

Question 1

**Given below is a dictionary having two keys ‘Boys’ and ‘Girls’ and having two lists of heights of five Boys and five Girls respectively as values associated with these keys.**

**Original dictionary of lists:**

**{'Boys': [72, 68, 70, 69, 74], 'Girls': [63, 65, 69, 62, 61]}**

**From the given dictionary of lists create the following list of dictionaries:**

**[{'Boys': 72, 'Girls': 63}, {'Boys': 68, 'Girls': 65}, {'Boys': 70, 'Girls': 69}, {'Boys': 69, 'Girls': 62}, {‘Boys’:74, ‘Girls’:61}]**

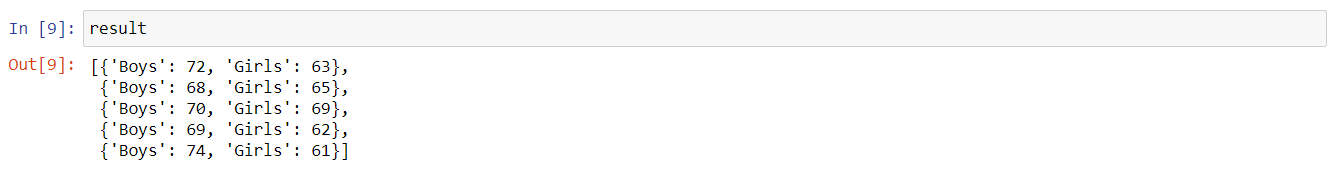
CODE

height = {'Boys': [72, 68, 70, 69, 74], 'Girls': [63, 65, 69, 62, 61]}

result = [dict(zip(height.keys(), i)) for i in zip(\*height.values())]

result

OUTPUT



Question 2

**Write programs in Python using NumPy library to do the following:**

1. **Compute the mean, standard deviation, and variance of a two dimensional random integer array along the second axis.**
2. **Get the indices of the sorted elements of a given array.**

**B = [56, 48, 22, 41, 78, 91, 24, 46, 8, 33]**

1. **Create a 2-dimensional array of size m x n integer elements, also print the shape, type and data type of the array and then reshape it into an n x m array, n and m are user inputs given at the run time.**
2. **Test whether the elements of a given array are zero, non-zero and NaN. Record the indices of these elements in three separate arrays.**
3. import numpy as np

array2d = np.random.randint(100, size = (3, 4))

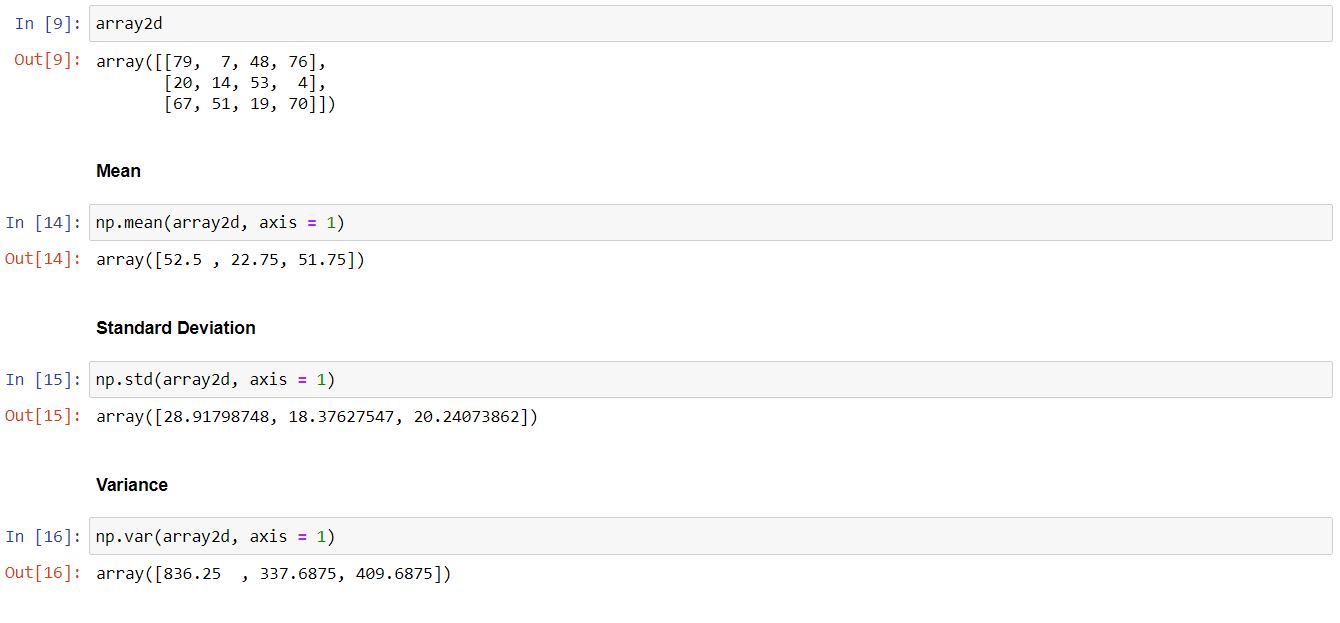
array2d

np.mean(array2d, axis = 1)

np.std(array2d, axis = 1)

np.var(array2d, axis = 1)

OUTPUT



1. b = [56, 48, 22, 41, 78, 91, 24, 46, 8, 33]

sorted\_indices = np.argsort(b)

print(sorted\_indices)

OUTPUT



1. m = int(input('Enter the value of m: '))

n = int(input('Enter the value of n: '))

arr2d = np.random.randint(100, size = (m, n))

print(arr2d)

print(arr2d.shape)

print(arr2d.ndim)

print(arr2d.dtype)

arr\_1 = np.reshape(arr2d, (n, m))

print(arr\_1)

OUTPUT





1. arr\_2 = np.array([[0, 2, 3], [4, 1, 0], [0, 0, 2], [np.nan, 3, np.nan]])

print(arr\_2)

indices\_zero = np.argwhere(arr\_2 == 0)

print(indices\_zero)

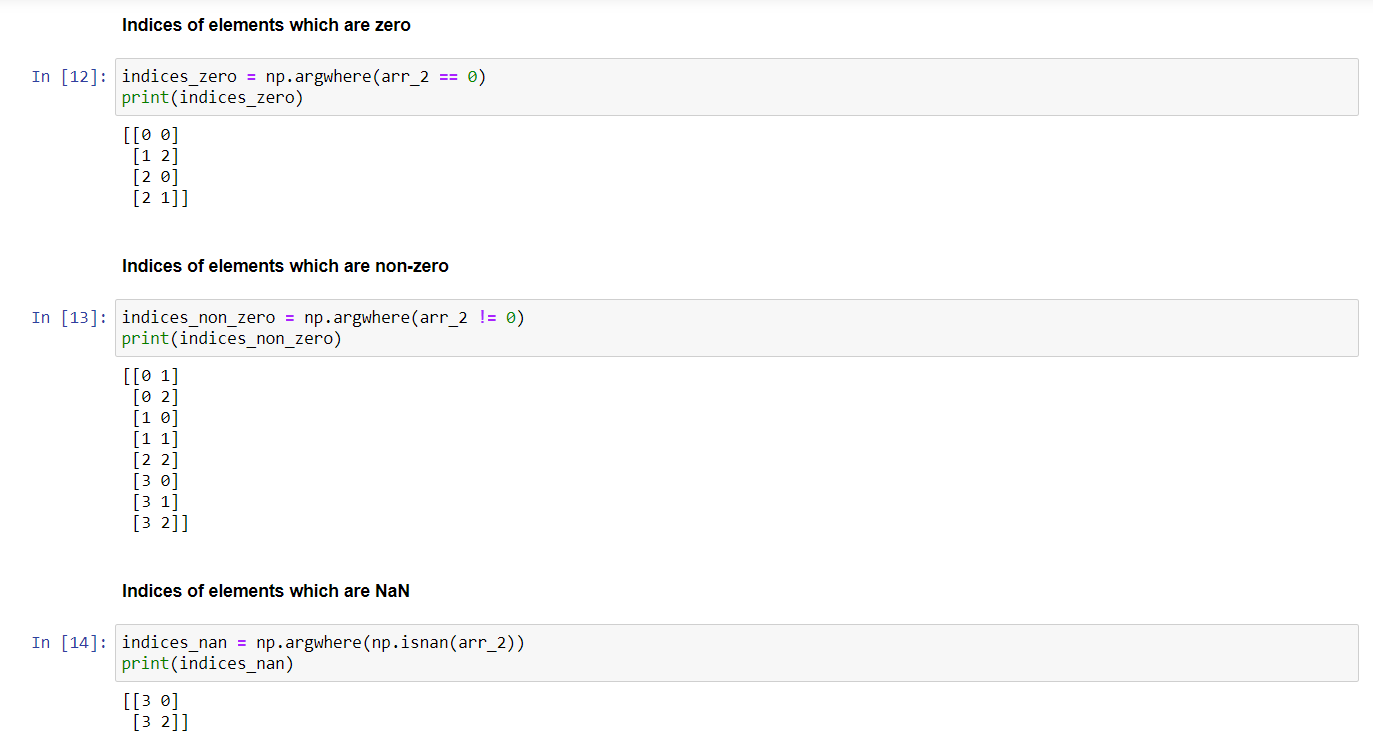
indices\_non\_zero = np.argwhere(arr\_2 != 0)

print(indices\_non\_zero)

indices\_nan = np.argwhere(np.isnan(arr\_2))

print(indices\_nan)

OUTPUT



Question 3

**Create a dataframe having at least 3 columns and 50 rows to store numeric data generated using a random function. Replace 10% of the values by null values whose index positions are generated using random function. Do the following:**

1. **Identify and count missing values in a dataframe.**
2. **Drop the column having more than 5 null values.**
3. **Identify the row label having maximum of the sum of all values in a row and drop that row.**
4. **Sort the dataframe on the basis of the first column.**
5. **Remove all duplicates from the first column.**
6. **Find the correlation between first and second column and covariance between second and third column.**
7. **Discretize the second column and create 5 bins.**

import pandas as pd

import numpy as np

frame = pd.DataFrame(np.random.randint(0, 50, size=(50, 3)), columns=list('ABC'))

rows = len(frame)

cols = len(frame.columns)

no\_of\_elements\_to\_replace = int(rows \* cols \* 0.1)

while no\_of\_elements\_to\_replace != 0:

i = np.random.randint(rows)

j = np.random.randint(cols)

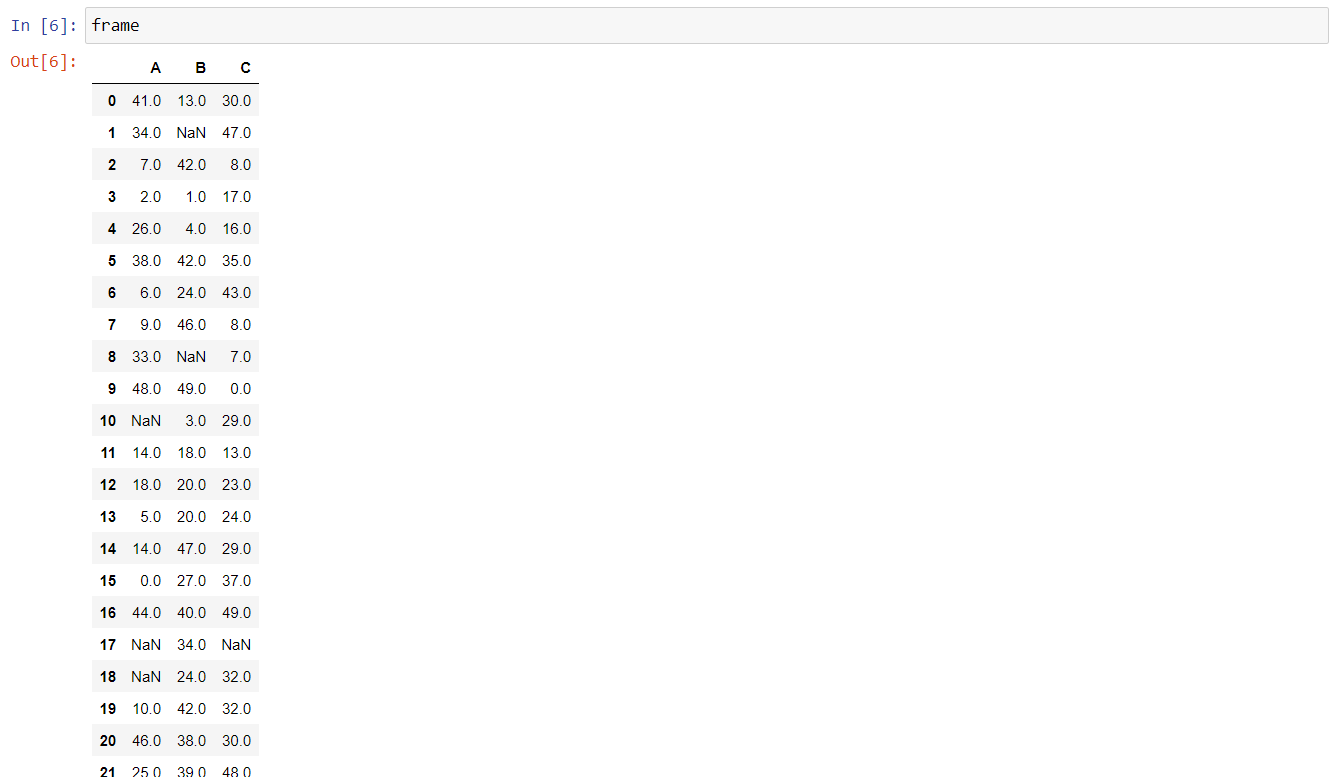
if frame.iloc[i, j] != np.nan:

frame.iat[i, j] = np.nan

no\_of\_elements\_to\_replace -= 1

frame

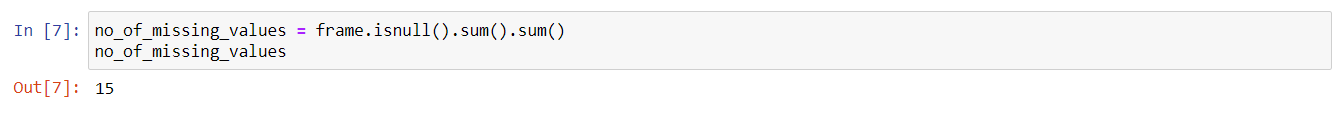
OUTPUT



1. no\_of\_missing\_values = frame.isnull().sum().sum()

no\_of\_missing\_values

OUTPUT



1. frame.dropna(axis=1, how='any', thresh=rows-5)

OUTPUT





1. row\_to\_drop = frame.sum(axis=1).idxmax()

frame.drop(row\_to\_drop)

OUTPUT





1. frame.sort\_values(by=frame.columns[0])

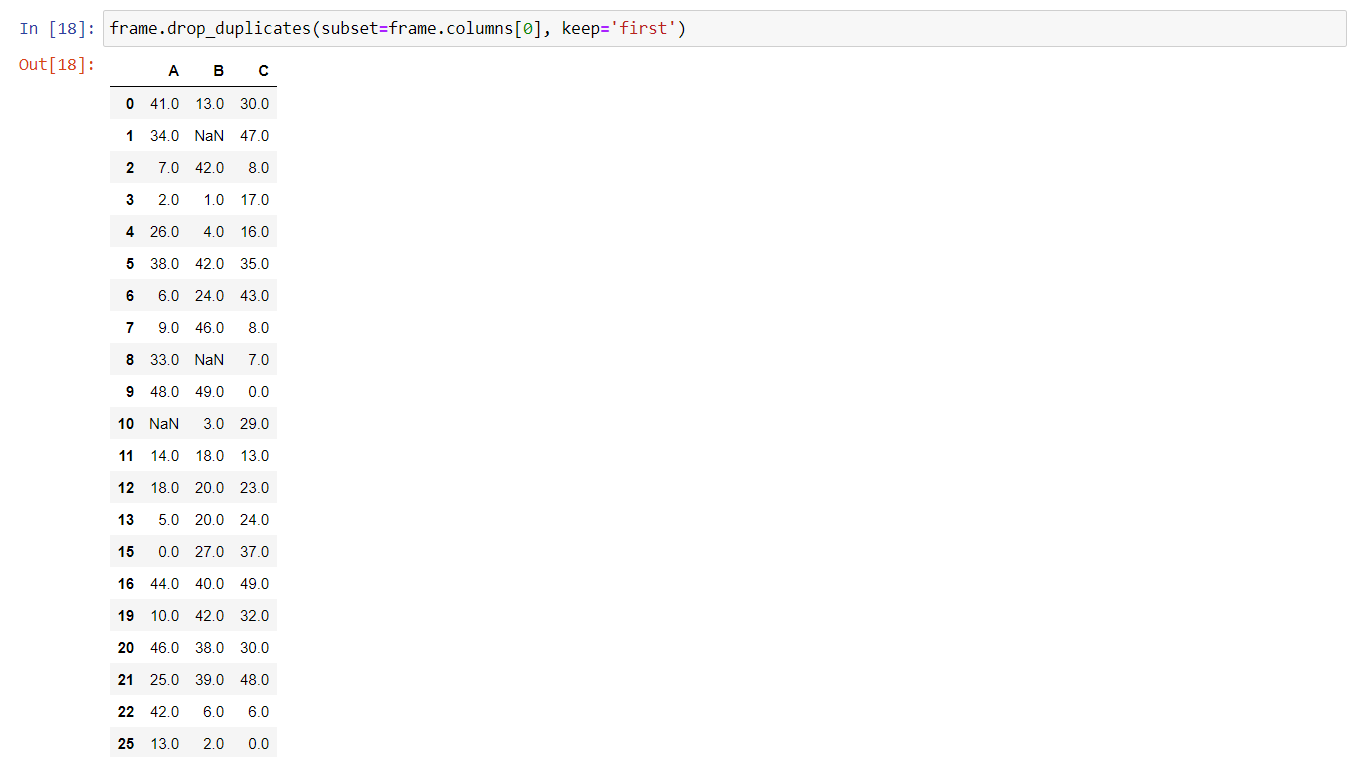
OUTPUT





1. frame.drop\_duplicates(subset=frame.columns[0], keep='first')

OUTPUT



1. correlation = frame['A'].corr(frame['B'])

correlation

covariance = frame['B'].cov(frame['C'])

covariance

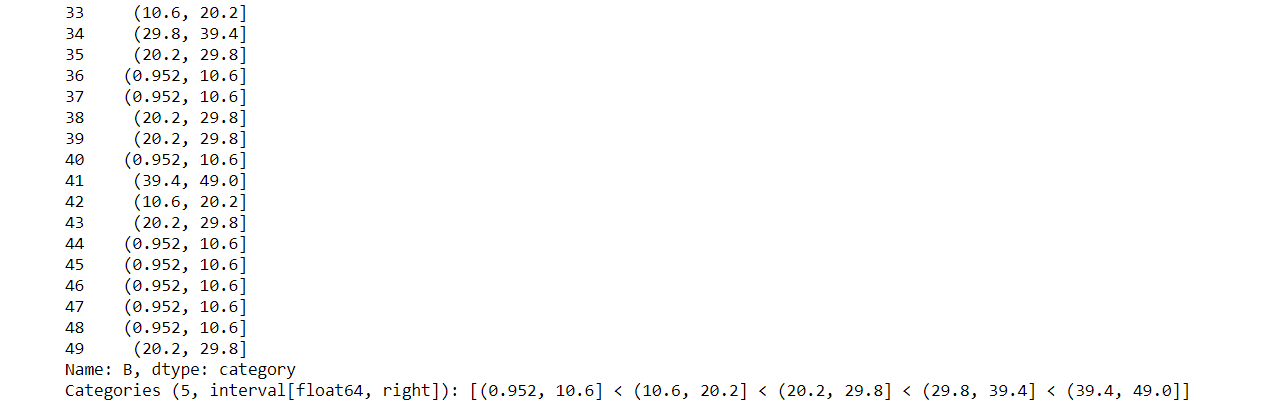
OUTPUT



1. pd.cut(frame['B'], 5)

OUTPUT





Question 4

**Consider two excel files having attendance of a workshop’s participants for two days. Each file has three fields ‘Name’, ‘Time of joining’, duration (in minutes) where names are unique within a file. Note that duration may take one of three values (30, 40, 50) only. Import the data into two dataframes and do the following:**

1. **Perform merging of the two dataframes to find the names of students who had attended the workshop on both days.**
2. **Find names of all students who have attended workshop on either of the days.**
3. **Merge two data frames row-wise and find the total number of records in the data frame.**

import pandas as pd

first\_workshop = pd.read\_excel('Attendence-1.xlsx')

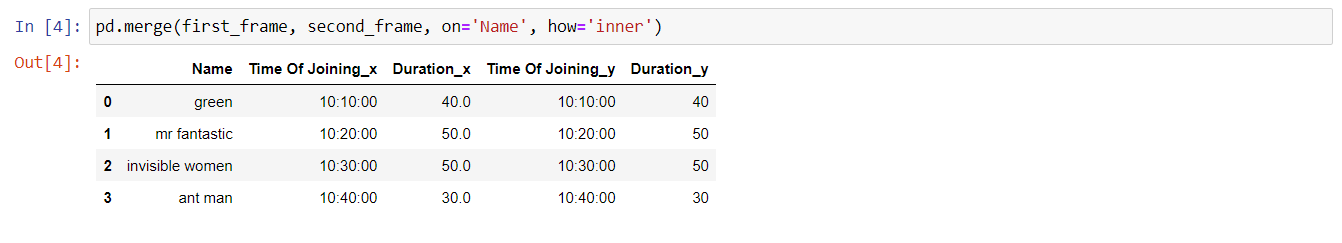
second\_workshop = pd.read\_excel('Attendence-2.xlsx')

first\_frame = pd.DataFrame(first\_workshop)

second\_frame = pd.DataFrame(second\_workshop)

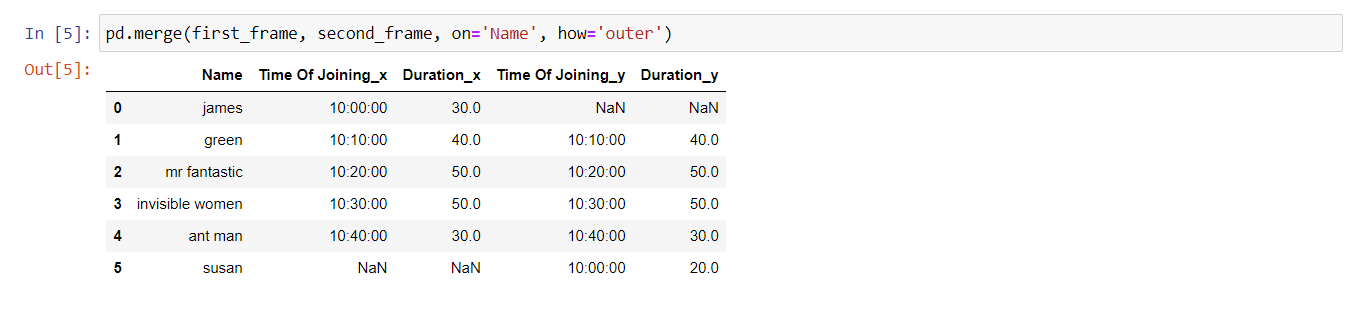
1. pd.merge(first\_frame, second\_frame, on='Name', how='inner')

OUTPUT



1. pd.merge(first\_frame, second\_frame, on='Name', how='outer')

OUTPUT

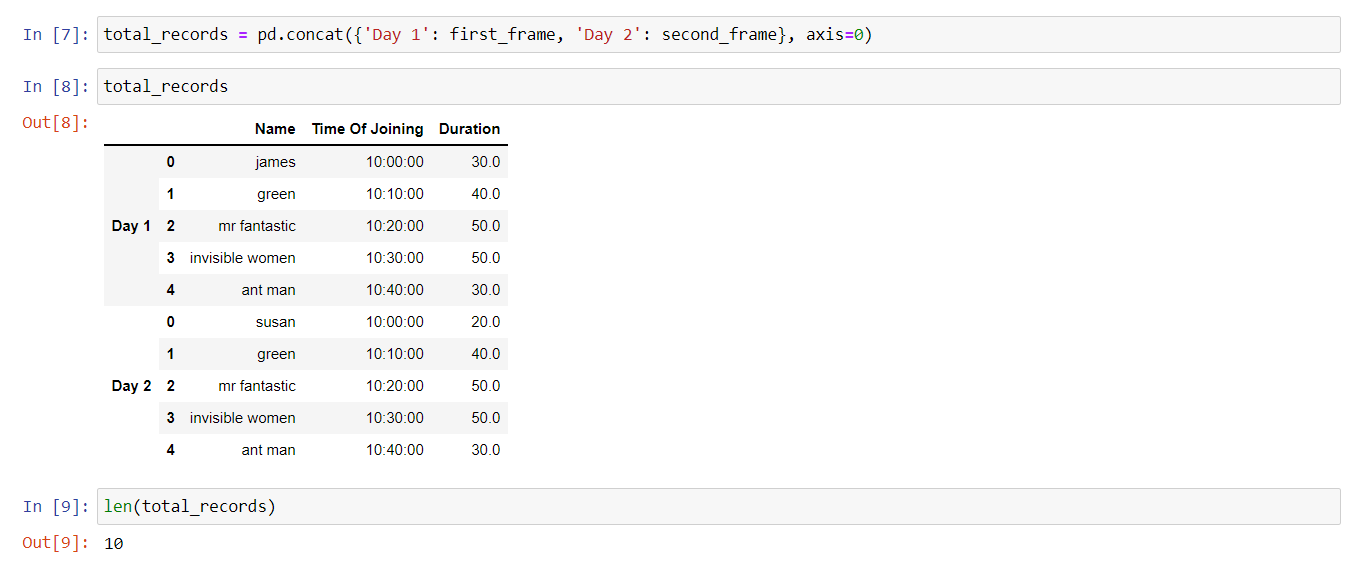


1. total\_records = pd.concat({'Day 1': first\_frame, 'Day 2': second\_frame}, axis=0)

total\_records

len(total\_records)

OUTPUT



Question 5

**Taking Iris data, plot the following with proper legend and axis labels: (Download IRIS data from: https://archive.ics.uci.edu/ml/datasets/iris or import it from sklearn.datasets)**

1. **Plot bar chart to show the frequency of each class label in the data.**
2. **Draw a scatter plot for Petal width vs sepal width.**
3. **Plot density distribution for feature petal length.**
4. **Use a pair plot to show pairwise bivariate distribution in the Iris Dataset.**

import pandas as pd

import seaborn as sns

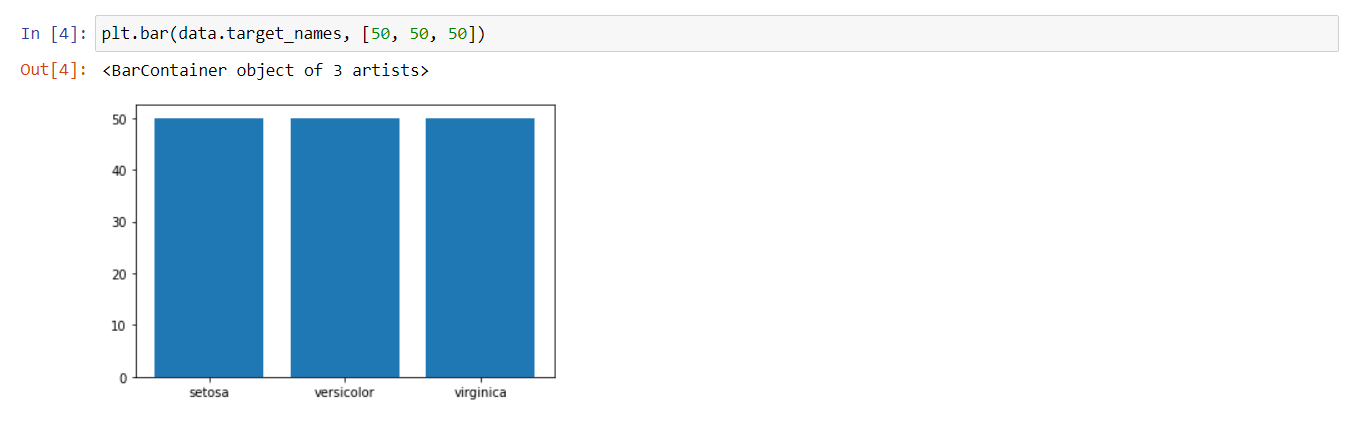
import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

data = load\_iris()

1. plt.bar(data.target\_names, [50, 50, 50])

OUTPUT



1. frame = pd.DataFrame(data.data, columns=data.feature\_names)

frame.head()

sepalWidth = data.feature\_names[1]

petalWidth = data.feature\_names[3]

plt.scatter(frame[petalWidth], frame[sepalWidth])

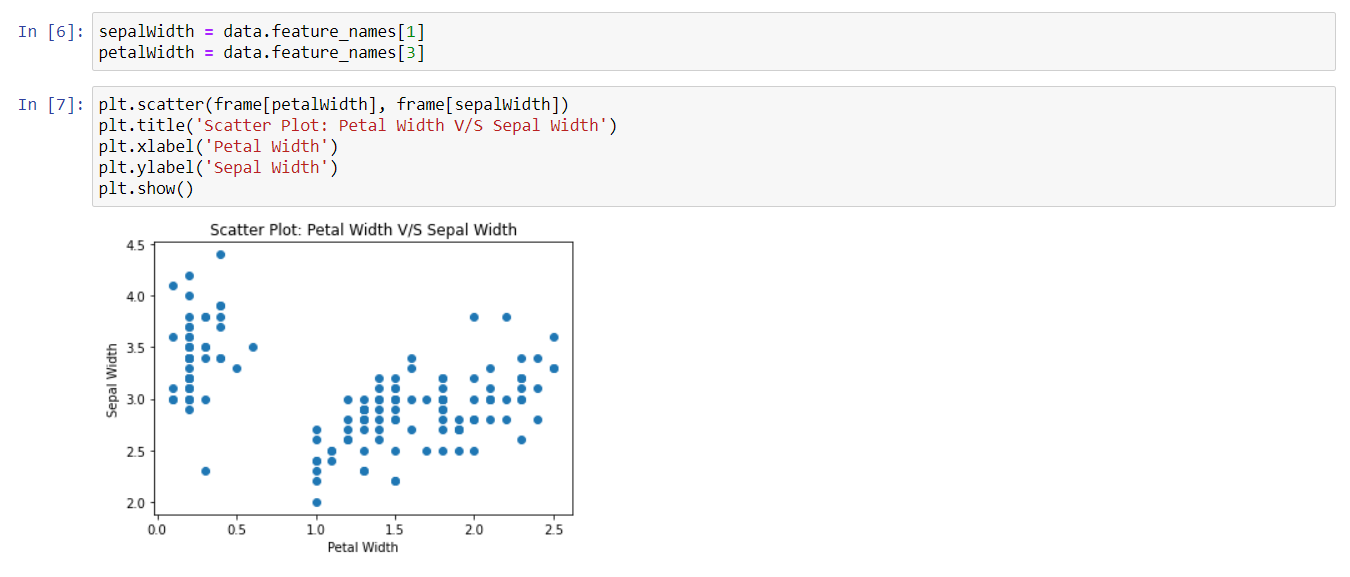
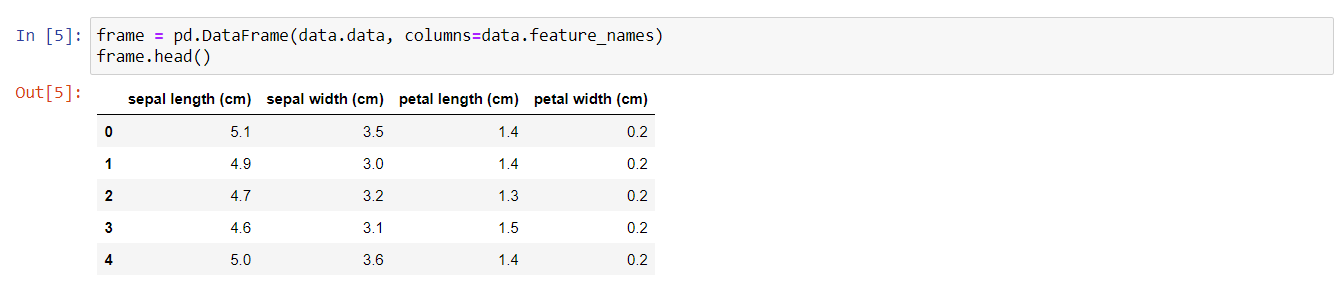
plt.title('Scatter Plot: Petal Width V/S Sepal Width')

plt.xlabel('Petal Width')

plt.ylabel('Sepal Width')

plt.show()

OUTPUT



1. petalLength = data.feature\_names[2]

fig = plt.figure()

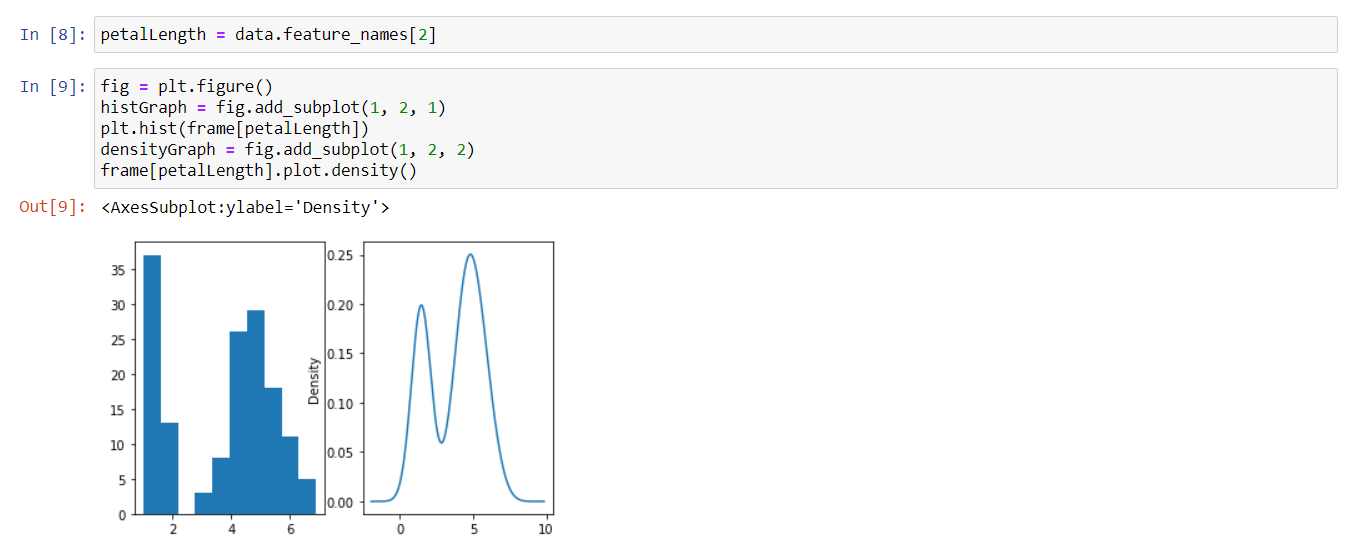
histGraph = fig.add\_subplot(1, 2, 1)

plt.hist(frame[petalLength])

densityGraph = fig.add\_subplot(1, 2, 2)

frame[petalLength].plot.density()

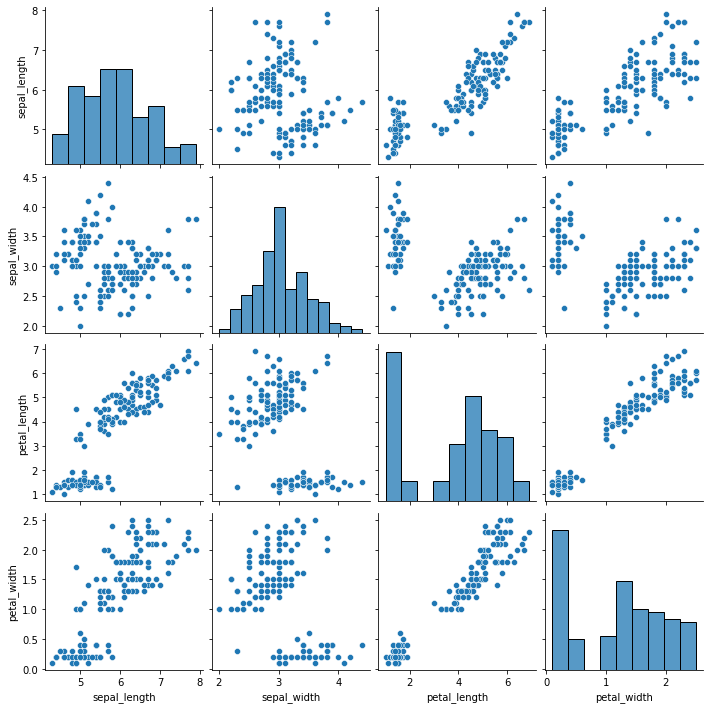
OUTPUT



1. data = sns.load\_dataset('iris')

sns.pairplot(data)

OUTPUT



Question 6

**Consider any sales training/ weather forecasting dataset.**

1. **Compute the mean of a series grouped by another series.**
2. **Fill an intermittent time series to replace all missing dates with values of previous non-missing dates.**
3. **Perform appropriate year-month string to dates conversion.**
4. **Split a dataset to group by two columns and then sort the aggregated results within the groups.**
5. **Split a given dataframe into groups with bin counts.**

import pandas as pd

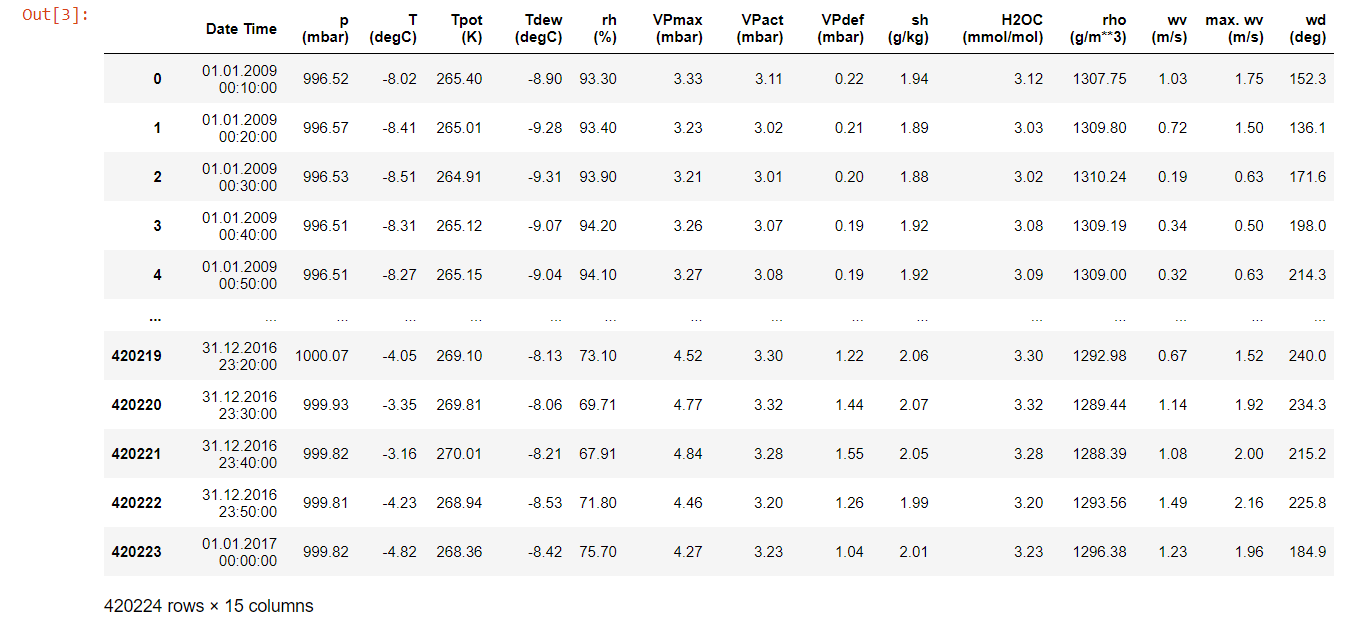
import numpy as np

frame = pd.read\_csv('climate.csv')

dataset = frame.drop\_duplicates(subset=['DateTime']).reset\_index(drop=True)

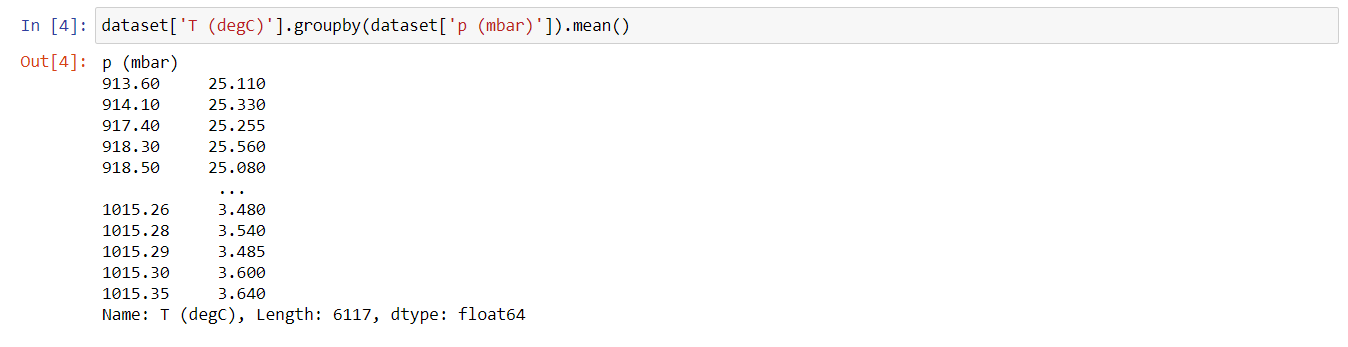
dataset

OUTPUT



1. dataset['T (degC)'].groupby(dataset['p (mbar)']).mean()

OUTPUT



1. rows\_to\_drop = np.random.choice(dataset.index, int(dataset.shape[0]\*25/100), replace=False)

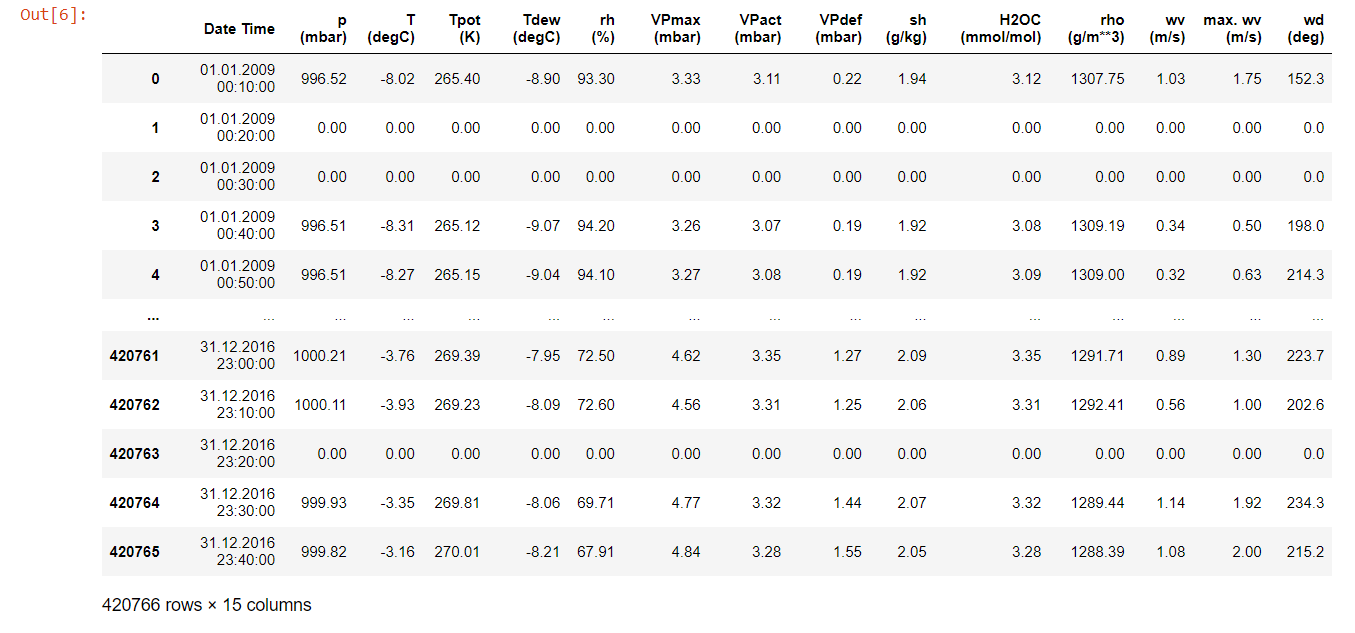
frame = dataset.drop(rows\_to\_drop).copy()

time\_series = pd.date\_range(frame['Date Time'].min(), frame['Date Time'].max(), freq='10T').strftime('%d.%m.%Y %H:%M:%S')

frame = frame.set\_index('Date Time').reindex(time\_series, fill\_value=0.0).rename\_axis('Date Time').reset\_index()

frame

OUTPUT



1. frame = dataset.copy()

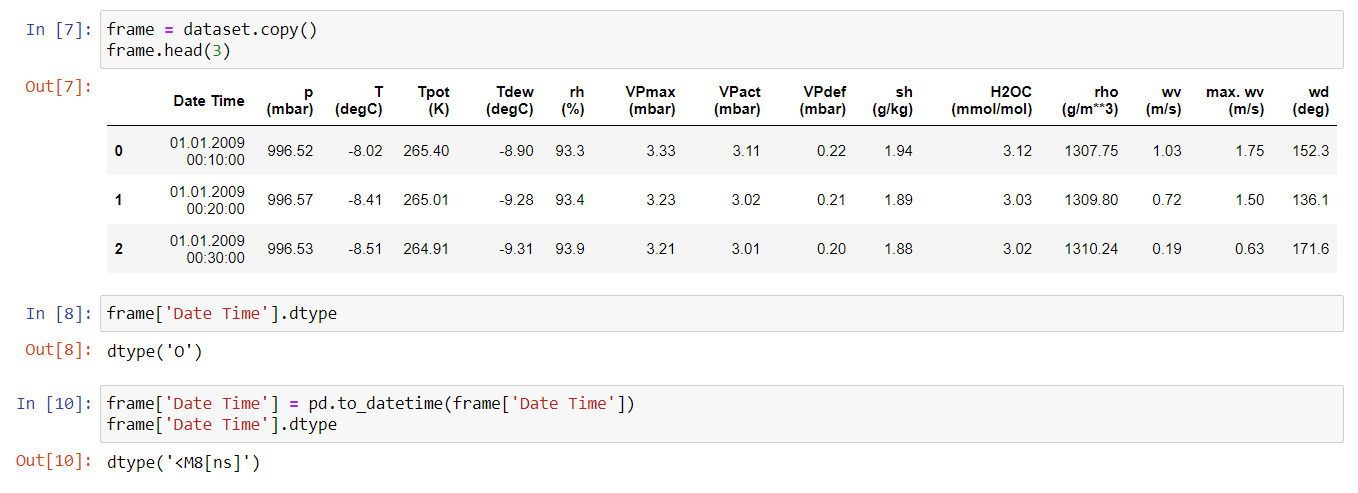
frame.head(3)

frame['Date Time'].dtype

frame['Date Time'] = pd.to\_datetime(frame['Date Time'])

frame['Date Time'].dtype

OUTPUT



1. frame.insert(1, 'Year', pd.DatetimeIndex(frame['Date Time']).year)

frame.insert(1, 'Month', pd.DatetimeIndex(frame['Date Time']).month\_name())

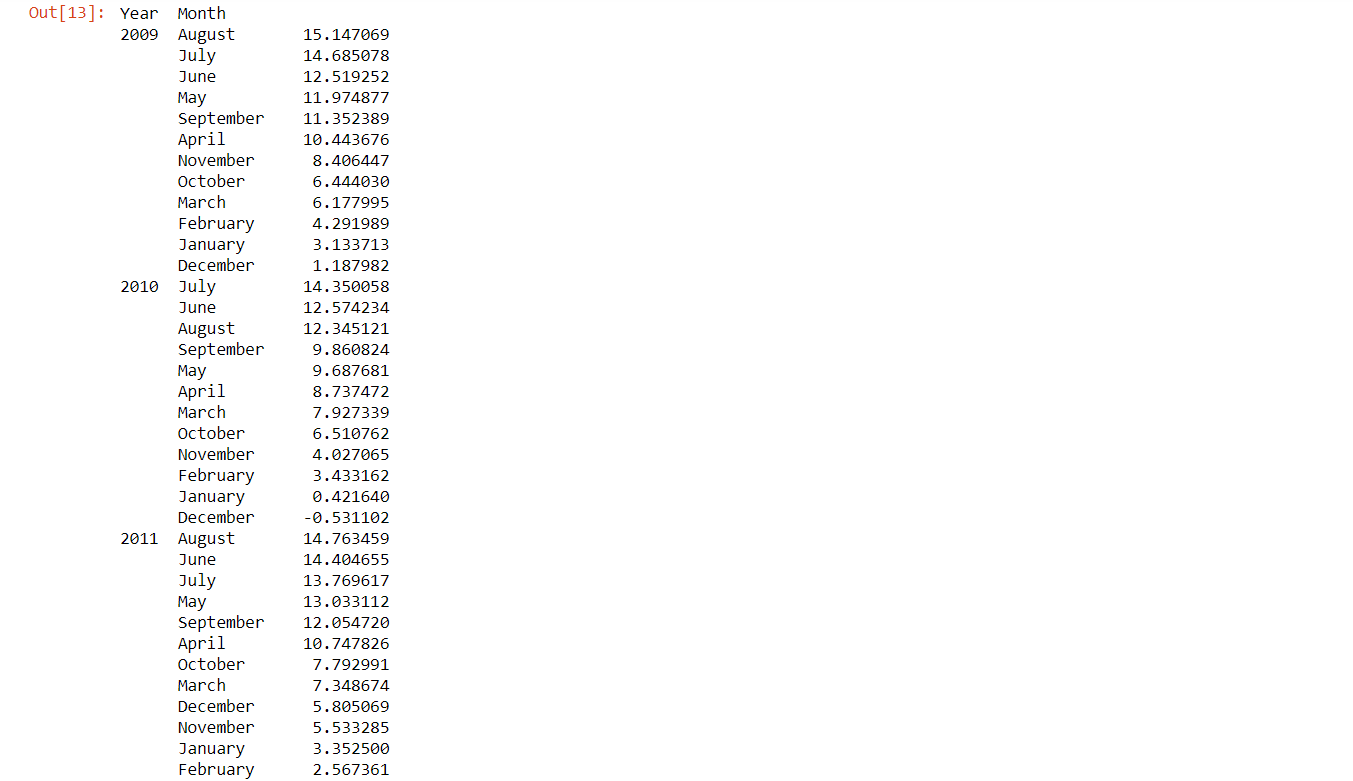
aggregate\_frame = frame.groupby(['Year', 'Month']).agg({'T (degC)': 'mean'})

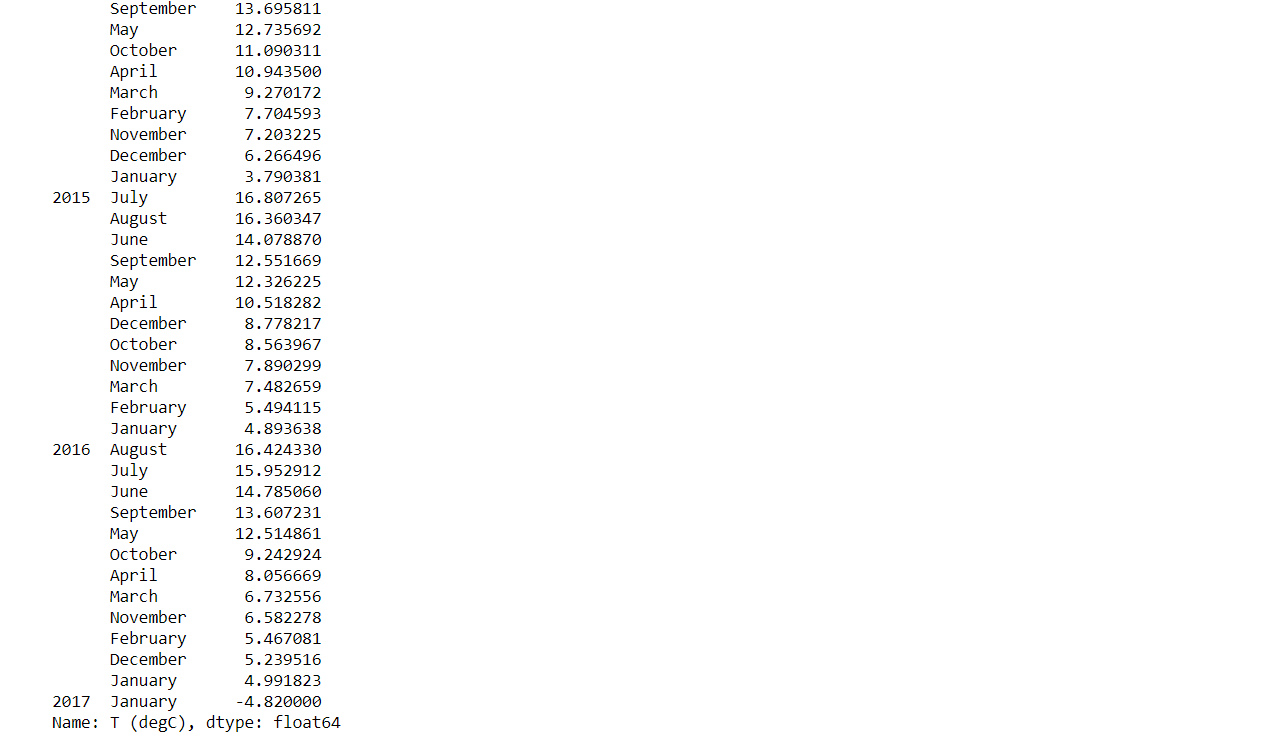
result = aggregate\_frame['T (degC)'].groupby(level=0, group\_keys=False)

pd.set\_option('display.max\_rows()', None)

result.nlargest(12)

OUTPUT





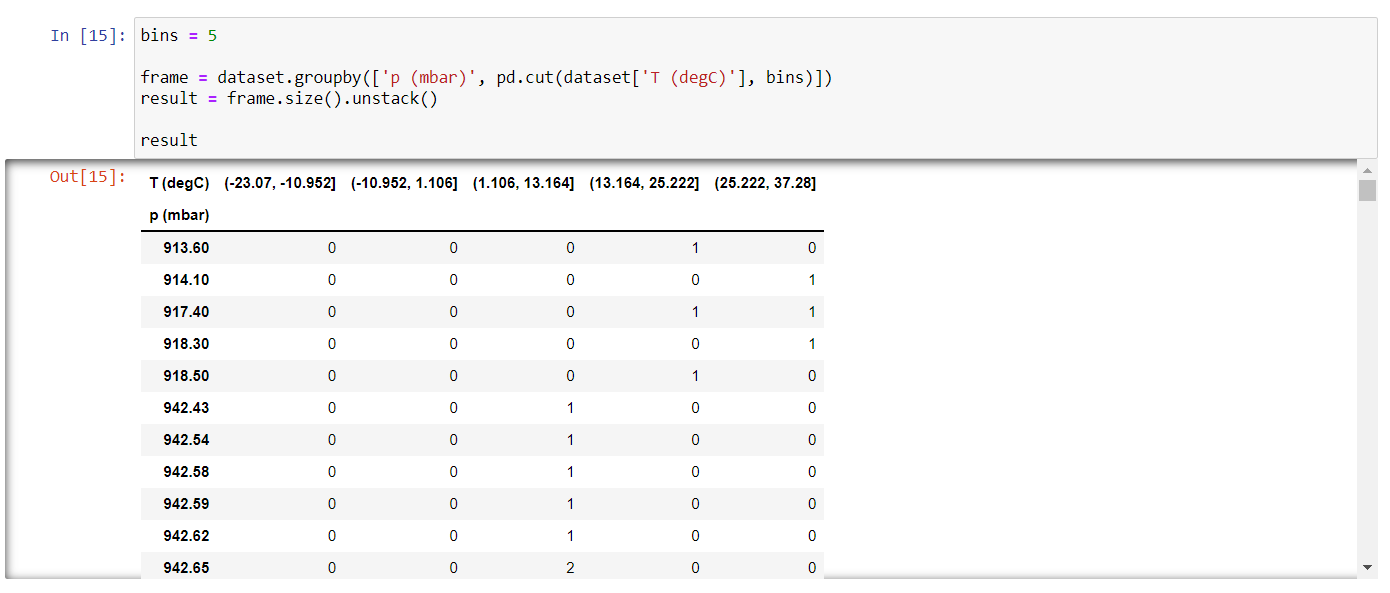
1. bins = 5

frame = dataset.groupby(['p (mbar)', pd.cut(dataset['T (degC)'], bins)])

result = frame.size().unstack()

result

OUTPUT



Question 7

**Consider a data frame containing data about students i.e. name, gender and passing division:**

1. **Perform one hot encoding of the last two columns of categorical data using the get\_dummies() function.**
2. **Sort this data frame on the “Birth Month” column (i.e. January to December). Hint: Convert Month to Categorical.**

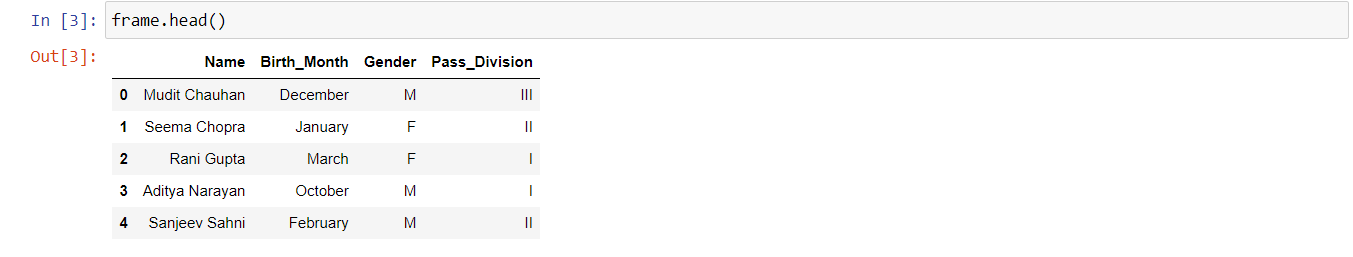
import pandas as pd

import numpy as np

frame = pd.read\_excel('students.xlsx')

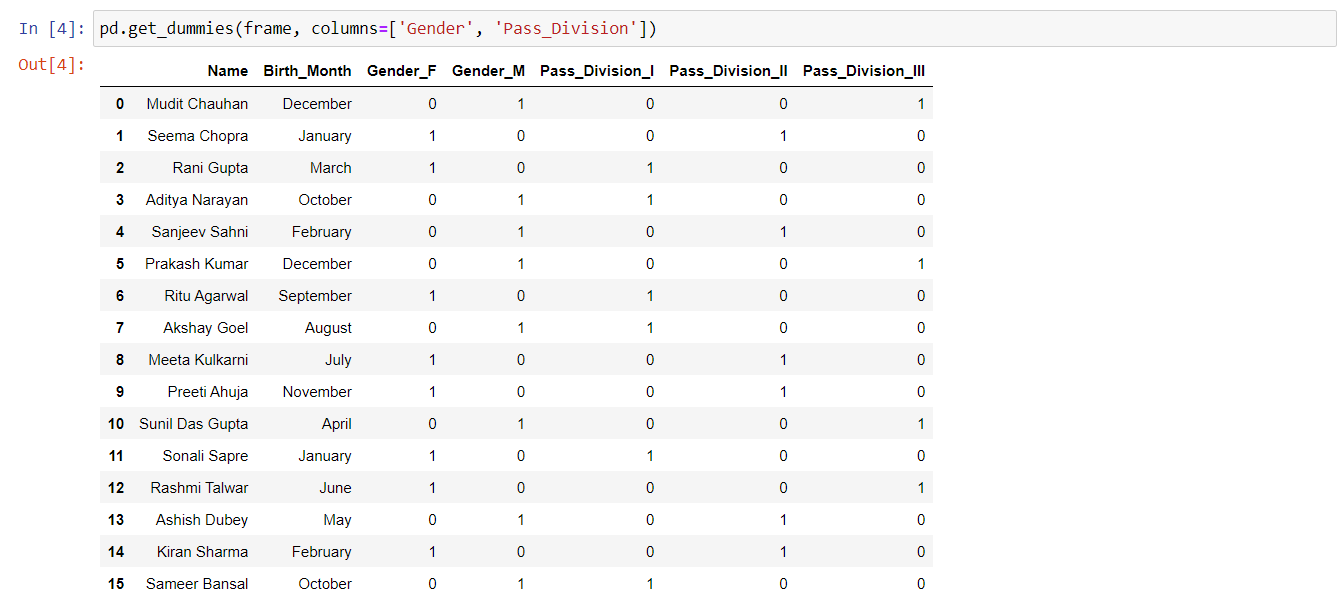
frame.head()

OUTPUT



1. pd.get\_dummies(frame, columns=['Gender', 'Pass\_Division'])

OUTPUT



1. frame['Birth\_Month'] = pd.Categorical(frame['Birth\_Month'],

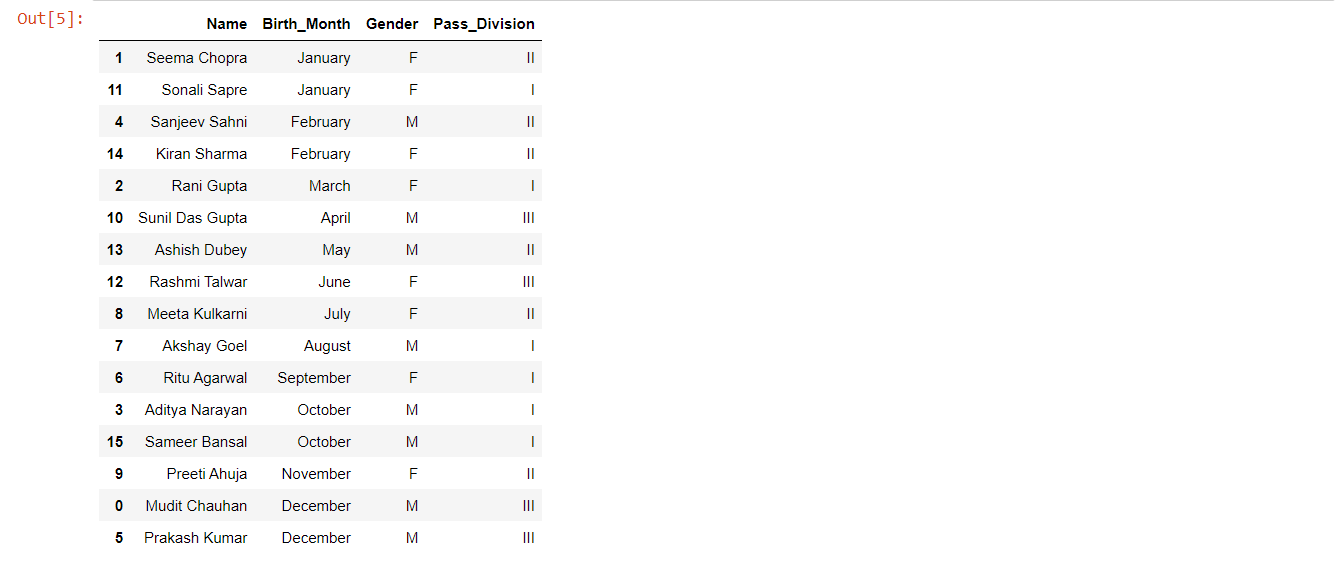
categories=['December', 'November', 'October', 'September', 'August', 'July', 'June', 'May', 'April', 'March', 'February', 'January'],

ordered=True)

frame = frame.sort\_values('Birth\_Month', ascending=False)

frame

OUTPUT



Question 8

**Consider the following data frame containing a family name, gender of the family member and her/his monthly income in each record.**

**Write a program in Python using Pandas to perform the following:**

1. **Calculate and display familywise gross monthly income.**
2. **Calculate and display the member with the highest monthly income in a family.**
3. **Calculate and display monthly income of all members with income greater than Rs. 60000.00.**
4. **Calculate and display the average monthly income of the female members in the Shah family.**

import numpy as np

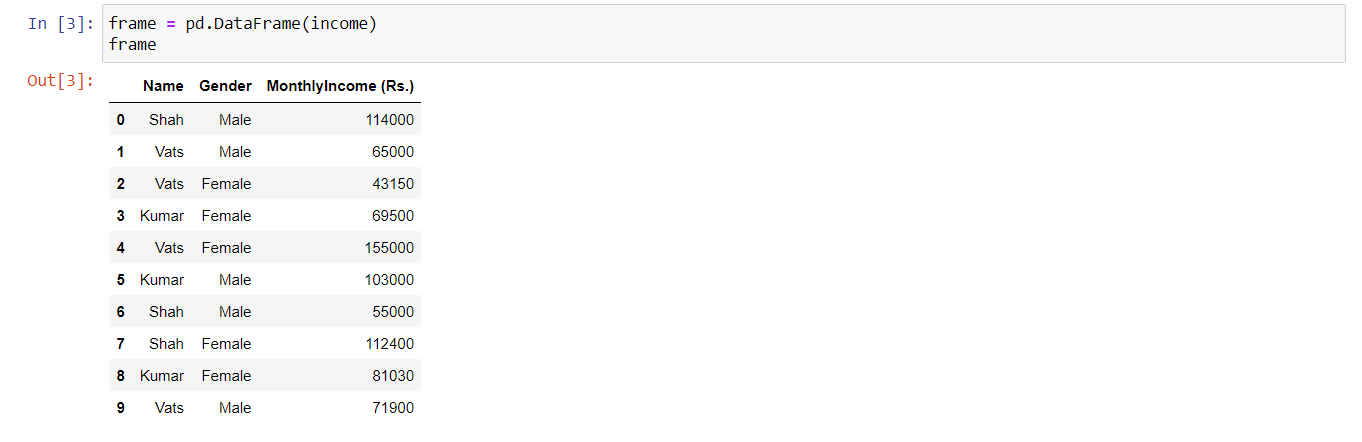
import pandas as pd

income = pd.read\_excel('Monthly-Income.xlsx')

frame = pd.DataFrame(income)

frame

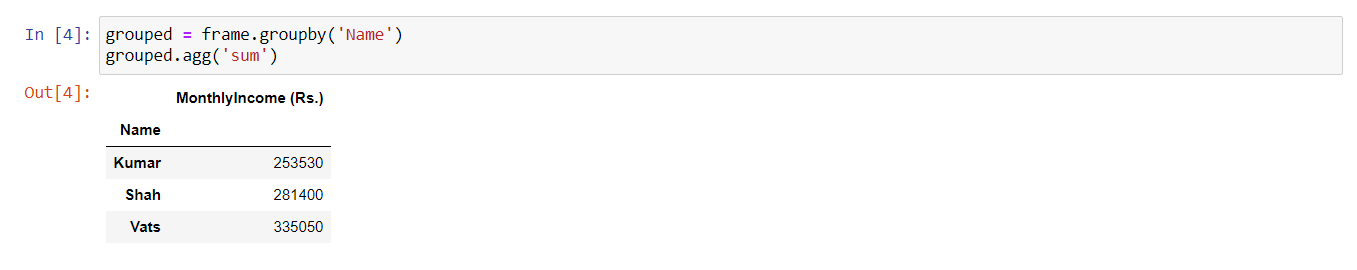
OUTPUT



1. grouped = frame.groupby('Name')

grouped.agg('sum')

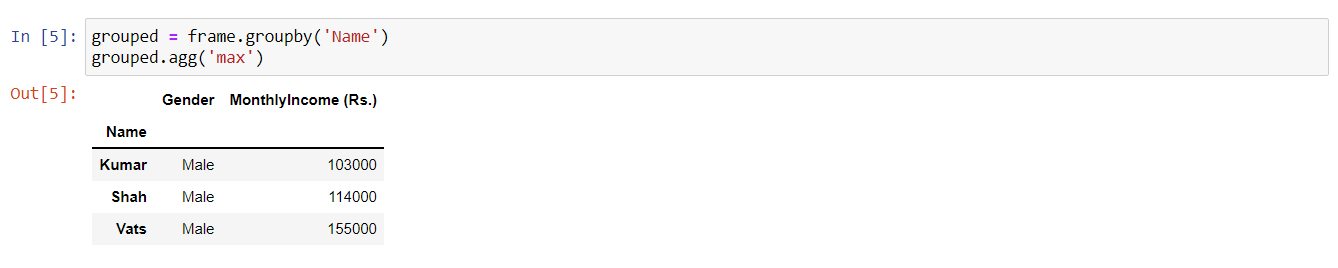
OUTPUT



1. grouped = frame.groupby('Name')

grouped.agg('max')

OUTPUT



1. frame[frame['MonthlyIncome (Rs.)'] > 60000.00]

OUTPUT



1. frame[(frame['Name'] == 'Shah') & (frame['Gender'] == 'Female')].mean()

OUTPUT

