**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]**

**Solution :**

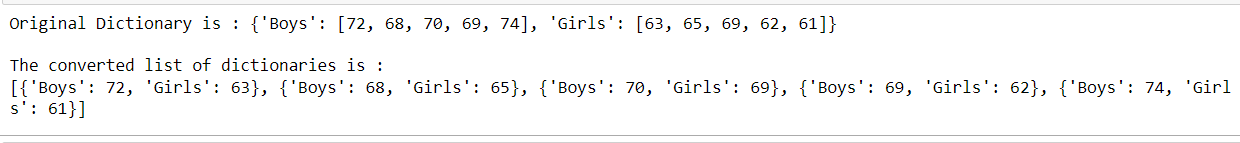
test\_dict = {'Boys':[72,68,70,69,74],'Girls':[63,65,69,62,61]}

print("Original Dictionary is : " + str(test\_dict))

res = [{key:value[i] for key,value in test\_dict.items()} for i in range(5)]

print("The converted list of dictionaries is : " + str(res))

**Output :**



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

**a. Compute the mean, standard deviation, and variance of a two dimensional random integer array along the second axis.**

**Solution :**

import numpy as np

from numpy import random

arr = np.random.randint(6, size=(3,4))

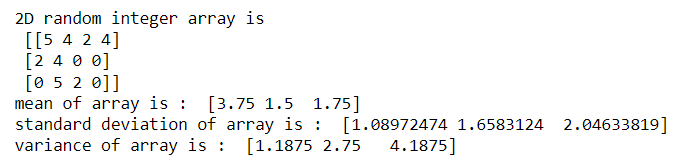
print("2D random integer array is \n", arr)

print("mean of array is : ", arr.mean(axis= 1) )

print("standard deviation of array is : ", arr.std(axis= 1) )

print("variance of array is : ", arr.var(axis= 1) )

**Output :**



**b. Get the indices of the sorted elements of a given array. a. B = [56, 48, 22, 41, 78, 91, 24, 46, 8, 33]**

**Solution :**

import numpy as np

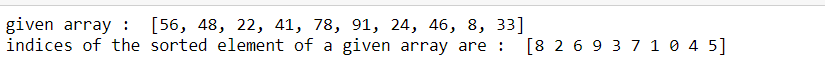
B = [56, 48, 22, 41, 78, 91, 24, 46, 8, 33]

print("given array : ", B)

indices = np.argsort(B)

print("indices of the sorted element of a given array are : ", indices)

**Output :**



**c. 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 nx m array, n and m are user inputs given at the run time.**

**Solution :**

import numpy as np

from numpy import random

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

print("2D array of dimension 3x4 : \n",arr)

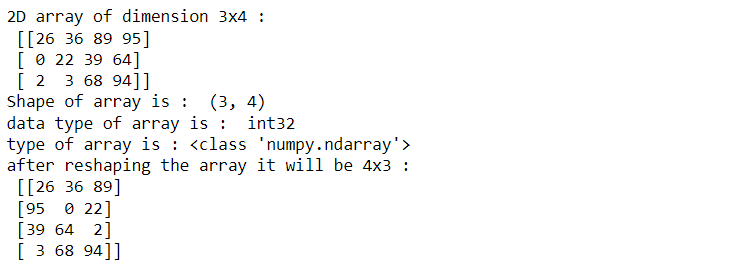
print("Shape of array is : ", np.shape(arr))

print("data type of array is : ", arr.dtype)

print("type of array is :", type(arr))

print("after reshaping the array it will be 4x3:\n", arr.reshape(4,3))

**Output :**

****

**d. Test whether the elements of a given array are zero, non-zero and NaN. Record the indices of these elements in three separate arrays.**

**Solution :**

import numpy as np

arr = np.array([1, 0, 2, 0, 3, np.nan, 0, 5 ,np.nan])

print("original array is : ", arr)

res = np.where(arr== 0)[0]

array1 = np.array(res)

print("indices of zero elements in given array : ", array1)

res1 = np.where(arr !=0)[0]

array2 = np.array(res1)

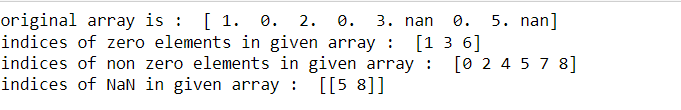
print("indices of non zero elements in given array : ",array2)

res2 = np.where(np.isnan(arr))

array3 = np.array(res2)

print("indices of NaN in given array : ",array3)

**Output**



**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:**

**Solution**

import pandas as pd

import numpy as np

from numpy import random

df = pd.DataFrame(np.random.randint(0,100,size=(50, 3)),columns=list('ABC'))

ncols = len(df.columns)

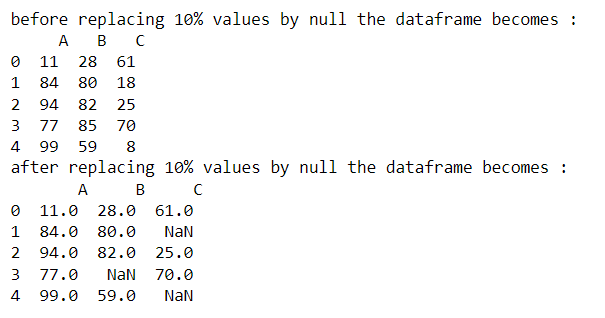
print("before replacing 10% values by null the dataframe becomes :\n", df.head())

while df.isnull().sum().sum() != (ncols\*50//10):

df.iloc[random.randint(50),random.randint(ncols)] = None

print("after replacing 10% values by null the dataframe becomes :\n", df.head())

**Output :**

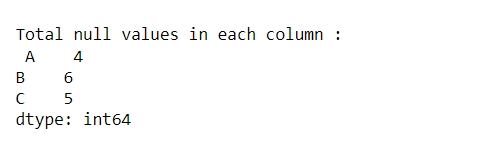


1. **Identify and count missing values in a dataframe.**

**Solution:**

print(" \nTotal null values in each column :\n",df.isnull().sum())

**Output :**



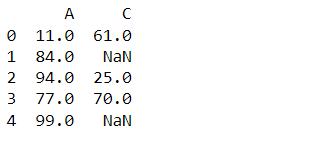
**b) Drop the column having more than 5 null values**.

**Solution :**

df1 = df.dropna(thresh=45,axis=1 )

print(df1)

**Output :**



1. **Identify the row label having maximum of the sum of all values in a row and drop that row.**

**Solution :**

sum=df.sum(axis=1)

print("Sum of rows :\n",sum.head())

print("\nMaximum Sum is :",sum.max())

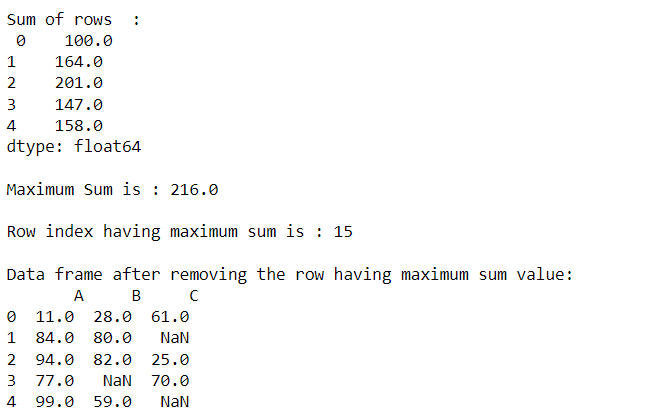
max\_sum\_row = df.sum(axis=1).idxmax()

print("\nRow index having maximum sum is :" ,max\_sum\_row)

df = df.drop(max\_sum\_row ,axis =0)

print("\nData frame after removing the row having maximum sum value: \n",df.head())

**Output :**



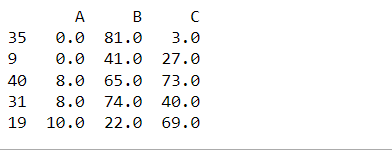
1. **Sort the dataframe on the basis of the first column.**

**Solution:**

sorted\_df = df.sort\_values(by='A')

print(sorted\_df.head())

**Output :**



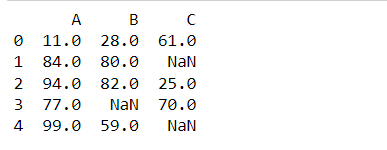
1. **Remove all duplicates from the first column.**

**Solution :**

df.drop\_duplicates(subset="A")

print(df.head())

**Output :**



1. **Find the correlation between first and second column and covariance between second and third column.**

**Solution :**

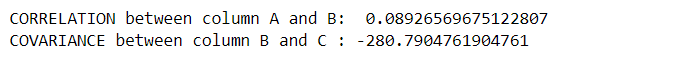
correlation **=** df['A']**.**corr(df['B'])

print("CORRELATION between column A and B: ", correlation)

covariance **=** df['B']**.**cov(df['C'])

print("COVARIANCE between column B and C :",covariance)

**Output :**



1. **Detect the outliers and remove the rows having outliers.**

**Solution:**

from scipy import stats

import numpy as np

z = np.abs(stats.zscore(df['A']))

print(z)

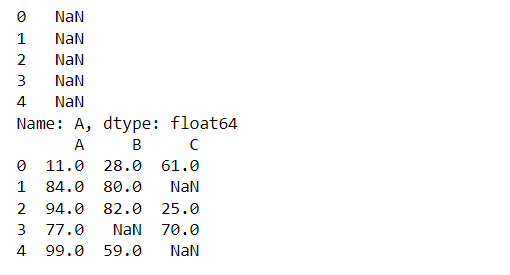
threshold = 1

outlier\_position = np.where(z > 1)

df.drop(outlier\_position[0])

print(df)

**Output :**



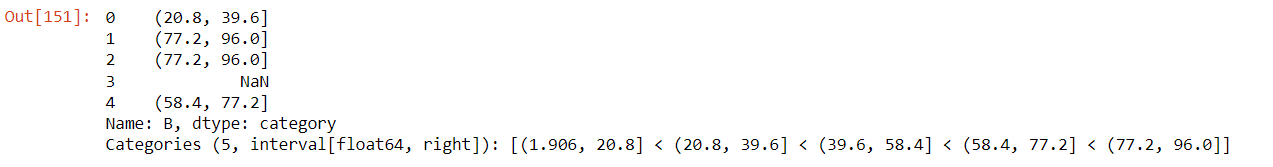
1. **Discretize second column and create 5 bins**

**Solution :**

df1 **=** pd**.**cut(df['B'],bins**=**5)**.**head()

df1

**Output :**



**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:**

**Solution :**

import numpy as np

import pandas as pd

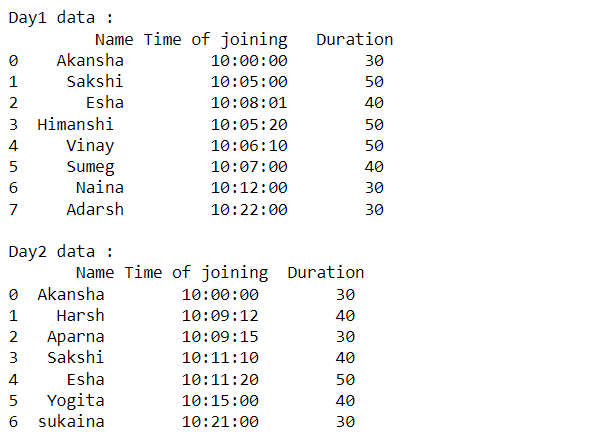
d1\_df = pd.read\_excel(r'C:\Users\MOHD SHOHEL\Downloads\day1.xlsx')

d2\_df = pd.read\_excel(r'C:\Users\MOHD SHOHEL\Downloads\day2.xlsx')

print("Day1 data : \n",d1\_df)

print("\nDay2 data : \n",d2\_df)

**Output :**

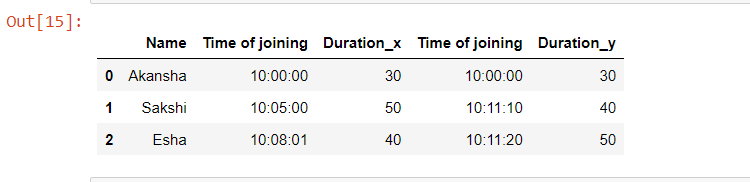


1. **Perform merging of the two dataframes to find the names of students who had attended the workshop on both days.**

**Solution :**

pd.merge(d1\_df,d2\_df,how='inner',on='Name')

**Output :**



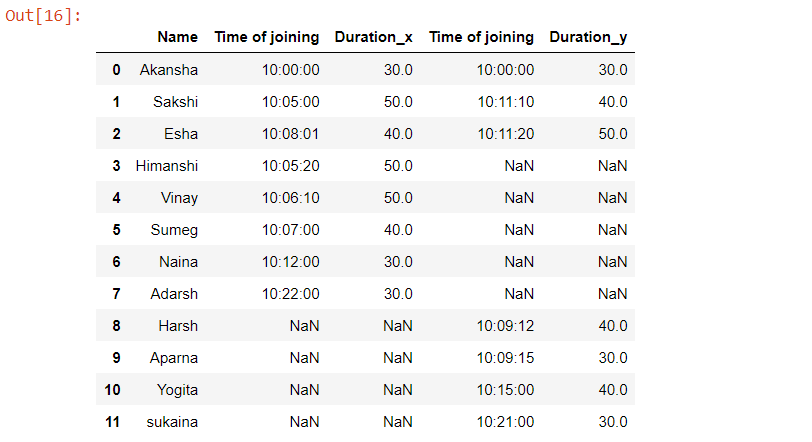
1. **Find names of all students who have attended workshop on either of the days.**

**Solution :**

either\_day = pd.merge(d1\_df,d2\_df,how='outer',on='Name')

either\_day

**Output :**

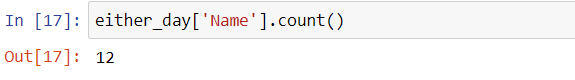


1. **Merge two data frames row-wise and find the total number of records in the data frame.**

**Solution :**

either\_day['Name'].count()

**Output :**



1. **Merge two data frames and use two columns names and duration as multi-row indexes. Generate descriptive statistics for this multi-index.**

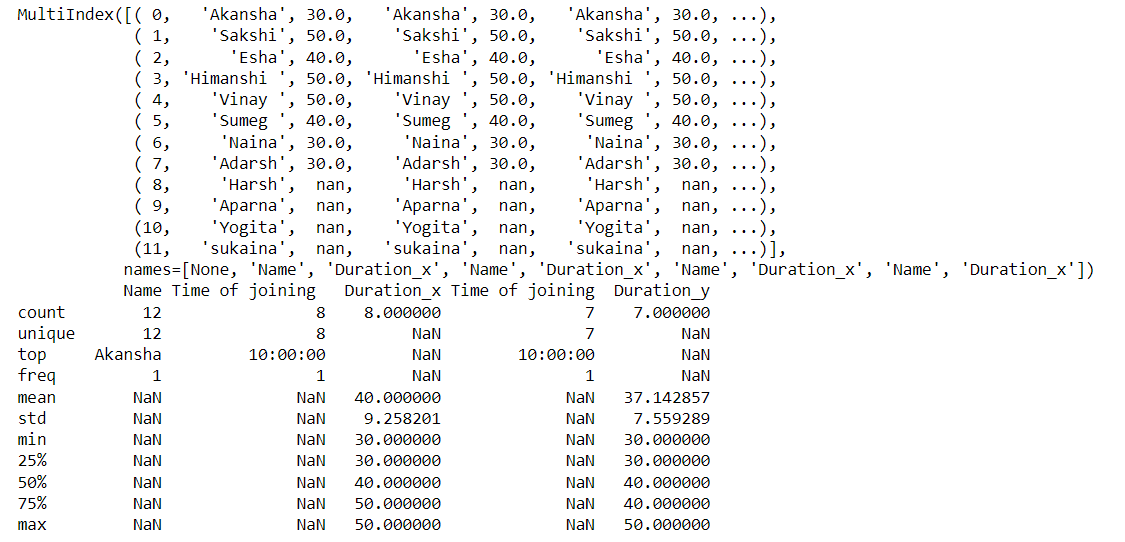
**Solution :**

either\_day.set\_index(['Name','Duration\_x'],inplace = True,append = True,drop = False)

print(either\_day.index)

print(either\_day.describe(include='all')) #descriptive statistic of multi index multi index

**Output :**



**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)**

**Solution :**

import pandas as pd

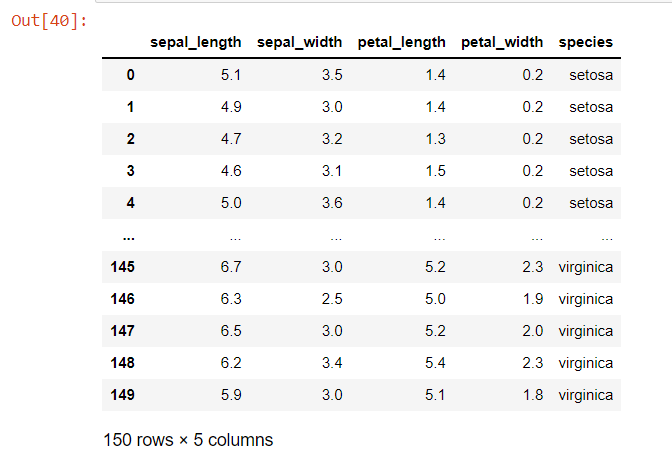
import matplotlib.pyplot as plt

import seaborn as sns

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

iris

**Output :**



1. **Plot bar chart to show the frequency of each class label in the data.**

**Solution :**

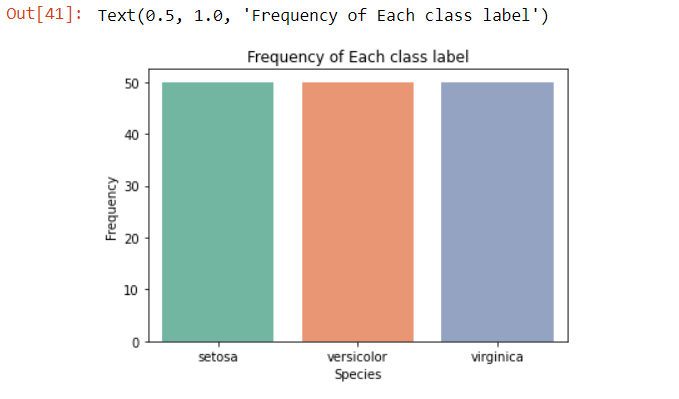
sns.countplot(x='species',data=iris,palette='Set2')

plt.xlabel('Species')

plt.ylabel('Frequency')

plt.title('Frequency of Each class label')

**Output :**



1. **Draw a scatter plot for Petal width vs sepal width.**

**Solution :**

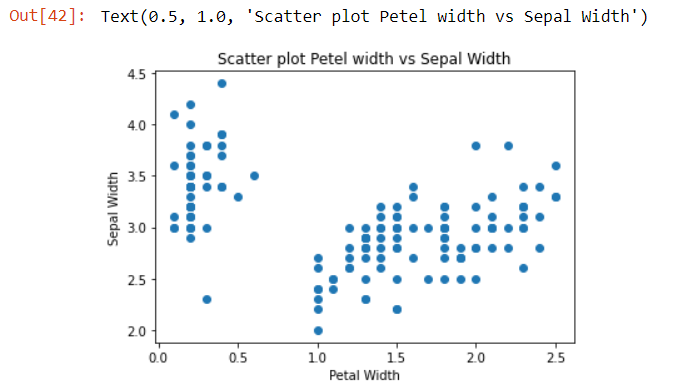
plt.scatter(x='petal\_width',y='sepal\_width',data=iris)

plt.xlabel('Petal Width')

plt.ylabel('Sepal Width')

plt.title("Scatter plot Petel width vs Sepal Width")

**Output :**

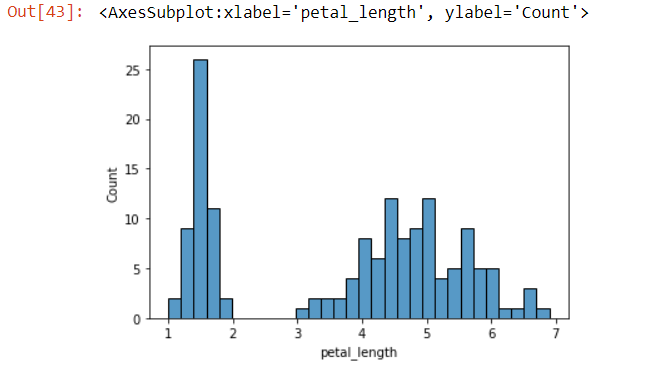


1. **Plot density distribution for feature petal length.**

**Solution :**

sns.histplot(iris['petal\_length'],kde=False,bins=30)

**Output :**

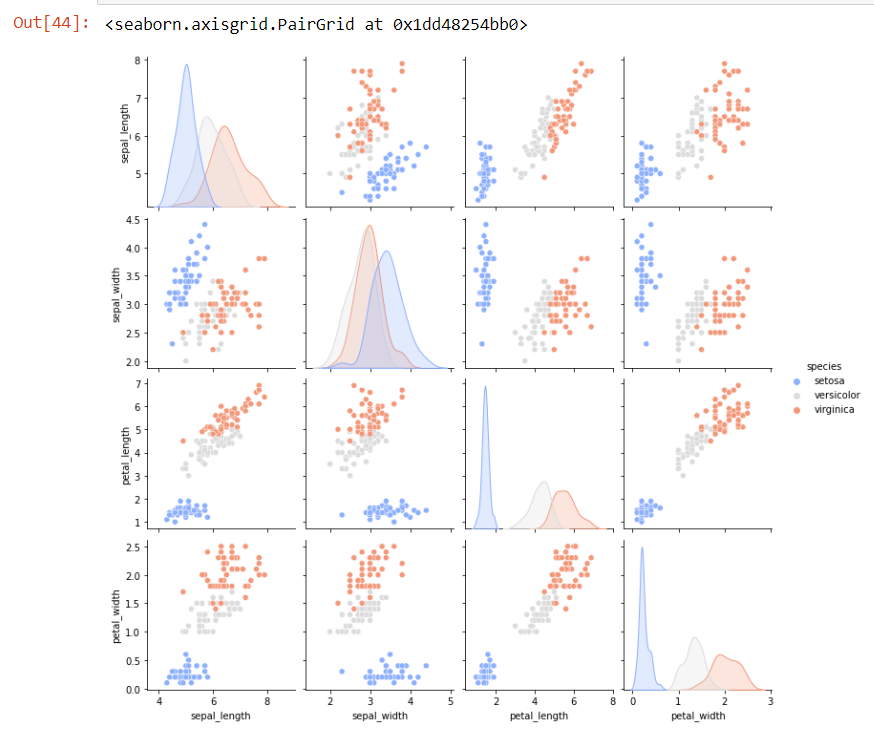


1. **Use a pair plot to show pairwise bivariate distribution in the Iris Dataset.**

**Solution :**

sns.pairplot(iris,hue='species',palette='coolwarm')

**Output :**



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

- > I take retail sales forecasting dataset from Kaggle [Retail Sales Forecasting | Kaggle](https://www.kaggle.com/datasets/tevecsystems/retail-sales-forecasting) …

**Solution :**

import pandas as pd

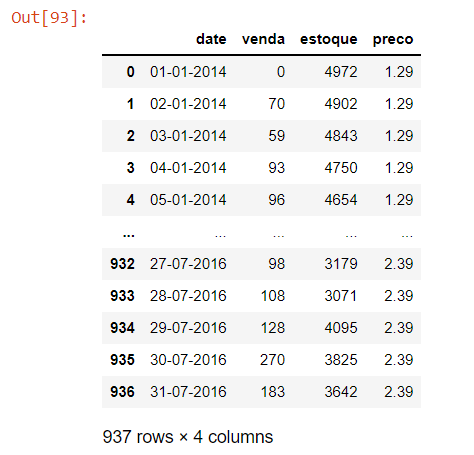
import numpy as np

data = pd.read\_csv(r"C:\Users\MOHD SHOHEL\Downloads\mock\_kaggle.csv")

df = pd.DataFrame(data)

df

**Output :**



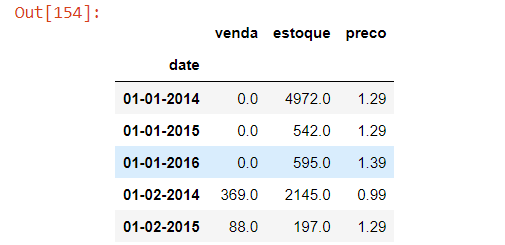
1. **Compute mean of a series grouped by another series**

**Solution :**

mean\_series = df.groupby(['date']).mean()

mean\_series.head()

**Output :**



1. **Fill an intermittent time series to replace all missing dates with values of previous non-missing date.**

**Solution :**

df = df.set\_index('date')

# to\_datetime() method converts string

# format to a DateTime object

df.index = pd.to\_datetime(df.index)

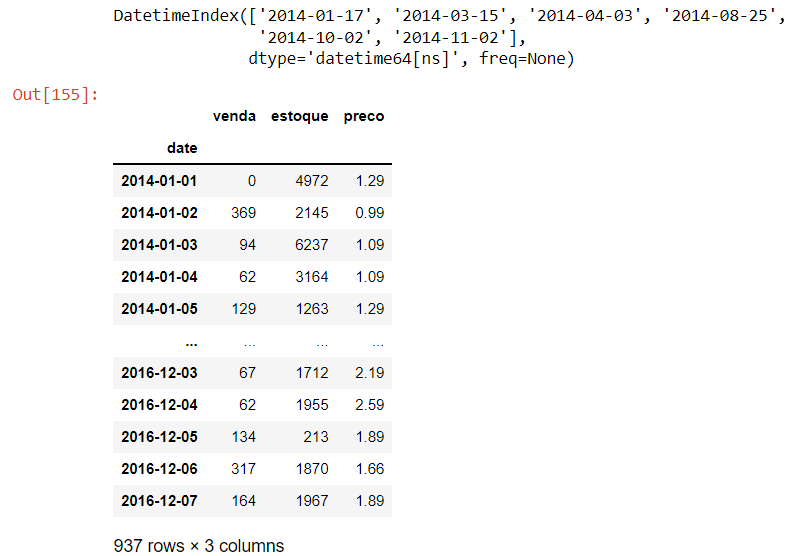
# dates which are not in the sequence

# are returned

print(pd.date\_range (start="2014-1-1", end="2015-12-31").difference(df.index))

df.sort\_values(['date','venda','estoque','preco']).groupby('date').ffill()

**Output:**



1. **Perform appropriate year-month string to dates conversion**.

**Solution:**

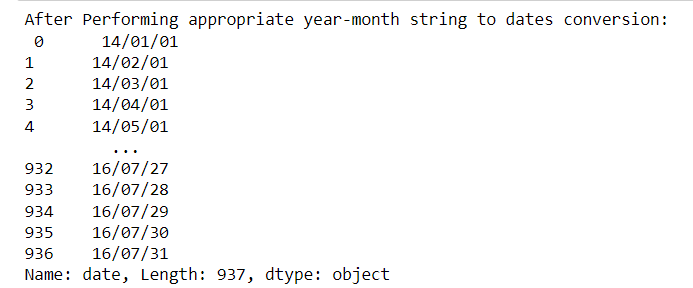
from datetime import datetime

df['date'] = pd.to\_datetime(df['date'])

change\_format = df['date'].dt.strftime('%y/%m/%d')

print("After Performing appropriate year-month string to dates conversion: \n",change\_format)

**Output:**



1. **Split a dataset to group by two columns and then sort the aggregated results within the groups.**

**Solution :**

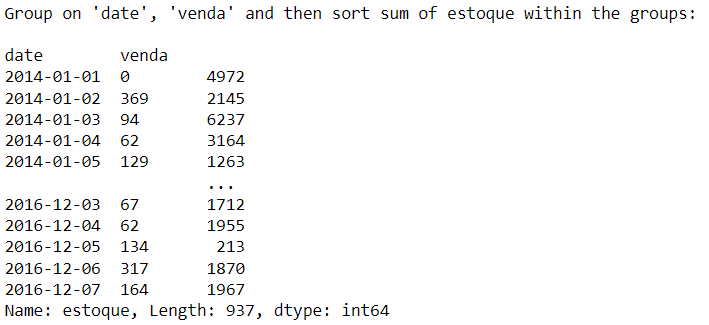
df\_agg = df.groupby(['date','venda']).agg({'estoque':sum})

result = df\_agg['estoque'].groupby(level=0, group\_keys=False)

print("\nGroup on 'date', 'venda' and then sort sum of estoque within the groups:\n")

print(result.nlargest())

**Output :**



1. **Split a given dataframe into groups with bin counts.**

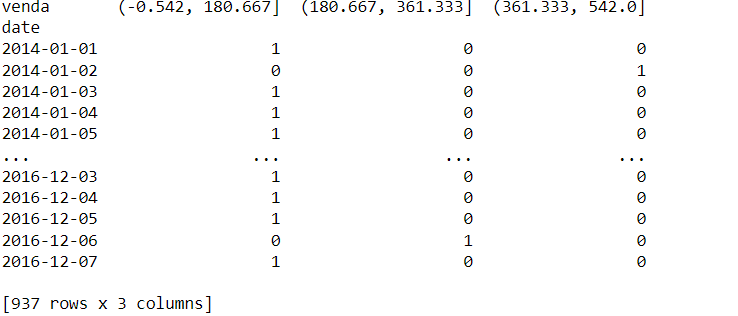
**Solution:**

groups = df.groupby(['date', pd.cut(df.venda, 3)])

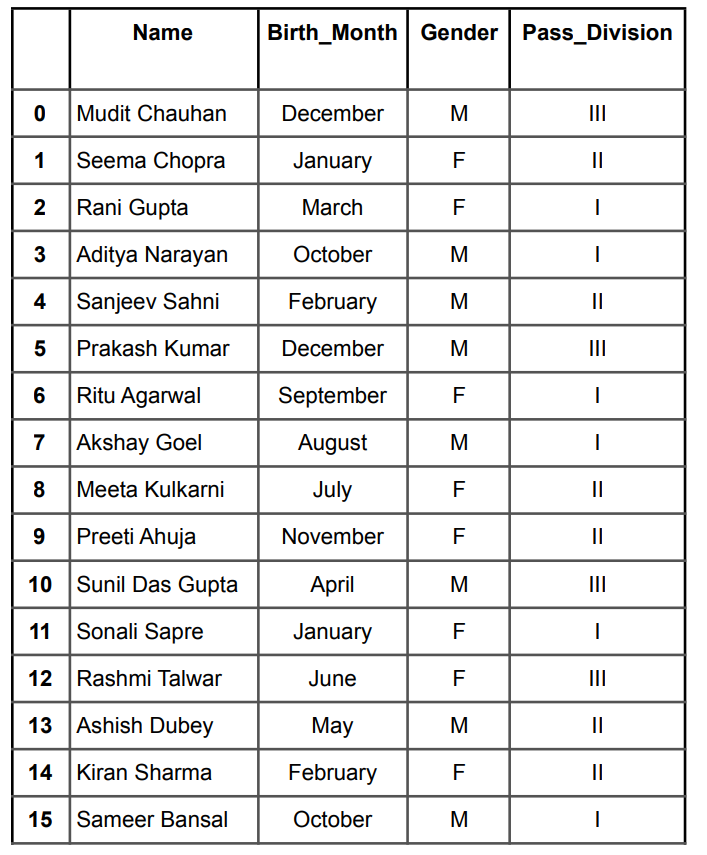
result = groups.size().unstack()

print(result)

**Output:**



**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.**

**Solution :**

import pandas as pd

#categorical data

categorical\_cols = ['Gender','pass\_division']

df = pd.get\_dummies(data, columns = categorical\_cols)

1. **Sort this data frame on the “Birth Month” column (i.e. January to December). Hint: Convert Month to Categorical.**

**Solution:**

import pandas as pd

dates\_in\_order = pd.date\_range(start='2022-01-01',end='2022-12-01',freq='MS')

months\_in\_order= dates\_in\_order.map(lambda x : x.month\_name()).to\_list()

df['month'] = pd.Categorical(df['month'],

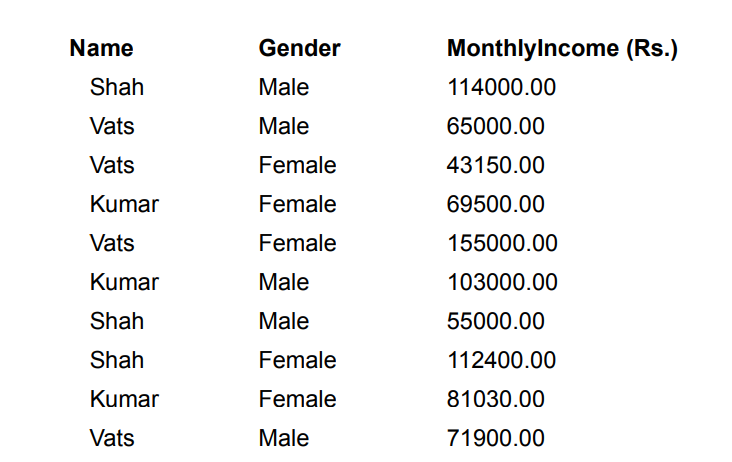
categories=months\_in\_order,

ordered=True)

df.sort\_values('month')

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

**Solution :**



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

1. **Calculate and display familywise gross monthly income**

**Solution :**

import pandas as pd

df = pd.DataFrame({

'Name': ['Shah','Vats','Vats','Kumar','Vats','Kumar','Shah','Shah','Kumar','Shah'],

'Gender': ['Male','Male' ,'Female','Female','Female','Male','Male','Female','Fem ale','Male'],

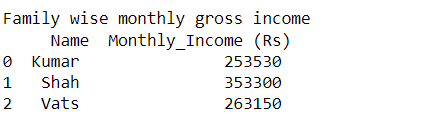
'Monthly\_Income ': [114000,65000,43150,69500,155000,103000,55000,112400,81030,71900]})

df

gross\_salary = df.groupby(by=['Name'], as\_index=False)['Monthly\_Income (Rs)'].sum()

print (gross\_salary)

**Output :**



1. **Calculate and display the member with the highest monthly income in a family.**

**Solution :**

max\_salary = data.groupby(by=['Name','Gender']).apply(lambda x : x[x['MonthlyIncome'] == x['MonthlyIncome'].max()])

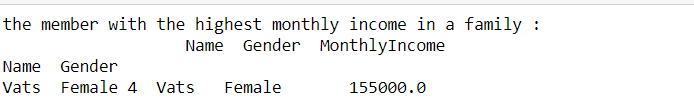
max\_salary

s = max(max\_salary['MonthlyIncome'])

res = max\_salary[max\_salary['MonthlyIncome'] == s ]

print("the member with the highest monthly income in a family :\n ",res)

**Output :**



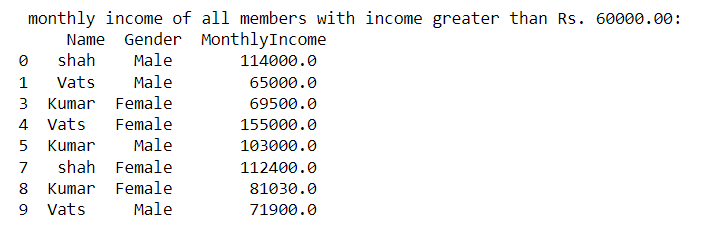
1. **Calculate and display monthly income of all members with income greater than Rs. 60000.00.**

**Solution :**

greater\_income = data[data['MonthlyIncome'] > 60000.00]

print(" monthly income of all members with income greater than Rs. 60000.00: \n",greater\_income)

**Output:**



1. **Calculate and display the average monthly income of the female members in the Shah family.**

**Solution :**

average = data[(data['Name']== 'shah') & (data['Gender']=='Female')].mean()

print("average monthly income of the female members in the Shah family: \n ",average)

**Output :**

