In [66]:
```
import pandas as pd
df=pd.read_csv ("E:\chrome downloads\iris\iris.data", header=None, names=cc)
#df_xlsx=pd.read_excel('')
#df_txt=pd.read_csv ("E:\chrome downloads\iris\iris.txt, delimeter='\t'")
print(df)
```

```
       1    2    3    4               5
0    5.1  3.5  1.4  0.2     Iris-setosa
1    4.9  3.0  1.4  0.2     Iris-setosa
2    4.7  3.2  1.3  0.2     Iris-setosa
3    4.6  3.1  1.5  0.2     Iris-setosa
4    5.0  3.6  1.4  0.2     Iris-setosa
..   ...  ...  ...  ...             ...
145  6.7  3.0  5.2  2.3  Iris-virginica
146  6.3  2.5  5.0  1.9  Iris-virginica
147  6.5  3.0  5.2  2.0  Iris-virginica
148  6.2  3.4  5.4  2.3  Iris-virginica
149  5.9  3.0  5.1  1.8  Iris-virginica

[150 rows x 5 columns]
```

In [65]:
```
cc=["1","2","3","4","5"]
cc
```

Out[65]:
```
['1', '2', '3', '4', '5']
```

In [67]:
```
#to read headers
df.columns
```

Out[67]:
```
Index(['1', '2', '3', '4', '5'], dtype='object')
```

In [17]:
```
# to read each clm
print(df[["3","5"]])
```

```
       3               5
0    1.4     Iris-setosa
1    1.4     Iris-setosa
2    1.3     Iris-setosa
3    1.5     Iris-setosa
4    1.4     Iris-setosa
..   ...             ...
145  5.2  Iris-virginica
146  5.0  Iris-virginica
147  5.2  Iris-virginica
148  5.4  Iris-virginica
149  5.1  Iris-virginica

[150 rows x 2 columns]
```

In [31]:
```
#each row
#df.iloc[0]
#for index ,row in df.iterrows():
#    print(index,row['5'])
df.loc[df['5']=="Iris-setosa"]
```

Out[31]:

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| **5** | 5.4 | 3.9 | 1.7 | 0.4 | Iris-setosa |
| **6** | 4.6 | 3.4 | 1.4 | 0.3 | Iris-setosa |
| **7** | 5.0 | 3.4 | 1.5 | 0.2 | Iris-setosa |
| **8** | 4.4 | 2.9 | 1.4 | 0.2 | Iris-setosa |
| **9** | 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa |
| **10** | 5.4 | 3.7 | 1.5 | 0.2 | Iris-setosa |
| **11** | 4.8 | 3.4 | 1.6 | 0.2 | Iris-setosa |
| **12** | 4.8 | 3.0 | 1.4 | 0.1 | Iris-setosa |
| **13** | 4.3 | 3.0 | 1.1 | 0.1 | Iris-setosa |
| **14** | 5.8 | 4.0 | 1.2 | 0.2 | Iris-setosa |
| **15** | 5.7 | 4.4 | 1.5 | 0.4 | Iris-setosa |
| **16** | 5.4 | 3.9 | 1.3 | 0.4 | Iris-setosa |
| **17** | 5.1 | 3.5 | 1.4 | 0.3 | Iris-setosa |
| **18** | 5.7 | 3.8 | 1.7 | 0.3 | Iris-setosa |
| **19** | 5.1 | 3.8 | 1.5 | 0.3 | Iris-setosa |
| **20** | 5.4 | 3.4 | 1.7 | 0.2 | Iris-setosa |
| **21** | 5.1 | 3.7 | 1.5 | 0.4 | Iris-setosa |
| **22** | 4.6 | 3.6 | 1.0 | 0.2 | Iris-setosa |
| **23** | 5.1 | 3.3 | 1.7 | 0.5 | Iris-setosa |
| **24** | 4.8 | 3.4 | 1.9 | 0.2 | Iris-setosa |
| **25** | 5.0 | 3.0 | 1.6 | 0.2 | Iris-setosa |
| **26** | 5.0 | 3.4 | 1.6 | 0.4 | Iris-setosa |
| **27** | 5.2 | 3.5 | 1.5 | 0.2 | Iris-setosa |
| **28** | 5.2 | 3.4 | 1.4 | 0.2 | Iris-setosa |
| **29** | 4.7 | 3.2 | 1.6 | 0.2 | Iris-setosa |
| **30** | 4.8 | 3.1 | 1.6 | 0.2 | Iris-setosa |
| **31** | 5.4 | 3.4 | 1.5 | 0.4 | Iris-setosa |
| **32** | 5.2 | 4.1 | 1.5 | 0.1 | Iris-setosa |
| **33** | 5.5 | 4.2 | 1.4 | 0.2 | Iris-setosa |
| **34** | 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa |
| **35** | 5.0 | 3.2 | 1.2 | 0.2 | Iris-setosa |

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| **36** | 5.5 | 3.5 | 1.3 | 0.2 | Iris-setosa |
| **37** | 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa |
| **38** | 4.4 | 3.0 | 1.3 | 0.2 | Iris-setosa |
| **39** | 5.1 | 3.4 | 1.5 | 0.2 | Iris-setosa |
| **40** | 5.0 | 3.5 | 1.3 | 0.3 | Iris-setosa |
| **41** | 4.5 | 2.3 | 1.3 | 0.3 | Iris-setosa |
| **42** | 4.4 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| **43** | 5.0 | 3.5 | 1.6 | 0.6 | Iris-setosa |
| **44** | 5.1 | 3.8 | 1.9 | 0.4 | Iris-setosa |
| **45** | 4.8 | 3.0 | 1.4 | 0.3 | Iris-setosa |
| **46** | 5.1 | 3.8 | 1.6 | 0.2 | Iris-setosa |
| **47** | 4.6 | 3.2 | 1.4 | 0.2 | Iris-setosa |
| **48** | 5.3 | 3.7 | 1.5 | 0.2 | Iris-setosa |
| **49** | 5.0 | 3.3 | 1.4 | 0.2 | Iris-setosa |

In [26]:
```
#read specific location(r,c)
df.iloc[148,4]
```

Out[26]: 'Iris-virginica'

In [32]:
```
df.describe()
```

Out[32]:

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| **count** | 150.000000 | 150.000000 | 150.000000 | 150.000000 |
| **mean** | 5.843333 | 3.054000 | 3.758667 | 1.198667 |
| **std** | 0.828066 | 0.433594 | 1.764420 | 0.763161 |
| **min** | 4.300000 | 2.000000 | 1.000000 | 0.100000 |
| **25%** | 5.100000 | 2.800000 | 1.600000 | 0.300000 |
| **50%** | 5.800000 | 3.000000 | 4.350000 | 1.300000 |
| **75%** | 6.400000 | 3.300000 | 5.100000 | 1.800000 |
| **max** | 7.900000 | 4.400000 | 6.900000 | 2.500000 |

In [37]:
```
df.sort_values("4", ascending=False)
```

Out[37]:

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| **100** | 6.3 | 3.3 | 6.0 | 2.5 | Iris-virginica |
| **109** | 7.2 | 3.6 | 6.1 | 2.5 | Iris-virginica |
| **144** | 6.7 | 3.3 | 5.7 | 2.5 | Iris-virginica |
| **114** | 5.8 | 2.8 | 5.1 | 2.4 | Iris-virginica |
| **140** | 6.7 | 3.1 | 5.6 | 2.4 | Iris-virginica |
| **...** | ... | ... | ... | ... | ... |
| **13** | 4.3 | 3.0 | 1.1 | 0.1 | Iris-setosa |
| **37** | 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa |
| **32** | 5.2 | 4.1 | 1.5 | 0.1 | Iris-setosa |
| **34** | 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa |
| **9** | 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa |

150 rows × 5 columns

In [48]:
```
#making changes to data
df['Total'] = df['1'] + df['2'] + df['3'] + df['4']
df
```

Out[48]:

| | 1 | 2 | 3 | 4 | 5 | Total |
|---|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa | 10.2 |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa | 9.5 |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa | 9.4 |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa | 9.4 |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa | 10.2 |
| **...** | ... | ... | ... | ... | ... | ... |
| **145** | 6.7 | 3.0 | 5.2 | 2.3 | Iris-virginica | 17.2 |
| **146** | 6.3 | 2.5 | 5.0 | 1.9 | Iris-virginica | 15.7 |
| **147** | 6.5 | 3.0 | 5.2 | 2.0 | Iris-virginica | 16.7 |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 | Iris-virginica | 17.3 |
| **149** | 5.9 | 3.0 | 5.1 | 1.8 | Iris-virginica | 15.8 |

150 rows × 6 columns

In [49]:
```
df[df.duplicated()]
```

Out[49]:

| | 1 | 2 | 3 | 4 | 5 | Total |
|---|---|---|---|---|---|---|
| **34** | 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa | 9.6 |
| **37** | 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa | 9.6 |
| **142** | 5.8 | 2.7 | 5.1 | 1.9 | Iris-virginica | 15.5 |

In [50]:
```
df.info
```

Out[50]:
```
<bound method DataFrame.info of        1    2    3    4                5  Tota
l
0    5.1  3.5  1.4  0.2     Iris-setosa   10.2
1    4.9  3.0  1.4  0.2     Iris-setosa    9.5
2    4.7  3.2  1.3  0.2     Iris-setosa    9.4
3    4.6  3.1  1.5  0.2     Iris-setosa    9.4
4    5.0  3.6  1.4  0.2     Iris-setosa   10.2
..   ...  ...  ...  ...             ...    ...
145  6.7  3.0  5.2  2.3  Iris-virginica   17.2
146  6.3  2.5  5.0  1.9  Iris-virginica   15.7
147  6.5  3.0  5.2  2.0  Iris-virginica   16.7
148  6.2  3.4  5.4  2.3  Iris-virginica   17.3
149  5.9  3.0  5.1  1.8  Iris-virginica   15.8

[150 rows x 6 columns]>
```

In [27]:
```
import pandas as pd
data = {'firstName':['Aryan','Rohan','Riya','Yash','Siddant'],'LastName':['Sin
df=pd.DataFrame(data)
df
```

Out[27]:

| | firstName | LastName | Type | Department | salary | yoe |
|---|---|---|---|---|---|---|
| **0** | Aryan | Singh | Full time | Admin | 20000 | 2 |
| **1** | Rohan | Agarval | intern | Tech | 5000 | 3 |
| **2** | Riya | Shah | Full time | Admin | 10000 | 5 |
| **3** | Yash | Bhatia | part time | Tech | 10000 | 7 |
| **4** | Siddant | Khanna | Full time | Management | 20000 | 6 |

In [7]:
```
avg_sal=df.pivot_table(values='salary',index='Department',columns='Type',aggfu
avg_sal
```

Out[7]:

| Type | Full time | intern | part time |
|---|---|---|---|
| **Department** | | | |
| **Admin** | 1.000005e+09 | NaN | NaN |
| **Management** | 2.000000e+04 | NaN | NaN |
| **Tech** | NaN | 5000.0 | 10000.0 |

In [9]:
```
sum_mean=df.pivot_table(values='salary',index='Type',aggfunc=['sum','mean','co
sum_mean
```

Out[9]:

| | sum | mean | count |
|---|---|---|---|
| | salary | salary | salary |
| **Type** | | | |
| **Full time** | 200001000020000 | 6.666700e+13 | 3 |
| **intern** | 5000 | 5.000000e+03 | 1 |
| **part time** | 10000 | 1.000000e+04 | 1 |

In [10]:
```
std_df=df.pivot_table(values='salary',index='Type',aggfunc='std')
std_df
```

Out[10]:

| | salary |
|---|---|
| **Type** | |
| **Full time** | 5773.502692 |

In [29]:
```
seriesA=pd.Series([10,20,30,40,50,60])
seriesB=pd.Series([40,50,60,70,80,90])
not_common=seriesA.append(seriesB).unique()
not_common
```

```
C:\Users\NUTHAN SM\AppData\Local\Temp\ipykernel_6636\2351936300.py:3: FutureW
arning: The series.append method is deprecated and will be removed from panda
s in a future version. Use pandas.concat instead.
  not_common=seriesA.append(seriesB).unique()
```

Out[29]:
```
array([10, 20, 30, 40, 50, 60, 70, 80, 90], dtype=int64)
```

In [31]:
```
smallest=seriesA.min()
print(smallest)
largest=seriesB.max()
largest
```

```
10
```

Out[31]:
```
90
```

In [32]:
```
sumb=seriesB.sum()
sumb
```

Out[32]:
```
390
```

In [35]:
```
avg_A=seriesA.mean()
avg_A
```

Out[35]:
```
35.0
```

In [36]:
```
medi_B=seriesB.median()
medi_B
```

Out[36]: 65.0

In [47]:
```
auto_mpg=pd.read_csv('C://datasets/auto-mpg.csv')
auto_mpg.head()
```

Out[47]:

| | mpg | cylinders | displacement | horsepower | weight | acceleration | model year | origin | car name |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 18.0 | 8 | 307.0 | 130 | 3504 | 12.0 | 70 | 1 | chevrolet chevelle malibu |
| **1** | 15.0 | 8 | 350.0 | 165 | 3693 | 11.5 | 70 | 1 | buick skylark 320 |
| **2** | 18.0 | 8 | 318.0 | 150 | 3436 | 11.0 | 70 | 1 | plymouth satellite |
| **3** | 16.0 | 8 | 304.0 | 150 | 3433 | 12.0 | 70 | 1 | amc rebel sst |
| **4** | 17.0 | 8 | 302.0 | 140 | 3449 | 10.5 | 70 | 1 | ford torino |

In [50]:
```
auto_mpg.describe()
```

Out[50]:

| | mpg | cylinders | displacement | weight | acceleration | model year | origin |
|---|---|---|---|---|---|---|---|
| **count** | 398.000000 | 398.000000 | 398.000000 | 398.000000 | 398.000000 | 398.000000 | 398.000000 |
| **mean** | 23.514573 | 5.454774 | 193.425879 | 2970.424623 | 15.568090 | 76.010050 | 1.572864 |
| **std** | 7.815984 | 1.701004 | 104.269838 | 846.841774 | 2.757689 | 3.697627 | 0.802055 |
| **min** | 9.000000 | 3.000000 | 68.000000 | 1613.000000 | 8.000000 | 70.000000 | 1.000000 |
| **25%** | 17.500000 | 4.000000 | 104.250000 | 2223.750000 | 13.825000 | 73.000000 | 1.000000 |
| **50%** | 23.000000 | 4.000000 | 148.500000 | 2803.500000 | 15.500000 | 76.000000 | 1.000000 |
| **75%** | 29.000000 | 8.000000 | 262.000000 | 3608.000000 | 17.175000 | 79.000000 | 2.000000 |
| **max** | 46.600000 | 8.000000 | 455.000000 | 5140.000000 | 24.800000 | 82.000000 | 3.000000 |

In [51]:
```
eight=auto_mpg[auto_mpg['cylinders']==8]
eight
```

Out[51]:

| | mpg | cylinders | displacement | horsepower | weight | acceleration | model year | origin | car name |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 18.0 | 8 | 307.0 | 130 | 3504 | 12.0 | 70 | 1 | chevrolet chevelle malibu |
| **1** | 15.0 | 8 | 350.0 | 165 | 3693 | 11.5 | 70 | 1 | buick skylark 320 |
| **2** | 18.0 | 8 | 318.0 | 150 | 3436 | 11.0 | 70 | 1 | plymouth satellite |
| **3** | 16.0 | 8 | 304.0 | 150 | 3433 | 12.0 | 70 | 1 | amc rebel sst |
| **4** | 17.0 | 8 | 302.0 | 140 | 3449 | 10.5 | 70 | 1 | ford torino |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **291** | 19.2 | 8 | 267.0 | 125 | 3605 | 15.0 | 79 | 1 | chevrolet malibu classic (sw) |
| **292** | 18.5 | 8 | 360.0 | 150 | 3940 | 13.0 | 79 | 1 | chrysler lebaron town @ country (sw) |
| **298** | 23.0 | 8 | 350.0 | 125 | 3900 | 17.4 | 79 | 1 | cadillac eldorado |
| **300** | 23.9 | 8 | 260.0 | 90 | 3420 | 22.2 | 79 | 1 | oldsmobile cutlass salon brougham |
| **364** | 26.6 | 8 | 350.0 | 105 | 3725 | 19.0 | 81 | 1 | oldsmobile cutlass ls |

103 rows × 9 columns

In [54]:
```
cars_new=auto_mpg.groupby('model year')['car name'].count()
cars_new
```

Out[54]:
```
model year
70    29
71    28
72    28
73    40
74    27
75    30
76    34
77    28
78    36
79    29
80    29
81    29
82    31
Name: car name, dtype: int64
```

In [62]:
```
import numpy as np
data = np.array([[1, 6012], [2, 4079], [3, 6386], [4, 5230], [5, 4598], [6,
5564], [7, 6971], [8, 7763], [9, 8032], [10, 8569]])
print(data)
```

```
[[   1 6012]
 [   2 4079]
 [   3 6386]
 [   4 5230]
 [   5 4598]
 [   6 5564]
 [   7 6971]
 [   8 7763]
 [   9 8032]
 [  10 8569]]
```

In [63]:
```
steps_more_than_9000 = data[data[:, 1] > 9000]
steps_more_than_9000
```

Out[63]:
```
array([], shape=(0, 2), dtype=int32)
```

In [70]:
```
df.head(5)
```

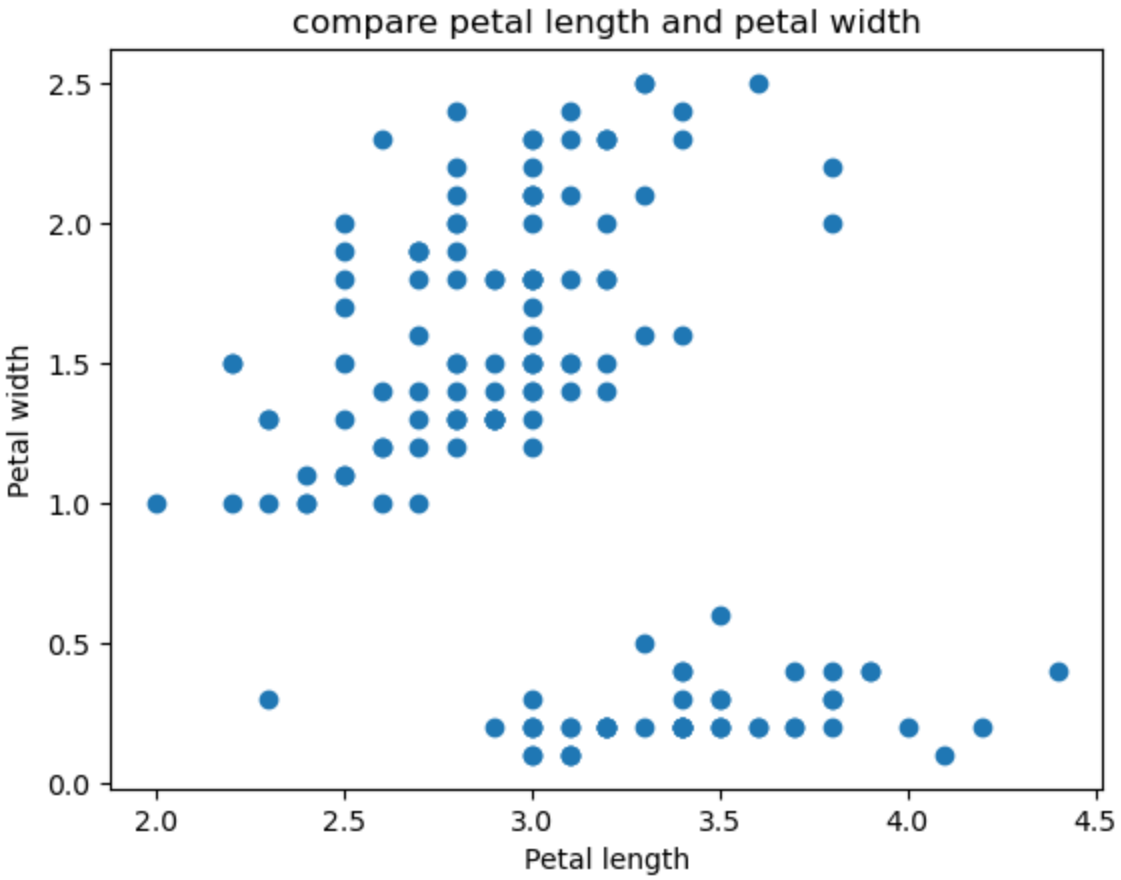
Out[70]:

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |

In [76]:
```
print(df.shape)
```

```
(150, 5)
```

In [86]:
```
import matplotlib.pyplot as plt
plt.scatter(df['2'], df['4'])
plt.title('compare petal length and petal width')
plt.xlabel('Petal length')
plt.ylabel('Petal width')
plt.show()
```




In [88]:
```
print(df.isnull().sum())
```

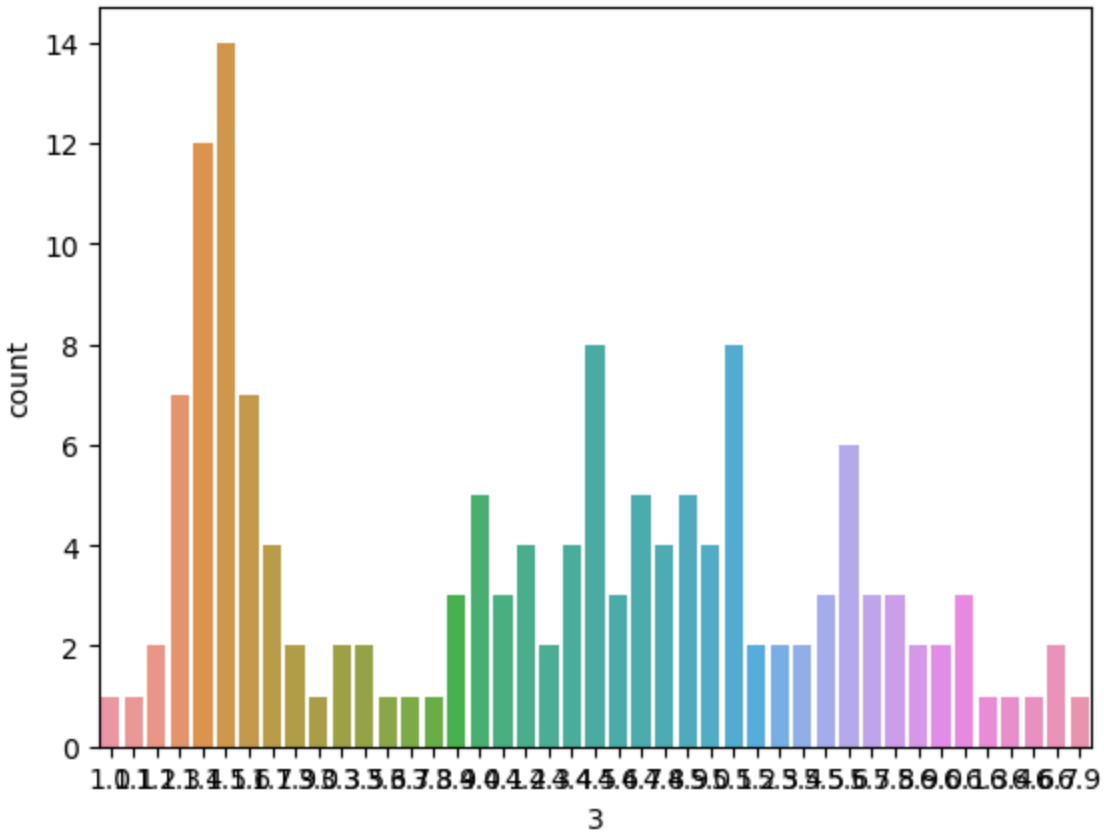
```
1    0
2    0
3    0
4    0
5    0
dtype: int64
```

In [89]:
```
df.describe()
```

Out[89]:

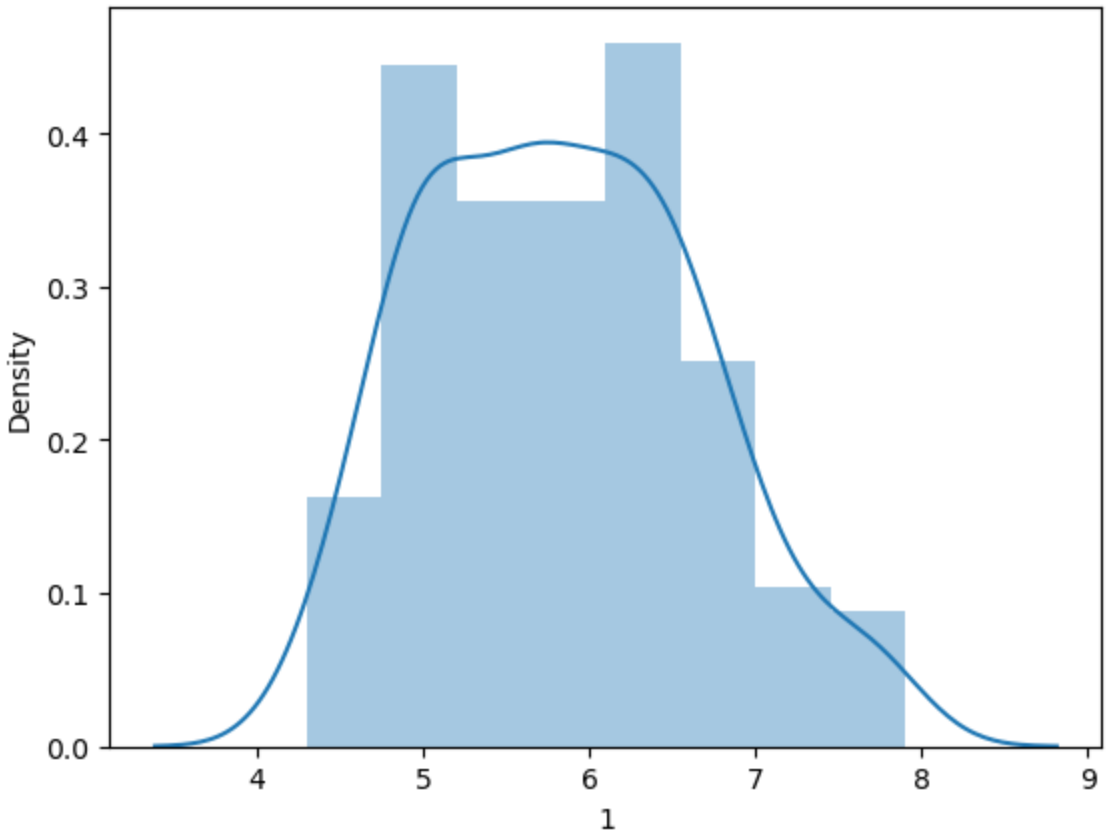
| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| **count** | 150.000000 | 150.000000 | 150.000000 | 150.000000 |
| **mean** | 5.843333 | 3.054000 | 3.758667 | 1.198667 |
| **std** | 0.828066 | 0.433594 | 1.764420 | 0.763161 |
| **min** | 4.300000 | 2.000000 | 1.000000 | 0.100000 |
| **25%** | 5.100000 | 2.800000 | 1.600000 | 0.300000 |
| **50%** | 5.800000 | 3.000000 | 4.350000 | 1.300000 |
| **75%** | 6.400000 | 3.300000 | 5.100000 | 1.800000 |
| **max** | 7.900000 | 4.400000 | 6.900000 | 2.500000 |

In [93]:
```
import seaborn as sns
sns.countplot(x='3',data=df)
#plt.xtricks(rotation=90)
plt.show()
```

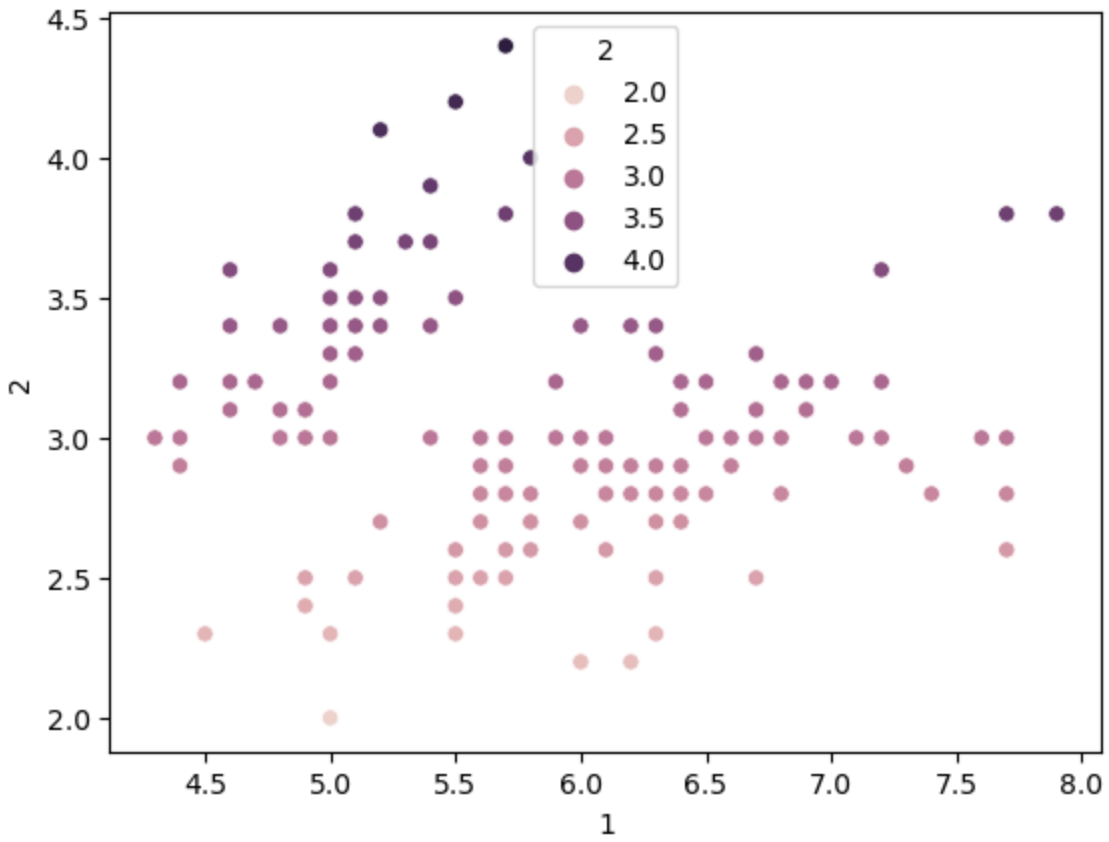


In [95]:
```
sns.distplot(df['1'])
plt.show()
```

```
C:\Users\NUTHAN SM\AppData\Local\Temp\ipykernel_6636\2008306444.py:1: UserWar
ning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gis
t.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(df['1'])
```



In [96]:
```
sns.scatterplot(x='1',y='2',hue='2',data=df)
plt.show()
```



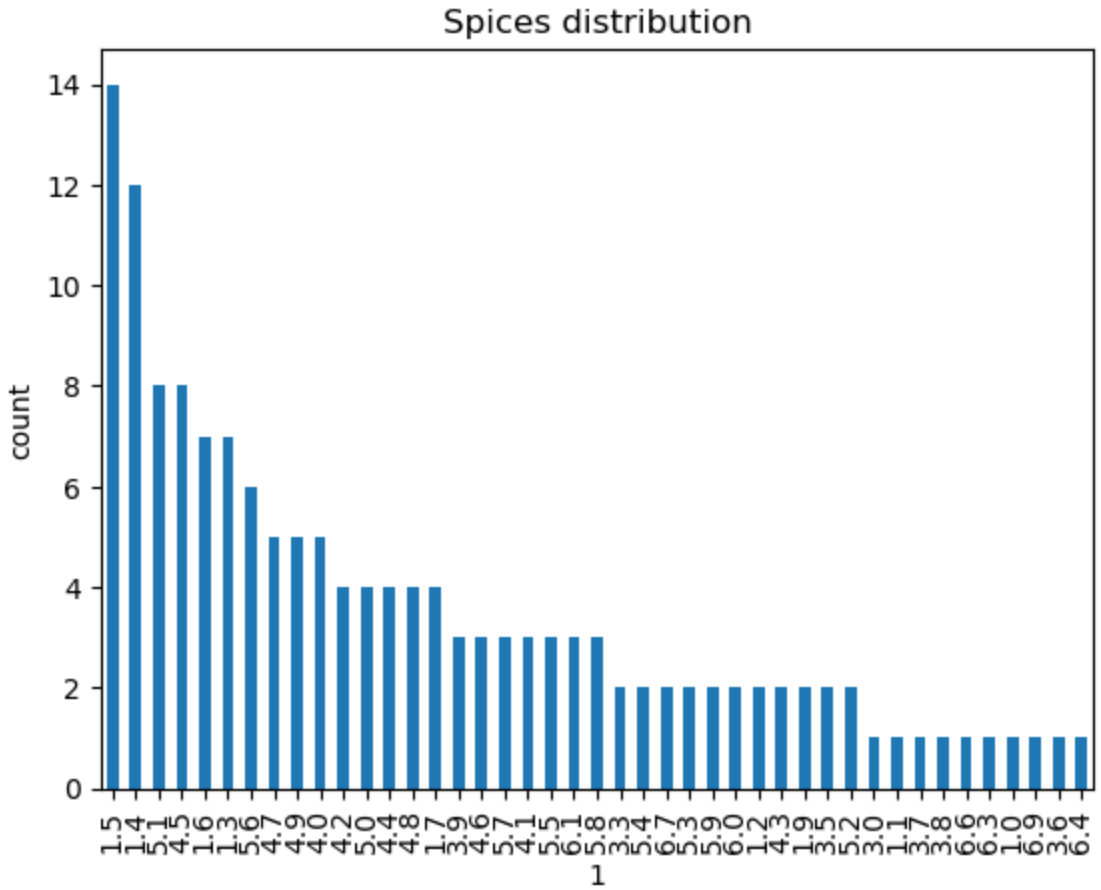

In [97]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   1       150 non-null    float64
 1   2       150 non-null    float64
 2   3       150 non-null    float64
 3   4       150 non-null    float64
 4   5       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [103]:
```
df.info
```

Out[103]:
```
<bound method DataFrame.info of        1    2    3    4               5
0    5.1  3.5  1.4  0.2     Iris-setosa
1    4.9  3.0  1.4  0.2     Iris-setosa
2    4.7  3.2  1.3  0.2     Iris-setosa
3    4.6  3.1  1.5  0.2     Iris-setosa
4    5.0  3.6  1.4  0.2     Iris-setosa
..   ...  ...  ...  ...             ...
145  6.7  3.0  5.2  2.3  Iris-virginica
146  6.3  2.5  5.0  1.9  Iris-virginica
147  6.5  3.0  5.2  2.0  Iris-virginica
148  6.2  3.4  5.4  2.3  Iris-virginica
149  5.9  3.0  5.1  1.8  Iris-virginica

[150 rows x 5 columns]>
```

In [108]:
```
df['3'].value_counts().plot(kind='bar')
plt.xlabel('1')
plt.ylabel('count')
plt.title('Spices distribution')
plt.show()
```



In [147]:
```
station={'number':['1','2','3','4','5'],'Pencil':['300','350','400','500','520
dfr=pd.DataFrame(station)
dfr
```

Out[147]:

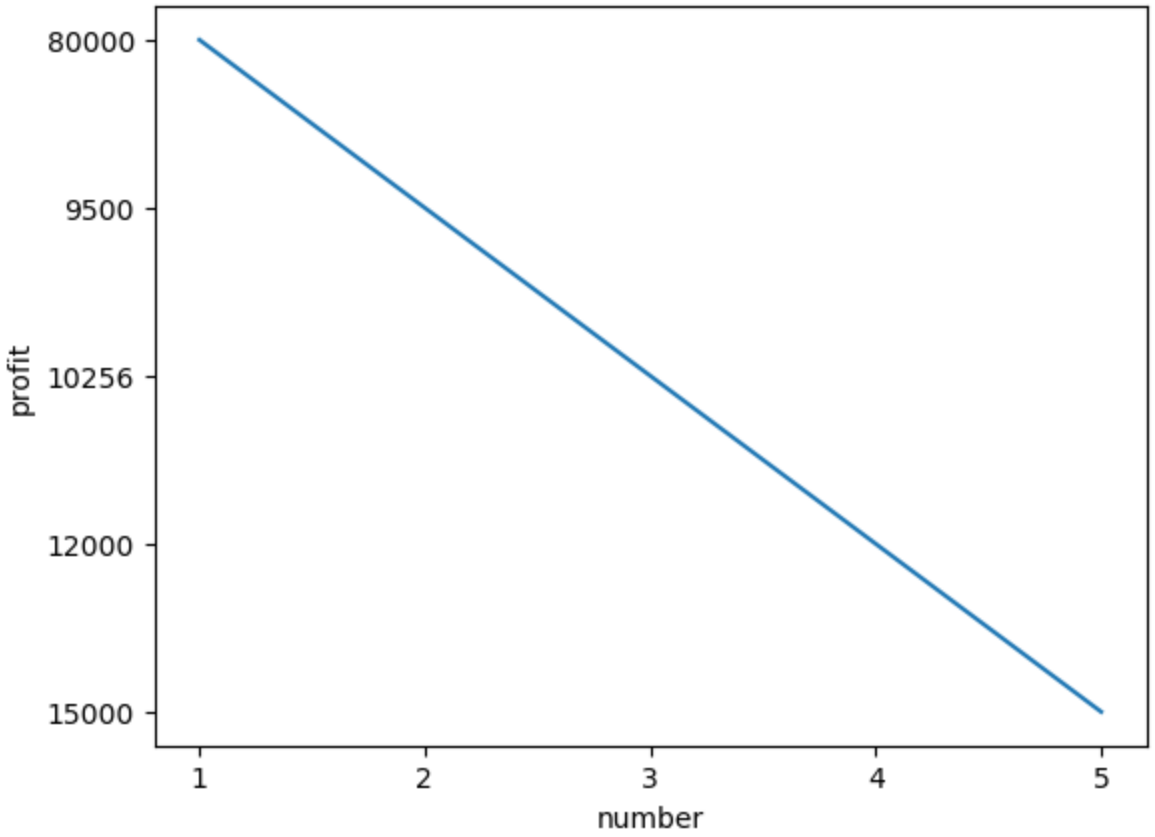
| | number | Pencil | textbooks | drawing sheets | Total units | profit |
|---|---|---|---|---|---|---|
| **0** | 1 | 300 | 250 | 100 | 700 | 80000 |
| **1** | 2 | 350 | 350 | 125 | 1075 | 9500 |
| **2** | 3 | 400 | 400 | 190 | 1320 | 10256 |
| **3** | 4 | 500 | 420 | 210 | 1510 | 12000 |
| **4** | 5 | 520 | 500 | 250 | None | 15000 |

In [148]:
```
dfr.describe()
```

Out[148]:

| | number | Pencil | textbooks | drawing sheets | Total units | profit |
|---|---|---|---|---|---|---|
| **count** | 5 | 5 | 5 | 5 | 5 | 5 |
| **unique** | 5 | 5 | 5 | 5 | 5 | 5 |
| **top** | 1 | 300 | 250 | 100 | 700 | 80000 |
| **freq** | 1 | 1 | 1 | 1 | 1 | 1 |

In [149]:
```
'''plt.plot(number,profit, marker='o',linestyle='-')
plt.title('line plot showing total profit on y axis and number column on x axi
sns.lineplot(x='number',y='profit',data=dfr)
plt.show()
```




In [150]:
```
tprofit=dfr['profit'].sum()
tprofit
```

Out[150]: '800009500102561200015000'

In [151]:
```
dfr['drawing sheets'].max()
```

Out[151]: '250'

In [155]:
```
meanimp=dfr.fillna(dfr.mean(),inplace=True)
meanimp
```

```
C:\Users\NUTHAN SM\AppData\Local\Temp\ipykernel_6636\643901086.py:1: FutureWa
rning: The default value of numeric_only in DataFrame.mean is deprecated. In
a future version, it will default to False. In addition, specifying 'numeric_
only=None' is deprecated. Select only valid columns or specify the value of n
umeric_only to silence this warning.
  meanimp=dfr.fillna(dfr.mean(),inplace=True)
```

In [ ]:
```
```