December 8, 2023

1 Question 1

[1]:
```python
OG={'Boys':[72,68,70,69,74],'Girls':[63,65,69,62,61]}
temp={}
out=[]
for i in range(5):
  temp['Boys']=OG['Boys'][i]
  temp['Girls']=OG['Girls'][i]
  out.append(temp)
  temp={}
out
```

[1]:
```
[{'Boys': 72, 'Girls': 63},
 {'Boys': 68, 'Girls': 65},
 {'Boys': 70, 'Girls': 69},
 {'Boys': 69, 'Girls': 62},
 {'Boys': 74, 'Girls': 61}]
```

2 Question 2

A>

[2]:
```python
import numpy as np
arr=np.random.randint(1,10,(3,3))
print("Mean along second axis:",arr.mean(axis=1).round(2))
print("Standard Deviation along second axis:",arr.std(axis=1).round(2))
print("Variance along second axis:",arr.var(axis=1).round(2))
```

```
Mean along second axis: [4.   6.   4.33]
Standard Deviation along second axis: [2.45 1.41 2.36]
Variance along second axis: [6.   2.   5.56]
```

B>

[3]:
```python
B=[56,48,22,41,78,91,24,46,8,33]
A=np.sort(B)
out=[]
for i in A:
  loc=B.index(i)
  out.append(loc)
print("Indices of sorted elements of array:",out)
```

```
Indices of sorted elements of array: [8, 2, 6, 9, 3, 7, 1, 0, 4, 5]
```

C>

[4]:

```
#Q2c
m=int(input("Enter number of rows: "))
n=int(input("Enter number of columns: "))
out=np.random.randint(0,10,(m,n))
print("Original array: \n",out)
print("\nShape of array:",out.shape)
print("Type of array:",type(out))
print("Datatype of array:",out.dtype)
out1=out.reshape(n,m)
print("\nReshaped array \n",out1)
```

```
Enter number of rows: 4
Enter number of columns: 3
Original array:
 [[4 0 3]
 [7 8 3]
 [1 3 3]
 [8 9 8]]

Shape of array: (4, 3)
Type of array: <class 'numpy.ndarray'>
Datatype of array: int64

Reshaped array
 [[4 0 3 7]
 [8 3 1 3]
 [3 8 9 8]]
```

D>

[5]:

```
#Q2d
zeros=np.argwhere(out == 0)
non_zeros=np.argwhere(out != 0)
NaN=np.argwhere(out == np.NaN)
print("Number of zeros in given array:",zeros)
print("Number of zeros in given array:",non_zeros)
print("Number of zeros in given array:",NaN)
```

```
Number of zeros in given array: [[0 1]]
Number of zeros in given array: [[0 0]
 [0 2]
 [1 0]
 [1 1]
 [1 2]
 [2 0]
 [2 1]
 [2 2]
 [3 0]
 [3 1]
 [3 2]]
Number of zeros in given array: []
```

3 Q3

[9]:

```
import pandas as pd
import random
arr1=np.random.randint(1,20,(50,4))
data=pd.DataFrame(arr1)
X=random.sample(range(0,50),20)
Y=np.random.randint(0,4,20)
index=[]
for i in range(20):
  index.append((X[i],Y[i]))
  data.iloc[X[i],Y[i]]=np.NaN
```

A>

[10]:

```
#Q3a
print("Indices of missing values:",index)
print("Number of missing values: ",data.isnull().sum().sum())
```

```
Indices of missing values: [(10, 2), (30, 3), (28, 3), (20, 1), (25, 3), (29,
3), (4, 3), (49, 3), (24, 1), (47, 3), (31, 1), (17, 2), (35, 0), (32, 1), (43,
2), (40, 1), (18, 3), (37, 0), (26, 1), (0, 1)]
Number of missing values:  20
```

B>

[11]:

```
#Q3b
for i in range(4):
  if data.iloc[:,i].isnull().sum()>5:
    data1=data.drop(i,axis=1)
data1
```

[11]:

```
      0     1     2
0   1.0   NaN  19.0
1  14.0   5.0  13.0
2  18.0   6.0   6.0
3  12.0  14.0  11.0
```

```
4    8.0  13.0  18.0
5    2.0   5.0  14.0
6    8.0   2.0   5.0
7   16.0  10.0   1.0
8   18.0   8.0  17.0
9    3.0  16.0   7.0
10   1.0   3.0   NaN
11   8.0  15.0   6.0
12   5.0  12.0   7.0
13   3.0   1.0   4.0
14   5.0   8.0  13.0
15  14.0   1.0  10.0
16   8.0   2.0   9.0
17   7.0  14.0   NaN
18  12.0  12.0   8.0
19  16.0  16.0   2.0
20  11.0   NaN   1.0
21   2.0   9.0   2.0
22  18.0   2.0  13.0
23   9.0   1.0  14.0
24   3.0   NaN  17.0
25  19.0  11.0  11.0
26   3.0   NaN  13.0
27   8.0  12.0   9.0
28   8.0   4.0   3.0
29  10.0   8.0  19.0
30  18.0  18.0   1.0
31   7.0   NaN  18.0
32  15.0   NaN  12.0
33  18.0  10.0   8.0
34   9.0  14.0  10.0
35   NaN  10.0   7.0
36  15.0   9.0   2.0
37   NaN   2.0   7.0
38   2.0   8.0  19.0
39   4.0   5.0  16.0
40  12.0   NaN  11.0
41  11.0  18.0   4.0
42  12.0  16.0  19.0
43  11.0  10.0   NaN
44  12.0   5.0  16.0
45  10.0  19.0   2.0
46  16.0  13.0   9.0
47  13.0  19.0  13.0
48   4.0   5.0   7.0
49  19.0  17.0  15.0
```

C>

[12]:

```
#Q3c
sums=[]
for i in range(50):
  sums.append(data.iloc[i].sum())
data2=data.drop(sums.index(max(sums)))
data2
```

[12]:

```
       0     1     2     3
0    1.0   NaN  19.0  18.0
1   14.0   5.0  13.0  13.0
2   18.0   6.0   6.0   6.0
3   12.0  14.0  11.0  17.0
4    8.0  13.0  18.0   NaN
5    2.0   5.0  14.0  18.0
6    8.0   2.0   5.0   2.0
7   16.0  10.0   1.0  12.0
8   18.0   8.0  17.0   1.0
9    3.0  16.0   7.0  14.0
10   1.0   3.0   NaN   9.0
11   8.0  15.0   6.0  15.0
12   5.0  12.0   7.0  10.0
13   3.0   1.0   4.0   2.0
14   5.0   8.0  13.0  19.0
15  14.0   1.0  10.0  13.0
16   8.0   2.0   9.0  17.0
17   7.0  14.0   NaN   9.0
18  12.0  12.0   8.0   NaN
19  16.0  16.0   2.0  12.0
20  11.0   NaN   1.0   3.0
21   2.0   9.0   2.0  11.0
22  18.0   2.0  13.0  19.0
23   9.0   1.0  14.0   3.0
24   3.0   NaN  17.0   9.0
25  19.0  11.0  11.0   NaN
26   3.0   NaN  13.0   2.0
27   8.0  12.0   9.0   8.0
28   8.0   4.0   3.0   NaN
29  10.0   8.0  19.0   NaN
30  18.0  18.0   1.0   NaN
31   7.0   NaN  18.0  18.0
32  15.0   NaN  12.0  16.0
33  18.0  10.0   8.0   3.0
34   9.0  14.0  10.0  15.0
35   NaN  10.0   7.0  10.0
36  15.0   9.0   2.0   8.0
37   NaN   2.0   7.0  18.0
```

```
38   2.0   8.0  19.0   9.0
39   4.0   5.0  16.0  14.0
40  12.0   NaN  11.0  14.0
41  11.0  18.0   4.0  11.0
43  11.0  10.0   NaN  16.0
44  12.0   5.0  16.0  18.0
45  10.0  19.0   2.0  11.0
46  16.0  13.0   9.0   8.0
47  13.0  19.0  13.0   NaN
48   4.0   5.0   7.0   4.0
49  19.0  17.0  15.0   NaN
```

D>

[13]:
```
#Q3d
sort_data=data.sort_values(by=0)
sort_data
```

[13]:
```
       0     1     2     3
0    1.0   NaN  19.0  18.0
10   1.0   3.0   NaN   9.0
38   2.0   8.0  19.0   9.0
5    2.0   5.0  14.0  18.0
21   2.0   9.0   2.0  11.0
13   3.0   1.0   4.0   2.0
24   3.0   NaN  17.0   9.0
26   3.0   NaN  13.0   2.0
9    3.0  16.0   7.0  14.0
48   4.0   5.0   7.0   4.0
39   4.0   5.0  16.0  14.0
12   5.0  12.0   7.0  10.0
14   5.0   8.0  13.0  19.0
31   7.0   NaN  18.0  18.0
17   7.0  14.0   NaN   9.0
16   8.0   2.0   9.0  17.0
11   8.0  15.0   6.0  15.0
4    8.0  13.0  18.0   NaN
28   8.0   4.0   3.0   NaN
6    8.0   2.0   5.0   2.0
27   8.0  12.0   9.0   8.0
34   9.0  14.0  10.0  15.0
23   9.0   1.0  14.0   3.0
45  10.0  19.0   2.0  11.0
29  10.0   8.0  19.0   NaN
20  11.0   NaN   1.0   3.0
43  11.0  10.0   NaN  16.0
41  11.0  18.0   4.0  11.0
```

```
18  12.0  12.0   8.0   NaN
44  12.0   5.0  16.0  18.0
42  12.0  16.0  19.0  16.0
40  12.0   NaN  11.0  14.0
3   12.0  14.0  11.0  17.0
47  13.0  19.0  13.0   NaN
15  14.0   1.0  10.0  13.0
1   14.0   5.0  13.0  13.0
36  15.0   9.0   2.0   8.0
32  15.0   NaN  12.0  16.0
7   16.0  10.0   1.0  12.0
46  16.0  13.0   9.0   8.0
19  16.0  16.0   2.0  12.0
22  18.0   2.0  13.0  19.0
33  18.0  10.0   8.0   3.0
30  18.0  18.0   1.0   NaN
8   18.0   8.0  17.0   1.0
2   18.0   6.0   6.0   6.0
25  19.0  11.0  11.0   NaN
49  19.0  17.0  15.0   NaN
35   NaN  10.0   7.0  10.0
37   NaN   2.0   7.0  18.0
```

E>

[14]:

```
#Q3e
data4=data.drop_duplicates(subset=0)
data4
```

[14]:

```
       0     1     2     3
0    1.0   NaN  19.0  18.0
1   14.0   5.0  13.0  13.0
2   18.0   6.0   6.0   6.0
3   12.0  14.0  11.0  17.0
4    8.0  13.0  18.0   NaN
5    2.0   5.0  14.0  18.0
7   16.0  10.0   1.0  12.0
9    3.0  16.0   7.0  14.0
12   5.0  12.0   7.0  10.0
17   7.0  14.0   NaN   9.0
20  11.0   NaN   1.0   3.0
23   9.0   1.0  14.0   3.0
25  19.0  11.0  11.0   NaN
29  10.0   8.0  19.0   NaN
32  15.0   NaN  12.0  16.0
35   NaN  10.0   7.0  10.0
39   4.0   5.0  16.0  14.0
```

```
47  13.0  19.0  13.0   NaN
```

F>

[15]:

```
#Q3f
print("Correlation between first and second column:",data[0].corr(data[1]))
print("Covariance between first and second column:",data[1].cov(data[2]))
```

```
Correlation between first and second column: 0.24493049043896592
Covariance between first and second column: -3.9724358974358966
```

G>

[18]:

```
#Q3g
z_scores = (data - data.mean()) / data.std()
outliers = (z_scores > 3) | (z_scores < -3)
new_df = data[~outliers.any(axis=1)]
print("\nDataFrame after removing rows with outliers:")
new_df
```

```
DataFrame after removing rows with outliers:
```

[18]:

```
        0     1     2     3
0     1.0   NaN  19.0  18.0
1    14.0   5.0  13.0  13.0
2    18.0   6.0   6.0   6.0
3    12.0  14.0  11.0  17.0
4     8.0  13.0  18.0   NaN
5     2.0   5.0  14.0  18.0
6     8.0   2.0   5.0   2.0
7    16.0  10.0   1.0  12.0
8    18.0   8.0  17.0   1.0
9     3.0  16.0   7.0  14.0
10    1.0   3.0   NaN   9.0
11    8.0  15.0   6.0  15.0
12    5.0  12.0   7.0  10.0
13    3.0   1.0   4.0   2.0
14    5.0   8.0  13.0  19.0
15   14.0   1.0  10.0  13.0
16    8.0   2.0   9.0  17.0
17    7.0  14.0   NaN   9.0
18   12.0  12.0   8.0   NaN
19   16.0  16.0   2.0  12.0
20   11.0   NaN   1.0   3.0
21    2.0   9.0   2.0  11.0
22   18.0   2.0  13.0  19.0
23    9.0   1.0  14.0   3.0
24    3.0   NaN  17.0   9.0
```

```
25  19.0  11.0  11.0   NaN
26   3.0   NaN  13.0   2.0
27   8.0  12.0   9.0   8.0
28   8.0   4.0   3.0   NaN
29  10.0   8.0  19.0   NaN
30  18.0  18.0   1.0   NaN
31   7.0   NaN  18.0  18.0
32  15.0   NaN  12.0  16.0
33  18.0  10.0   8.0   3.0
34   9.0  14.0  10.0  15.0
35   NaN  10.0   7.0  10.0
36  15.0   9.0   2.0   8.0
37   NaN   2.0   7.0  18.0
38   2.0   8.0  19.0   9.0
39   4.0   5.0  16.0  14.0
40  12.0   NaN  11.0  14.0
41  11.0  18.0   4.0  11.0
42  12.0  16.0  19.0  16.0
43  11.0  10.0   NaN  16.0
44  12.0   5.0  16.0  18.0
45  10.0  19.0   2.0  11.0
46  16.0  13.0   9.0   8.0
47  13.0  19.0  13.0   NaN
48   4.0   5.0   7.0   4.0
49  19.0  17.0  15.0   NaN
```

H>

```
[19]: #Q3h
      data['Bin'] = pd.cut(data[1], bins=5, labels=False)
```

4 Question 4

```
[20]: file1=pd.read_csv('file1.csv')
      file2=pd.read_csv('file2.csv')
```

```
[21]: file1,file2
```

```
[21]: (       Name Time of Joining  Duration
       0        Om         9:03:00        50
       1     Aryan         9:00:00        40
       2   Vaibhav         8:56:00        30
       3   Parvesh         8:59:00        30
       4    Girish         9:01:00        30
       5    Pankaj         9:03:00        50
       6  Abhigyan         9:05:00        40
```

```
 7    Suyash          8:57:00        40
 8    Mayank          9:00:00        50
 9    Akshit          9:01:00        30,
        Name Time of Joining   Duration
 0        Om          9:01:00        50
 1     Aryan          9:03:00        30
 2   Parvesh          8:56:00        40
 3    Aditya          8:57:00        30
 4    Girish          9:00:00        40
 5   Anushka          9:03:00        40
 6     Tanya          9:01:00        50
 7    Suyash          8:58:00        40
 8    Mayank          8:55:00        30
 9  Prikshit          9:00:00        30)
```

A>

[22]:
```
#Q4a
merge_both=pd.merge(file1,file2,on='Name',how='inner')
print("Names of students who had attended the workshop on both days:
 ↪\n",merge_both['Name'])
```

```
Names of students who had attended the workshop on both days:
 0         Om
1      Aryan
2    Parvesh
3     Girish
4     Suyash
5     Mayank
Name: Name, dtype: object
```

B>

[23]:
```
#Q4b
merge_either=pd.merge(file1,file2,on='Name',how='outer')
print("Names of students who had attended the workshop on either of the days:
 ↪\n",merge_either['Name'])
```

```
Names of students who had attended the workshop on either of the days:
 0           Om
1        Aryan
2      Vaibhav
3      Parvesh
4       Girish
5       Pankaj
6     Abhigyan
7       Suyash
8       Mayank
9       Akshit
```

```
10      Aditya
11     Anushka
12       Tanya
13    Prikshit
Name: Name, dtype: object
```

C>

[24]:
```
#Q4c
concat_df=pd.concat([file1,file2],ignore_index=True)
print("Total number of records in concatenated dataframe:",concat_df.shape[0])
```

```
Total number of records in concatenated dataframe: 20
```

D>

[25]:
```
#Q4d
merged_multi_index = pd.merge(file1, file2, on=['Name', 'Duration'],␣
 ↪how='inner')
multi_merge_stats = merged_multi_index.groupby(['Name', 'Duration']).describe()
multi_merge_stats
```

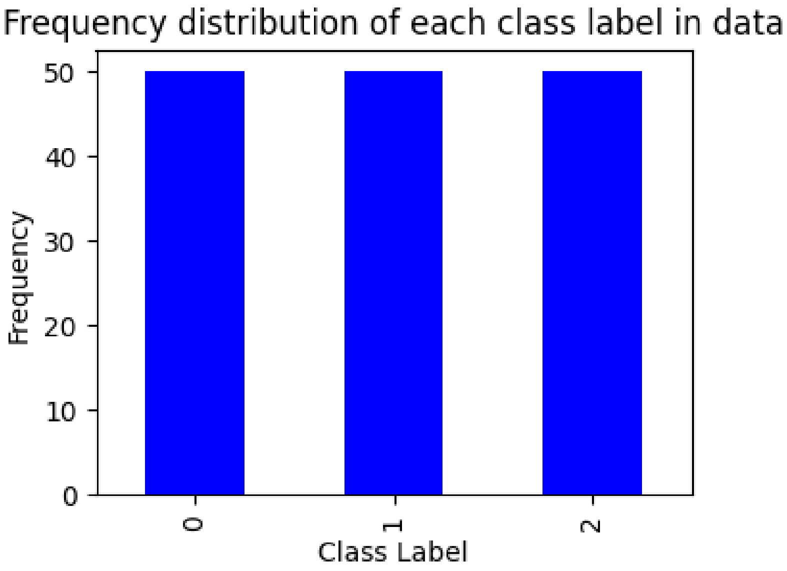
[25]:
```
                Time of Joining _x                        Time of Joining _y  \
                             count unique      top freq              count
Name   Duration
Om     50                        1      1  9:03:00    1                  1
Suyash 40                        1      1  8:57:00    1                  1


                unique      top freq
Name   Duration
Om     50            1  9:01:00    1
Suyash 40            1  8:58:00    1
```

5 Question 5

A>

[26]:
```
#Q5a
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
X,y=load_iris(return_X_y=True,as_frame=True)
count=y.value_counts()
plt.figure(figsize=(4, 3))
count.plot(kind='bar', color='blue')
plt.xlabel("Class Label")
plt.ylabel("Frequency")
plt.title("Frequency distribution of each class label in data")
plt.show()
```

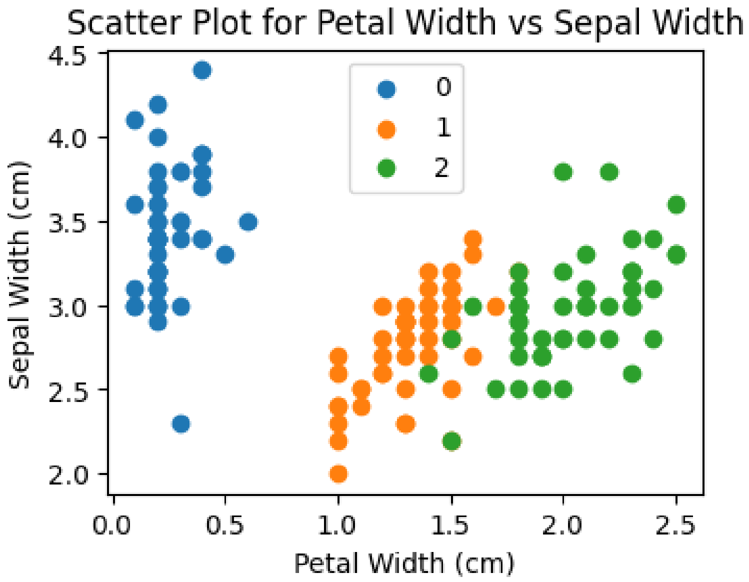
```
#Frequency of each class label in Iris dataset is 50
```



B>

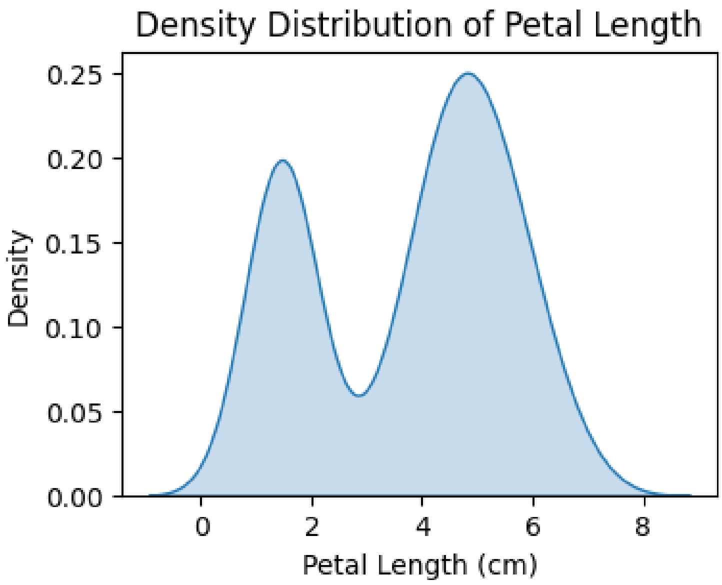
[27]:

```
#Q5b
plt.figure(figsize=(4, 3))
for i in y.unique():
    subset = X[y == i]
    plt.scatter(subset['petal width (cm)'], subset['sepal width (cm)'], label=i)
plt.ylabel('Sepal Width (cm)')
plt.xlabel('Petal Width (cm)')
plt.title("Scatter Plot for Petal Width vs Sepal Width")
plt.legend()
plt.show()
```



C>

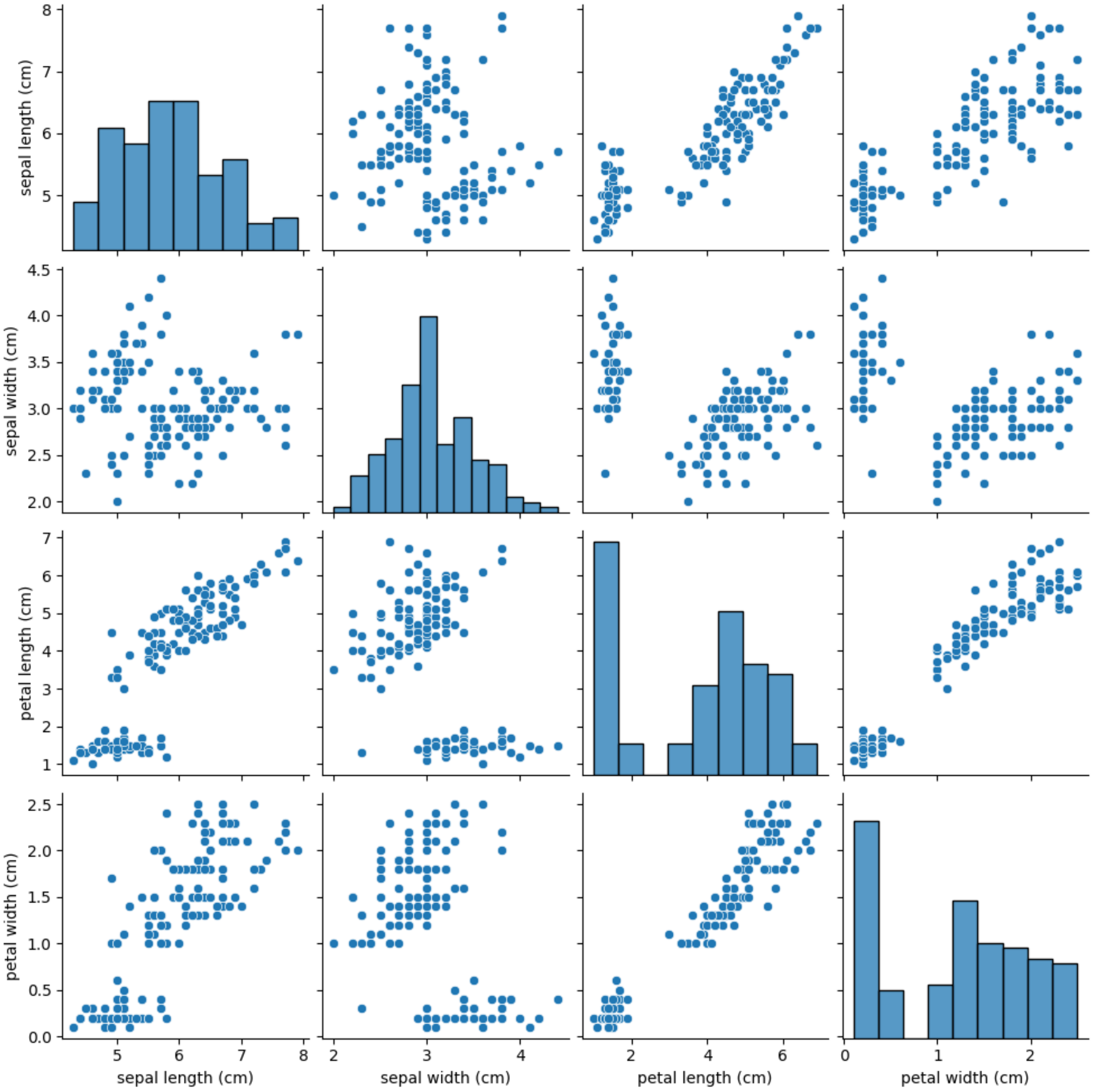
[28]:
```
#Q5c
import seaborn as sns
plt.figure(figsize=(4, 3))
sns.kdeplot(X['petal length (cm)'], fill=True)
plt.title('Density Distribution of Petal Length')
plt.xlabel('Petal Length (cm)')
plt.ylabel('Density')
plt.show()
```



D>

```
[29]: #Q5d
      sns.pairplot(X)
```

```
[29]: <seaborn.axisgrid.PairGrid at 0x7aa2daf01090>
```



6 Question 6

A>

```
[30]: #Q6a
      weather=pd.read_csv('DailyDelhiClimateTest.csv')
      weather.groupby('meanpressure')['wind_speed'].mean()
      weather.head()
```

```
[30]:         date   meantemp   humidity  wind_speed  meanpressure
      0  2017-01-01  15.913043  85.869565    2.743478     59.000000
      1  2017-01-02  18.500000  77.222222    2.894444   1018.277778
      2  2017-01-03  17.111111  81.888889    4.016667   1018.333333
      3  2017-01-04  18.700000  70.050000    4.545000   1015.700000
      4  2017-01-05  18.388889  74.944444    3.300000   1014.333333
```

B>

```
[ ]: #Q6b
     df_weather_filled = weather.set_index('date').asfreq('D', method='pad')
     print("DataFrame with Missing Dates Filled:")
     print(df_weather_filled)
```

C>

```
[32]: #Q6c
      weather['YearMonth'] = pd.to_datetime(weather['date'],format="%Y-%m-%d").dt.
       ↪to_period('M')
      print("Converted Year-Month:")
      print(weather[['date', 'YearMonth']])
```

```
Converted Year-Month:
           date YearMonth
0    2017-01-01   2017-01
1    2017-01-02   2017-01
2    2017-01-03   2017-01
3    2017-01-04   2017-01
4    2017-01-05   2017-01
..          …         …
109  2017-04-20   2017-04
110  2017-04-21   2017-04
111  2017-04-22   2017-04
112  2017-04-23   2017-04
113  2017-04-24   2017-04

[114 rows x 2 columns]
```

D>

```
[33]: #Q6d
      sorted_weather_by_pressure = weather.groupby(['meanpressure', 'YearMonth']).
       ↪agg({'meantemp':'mean','humidity':'mean'}).reset_index()
      sorted_weather_by_pressure
```

[33]:

```
     meanpressure YearMonth   meantemp   humidity
0       59.000000   2017-01  15.913043  85.869565
1      998.625000   2017-04  34.500000  27.500000
2      999.875000   2017-04  34.250000  39.375000
3     1000.875000   2017-04  33.500000  24.125000
4     1001.600000   2017-04  32.900000  40.900000
..            …       …         …          …
106   1021.375000   2017-02  16.875000  65.500000
107   1021.555556   2017-02  16.333333  67.000000
108   1021.789474   2017-01  15.263158  66.473684
109   1021.958333   2017-01  13.041667  78.333333
110   1022.809524   2017-01  14.619048  75.142857

[111 rows x 4 columns]
```

E>

[34]:

```python
#Q6e
temp_bins = [0, 15, 25, 35]
weather['TempBins'] = pd.cut(weather['meantemp'], bins=temp_bins)
groupby_bins = weather.groupby('TempBins')
print(groupby_bins.describe())
```

```
         meantemp                                                    \
            count       mean       std     min        25%        50%
TempBins
(0, 15]      13.0  13.398375  1.381566  11.000  12.111111  13.235294
(15, 25]     67.0  18.999372  2.790567  15.125  16.472222  18.631579
(25, 35]     34.0  30.239829  2.269097  25.625  29.132692  30.194444

                                humidity            … wind_speed            \
               75%        max      count       mean  …        75%        max
TempBins                                             …
(0, 15]   14.650000  14.863636      13.0  77.502871  …   9.772222  10.380000
(15, 25]  20.842857  25.000000      67.0  63.864985  …   9.473333  16.662500
(25, 35]  31.336806  34.500000      34.0  33.145938  …  12.939286  19.314286

         meanpressure                                                 \
                count         mean         std       min          25%
TempBins
(0, 15]          13.0  1017.641666    2.894354  1011.375  1016.368421
(15, 25]         67.0  1000.470917  116.827770    59.000  1011.830808
(25, 35]         34.0  1005.856092    3.299112   998.625  1003.473214


                50%          75%          max
TempBins
(0, 15]   1017.1500  1018.840000  1022.809524
(15, 25]   1015.2500  1017.676136  1021.789474
(25, 35]   1006.0625  1008.799107  1010.625000

[3 rows x 32 columns]
```

7 Question 7

[35]:
```
dic = {
'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(dic)
```

A>

[37]:
```
#Q7a
df_enc=pd.get_dummies(df,columns=['Gender','Pass_Division'])
df_enc
```

[37]:
```
               Name Birth_Month  Gender_F  Gender_M  Pass_Division_I  \
0     Mudit Chauhan    December         0         1                0
1      Seema Chopra     January         1         0                0
2        Rani Gupta       March         1         0                1
3    Aditya Narayan     October         0         1                1
4     Sanjeev Sahni    February         0         1                0
5     Prakash Kumar    December         0         1                0
6      Ritu Agarwal   September         1         0                1
7       Akshay Goel      August         0         1                1
8    Meeta Kulkarni        July         1         0                0
9      Preeti Ahuja    November         1         0                0
10  Sunil Das Gupta       April         0         1                0
11     Sonali Sapre     January         1         0                1
12    Rashmi Talwar        June         1         0                0
13     Ashish␣Dubey         May         0         1                0
14     Kiran Sharma    February         1         0                0
```

```
15    Sameer Bansal     October         0         1                 1

    Pass_Division_II  Pass_Division_III
0                  0                  1
1                  1                  0
2                  0                  0
3                  0                  0
4                  1                  0
5                  0                  1
6                  0                  0
7                  0                  0
8                  1                  0
9                  1                  0
10                 0                  1
11                 0                  0
12                 0                  1
13                 1                  0
14                 1                  0
15                 0                  0
```

B>

[38]:

```
#Q7b
months = ['January', 'February', 'March', 'April', 'May', 'June', 'July',␣
 ↪'August', 'September', 'October', 'November', 'December']
df['Birth_Month'] = pd.Categorical(df['Birth_Month'], categories=months,␣
 ↪ordered=True)
df.sort_values(by='Birth_Month')
```

[38]:

```
               Name Birth_Month Gender Pass_Division
1      Seema Chopra     January      F            II
11     Sonali Sapre     January      F             I
4     Sanjeev Sahni    February      M            II
14     Kiran Sharma    February      F            II
2        Rani Gupta       March      F             I
10  Sunil Das Gupta       April      M           III
13     Ashish␣Dubey         May      M            II
12   Rashmi Talwar        June      F           III
8    Meeta Kulkarni        July      F            II
7       Akshay Goel      August      M             I
6      Ritu Agarwal   September      F             I
3    Aditya Narayan     October      M             I
15    Sameer Bansal     October      M             I
9      Preeti Ahuja    November      F            II
0     Mudit Chauhan    December      M           III
5     Prakash Kumar    December      M           III
```

8 Question 8

A>

[39]:
```
#Q8a
df=pd.read_csv('q8.csv')
print("Familywise gross monthly income:\n")
df.groupby('Name ')['MonthlyIncome(Rs.)'].sum()
```

```
Familywise gross monthly income:
```

[39]:
```
Name
Kumar    253530
Shah     281400
Vats     335050
Name: MonthlyIncome(Rs.), dtype: int64
```

B>

[40]:
```
#Q8b
df.loc[df.groupby('Name ')["MonthlyIncome(Rs.)"].idxmax()]
```

[40]:
```
   Name   Gender  MonthlyIncome(Rs.)
5  Kumar    Male              103000
0   Shah    Male              114000
4   Vats  Female              155000
```

C>

[41]:
```
#Q8c
df[df['MonthlyIncome(Rs.)']>60000][['Name ','Gender','MonthlyIncome(Rs.)']]
```

[41]:
```
   Name   Gender  MonthlyIncome(Rs.)
0   Shah    Male              114000
1   Vats    Male               65000
3  Kumar  Female               69500
4   Vats  Female              155000
5  Kumar    Male              103000
7   Shah  Female              112400
8  Kumar  Female               81030
9   Vats    Male               71900
```

D>

[42]:
```
#Q8d
print("Average monthly income of the female members of 'Shah' Family")
df[(df['Name ']=='Vats') & (df['Gender']=='Female')]['MonthlyIncome(Rs.)'].
 ↪mean()
```

```
Average monthly income of the female members of 'Shah' Family
```

```
[42]: 99075.0
```