Deen Dayal Upadhyaya College
University of Delhi

Practical File Programs with Outputs
"***DATA ANALYSIS AND VISUALISATION"***

Bachelor of Science Computer Science Honors
Semester 5

Submitted To:
Mrs. Arpita Sharma

Submitted By:
Bhavana Kashyap
21HCS4131

Q1. 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]

Answer:-

```
def list_of_dict(heights):
    keys=heights.keys()
    # print(keys)
    values = zip(*[heights[k] for k in keys])
    # print(values)
    result = [dict(zip(keys,v )) for v in values]
    return result

heights = {'Boys':[72,68,70,69,74], 'Girls':[63,65,69,62,61]}
print("\n ORIGINAL DICTIONARY OF LISTS :" , heights)
print("\n NOW LIST OF DICTIONARIES : \n",list_of_dict(heights))
```

OUTPUT :

ORIGINAL DICTIONARY OF LISTS : {'Boys': [72, 68, 70, 69, 74], 'Girls': [63, 65, 69, 62, 61]}

NOW LIST OF DICTIONARIES :
[{'Boys': 72, 'Girls': 63}, {'Boys': 68, 'Girls': 65}, {'Boys': 70, 'Girls': 69}, {'Boys': 69, 'Girls': 62}, {'Boys': 74, 'Girls': 61}]

Q2. 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.

b. Get the indices of the sorted elements of a given array.

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

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.

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.

Answer:- (a)

```
import numpy as np
arr = np.random.randint(1,50,(4,6))
arr

#along the second axis
#Mean
print('Mean of the array: ',arr.mean(axis=1))
#standard deviation
print('Standard Deviation of the array: ',arr.std(axis=1))
#variance
print('Variance of the array: ',arr.var(axis=1))
```

(b)

```
B = [56, 48, 22, 41, 78, 91, 24, 46, 8, 33]
arr1 = np.array(B)
#arr1
print("Sorted array: ",np.sort(arr1))
print("Indices of the sorted elements of a given array:
",np.argsort(arr1))
```

(c)

```
m = int(input('Enter the number of rows(m): '))
n = int(input('Enter the number of columns(n): '))
arr2 = np.random.randint(1,100,(m,n))
print(arr2)
print('Shape: ',arr2.shape)
print('Type: ',type(arr2))
print('Data Type: ',arr2.dtype)
arr2 = arr2.reshape(n,m)
print('After reshaping: \n',arr2)
print('New Shape: ',arr2.shape)
```

(d)

```
x = np.array([1, 0, 3, 4])
```

```
print("ORIGINAL ARRAY ::-> ",x)
print("\nTest if none of the elements of the said array is zero
::-> ", np.all(x))

res = np.where(x == 0)[0]
print("The index of the  zero elements is :: ",res)


x = np.array([1, 0, 0, 3, 2, 0])
print("\n")
print("\nORIGINAL ARRAY ::-> ",x)
print("\nTest whether any of the elements of a given array is
non-zero ::",np.any(x))
res = np.where(x != 0)[0]
print("The index of the non- zero elements is :: ",res)
x = np.array([0, 0, 0, 0])


a = np.array([1, 0, np.nan, 3, np.nan])
print("\n")
print("\nORIGINAL ARRAY ::-> ",a)
print("\nTest element-wise for NaN :: ",np.isnan(a))

res = np.where(np.isnan(a) == True)[0]
print("The index of the  zero elements is :: ",res)
```

OUTPUT :

```
array([[17, 20, 31, 12, 16, 10],
       [44, 22, 32, 42, 30,  6],
       [49, 46, 33,  6,  3, 14],
       [34, 39, 35, 17, 29, 20]])
```

(a)

```
Mean of the array:  [17.66666667 29.33333333 25.16666667 29.
]
Standard Deviation of the array:  [ 6.79869268 12.78888406 18.46
994556  8.02080628]
Variance of the array:  [ 46.22222222 163.55555556 341.13888889
64.33333333]
```

(b)

```
Sorted array:  [ 8 22 24 33 41 46 48 56 78 91]
Indices of the sorted elements of a given array:  [8 2 6 9 3 7 1
0 4 5]
```

(c)

```
[[ 6 77 89]
 [55 43 24]]
Shape:  (2, 3)
Type: <class 'numpy.ndarray'>
Data Type:  int32
After reshaping:
 [[ 6 77]
 [89 55]
 [43 24]]
New Shape:  (3, 2)
```

(d)

```
ORIGINAL ARRAY ::->  [1 0 3 4]

Test if none of the elements of the said array is zero ::-> Fal
se
The index of the  zero elements is ::  [1]


ORIGINAL ARRAY ::->  [1 0 0 3 2 0]

Test whether any of the elements of a given array is non-zero ::
True
The index of the non- zero elements is ::  [0 3 4]



ORIGINAL ARRAY ::->  [ 1.  0. nan  3. nan]

Test element-wise for NaN ::  [False False  True False  True]
The index of the  zero elements is ::  [2 4]
```

Q3. 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:

- a. Identify and count missing values in a dataframe.
- b. Drop the column having more than 5 null values.
- c. Identify the row label having maximum of the sum of all values in a row and drop that row.

- d. Sort the dataframe on the basis of the first column.
- e. Remove all duplicates from the first column.
- f. Find the correlation between first and second column and covariance between second and third column.
- g. Detect the outliers and remove the rows having outliers.
- h. Discretize second column and create 5 bins

Answer:-

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randint(0,100,size=(50,3)),
columns=list('123'))
df.head()

for c in df.sample(int(df.shape[0]*df.shape[1]*0.10)).index:
    df.loc[c,str(np.random.randint(1,4))]=np.nan
df

(a)
print(df.isnull().sum().sum())

(b)
for col in df.columns:
    print(col,df[col].isnull().sum())
df.dropna(axis = 1,thresh=(df.shape[0]-5)).head()

(c)
sum=df.sum(axis=1)
print("SUM IS :\n",sum)
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")
df

(d)
sortdf=df.sort_values('1')
sortdf.head()

(e)
df =df.drop_duplicates(subset='1',keep = "first")
print(df)
```

(f)

```
correlation = df['1'].corr(df['2'])
print("CORRELATION between column 1 and 2 : ", correlation)
covariance = df['2'].cov(df['3'])
print("COVARIANCE between column 2 and 3 :",covariance)
```

(g)

```
df.plot.box()
```

(h)

```
df1 =  pd.cut(df['2'],bins=5).head()
df1
```

OUTPUT :

Out[1]:

	1	2	3
0	76	60	5
1	3	11	20
2	13	9	38
3	70	92	26
4	22	92	75

(a)

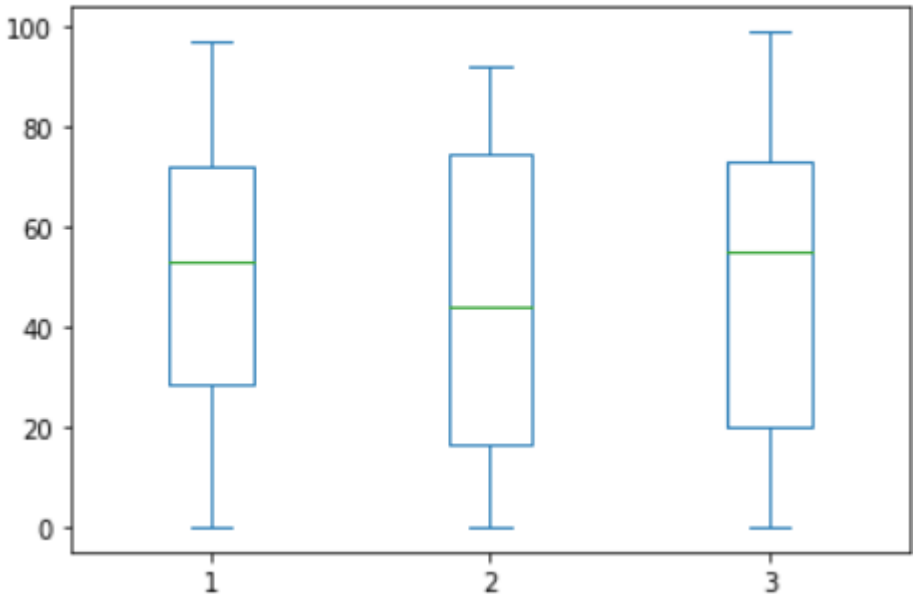
Out[2]:

	1	2	3
0	76.0	60.0	5.0
1	3.0	11.0	20.0
2	13.0	9.0	38.0
3	70.0	92.0	26.0
4	22.0	92.0	75.0
5	51.0	44.0	59.0
6	53.0	31.0	NaN
7	43.0	NaN	1.0
8	97.0	2.0	79.0
9	NaN	0.0	6.0
10	12.0	90.0	91.0
11	69.0	63.0	62.0
12	56.0	37.0	55.0
13	79.0	28.0	0.0
14	NaN	64.0	7.0
15	96.0	4.0	83.0
16	69.0	84.0	63.0
17	28.0	29.0	11.0
18	13.0	99.0	96.0
19	32.0	48.0	63.0
20	52.0	21.0	86.0
21	95.0	4.0	16.0
22	5.0	12.0	NaN
23	53.0	88.0	13.0
24	NaN	15.0	7.0
25	87.0	84.0	80.0
26	97.0	29.0	NaN
27	0.0	NaN	53.0
28	66.0	NaN	5.0
29	21.0	85.0	70.0
30	21.0	47.0	NaN
31	65.0	89.0	16.0
32	68.0	31.0	68.0
33	27.0	11.0	67.0
34	97.0	38.0	54.0
35	NaN	17.0	50.0
36	29.0	85.0	40.0
37	29.0	54.0	56.0
38	69.0	44.0	48.0
39	60.0	77.0	60.0
40	82.0	61.0	73.0
41	96.0	73.0	59.0
42	88.0	72.0	NaN
43	30.0	64.0	99.0
45	52.0	64.0	NaN
46	34.0	65.0	77.0

Out[4]:

	1	2
0	76.0	60.0
1	3.0	11.0
2	13.0	9.0
3	70.0	92.0
4	22.0	92.0

```
Out[9]: <AxesSubplot:>
```




```
Out[10]: 0      (55.2, 73.6]
         1    (-0.092, 18.4]
         2    (-0.092, 18.4]
         3      (73.6, 92.0]
         4      (73.6, 92.0]
         Name: 2, dtype: category
         Categories (5, interval[float64, right]): [(-0.092, 18.4] < (18.4, 36.8] < (36.8, 55.2] < (55.2, 73.6] < (73.6, 92.0]]
```

Q4. 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:

- a.Perform merging of the two dataframes to find the names of students who had attended the workshop on both days.
- b. Find names of all students who have attended workshop on either of the days.
- c. Merge two data frames row-wise and find the total number of records in the data frame.
- d. Merge two data frames and use two columns names and duration as multi-row indexes. Generate descriptive statistics for this multi-index.

Answer:-

```
import numpy as np
import pandas as pd
dfDay1 = pd.read_excel('Day1_anirbn.xlsx')
dfDay2 = pd.read_excel('Day2_anirbn.xlsx')
print(dfDay1.head(),"\n")
print(dfDay2.head())
```

(a)

```
pd.merge(dfDay1,dfDay2,how='inner',on='Name')
```

(b)

```
either_day = pd.merge(dfDay1,dfDay2,how='outer',on='Name')
either_day
```

(c)

```
either_day['Name'].count()
```

(d)

```
both_days =
pd.merge(dfDay1,dfDay2,how='outer',on=['Name','Duration']).copy(
) #  creates a copy of an existing list

both_days.fillna(value='-',inplace=True) # to fill out the
missing values in the given series object

both_days.set_index(['Name','Duration']) # a method to set a
List as index of a Data Frame
```

OUTPUT :

	Name	Time of Joining	Duration
0	Abhimanyu	11:00:00	40
1	Abhishek	11:04:00	30
2	Aasif	11:08:00	30
3	Aman	11:01:00	40
4	Anand	11:12:00	50

	Name	Time of Joining	Duration
0	Abhimanyu	11:00:00	40
1	Abhishek	11:06:00	30
2	Deepanshu	11:10:00	40
3	Aman	11:09:00	40
4	Anubhav	11:10:00	50

Out[4]:

	Name	Time of Joining_x	Duration_x	Time of Joining_y	Duration_y
0	Abhimanyu	11:00:00	40	11:00:00	40
1	Abhishek	11:04:00	30	11:06:00	30
2	Aman	11:01:00	40	11:09:00	40
3	Anubhav	11:10:00	30	11:10:00	50
4	Anurag	11:11:00	30	11:08:00	30
5	Arpit	11:07:00	40	11:08:00	40
6	Bhavana	11:15:00	30	11:14:00	30
7	Deepanshu	11:02:00	40	11:10:00	40
8	Ishant	11:03:00	30	11:00:00	30
9	Harshit	11:13:00	40	11:09:00	40

Out[5]:

	Name	Time of Joining_x	Duration_x	Time of Joining_y	Duration_y
0	Abhimanyu	11:00:00	40.0	11:00:00	40.0
1	Abhishek	11:04:00	30.0	11:06:00	30.0
2	Aasif	11:08:00	30.0	NaN	NaN
3	Aman	11:01:00	40.0	11:09:00	40.0
4	Anand	11:12:00	50.0	NaN	NaN
5	Anubhav	11:10:00	30.0	11:10:00	50.0
6	Anurag	11:11:00	30.0	11:08:00	30.0
7	Arpit	11:07:00	40.0	11:08:00	40.0
8	Akanksha	11:08:00	50.0	NaN	NaN
9	Bhavana	11:15:00	30.0	11:14:00	30.0
10	Deepanshu	11:02:00	40.0	11:10:00	40.0
11	Ishant	11:03:00	30.0	11:00:00	30.0
12	Gourav	11:19:00	30.0	NaN	NaN
13	Harshit	11:13:00	40.0	11:09:00	40.0
14	Kartikey	11:05:00	50.0	NaN	NaN
15	Bharat	NaN	NaN	11:12:00	30.0
16	Divyanshu	NaN	NaN	11:13:00	40.0
17	Deepak	NaN	NaN	11:02:00	50.0
18	Jayesh	NaN	NaN	11:08:00	30.0
19	Jeeva	NaN	NaN	11:06:00	30.0

Out[7]:

Name	Duration	Time of Joining_x	Time of Joining_y
Abhimanyu	**40**	11:00:00	11:00:00
Abhishek	**30**	11:04:00	11:06:00
Aasif	**30**	11:08:00	-
Aman	**40**	11:01:00	11:09:00
Anand	**50**	11:12:00	-
Anubhav	**30**	11:10:00	-
Anurag	**30**	11:11:00	11:08:00
Arpit	**40**	11:07:00	11:08:00
Akanksha	**50**	11:08:00	-
Bhavana	**30**	11:15:00	11:14:00
Deepanshu	**40**	11:02:00	11:10:00
Ishant	**30**	11:03:00	11:00:00
Gourav	**30**	11:19:00	-
Harshit	**40**	11:13:00	11:09:00
Kartikey	**50**	11:05:00	-
Anubhav	**50**	-	11:10:00
Bharat	**30**	-	11:12:00
Divyanshu	**40**	-	11:13:00
Deepak	**50**	-	11:02:00
Jayesh	**30**	-	11:08:00
Jeeva	**30**	-	11:06:00

Q5. 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)

- a. Plot bar chart to show the frequency of each class label in the data.
- b. Draw a scatter plot for Petal width vs sepal width.
- c. Plot density distribution for feature petal length.
- d. Use a pair plot to show pairwise bivariate distribution in the Iris Dataset.

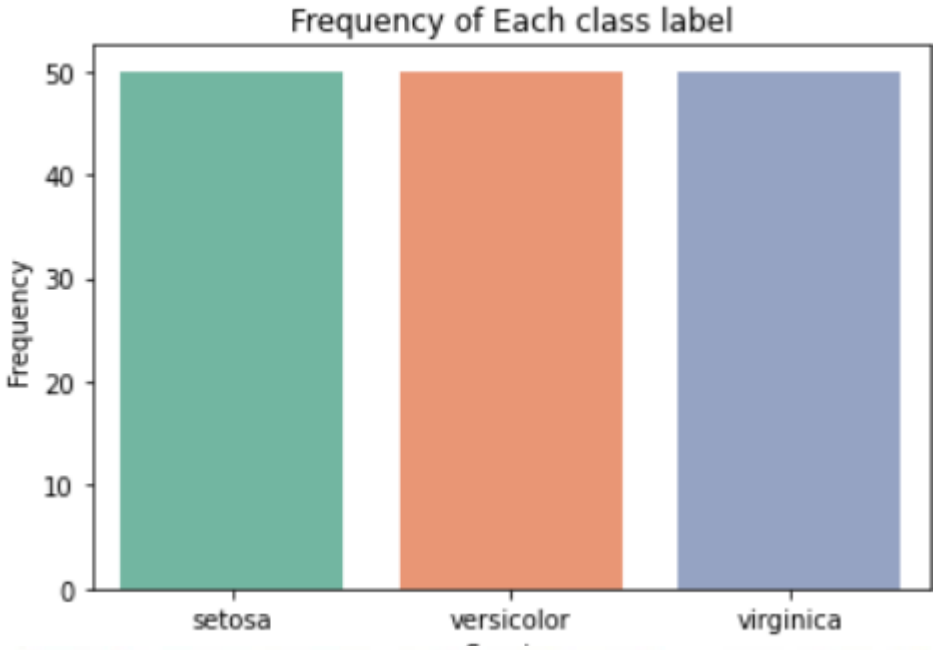
Answer:-

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
iris = sns.load_dataset('iris')

(a)
sns.countplot(x='species',data=iris,palette='Set2')
plt.xlabel('Species')
plt.ylabel('Frequency')
plt.title('Frequency of Each class label')

(b)
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")

(c)
sns.histplot(iris['petal_length'],kde=False,bins=30)

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

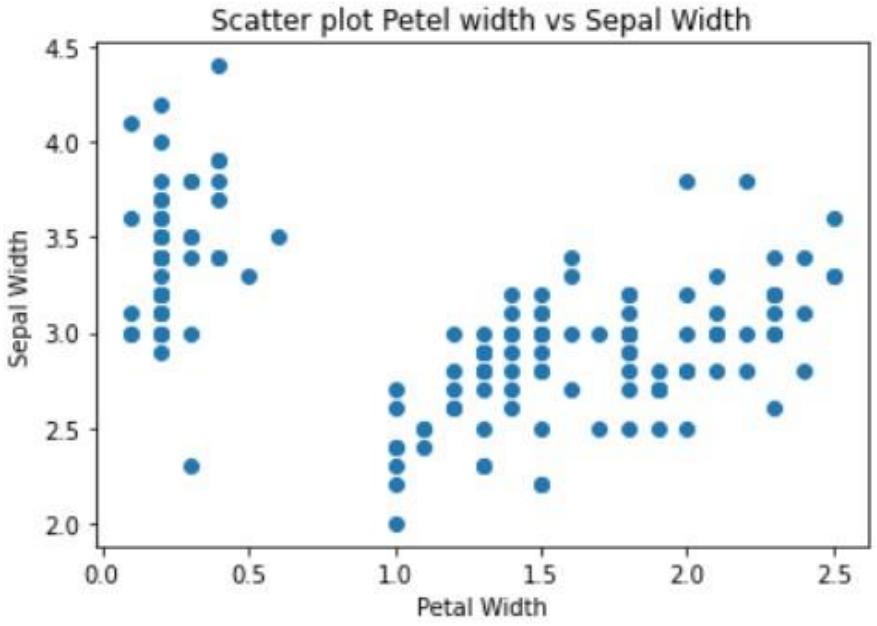
OUTPUT :

Next Page------→

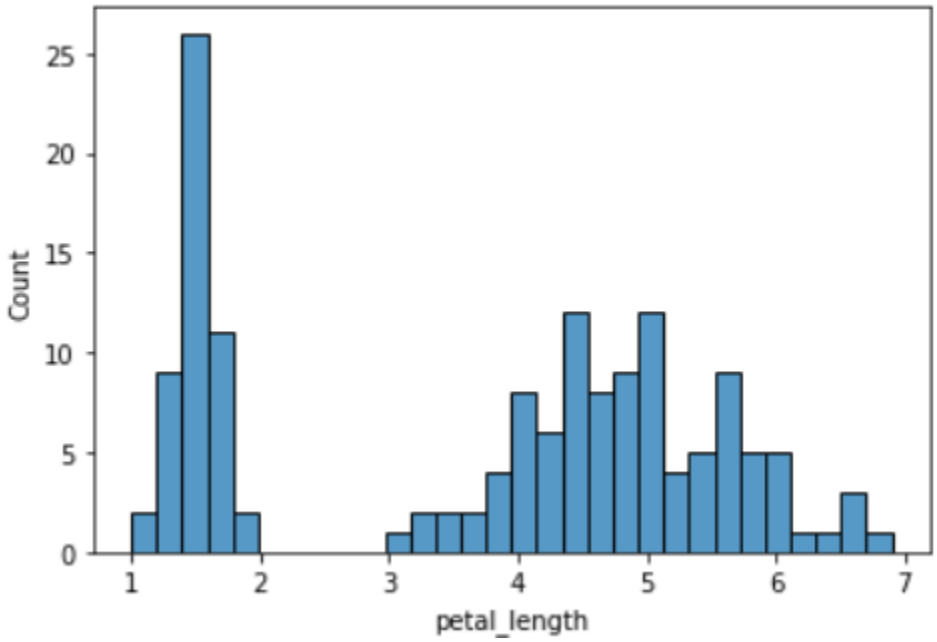
Out[3]: Text(0.5, 1.0, 'Frequency of Each class label')



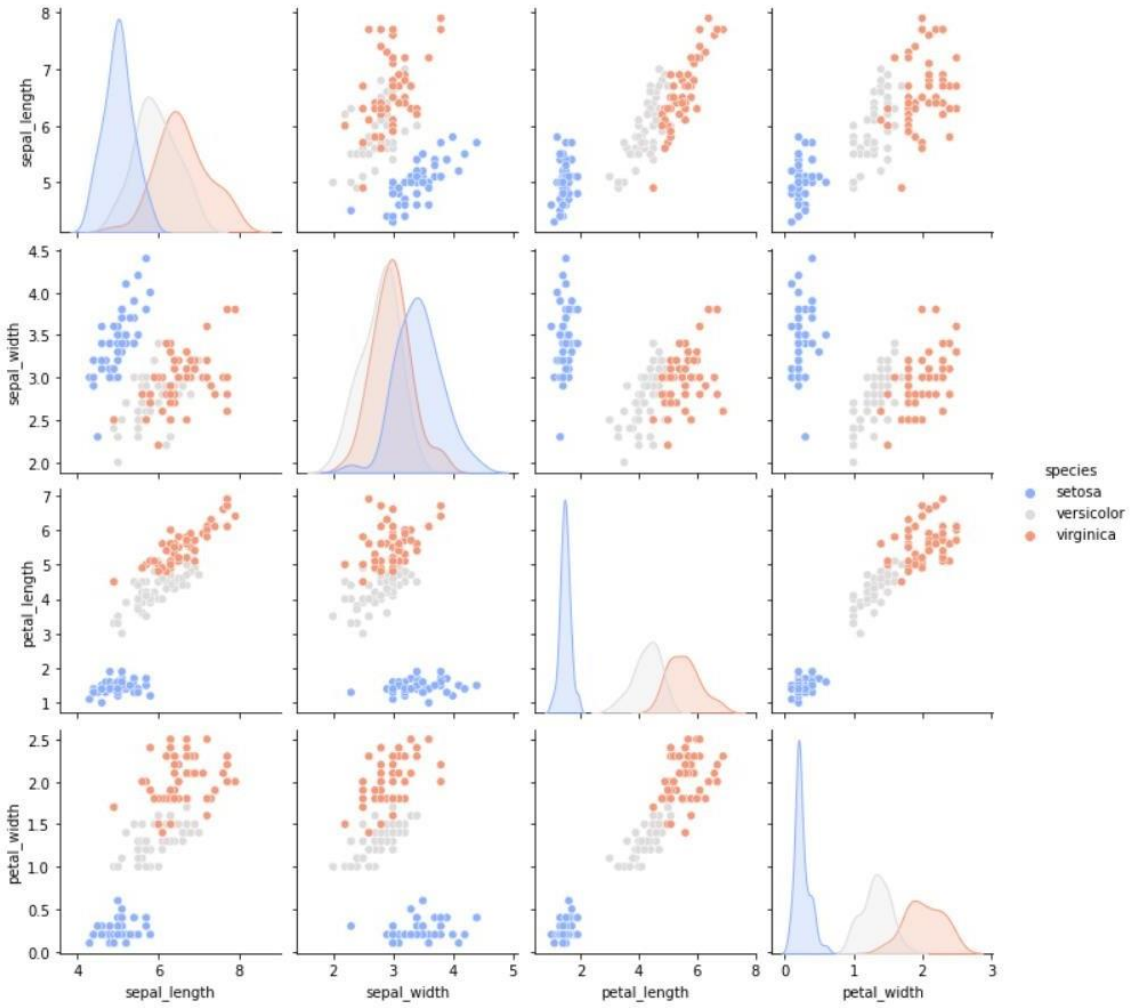
Out[4]: Text(0.5, 1.0, 'Scatter plot Petel width vs Sepal Width')



Out[6]: <AxesSubplot:xlabel='petal_length', ylabel='Count'>



Out[7]: <seaborn.axisgrid.PairGrid at 0x26728f1da08>



Q6. Consider any sales training/ weather forecasting dataset

a. Compute mean of a series grouped by another series

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

c. Perform appropriate year-month string to dates conversion.

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

e. Split a given dataframe into groups with bin counts.

Answer:-

```
import pandas as pd
import numpy as np

data = {
    'Date': pd.date_range(start='2022-01-01', end='2022-01-10'),
    'Sales': [100, 120, np.nan, 150, 200, 180, np.nan, 220, 250, 300],
    'Product': ['A', 'B', 'A', 'B', 'A', 'B', 'A', 'B', 'A', 'B']
}

df = pd.DataFrame(data)
print("Original Dataset:")
print(df)
print("\n")

# a. Compute mean of 'Sales' grouped by 'Product'
mean_sales = df.groupby('Product')['Sales'].mean()
print("Mean Sales Grouped by Product:")
print(mean_sales)
print("\n")

# b. Fill missing values in 'Sales' with the previous non-missing date
df['Date'] = pd.to_datetime(df['Date'])
df = df.set_index('Date')
df = df.resample('D').ffill()
print("Dataset after filling missing values:")
```

```
print(df)
print("\n")


# c. Perform year-month string to date conversion
df['YearMonth'] =
pd.to_datetime(df.index.to_period('M')).to_timestamp()
print("Dataset after year-month string to date conversion:")
print(df)
print("\n")


# d. Split a dataset to group by two columns and then sort the
aggregated results within the groups
sorted_sales = df.groupby(['Product',
'Date'])['Sales'].sum().sort_values(ascending=False)
print("Sorted Sales Grouped by Product and Date:")
print(sorted_sales)
print("\n")


# e. Split a given dataframe into groups with bin counts
num_bins = 3
df['SalesBins'] = pd.cut(df['Sales'], bins=num_bins)
bin_counts = df.groupby('SalesBins').size()
print("Bin Counts for Sales:")
print(bin_counts)
```

OUTPUT:-

```
Original Dataset:
        Date  Sales Product
```

```
0 2022-01-01  100.0       A
1 2022-01-02  120.0       B
2 2022-01-03    NaN       A
3 2022-01-04  150.0       B
4 2022-01-05  200.0       A
5 2022-01-06  180.0       B
6 2022-01-07    NaN       A
7 2022-01-08  220.0       B
8 2022-01-09  250.0       A
9 2022-01-10  300.0       B


Mean Sales Grouped by Product:
Product
A    182.5
B    212.5
Name: Sales, dtype: float64


Dataset after filling missing values:
            Sales Product
Date
2022-01-01  100.0       A
2022-01-02  120.0       B
2022-01-03  120.0       A
2022-01-04  150.0       B
2022-01-05  200.0       A
2022-01-06  180.0       B
```

```
2022-01-07  180.0       A
2022-01-08  220.0       B
2022-01-09  250.0       A
2022-01-10  300.0       B


Dataset after year-month string to date conversion:
            Sales Product  YearMonth
Date
2022-01-01  100.0       A 2022-01-01
2022-01-02  120.0       B 2022-01-01
2022-01-03  120.0       A 2022-01-01
2022-01-04  150.0       B 2022-01-01
2022-01-05  200.0       A 2022-01-01
2022-01-06  180.0       B 2022-01-01
2022-01-07  180.0       A 2022-01-01
2022-01-08  220.0       B 2022-01-01
2022-01-09  250.0       A 2022-01-01
2022-01-10  300.0       B 2022-01-01


Sorted Sales Grouped by Product and Date:
Product  Date
B        2022-01-10    300.0
         2022-01-08    220.0
         2022-01-06    180.0
         2022-01-04    150.0
         2022-01-02    120.0
```

```
A         2022-01-09    250.0
          2022-01-05    200.0
          2022-01-01    100.0
          2022-01-07    180.0
          2022-01-03    120.0
Name: Sales, dtype: float64



Bin Counts for Sales:
SalesBins
(99.7, 140.0]     3
(140.0, 180.0]    4
(180.0, 300.0]    3
dtype: int64
```

Q7. Consider a data frame containing data about students i.e. name, gender and passing division:

a. Perform one hot encoding of the last two columns of categorical data using the get_dummies() function.

b. Sort this data frame on the "Birth Month" column (i.e. January to December). Hint: Convert Month to Categorical.

Answer: -

```
import pandas as pd


# Creating the student DataFrame
data = {
    'Name': ['Mudit Chauhan', 'Seema Chopra', 'Rani Gupta',
'Aditya Narayan', 'Sanjeev Sahni', 'Prakash Kumar',
```

```
            'Ritu Agarwal', 'Akshay Goel', 'Meeta Kulkarni',
'Preeti Ahuja', 'Sunil Das Gupta', 'Sonali Sapre',
            'Rashmi Talwar', 'Ashish Dubey', 'Kiran Sharma',
'Sameer Bansal'],
    'Birth_Month': ['December', 'January', 'March', 'October',
'February', 'December', 'September', 'August',
                    'July', 'November', 'April', 'January',
'June', 'May', 'February', 'October'],
    'Gender': ['M', 'F', 'F', 'M', 'M', 'M', 'F', 'M', 'F', 'F',
'M', 'F', 'F', 'M', 'F', 'M'],
    'Pass_Division': ['III', 'II', 'I', 'I', 'II', 'III', 'I',
'I', 'II', 'II', 'III', 'I', 'III', 'II', 'II', 'I']
}

df = pd.DataFrame(data)

# a. Perform one hot encoding of the last two columns using
get_dummies()
df_encoded = pd.get_dummies(df, columns=['Gender',
'Pass_Division'])

# b. Sort the DataFrame on the "Birth Month" column
month_order = ['January', 'February', 'March', 'April', 'May',
'June', 'July', 'August', 'September', 'October', 'November',
'December']
df_encoded['Birth_Month'] =
pd.Categorical(df_encoded['Birth_Month'],
categories=month_order, ordered=True)
df_encoded = df_encoded.sort_values(by='Birth_Month')

# Displaying the resulting DataFrame
print("DataFrame after one-hot encoding and sorting:")
print(df_encoded)
```

OUTPUT:-

Out[3]: DataFrame after one-hot encoding and sorting:

	Name	Birth_Month	Gender_F	Gender_M	Pass_Division_I	Pass_Division_II	Pass_Division_III
1	Seema Chopra	January	True	False	False	True	False
11	Sonali Sapre	January	True	False	True	False	False
4	Sanjeev Sahni	February	False	True	False	True	False
14	Kiran Sharma	February	True	False	False	True	False
2	Rani Gupta	March	True	False	True	False	False
10	Sunil Das Gupta	April	False	True	False	False	True
13	Ashish Dubey	May	False	True	False	True	False
12	Rashmi Talwar	June	True	False	False	False	True
8	Meeta Kulkarni	July	True	False	False	True	False
7	Akshay Goel	August	False	True	True	False	False
6	Ritu Agarwal	September	True	False	True	False	False
3	Aditya Narayan	October	False	True	True	False	False
15	Sameer Bansal	October	False	True	True	False	False
9	Preeti Ahuja	November	True	False	False	True	False
0	Mudit Chauhan	December	False	True	False	False	True
5	Prakash Kumar	December	False	True	False	False	True

Q8. 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:

a. Calculate and display familywise gross monthly income.

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

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

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

Answer:-

```
import pandas as pd


# Creating the DataFrame
data = {
    'Name': ['Shah', 'Vats', 'Vats', 'Kumar', 'Vats', 'Kumar', 'Shah', 'Shah', 'Kumar', 'Vats'],
    'Gender': ['Male', 'Male', 'Female', 'Female', 'Female', 'Male', 'Male', 'Female', 'Female', 'Male'],
    'MonthlyIncome': [114000.00, 65000.00, 43150.00, 69500.00, 155000.00, 103000.00, 55000.00, 112400.00, 81030.00, 71900.00]
}


df = pd.DataFrame(data)


# a. Calculate and display familywise gross monthly income
familywise_income = df.groupby('Name')['MonthlyIncome'].sum()
print("Familywise Gross Monthly Income:")
print(familywise_income)
```

```
print("\n")


# b. Calculate and display the member with the highest monthly
income in each family

max_income_member =
df.loc[df.groupby('Name')['MonthlyIncome'].idxmax()]

print("Member with the Highest Monthly Income in Each Family:")

print(max_income_member)

print("\n")


# c. Calculate and display monthly income of all members with
income greater than Rs. 60000.00

high_income_members = df[df['MonthlyIncome'] > 60000.00]

print("Monthly Income of Members with Income Greater than Rs.
60000.00:")

print(high_income_members[['Name', 'Gender', 'MonthlyIncome']])

print("\n")


# d. Calculate and display the average monthly income of the
female members in the Shah family

average_female_income_shah = df[(df['Name'] == 'Shah') &
(df['Gender'] == 'Female')]['MonthlyIncome'].mean()

print("Average Monthly Income of Female Members in the Shah
Family:")

print(average_female_income_shah)
```

OUTPUT:-

Out[4]:
```
Familywise Gross Monthly Income:
Name
Kumar    253530.0
Shah     281400.0
Vats     335050.0
Name: MonthlyIncome, dtype: float64


Member with the Highest Monthly Income in Each Family:
    Name  Gender  MonthlyIncome
5  Kumar    Male       103000.0
0   Shah    Male       114000.0
4   Vats  Female       155000.0


Monthly Income of Members with Income Greater than Rs. 60000.00:
    Name  Gender  MonthlyIncome
0   Shah    Male       114000.0
1   Vats    Male        65000.0
3  Kumar  Female        69500.0
4   Vats  Female       155000.0
5  Kumar    Male       103000.0
7   Shah  Female       112400.0
8  Kumar  Female        81030.0
9   Vats    Male        71900.0


Average Monthly Income of Female Members in the Shah Family:
112400.0
```