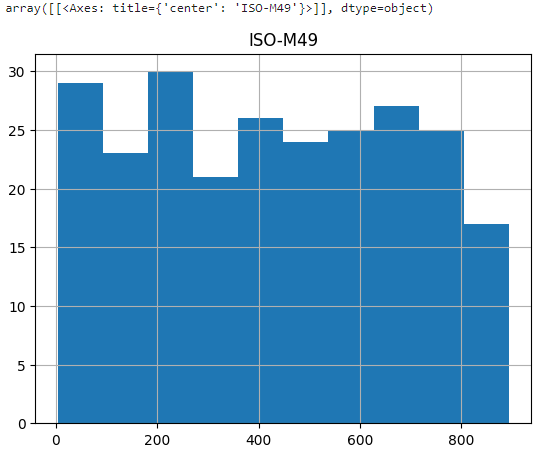
print("Skewness :")

print(skewness)



Q. Histogram

CurrencyData.hist(column=’ISO-M49’)

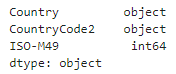


Q. Identify any loading characteristics (Columns Names, Data Types, Volumes).

CurrencyData.columns



CurrencyData.dtypes



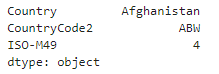
CurrencyData.shape



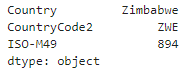
CurrencyData.size



CurrencyData.min()



CurrencyData.max()



# Practical No : 6 [20/08/24]

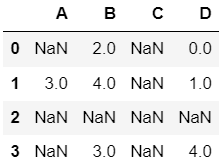
Access Superstep.

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],[np.nan,3,np.nan,4]], columns=list("ABCD"))

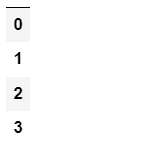
df

****

#Dropping columns including missing value

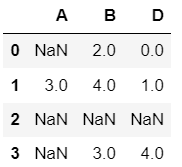
a = df.dropna(axis=1,how='any')

a

****

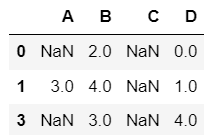
a = df.dropna(axis=1,how='all')

a

****

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

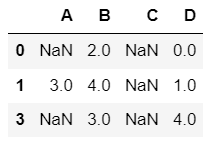
a

****

#dropping columns including a specific number of missing values

a = df[df.isnull().sum(axis=1) <=2]

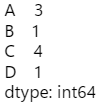
a

****

#Number of missing values in each row

a = df.isnull().sum()

a

****

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

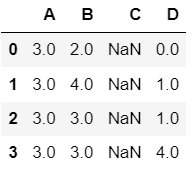
a = df.fillna(df.mean())

a

****

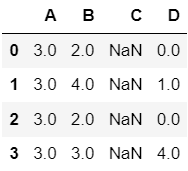
a = df.fillna(df.median())

a

****

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

**a**

****

# Practical No : 8 [20/08/24]

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']

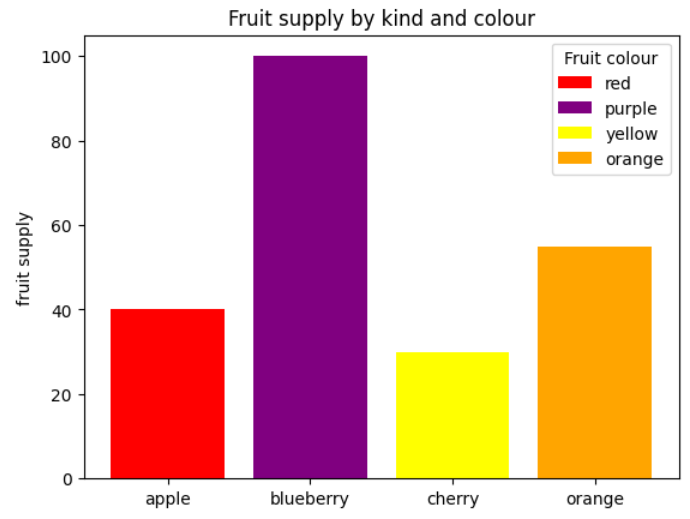
ax.bar(fruits, counts, label=bar\_labels, color=bar\_colors)

ax.set\_ylabel('fruit supply')

ax.set\_title('Fruit supply by kind and colour')

ax.legend(title='Fruit colour')

plt.show()

****

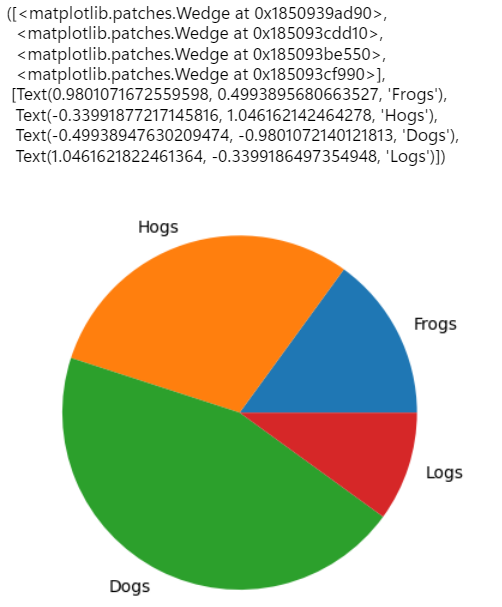
#Pie Chart

labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'

sizes = [15,30,45,10]

fig, ax = plt.subplots()

ax.pie(sizes, labels=labels)

****

#Line Graph

t = np.arange(0.0, 2.0, 0.01)

s = 1 + np.sin(2 \* np.pi \* t)

fig, ax = plt.subplots()

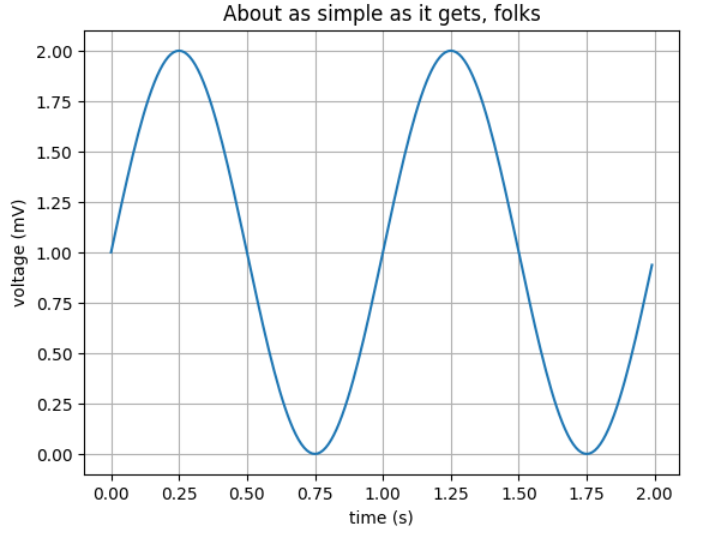
ax.plot(t, s)

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")

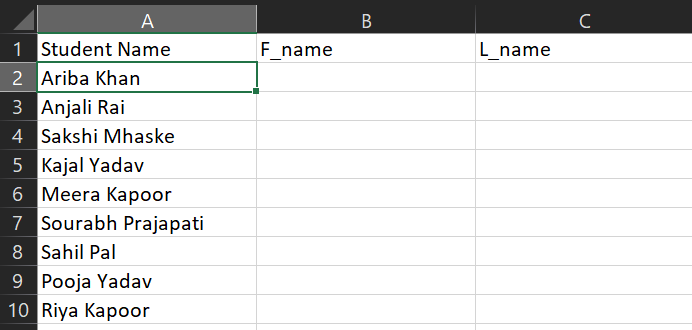
plt.show()

****

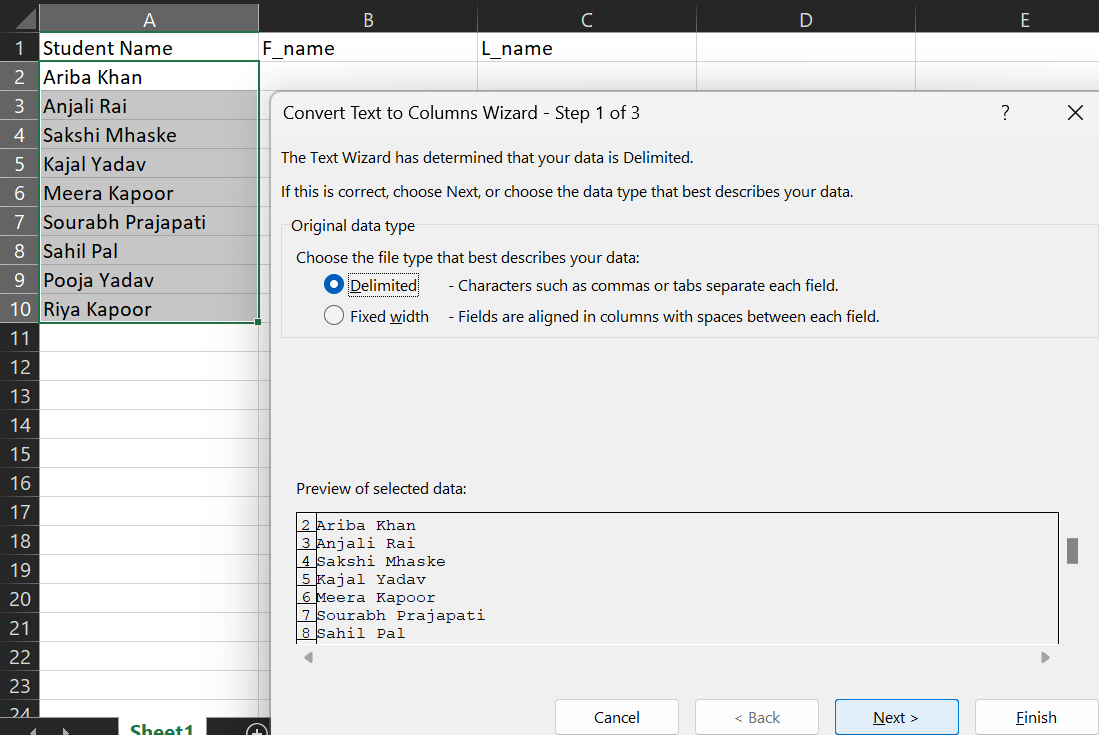
# Practical no : 9 []

Data Analysis using Excel

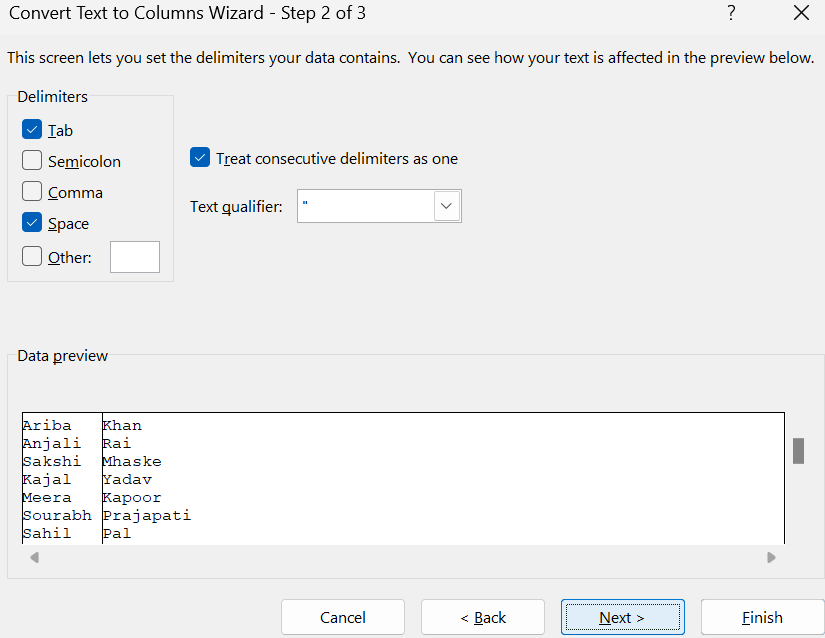
Create an excel sheet



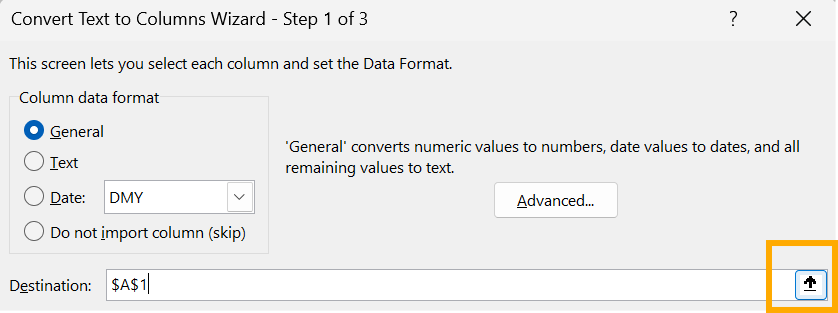
Select the values from Student Name column -> Data tab -> Text to Columns -> Click Next



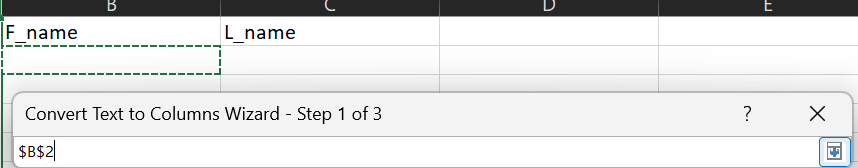
Click Next



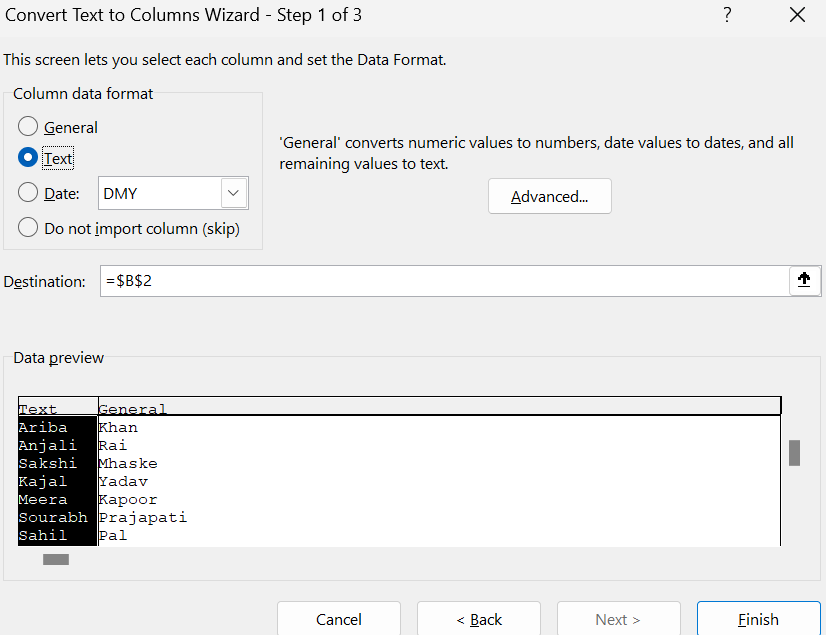
Click arrow to select destination cell



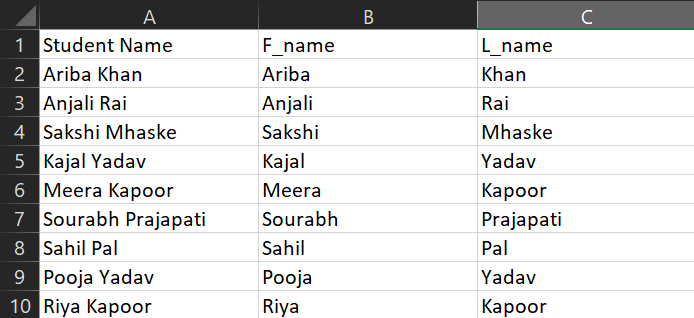
Select $B$2 -> Click down arrow



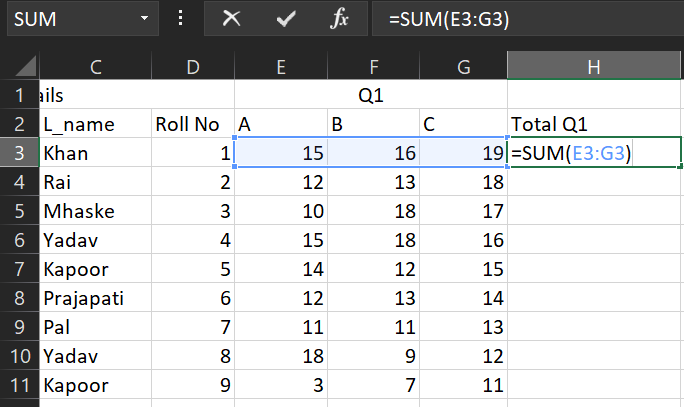
Select Text -> Click Finish



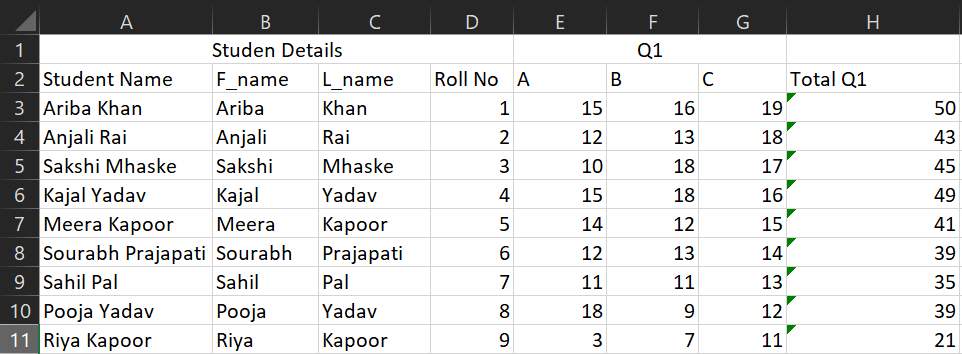
Output:



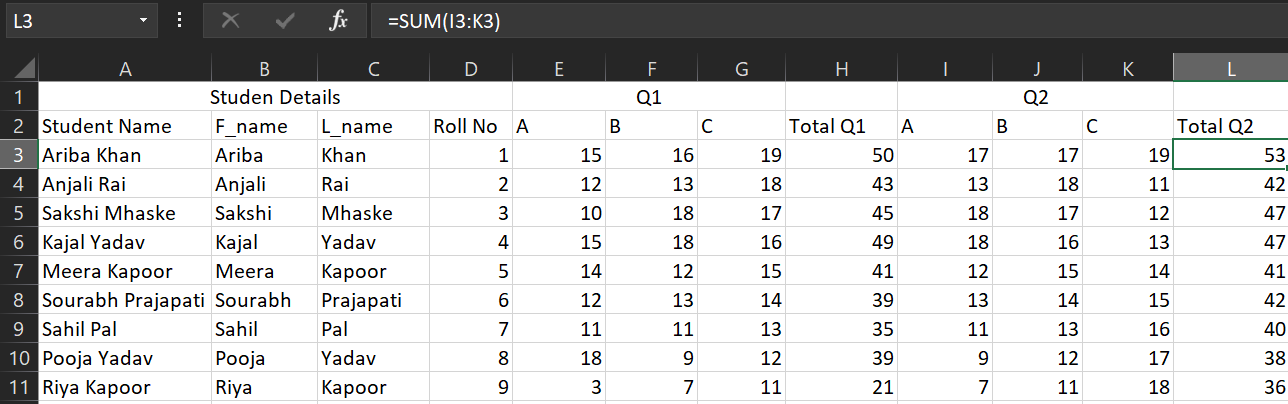
Create Q1 columns having A, B and C as sub columns -> CreateTotal Q1 column for sum value



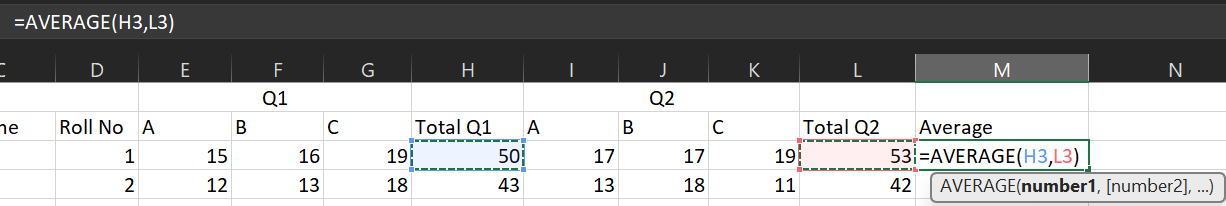
Result:

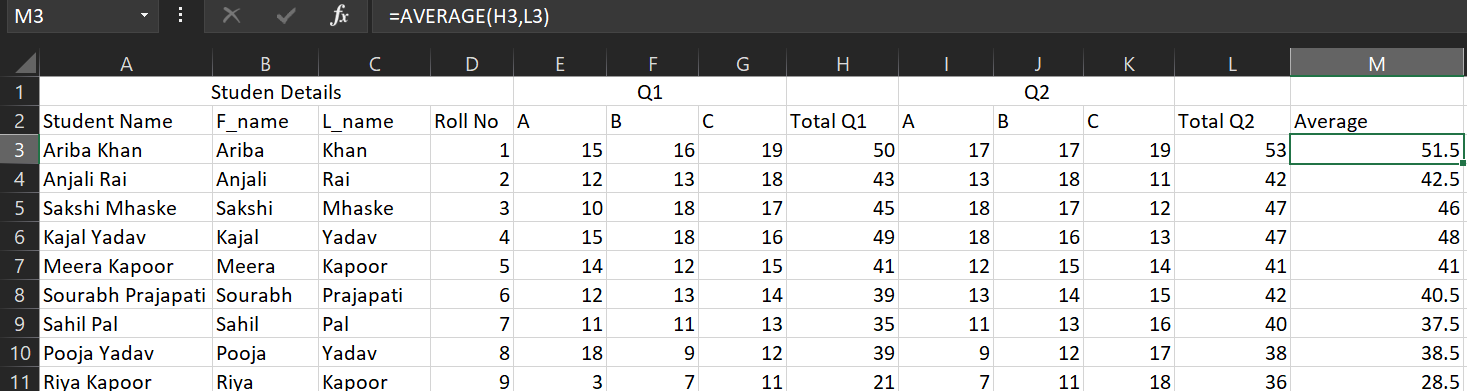


Similarly create a Q2 and Total Q2 columns

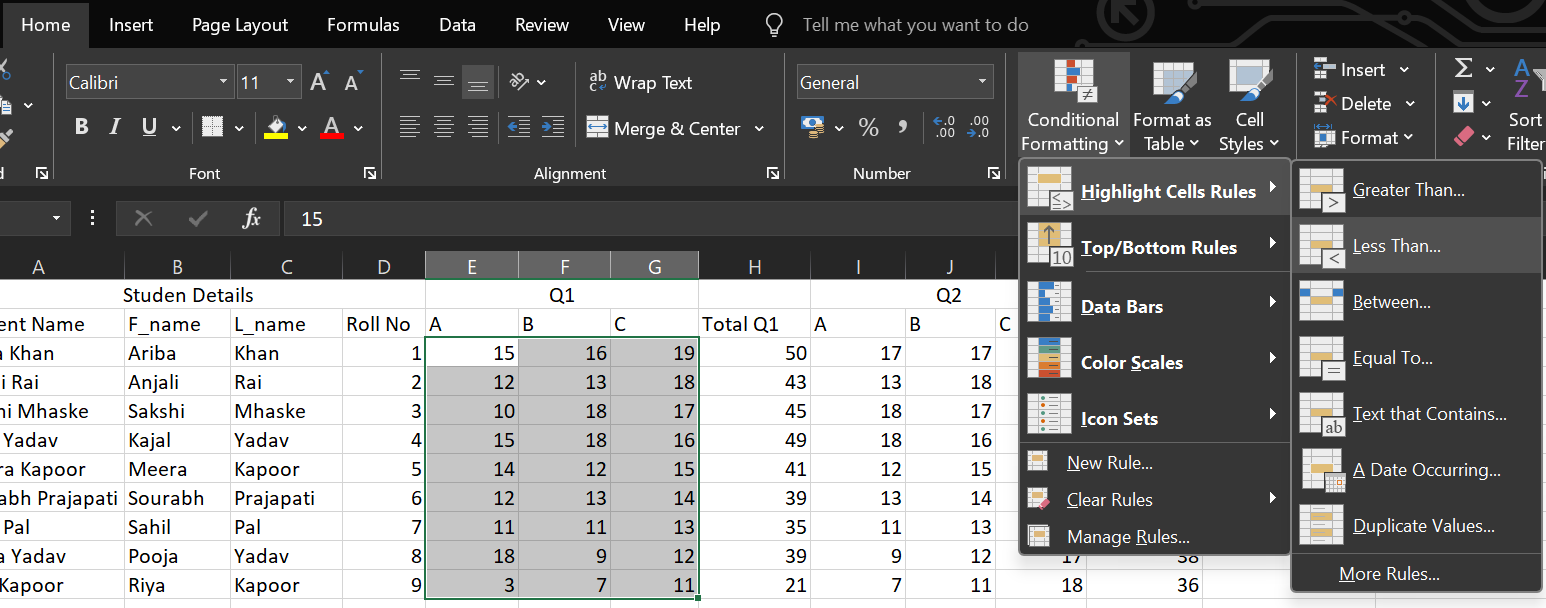


Create a column named Average for calculating average value of Q1 and Q2

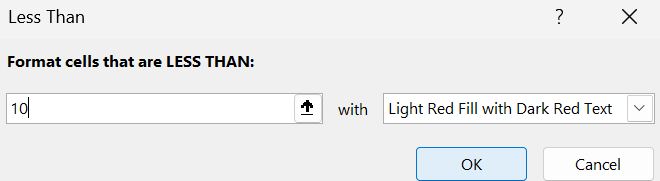




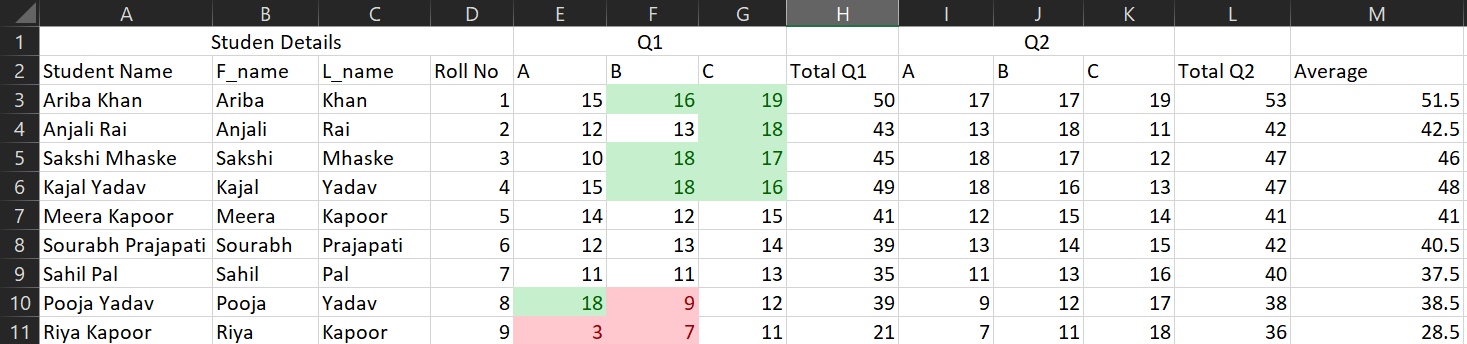
Apply Conditional Formatting (Less than) on values of Q1



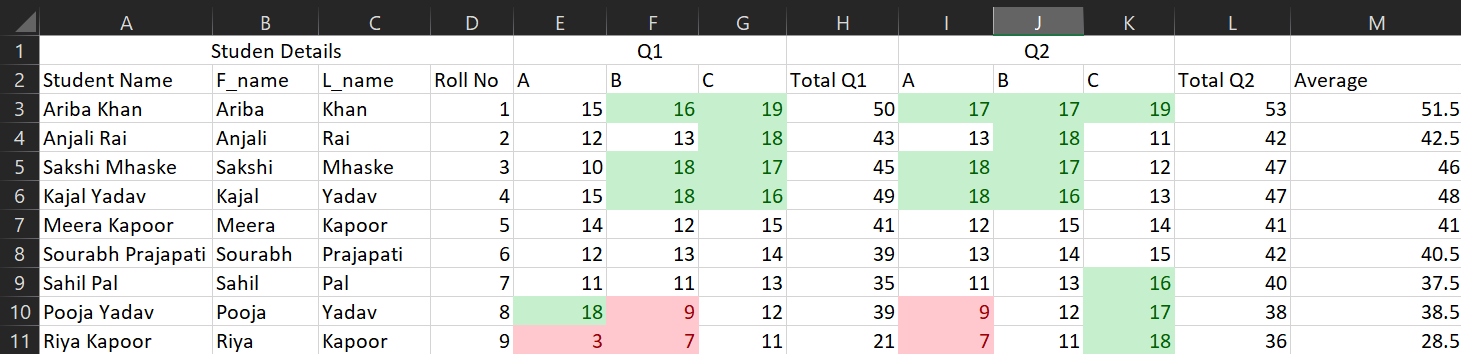
Enter a value -> Click OK



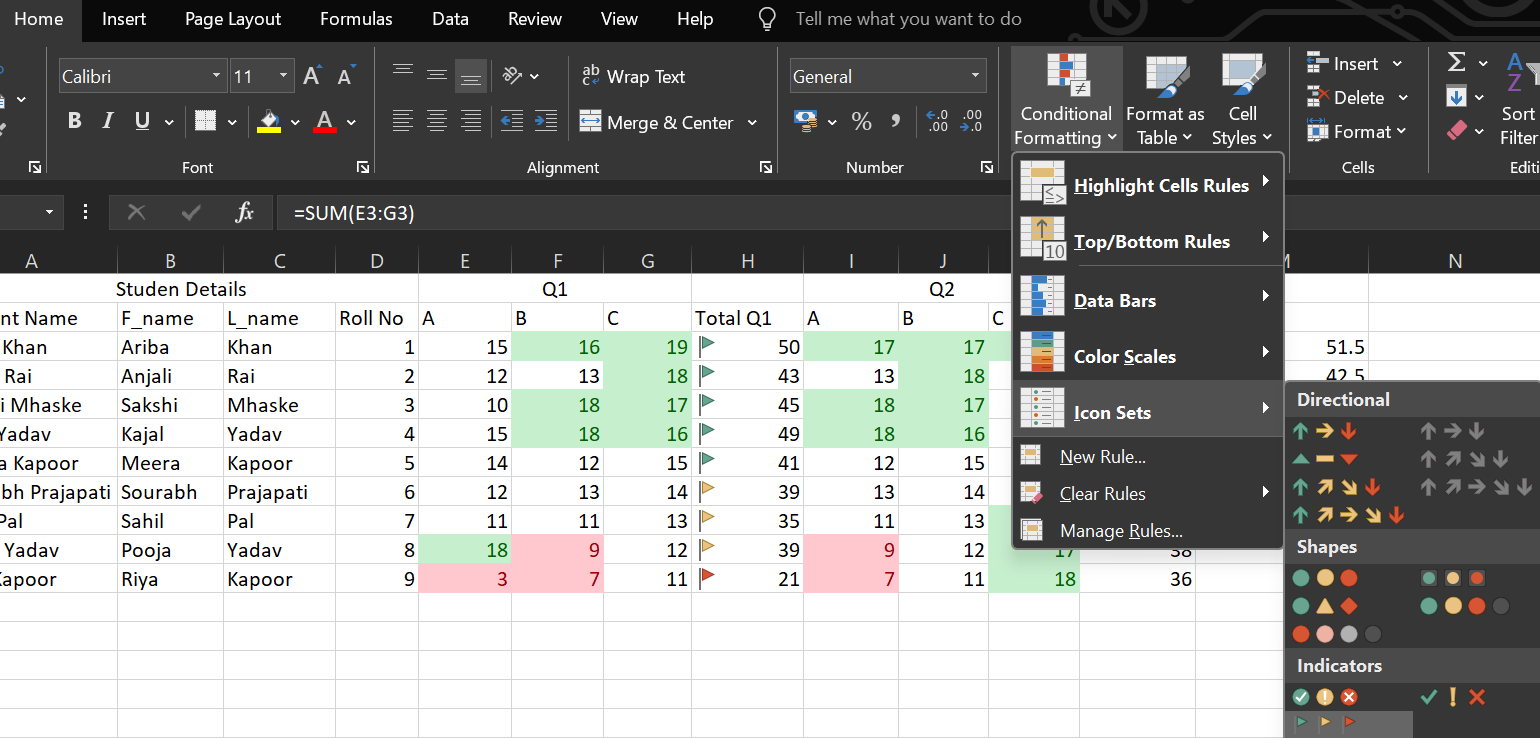
Similarly apply Greater than on same values



Do the same thing on values of Q2



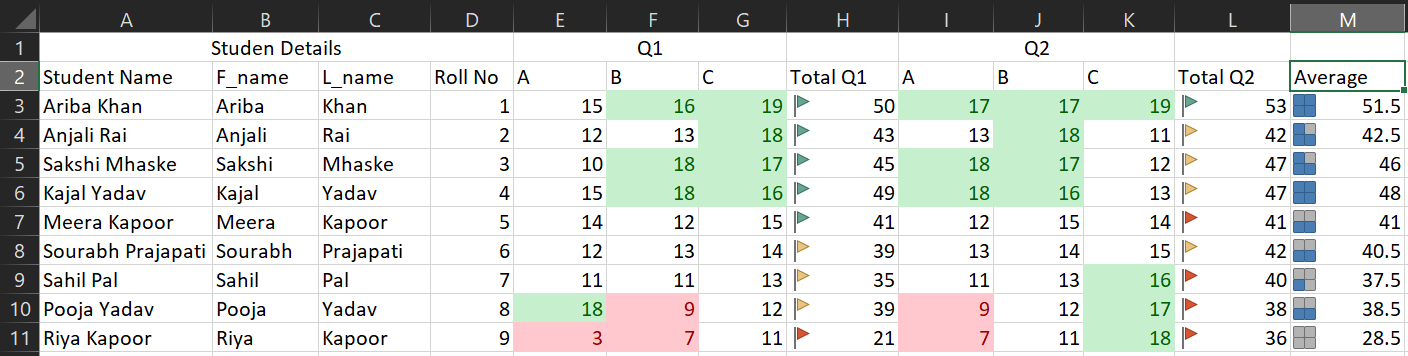
Select values of Total Q1 -> Home tab -> Conditional Formatting -> Icon Sets (any desirable icon set)



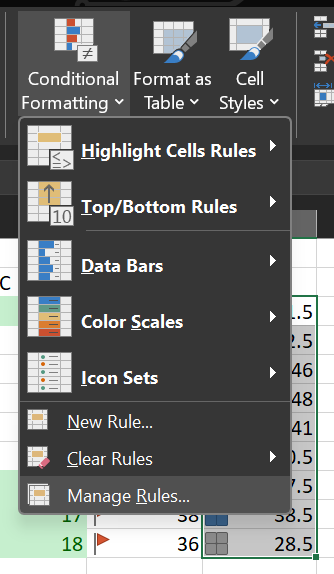
Similarly on Total Q2



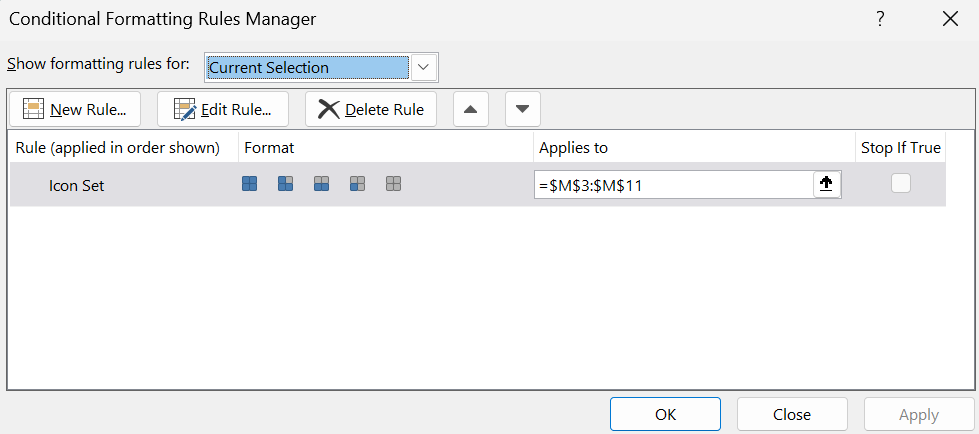
Do the same thing for Average column



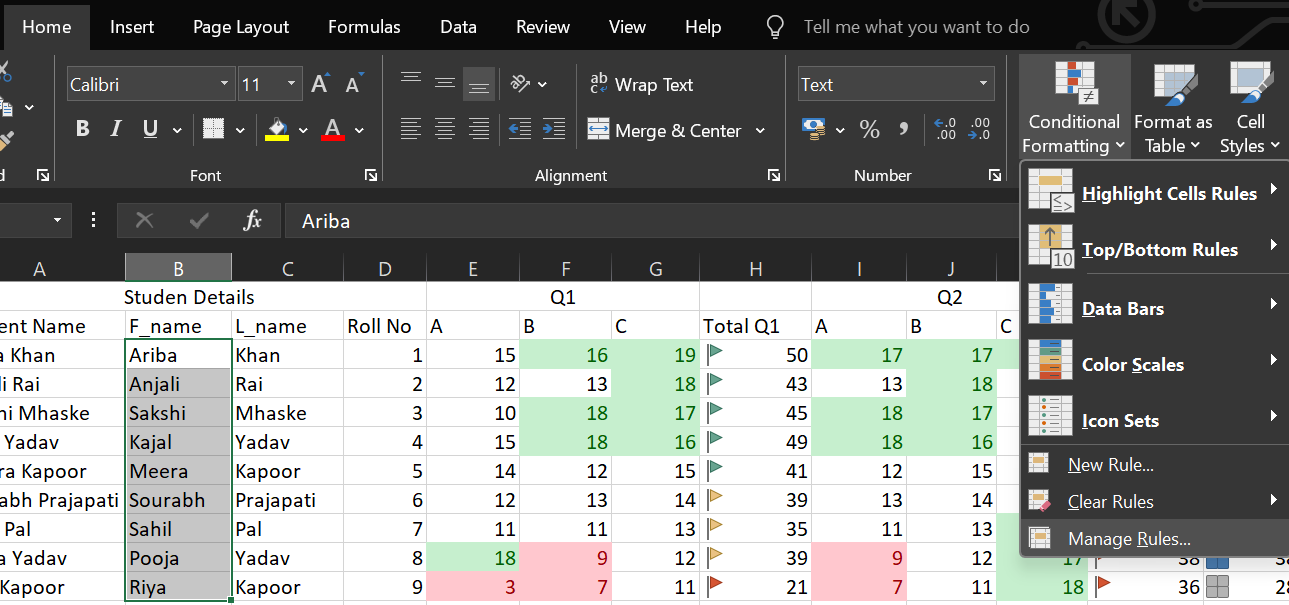
Select values of the Average column -> Conditional Formatting (Home tab) -> Click Manage Rules



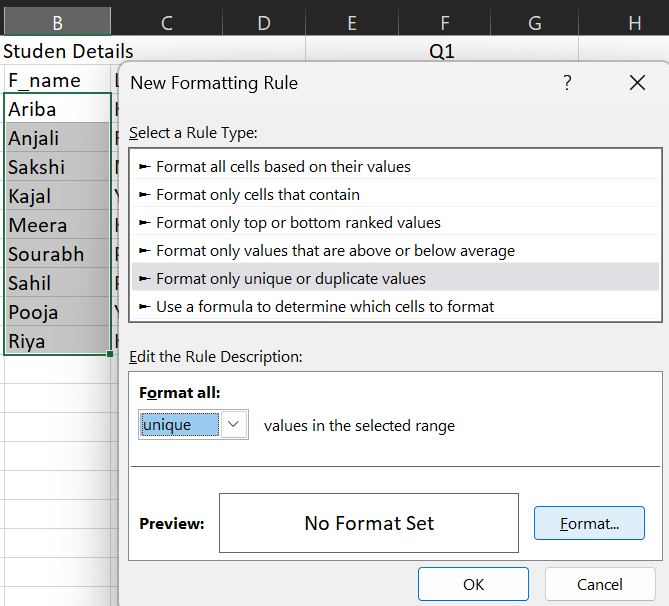
You can also edit from here



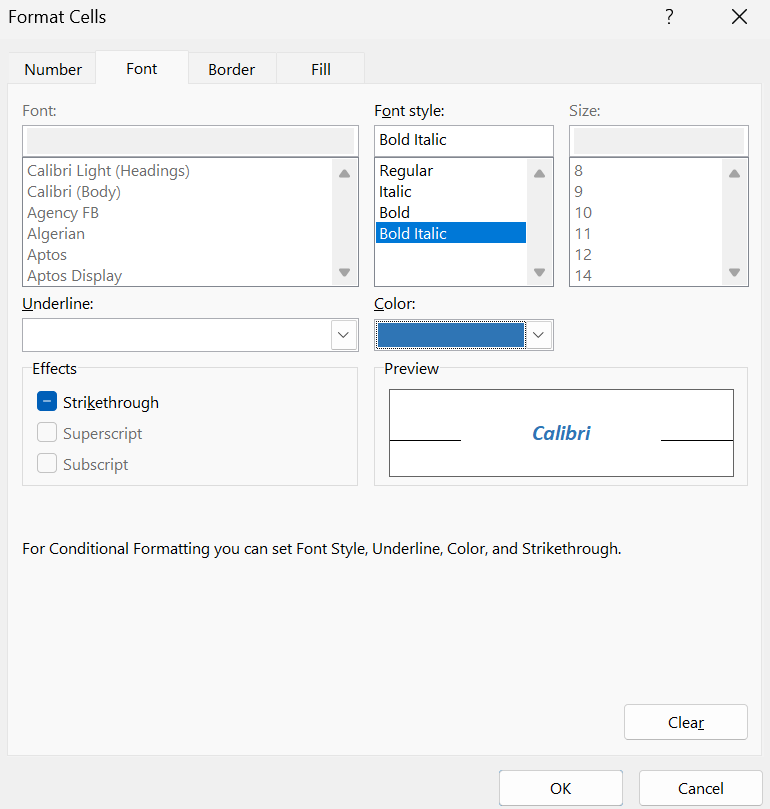
Select values of F\_name -> Conditional Formatting (Home tab) -> Manage rules -> New Rule



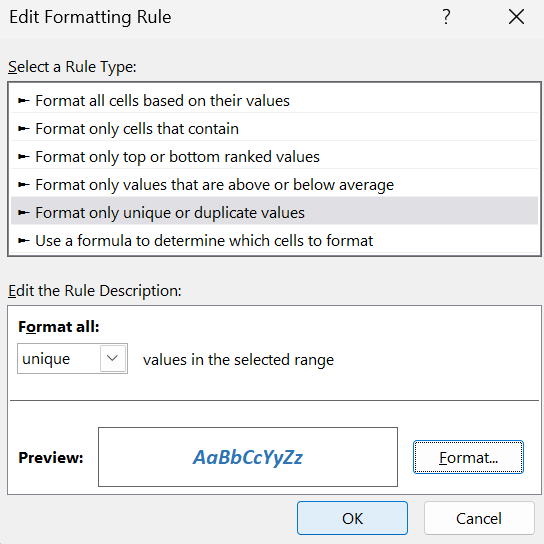
Select the following and click Format



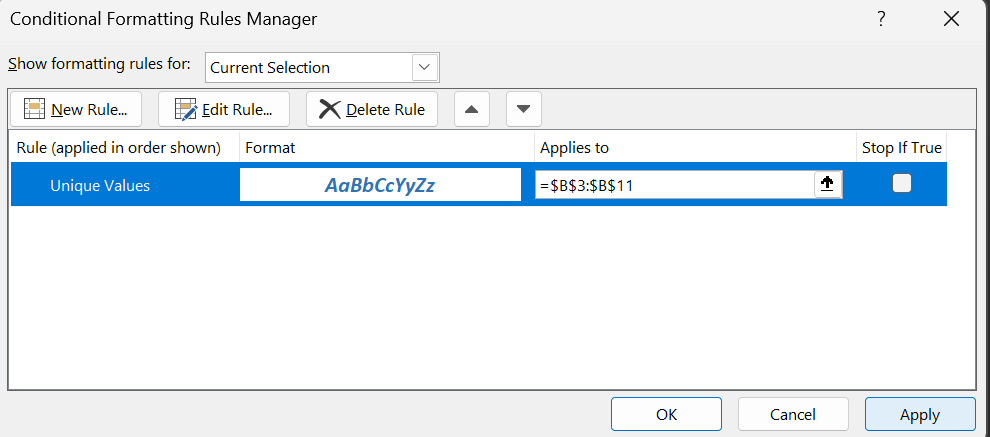
Make changes as you want -> Click OK



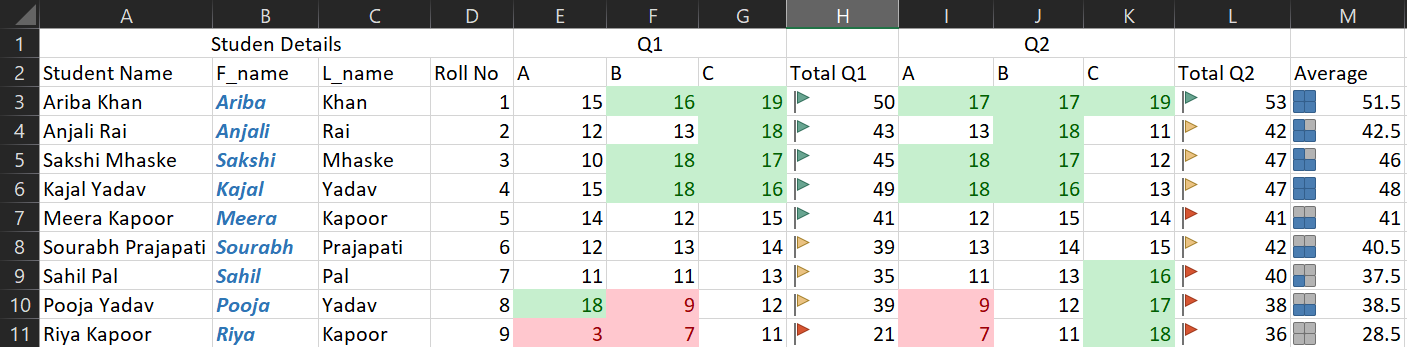
Click Ok



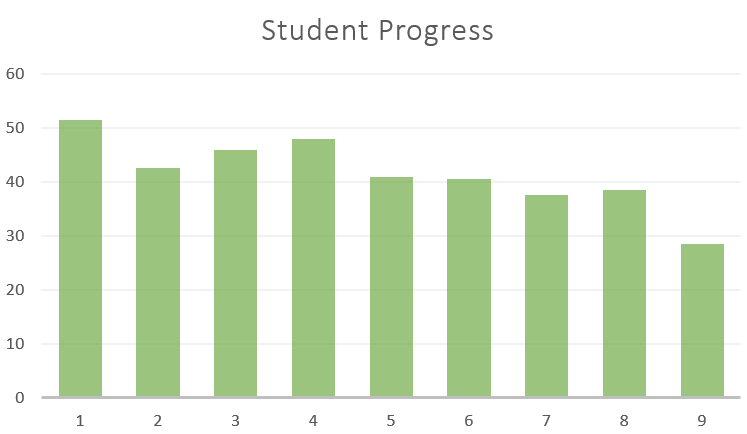
Click Apply -> Click OK



Result:

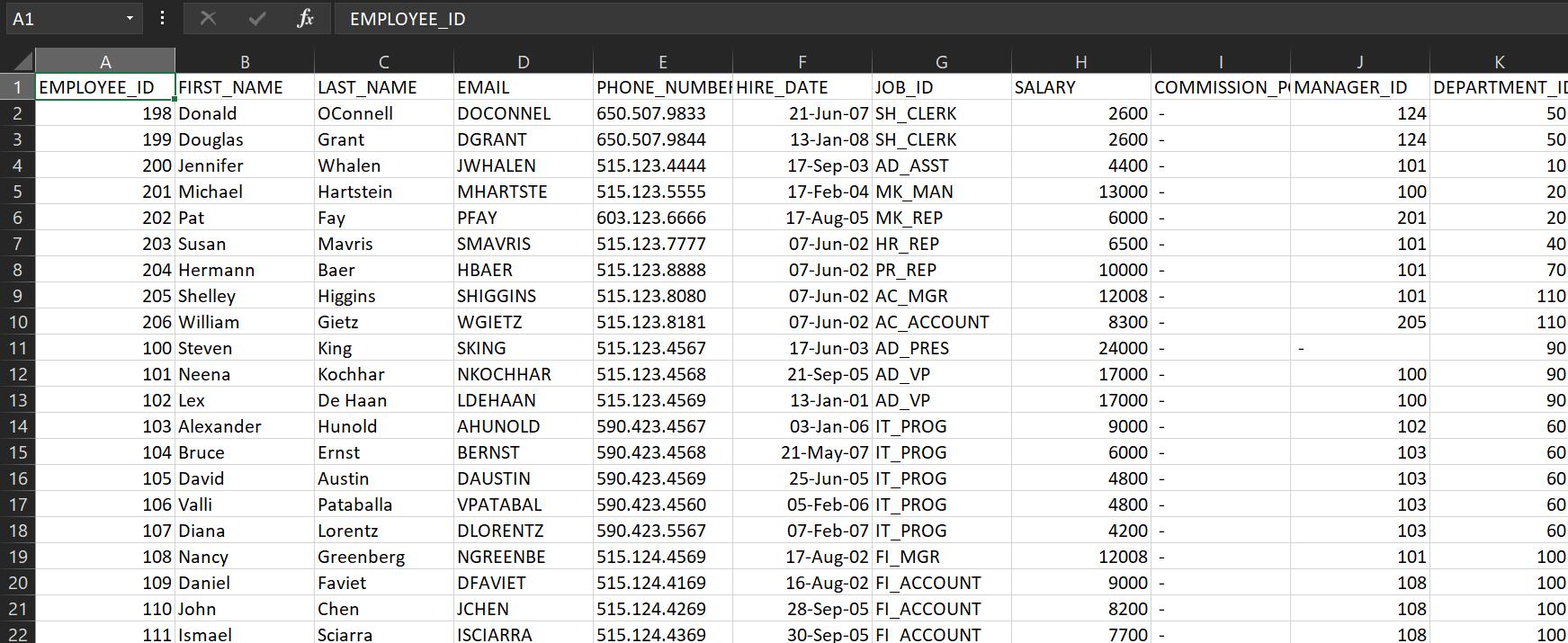


Line chart for Roll no and Average

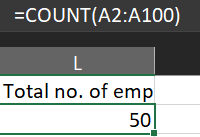


Employee data csv file : <https://gist.github.com/kevin336/acbb2271e66c10a5b73aacf82ca82784>

Open the csv file

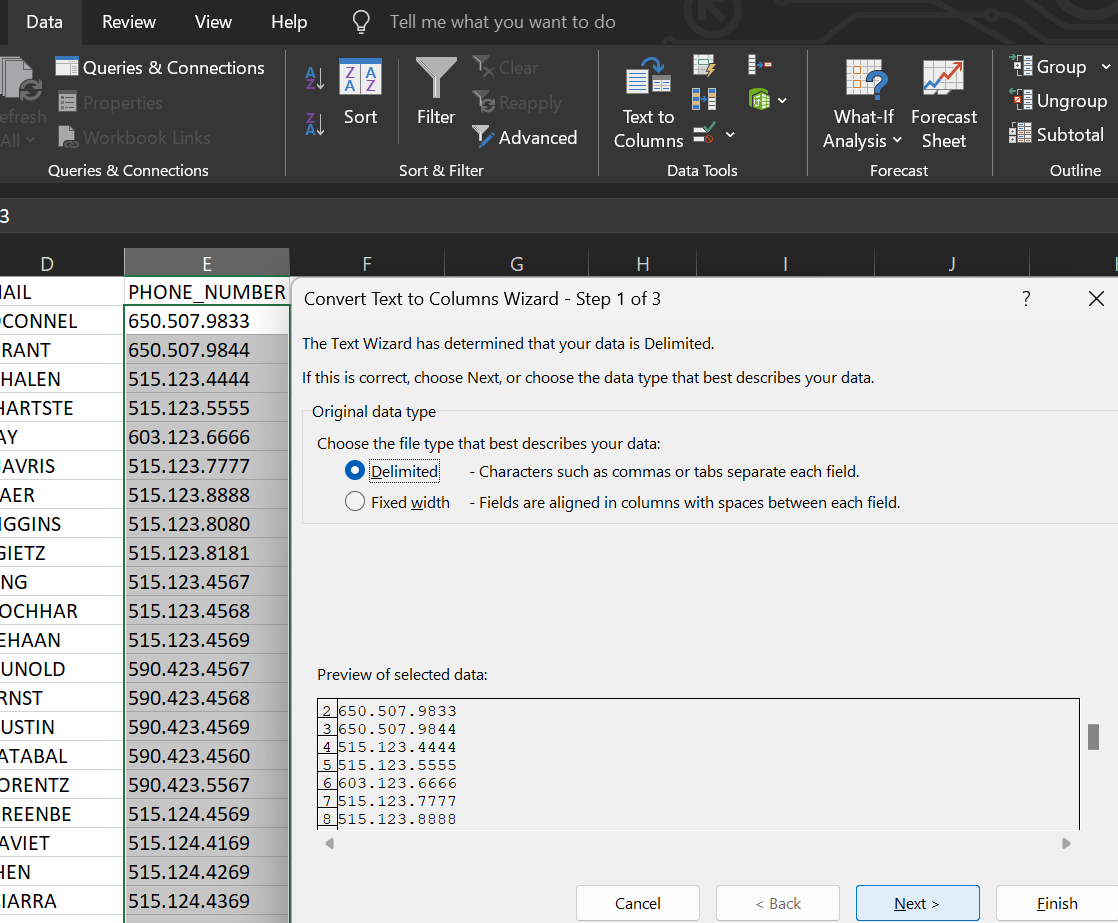


1. Total employee

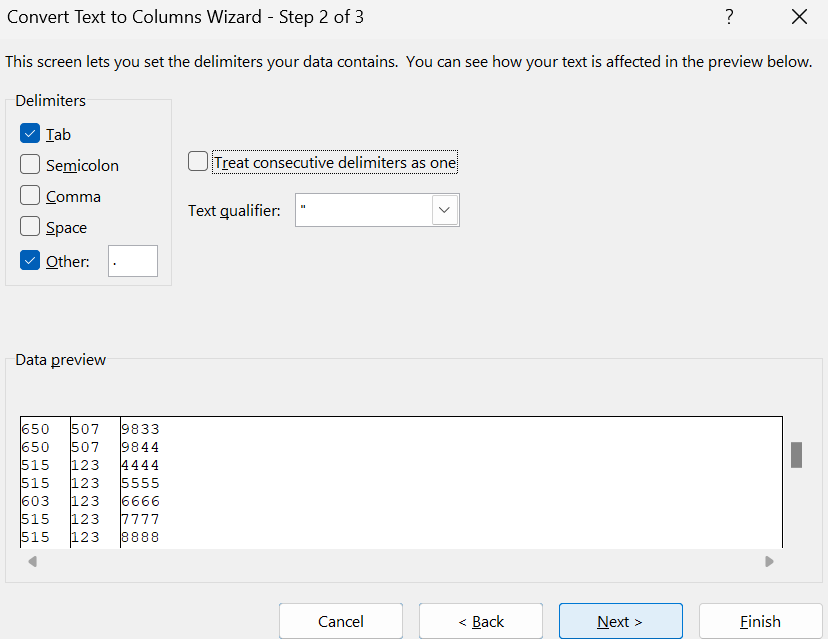


2. List the Emp with region code 650

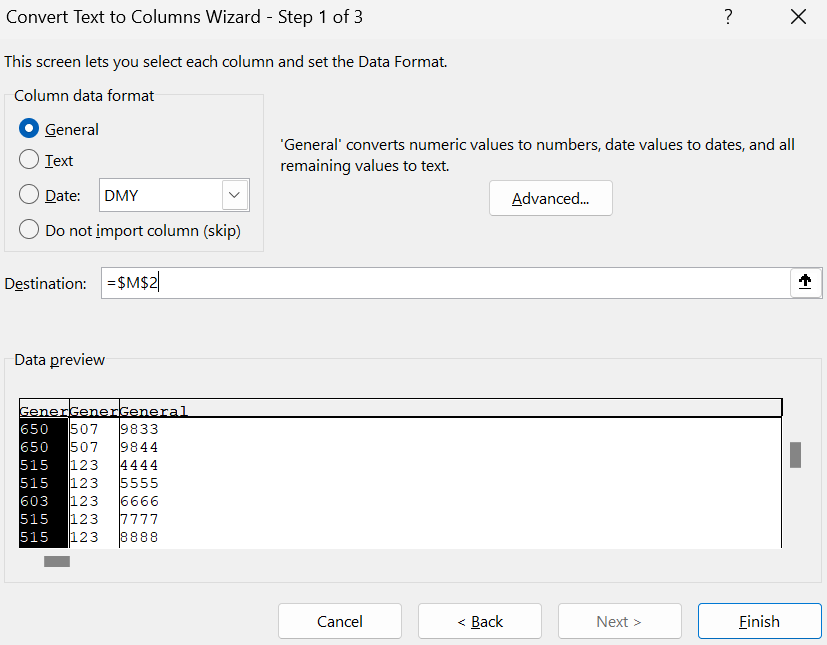
Do the following steps



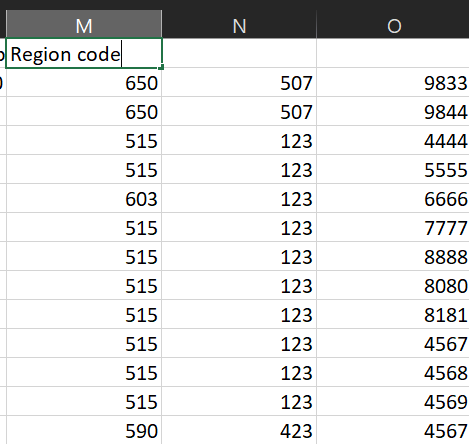
Enter dot in Other



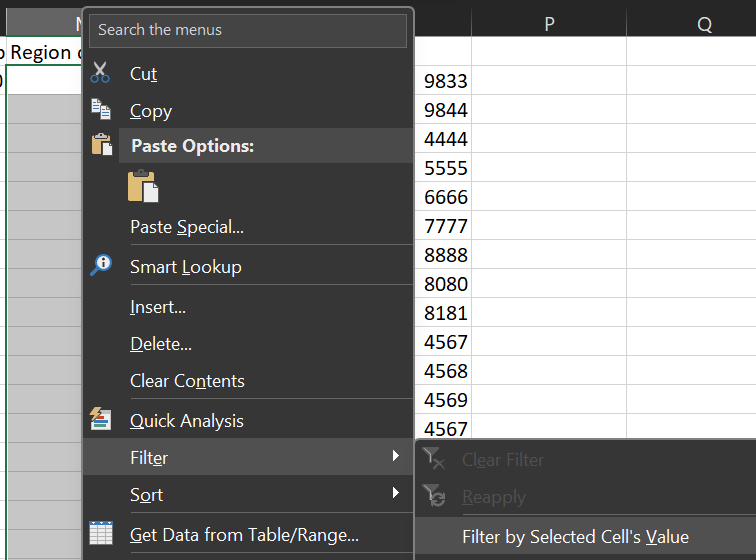
Enter destination



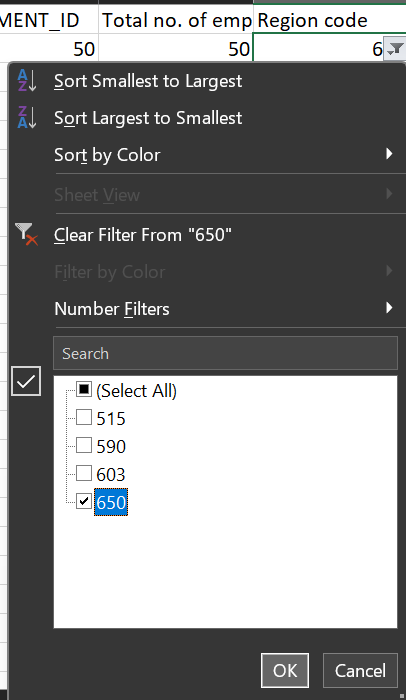
Give the column name as Region code



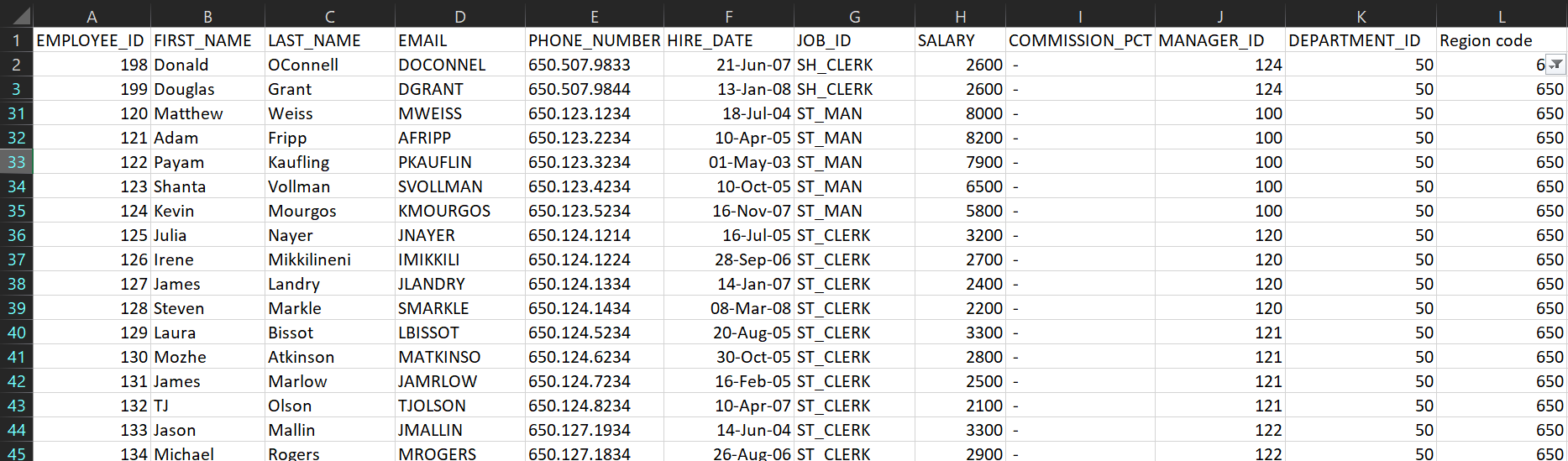
Select values of Region code column -> Right click -> Filter -> Filter by Selected Cell’s Value



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

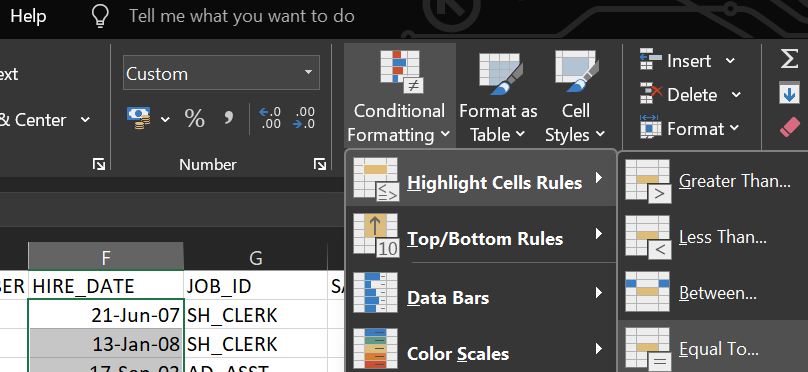


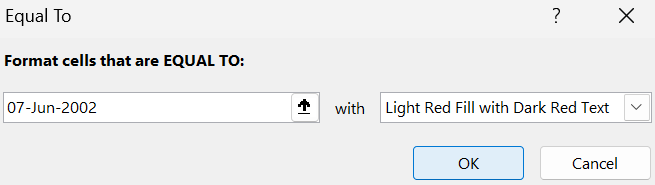
Results:



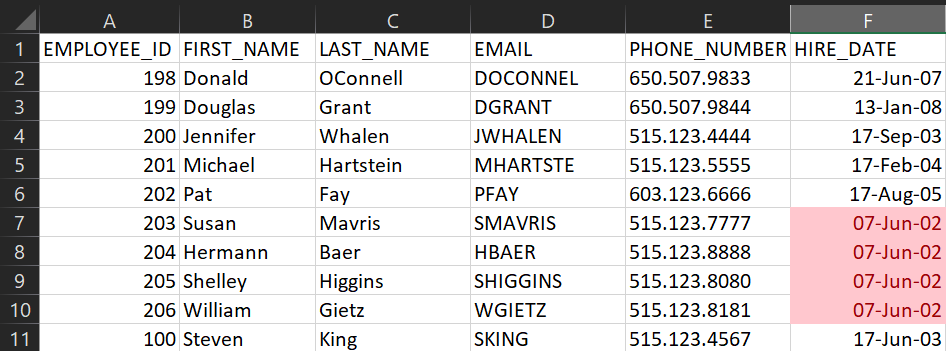
3. Highlight the emp with the joining date as 7 june 2002

Do the following steps



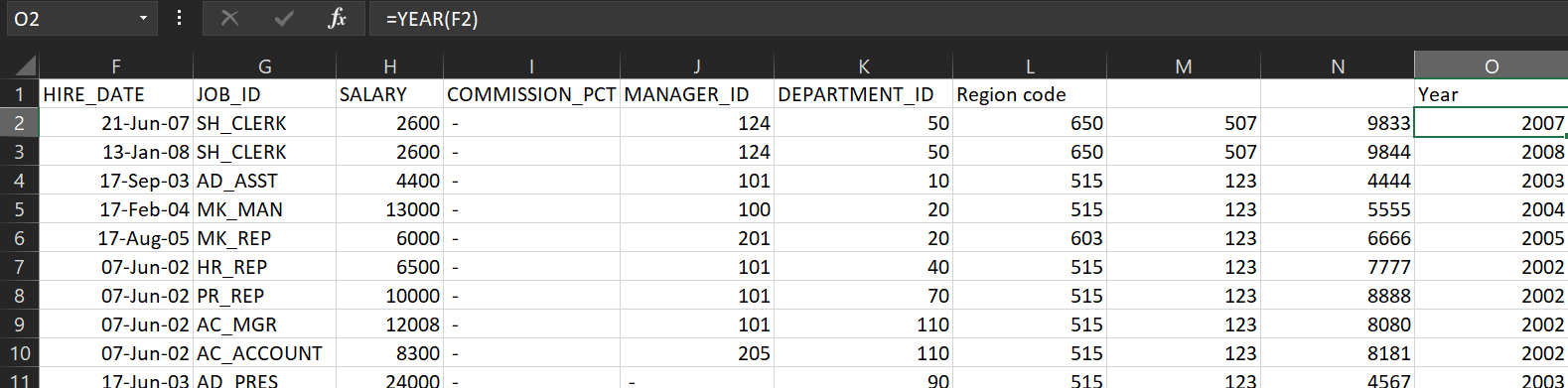


Results:

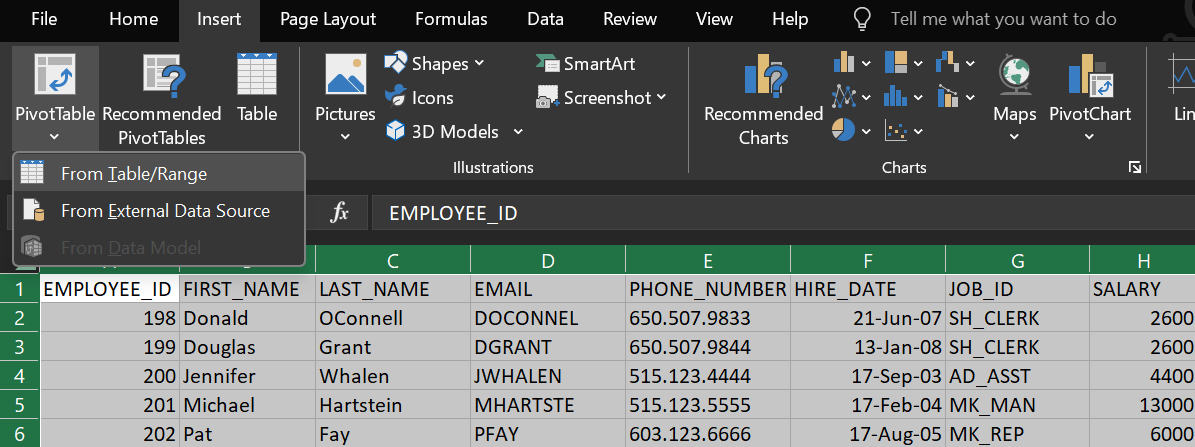


4. Get year wise salary trend

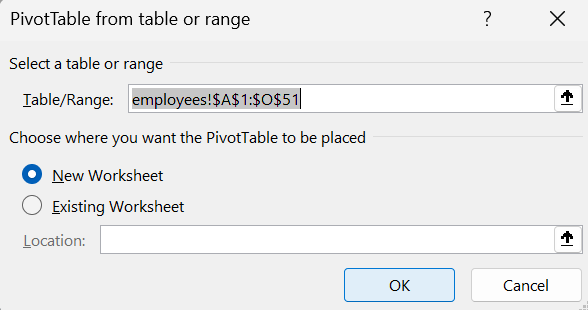
Create a new column Year



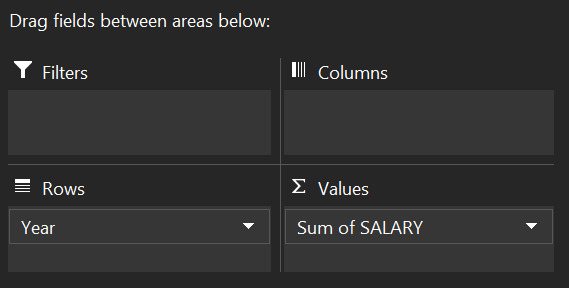
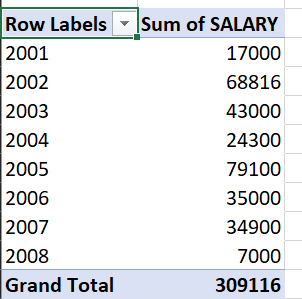
Select entire sheet -> Insert tab -> Pivot Table -> From Table/Range



Click OK



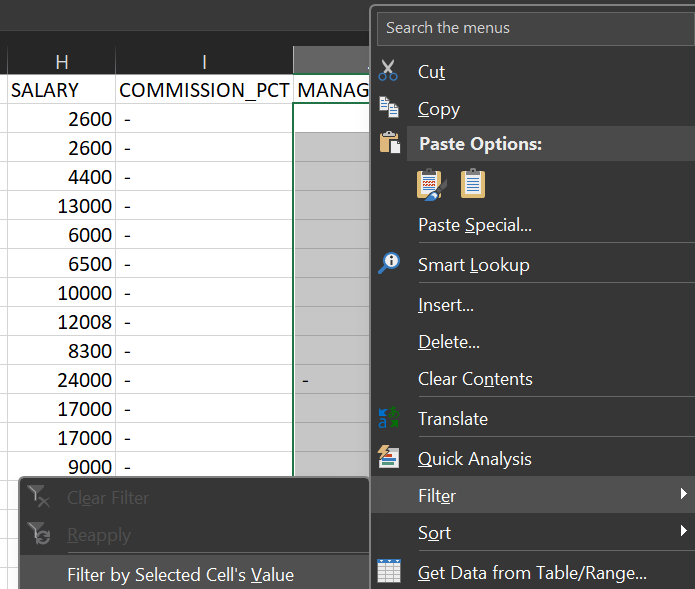
Drag Year column to rows and Salary column to Values

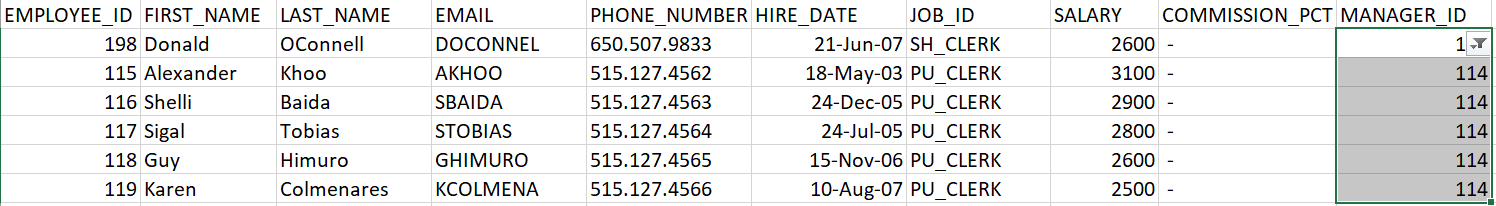


5. List Designations with Highest and Lowest Salary

6. List emp who are working under manager with manager id 114

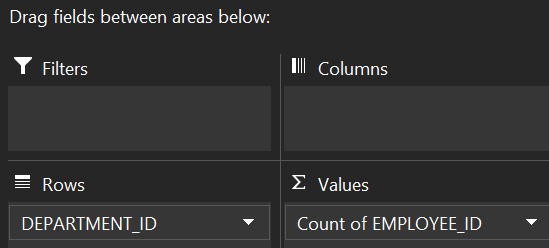
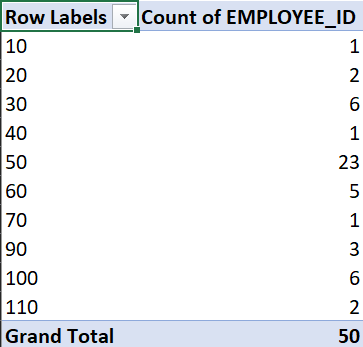
Do the following steps (similar to steps of filter)





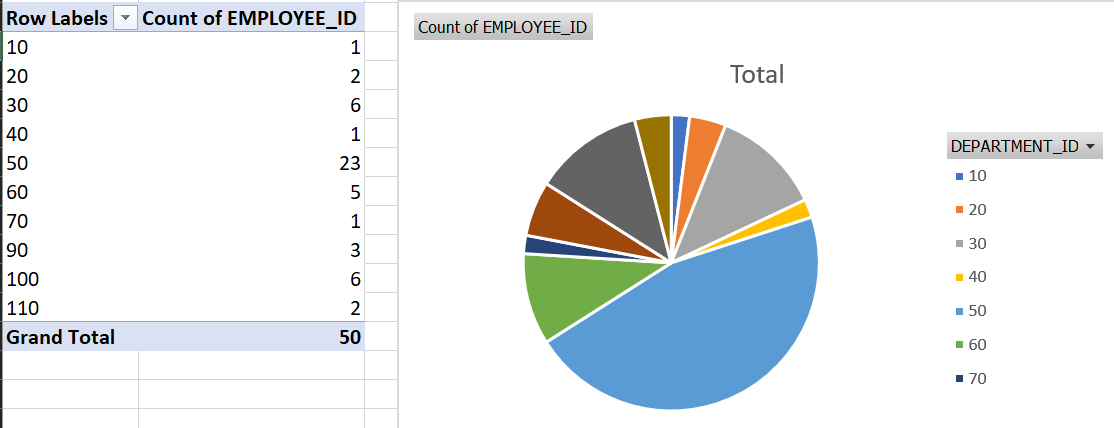
7. Count Employees in Each Department

Drag Department\_ID in rows and Count of Employee\_ID in values



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

