Wine Quality :- Lab cycle-1

1. Download the dataset winequality-red.csv file(each column is separated by a semicolon (;)) from the UCI Machine Learning Repository.

```
import numpy as np

import pandas as pd

arr = np.loadtxt(r"E:\Downloads\data-analysis\winequality-red.csv",delimiter=";", dtype=str,skiprows=1)

display(arr)
```

Output:

```
array([['7.4', '0.7', '0', ..., '0.56', '9.4', '5'],
       ['7.8', '0.88', '0', ..., '0.68', '9.8', '5'],
       ['7.8', '0.76', '0.04', ..., '0.65', '9.8', '5'],
       ...,
       ['5.9', '0.645', '0.12', ..., '0.71', '10.2', '5'],
       ['6', '0.31', '0.47', ..., '0.66', '11', '6']], dtype='<U16')
```

2. Convert it to numPy array, name it as wines (leave the first row of the list) and specify the data type of array as float.

```
wines=arr.astype('float64')

print(wines)
```

Output:

```
[[ 7.4    0.7    0.    ...  0.56   9.4    5.   ]
 [ 7.8    0.88   0.    ...  0.68   9.8    5.   ]
 [ 7.8    0.76   0.04  ...  0.65   9.8    5.   ]
 ...
 [ 6.3    0.51   0.13  ...  0.75  11.     6.   ]
 [ 5.9    0.645  0.12  ...  0.71  10.2    5.   ]
 [ 6.     0.31   0.47  ...  0.66  11.     6.   ]]
```

3. Identify the shape of the array.

```
np.shape(wines)
```

Output:

```
(1599, 12)
```

4. Display the element at row 3 and column 4.

```
wines[2,3]
```

Output:

```
2.3
```

In [48]:

5. Display the first three items from the fourth column.

```
wines[:3,3]
```

Ouput:

```
array([1.9, 2.6, 2.3])
```

6. Display third column from each row.

```
wines[:,2]
```

Output:

```
array([0.  , 0.  , 0.04, ..., 0.13, 0.12, 0.47])
```

In [21]:

7. Display fourth row.

```
wines[3,:]
```

Output:

```
array([11.2  ,  0.28 ,  0.56 ,  1.9  ,  0.075, 17.   , 60.   ,  0.998,
        3.16 ,  0.58 ,  9.8  ,  6.   ])
```

8. Assign value 10 to 2nd row and 6th column element.

```
wines[1,5] = 10

print(wines[1,5])
```

Output:

```
10.0
```

9. Take the 10th column from wines array and name that slice as slice_new and assign value 666 to all elements of slice_new

```
slice_new = wines[:,9]
slice_new[:] = 666
```

```
print(slice_new)
```

Output:

```
[666. 666. 666. ... 666. 666. 666.]
```

10. Display wines array.

```
print(wines)
```

Output:

```
[[7.40e+00 7.00e-01 0.00e+00 ... 6.66e+02 9.40e+00 5.00e+00]
 [7.80e+00 8.80e-01 0.00e+00 ... 6.66e+02 9.80e+00 5.00e+00]
 [7.80e+00 7.60e-01 4.00e-02 ... 6.66e+02 9.80e+00 5.00e+00]
 ...
 [6.30e+00 5.10e-01 1.30e-01 ... 6.66e+02 1.10e+01 6.00e+00]
 [5.90e+00 6.45e-01 1.20e-01 ... 6.66e+02 1.02e+01 5.00e+00]
 [6.00e+00 3.10e-01 4.70e-01 ... 6.66e+02 1.10e+01 6.00e+00]]
```

11. Find the data type of wines array and Change the data type to int.

```
print(wines.dtype)
wines = wines.astype('int')
print(wines.dtype)
print(wines)
```

Output:

```
float64
int32
[[  7   0   0 ... 666   9   5]
 [  7   0   0 ... 666   9   5]
 [  7   0   0 ... 666   9   5]
 ...
 [  6   0   0 ... 666  11   6]
 [  5   0   0 ... 666  10   5]
 [  6   0   0 ... 666  11   6]]
```

12. Add 10 points to each quality score.

```
wines[:-1] += 10
print(wines[:,-1])
```

Output:

```
[15 15 15 ... 16 15  6]
```

13. Find the sum of all the elements in an array

```
print(wines.sum())
print(sum(sum(wines)))
```

Output:

```
1402158
1402158
```

14. Find the sum of all the values in every column.

```
print(wines.sum(axis = 0))
```

Output:

```
[ 28569   16004   15981   19330   15980   41347   90281   16061   20750 10809
14   31949   24992]
```

15. Find the sum of all the values in every row.

```
print(wines.sum(axis = 1))
```

Output:

```
[856 889 881 ... 883 887 755]
```

16. Add the quality column to itself

```
wines[:,-1] += wines[:,-1]
print(wines[:,-1])
```

Output:

```
[30 30 30 ... 32 30 12]
```

17. Multiply alcohol by quality

```
print(wines[:,-2]* wines[:,-1])
```

Output:

```
[570 570 570 ... 672 600 132]
```

18. Display which wines have a quality rating higher than 5.

```
print(wines[wines[:,-1]>5])
```

Output:

```
[[ 17  10  10 ... 676  19  30]
 [ 17  10  10 ... 676  19  30]
 [ 17  10  10 ... 676  19  30]
 ...
 [ 16  10  10 ... 676  21  32]
 [ 15  10  10 ... 676  20  30]
 [  6   0   0 ... 666  11  12]]
```

19. Check if any wines have a quality rating equal to 10.

```
print(wines[wines[:,-1]==10])
```

Output:

```
[]
```

20. Select rows in wines where the quality is over 7

```
print(wines[wines[:,-1]>7])
```

Output:

```
[[ 17  10  10 ... 676  19  30]
```

```
 [ 17  10  10 ... 676  19  30]
 [ 17  10  10 ... 676  19  30]
 ...
 [ 16  10  10 ... 676  21  32]
 [ 15  10  10 ... 676  20  30]
 [  6   0   0 ... 666  11  12]]
```

21. Display wines with alcohol greater than 10 and quality greater than 7.

```
print(wines[(wines[:,-2]>10) & (wines[:,-1]>7)])
```

Output:

```
[[ 17  10  10 ... 676  19  30]
 [ 17  10  10 ... 676  19  30]
 [ 17  10  10 ... 676  19  30]
 ...
 [ 16  10  10 ... 676  21  32]
 [ 15  10  10 ... 676  20  30]
 [  6   0   0 ... 666  11  12]]
```

```
# 22. Change the shape of wines array.
wines = wines.reshape((533,36))
print(wines.shape)
```

Output:

```
(533, 36)
```

Iris dataset :- Lab cycle-2

1. Print the dataset iris.

```
import pandas as pd
file = pd.read_csv("E:\Downloads\data-analysis/iris.csv")
print(file)
```

Output:

```
     sepal_length  sepal_width  petal_length  petal_width        species
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]
```

2. Print the structure of the dataset iris.

```
print(file.info())
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
 5   Total         150 non-null    float64
dtypes: float64(5), object(1)
memory usage: 7.2+ KB
None
```

3. Print the summary of all the variables of the dataset iris.

print(file.describe())

Output:

```
  sepal_length  sepal_width  petal_length  petal_width       Total
count   150.000000   150.000000    150.000000   150.000000  150.000000
mean      5.843333     3.054000      3.758667     1.198667   13.854667
std       0.828066     0.433594      1.764420     0.763161    3.128149
min       4.300000     2.000000      1.000000     0.100000    8.400000
25%       5.100000     2.800000      1.600000     0.300000   10.700000
50%       5.800000     3.000000      4.350000     1.300000   14.300000
75%       6.400000     3.300000      5.100000     1.800000   16.250000
max       7.900000     4.400000      6.900000     2.500000   20.400000
```

4. How many of the variables (columns) are in the dataset iris.

len(file.keys())

Output:

```
6
```

5. How many observations (rows) are in the dataset iris.

len(file)

Output:

```
150
```

6. Use duplicated() function to print the logical vector indicating the duplicate values present in the dataset iris.

file.duplicated()

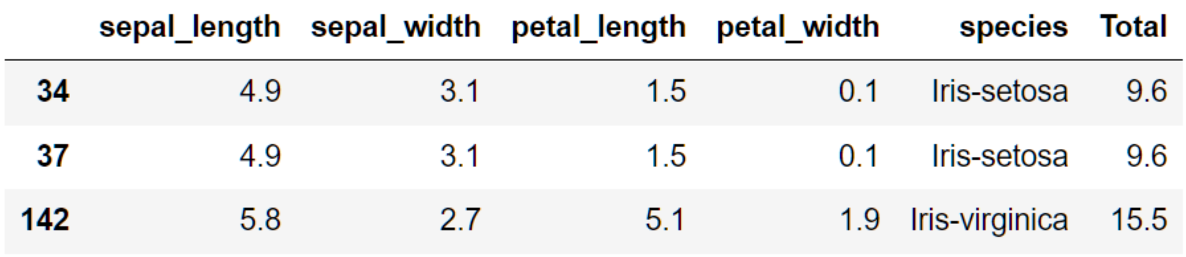
Output:

```
0      False
1      False
2      False
3      False
4      False
       ...
145    False
146    False
147    False
148    False
149    False
Length: 150, dtype: bool
```

7. Extract duplicate elements from the dataset iris.

```
file[file.duplicated()]
```

Output:

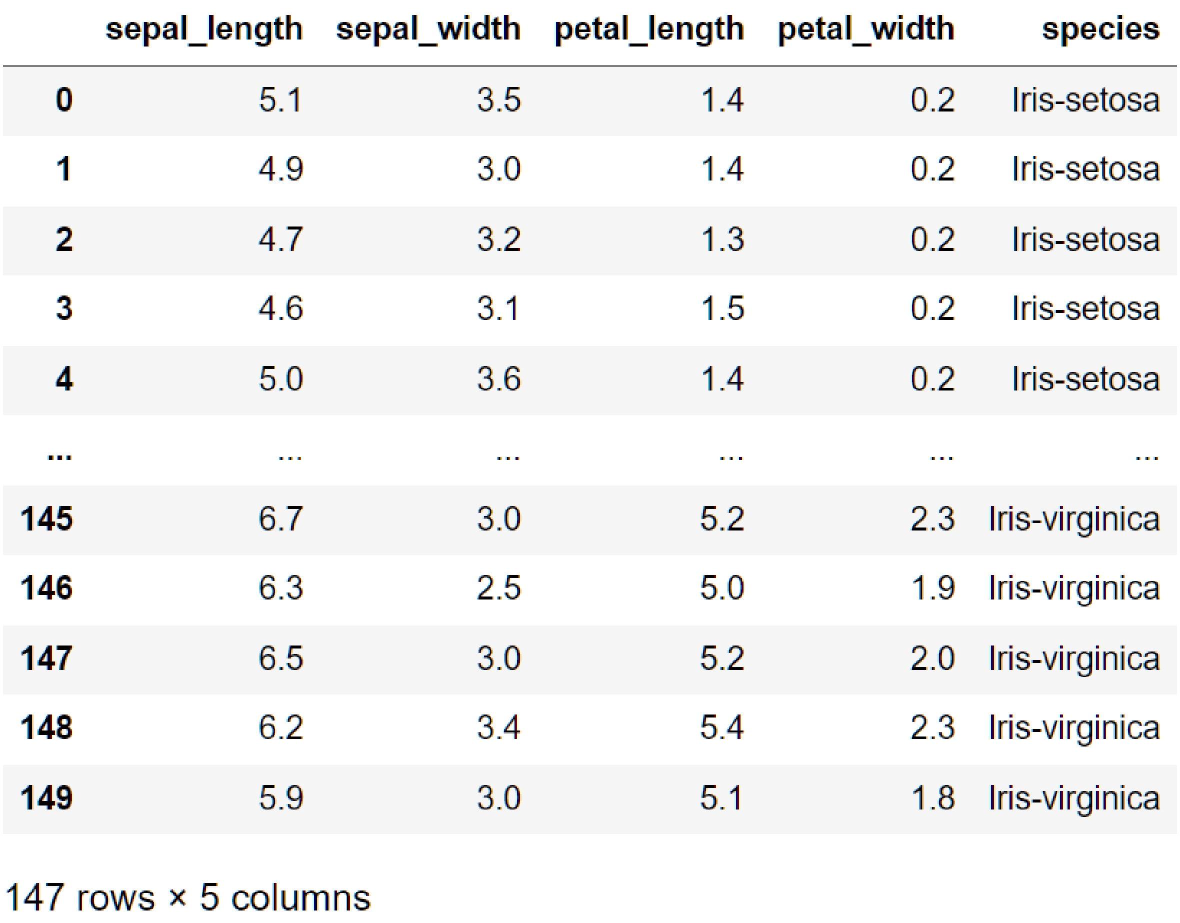


	sepal_length	sepal_width	petal_length	petal_width	species	Total
34	4.9	3.1	1.5	0.1	Iris-setosa	9.6
37	4.9	3.1	1.5	0.1	Iris-setosa	9.6
142	5.8	2.7	5.1	1.9	Iris-virginica	15.5

8. Extract unique elements from the dataset iris.

```
file.drop_duplicates()
```

Output:



	sepal_length	sepal_width	petal_length	petal_width	species
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

147 rows × 5 columns

9. Print the indices of duplicate elements in the dataset iris.

```
print(file[file.duplicated()].index.tolist())
```

Output:

```
[34, 37, 142]
```

10. Print the indices of unique elements in the dataset iris.

```
print(file.drop_duplicates().index.tolist())
```

Output:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 2
4, 25, 26, 27, 28, 29, 30, 31, 32, 33, 35, 36, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47
, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68,
69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 9
0, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108
, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124
, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140
, 141, 143, 144, 145, 146, 147, 148, 149]
```

11.How many unique elements are in the dataset iris.

```
len(file.drop_duplicates())
```

Output:

```
147
```

12.How many duplicate elements are in the dataset iris.

```
len(file[file.duplicated()])
```

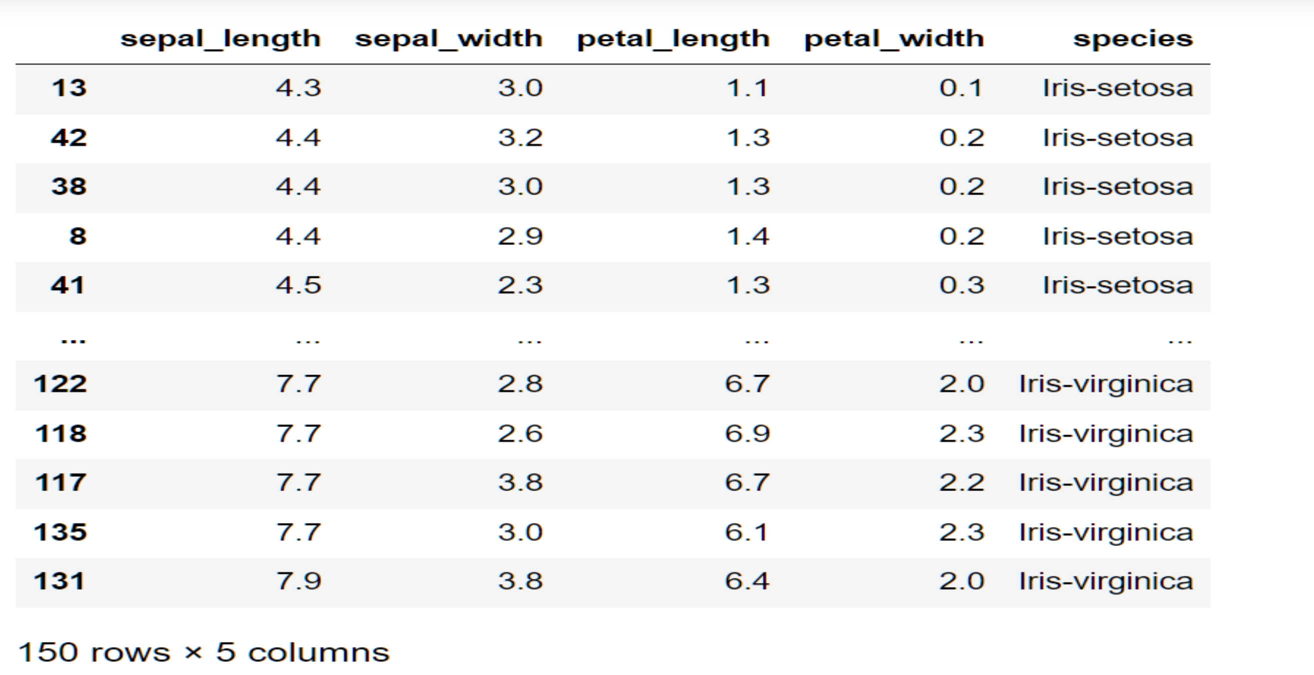
Output:

```
3
```

13.Print the sorted elements in the dataset iris(Ascending order).

```
file.sort_values('sepal_length', axis=0)
```

Output:



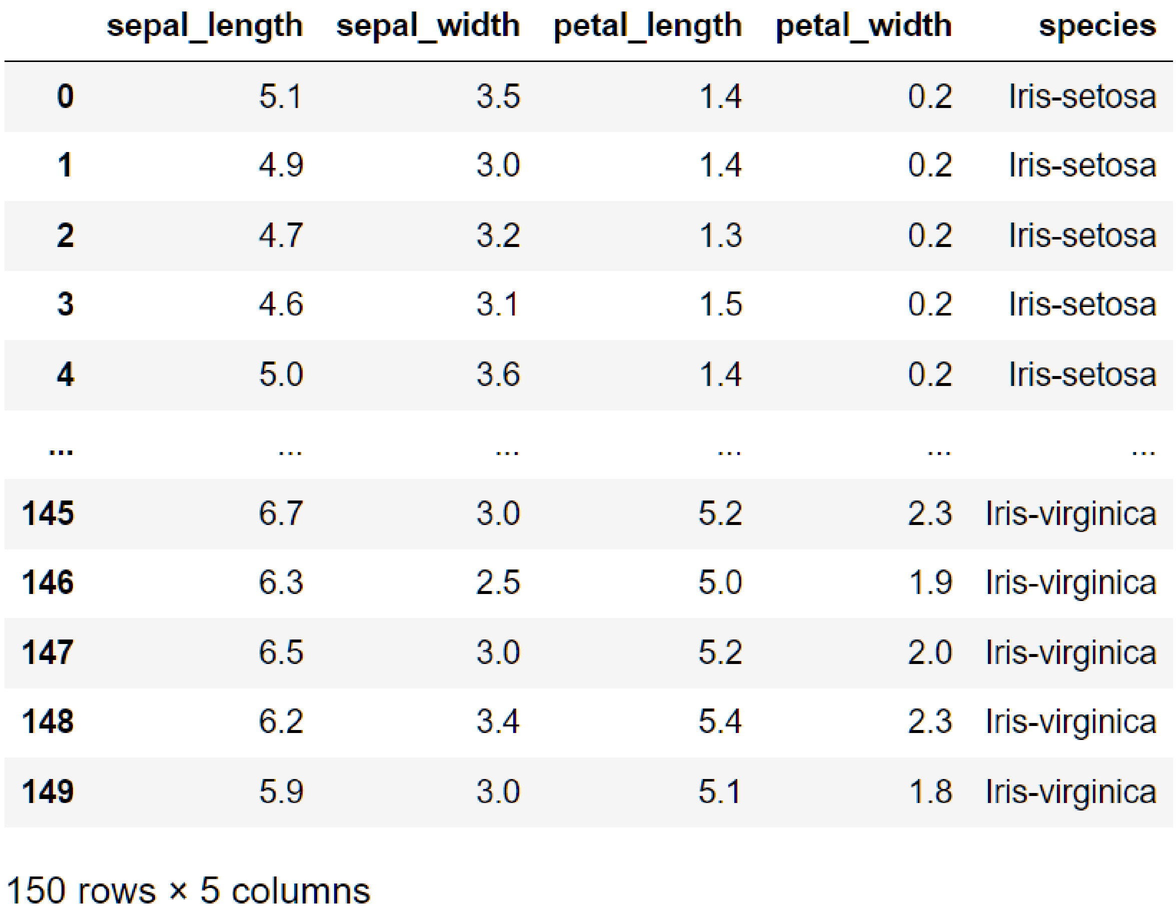
	sepal_length	sepal_width	petal_length	petal_width	species
13	4.3	3.0	1.1	0.1	Iris-setosa
42	4.4	3.2	1.3	0.2	Iris-setosa
38	4.4	3.0	1.3	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
41	4.5	2.3	1.3	0.3	Iris-setosa
...	...	...	...	...	...
122	7.7	2.8	6.7	2.0	Iris-virginica
118	7.7	2.6	6.9	2.3	Iris-virginica
117	7.7	3.8	6.7	2.2	Iris-virginica
135	7.7	3.0	6.1	2.3	Iris-virginica
131	7.9	3.8	6.4	2.0	Iris-virginica

150 rows × 5 columns

14.Find whether any missing values are in the dataset iris.

file.dropna()

Output:



	sepal_length	sepal_width	petal_length	petal_width	species
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 × 5 columns

15.Display how many missing values are present in each column.

file.isnull().sum()

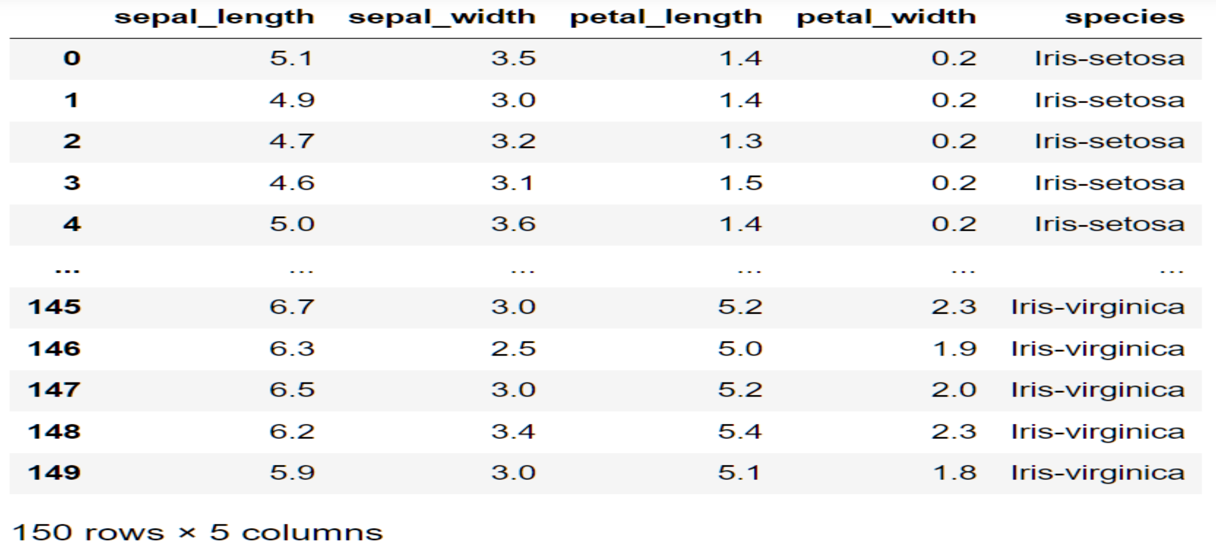
Output:

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

16.Replace all missing values with zero.

file.fillna(0)

Output:



	sepal_length	sepal_width	petal_length	petal_width	species
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 × 5 columns

17.Calculate Petal width mean ,median ,SD,Variance for the species setosa.

```
print(file[file['species']=='Iris-setosa']['petal_width'].mean())
print(file[file['species']=='Iris-setosa']['petal_width'].std())
print(file[file['species']=='Iris-setosa']['petal_width'].var())
```

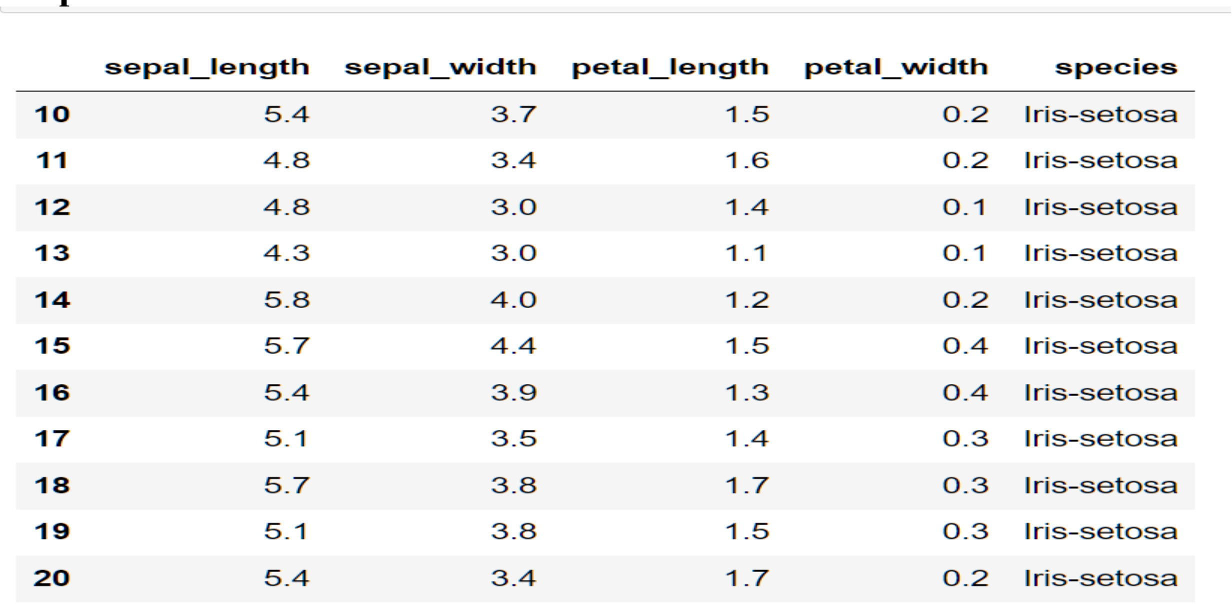
Output:

```
0.244
0.10720950308167840
0.011493877551020411
```

18.Print from 10th row to 20th row of iris dataset.

file[10:21]

Output:

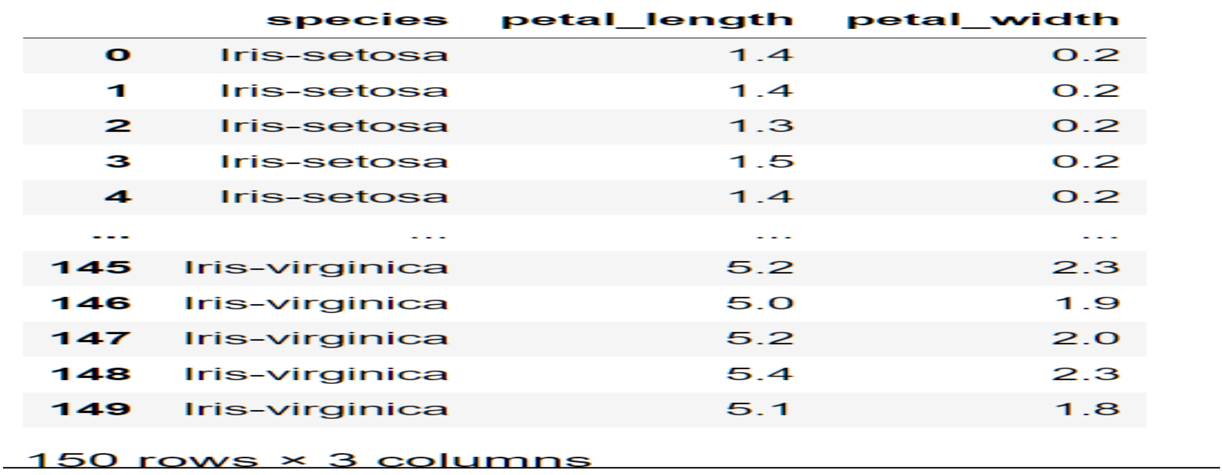


	sepal_length	sepal_width	petal_length	petal_width	species
10	5.4	3.7	1.5	0.2	Iris-setosa
11	4.8	3.4	1.6	0.2	Iris-setosa
12	4.8	3.0	1.4	0.1	Iris-setosa
13	4.3	3.0	1.1	0.1	Iris-setosa
14	5.8	4.0	1.2	0.2	Iris-setosa
15	5.7	4.4	1.5	0.4	Iris-setosa
16	5.4	3.9	1.3	0.4	Iris-setosa
17	5.1	3.5	1.4	0.3	Iris-setosa
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

19.Print Species and its corresponding Petal length and Width.

file[['species', 'petal_length', 'petal_width']]

Output:



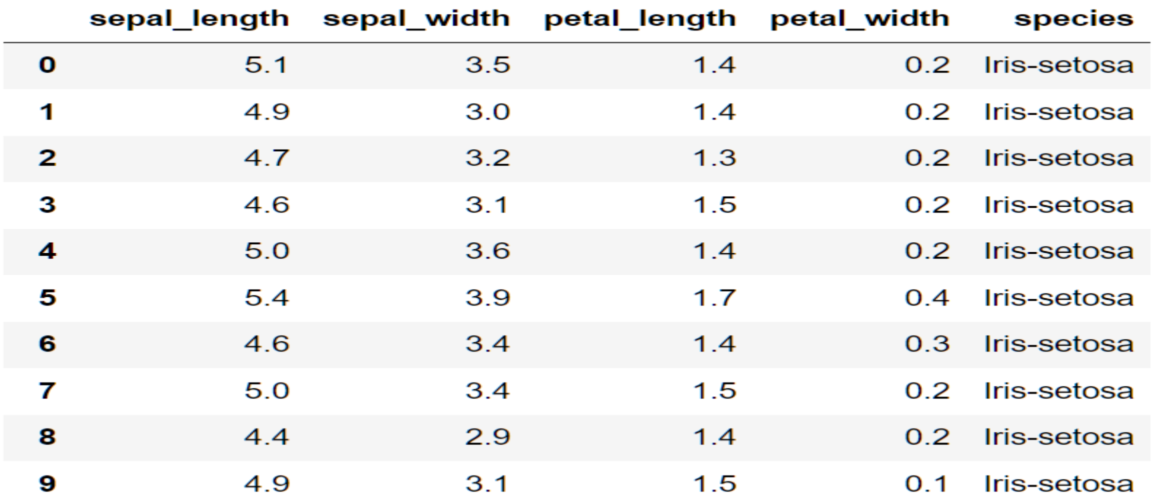
	species	petal_length	petal_width
0	Iris-setosa	1.4	0.2
1	Iris-setosa	1.4	0.2
2	Iris-setosa	1.3	0.2
3	Iris-setosa	1.5	0.2
4	Iris-setosa	1.4	0.2
...	...	...	...
145	Iris-virginica	5.2	2.3
146	Iris-virginica	5.0	1.9
147	Iris-virginica	5.2	2.0
148	Iris-virginica	5.4	2.3
149	Iris-virginica	5.1	1.8

150 rows × 3 columns

20.Display records only with species "Iris-setosa".

file[file['species']=="Iris-setosa"]

Output:



	sepal_length	sepal_width	petal_length	petal_width	species
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

21.Count number of times a particular species has occurred.

file['species'].value_counts()

Output:

```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: species, dtype: int64
```

22.Identifying minimum and maximum Value of Sepal width.

```
print(min(file['sepal_width']))
print(max(file['sepal_width']))
```

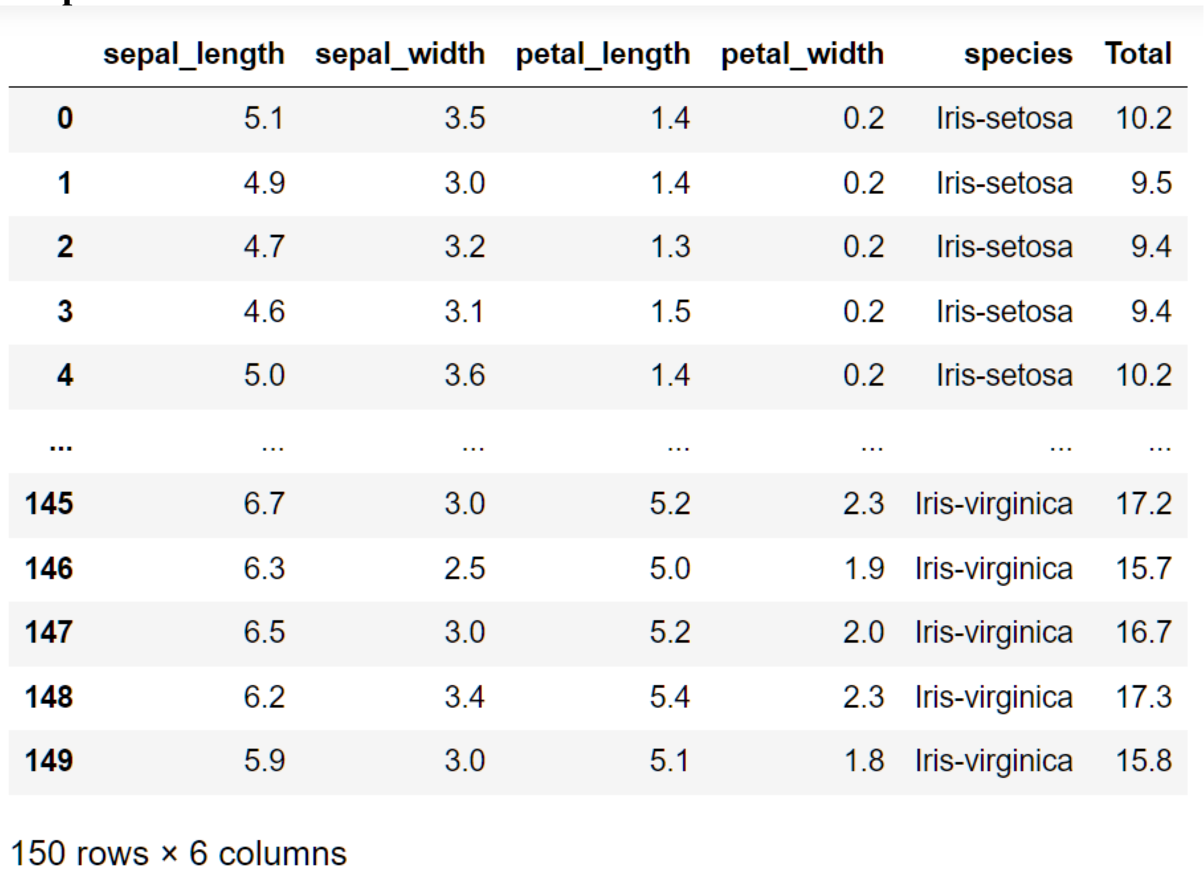
Output:

```
2.0
4.4
```

23.Add new column to store sum of first four column values

```
file['Total']=file[file.columns[0:4]].sum(axis=1)
print(file)
```

Output:



	sepal_length	sepal_width	petal_length	petal_width	species	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

Universities Ranking dataset :- Lab cycle-3

1. Find the universities in which undergraduate students were admitted.

```
import pandas as pd
file = pd.read_csv("E:\Downloads\data-
analysis/National_Universities_Rankings.csv")
file
```

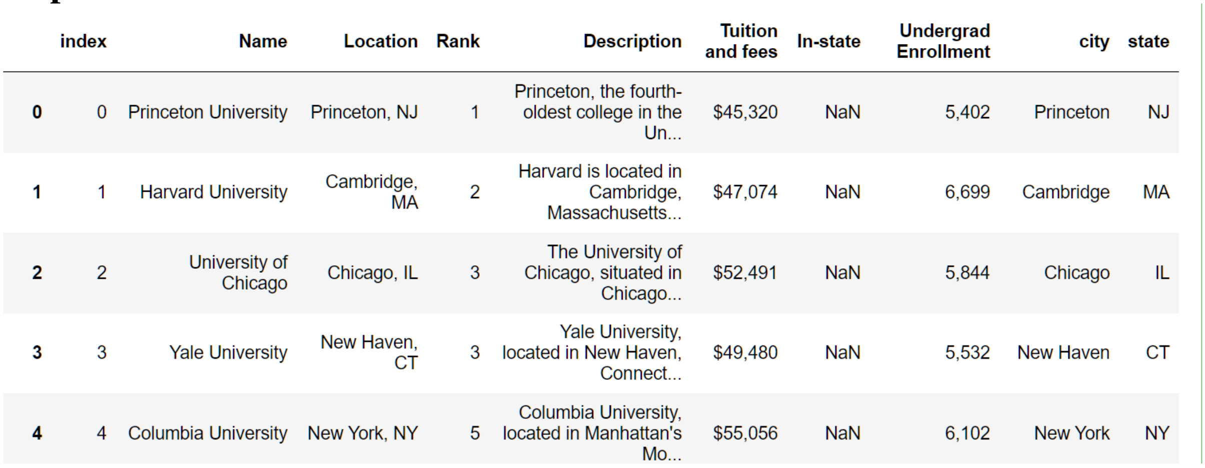
Output:

```
file['Name'][file['Undergrad Enrollment'].notnull()]
0                        Princeton University
1                          Harvard University
2                        University of Chicago
3                            Yale University
4                          Columbia University
                         ...
226     University of Massachusetts--Dartmouth
227          University of Missouri--St. Louis
228    University of North Carolina--Greensboro
229         University of Southern Mississippi
230                       Utah State University
Name: Name, Length: 231, dtype: object
```

2. List the states along with the cities in which universities located.

```
a = file['Location'].str.split(', ', expand=True)
file['city']=a[0]
file['state']=a[1]
file
```

Ouput:



	index	Name	Location	Rank	Description	Tuition and fees	In-state	Undergrad Enrollment	city	state
0	0	Princeton University	Princeton, NJ	1	Princeton, the fourth-oldest college in the Un...	$45,320	NaN	5,402	Princeton	NJ
1	1	Harvard University	Cambridge, MA	2	Harvard is located in Cambridge, Massachusetts...	$47,074	NaN	6,699	Cambridge	MA
2	2	University of Chicago	Chicago, IL	3	The University of Chicago, situated in Chicago...	$52,491	NaN	5,844	Chicago	IL
3	3	Yale University	New Haven, CT	3	Yale University, located in New Haven, Connect...	$49,480	NaN	5,532	New Haven	CT
4	4	Columbia University	New York, NY	5	Columbia University, located in Manhattan's Mo...	$55,056	NaN	6,102	New York	NY

3. List the cities &universities under each state.

```
list(file.groupby(['state', 'city', 'Name']).groups.keys())
```

Output:

```
[('AK', 'Fairbanks', 'University of Alaska--Fairbanks'),
 ('AL', 'Auburn', 'Auburn University'),
 ('AL', 'Birmingham', 'University of Alabama--Birmingham'),
 ('AL', 'Huntsville', 'University of Alabama--Huntsville'),
 ('AL', 'Tuscaloosa', 'University of Alabama'),
 ('AR', 'Fayetteville', 'University of Arkansas'),
 ('AZ', 'Tempe', 'Arizona State University--Tempe'),
 ('AZ', 'Tucson', 'University of Arizona'),
 ('CA', 'Azusa', 'Azusa Pacific University'),
 ('CA', 'Berkeley', 'University of California--Berkeley'),
….
('RI', 14, 'Brown University'),
 ('TN', 15, 'Vanderbilt University'),
 ('TX', 15, 'Rice University'),
 ('VA', 24, 'University of Virginia'),
 ('VA', 32, 'College of William & Mary'),
 ('WI', 44, 'University of Wisconsin--Madison')]
```

5. How many universities have both out-of state and in-state students?

```
len(file[(file['Tuition and fees'].notnull()) & (file['In-state'].notnull())])
```

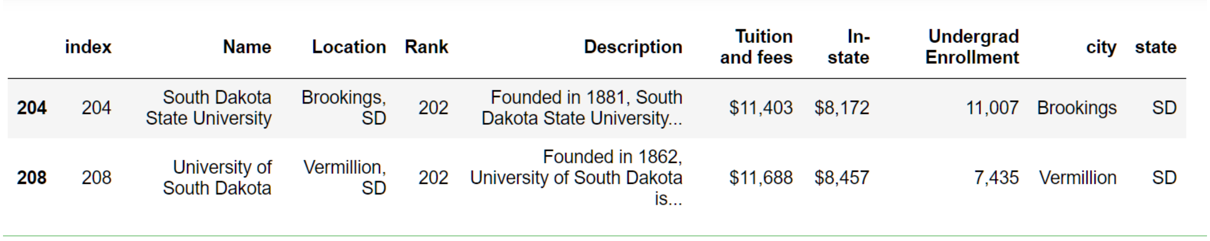
Ouput:

```
133
```

6. How many universities have marginal difference <=$5000 in in-state &out-of state tuition fees.

```
fee_Tu = file['Tuition and fees'].str.replace('\W', '' ,regex=True)
fee_Tu = fee_Tu.astype(int)
fee_In = file['In-state'].fillna('0')
fee_In = fee_In.str.replace('\W', '' ,regex=True)
fee_In = fee_In.astype(int)
file[abs(fee_Tu-fee_In)<=5000]
```

Output:

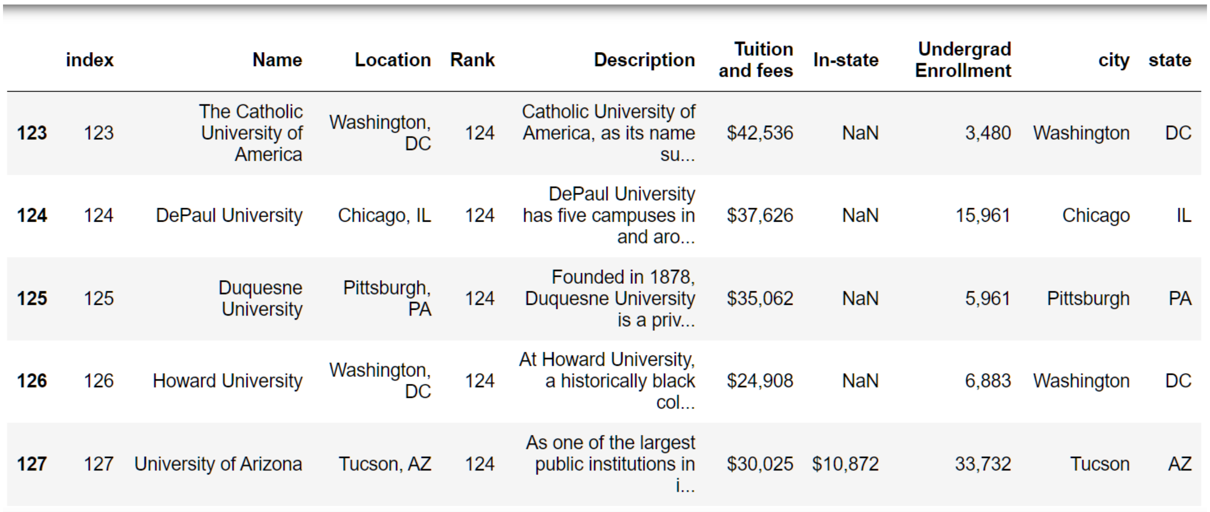


	index	Name	Location	Rank	Description	Tuition and fees	In-state	Undergrad Enrollment	city	state
204	204	South Dakota State University	Brookings, SD	202	Founded in 1881, South Dakota State University...	$11,403	$8,172	11,007	Brookings	SD
208	208	University of South Dakota	Vermillion, SD	202	Founded in 1862, University of South Dakota is...	$11,688	$8,457	7,435	Vermillion	SD

7. List the universities having tuition fee>15000$ and rank between 120 to 170.

```
file[(file['Tuition and fees'].str.replace('\W', '' ,regex=True).astype
(int)>15000)&(file['Rank']>120)& (file['Rank']<170)]
```

Output:

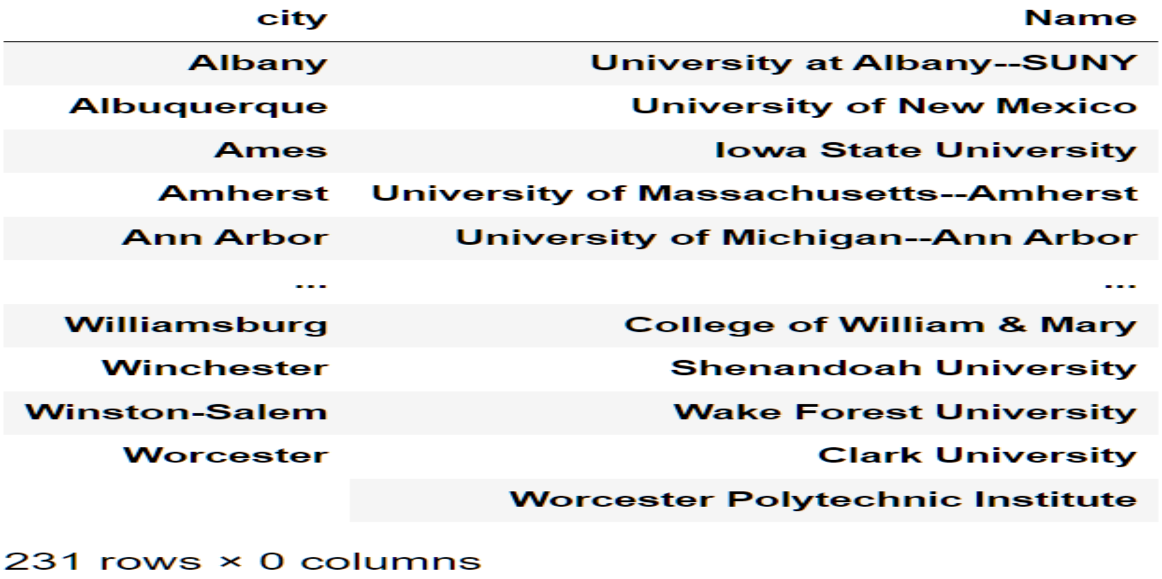


	index	Name	Location	Rank	Description	Tuition and fees	In-state	Undergrad Enrollment	city	state
123	123	The Catholic University of America	Washington, DC	124	Catholic University of America, as its name su...	$42,536	NaN	3,480	Washington	DC
124	124	DePaul University	Chicago, IL	124	DePaul University has five campuses in and aro...	$37,626	NaN	15,961	Chicago	IL
125	125	Duquesne University	Pittsburgh, PA	124	Founded in 1878, Duquesne University is a priv...	$35,062	NaN	5,961	Pittsburgh	PA
126	126	Howard University	Washington, DC	124	At Howard University, a historically black col...	$24,908	NaN	6,883	Washington	DC
127	127	University of Arizona	Tucson, AZ	124	As one of the largest public institutions in i...	$30,025	$10,872	33,732	Tucson	AZ

8. Find the campuses of universities located in different cities (multiple cities).

```
file[['city', 'Name']].groupby(['city', 'Name']).first()
```

Output:



city	Name
Albany	University at Albany--SUNY
Albuquerque	University of New Mexico
Ames	Iowa State University
Amherst	University of Massachusetts--Amherst
Ann Arbor	University of Michigan--Ann Arbor
...	...
Williamsburg	College of William & Mary
Winchester	Shenandoah University
Winston-Salem	Wake Forest University
Worcester	Clark University
	Worcester Polytechnic Institute

231 rows × 0 columns

9. Mention the states where out-of state fee is more than in-state students. Print Minimum and Maximum fees.

```
ft = file['Tuition and fees'].str.replace('\W', '',regex=True).astype(int)
fi = file['In-state'].fillna('0').str.replace('\W', '' ,regex=True).astype(int)
print("Min and max of Tuition and fees: " ,min(ft), max(ft))
print("Min and max of In-state fee: ", min(fi), max(fi))
```

```
file[ft>fi].drop_duplicates(['state'])['state']
```

Output:

```
Min and max of Tuition and fees:  5300 55056
Min and max of In-state fee:  0 18687
0      NJ
1      MA
2      IL
3      CT
4      NY
5      CA
….
200    NV
204    SD
205    AK
210    MT
Name: state, dtype: object
```

10.Find the cities locating top 100 universities.

```
file[file['Rank']<=100].drop_duplicates(['city'])['city']
```

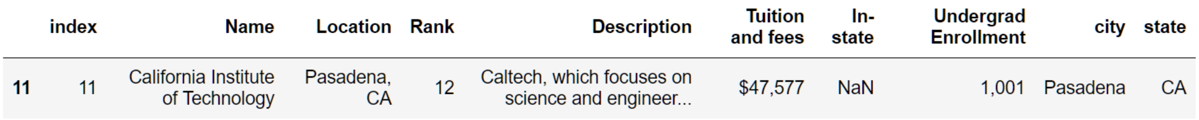
Output:

```
0       Princeton
1       Cambridge
2         Chicago
          ...
97     Stony Brook
98          Auburn
101         Buffalo
Name: city, Length: 87, dtype: object
```

11.Find universities with least no of undergraduate students.

```
file[file['Undergrad Enrollment']==file['Undergrad Enrollment'].min()]
```

Output:



	index	Name	Location	Rank	Description	Tuition and fees	In-state	Undergrad Enrollment	city	state
11	11	California Institute of Technology	Pasadena, CA	12	Caltech, which focuses on science and engineer...	$47,577	NaN	1,001	Pasadena	CA

12.Identifying correlations between enrollment numbers and university rank.

```
file['Rank'].corr(file['Undergrad Enrollment'].apply(lambda x:int(x.replace(',',
""))))
```

Output:

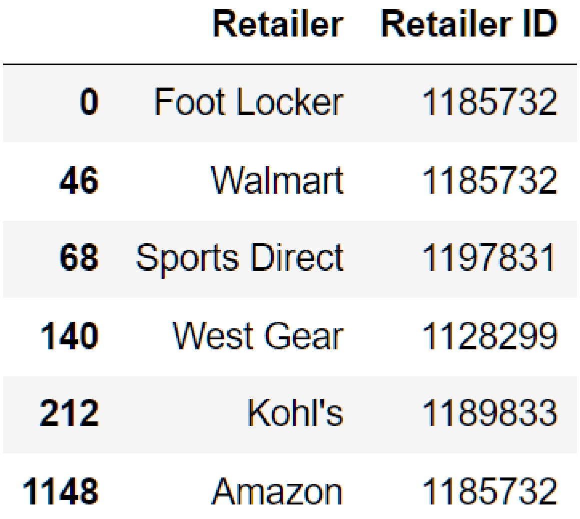
```
-0.040770935787747827
```

Adidas Sales dataset :- Lab cycle-4

1. List all the retailers with retailer id.

```
import pandas as pd
file = pd.read_excel("E:\Downloads\data-analysis/Adidas_US_Sales_Datasets_1_.xlsx")
file.drop_duplicates(['Retailer'])[['Retailer', 'Retailer ID']]
```

Output:



	Retailer	Retailer ID
0	Foot Locker	1185732
46	Walmart	1185732
68	Sports Direct	1197831
140	West Gear	1128299
212	Kohl's	1189833
1148	Amazon	1185732

2. List all the retailers in every region.

```
list(file.groupby(['Region', 'Retailer']).groups.keys())
```

Output:

```
[('Midwest', 'Amazon'),
 ('Midwest', 'Foot Locker'),
 ('Midwest', "Kohl's"),
 ('Midwest', 'Sports Direct'),
 ('Midwest', 'West Gear'),
 ('Northeast', 'Amazon'),
 ….
 ('West', 'Amazon'),
 ('West', 'Foot Locker'),
 ('West', "Kohl's"),
 ('West', 'Sports Direct'),
 ('West', 'Walmart'),
 ('West', 'West Gear')]
```

3. List the retailers in every city of a state.

```
list(file.groupby(['State', 'City', 'Retailer']).groups.keys())
```

Output:

```
[('Alabama', 'Birmingham', 'Amazon'),
 ('Alabama', 'Birmingham', 'Sports Direct'),
 ('Alaska', 'Anchorage', 'Amazon'),
 ('Alaska', 'Anchorage', 'Foot Locker'),
…
('Wilmington', "Women's Apparel"),
 ('Wilmington', "Women's Athletic Footwear"),
 ('Wilmington', "Women's Street Footwear")]
```

5. Find the total sales of every retailer.

```
file.groupby('Retailer')['Total Sales'].sum()
```

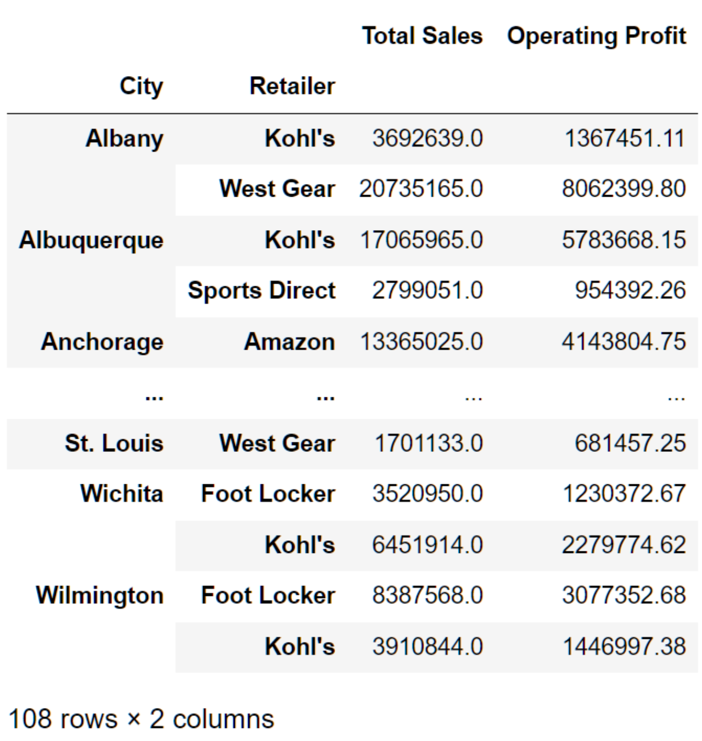
Output:

```
Retailer
Amazon          77698912.0
Foot Locker     220094720.0
Kohl's          102114753.0
Sports Direct   182470997.0
Walmart         74558410.0
West Gear       242964333.0
Name: Total Sales, dtype: float64
```

6. Find the total sales of the retailers in every city along with profit.

```
file.groupby(['City', 'Retailer'])[['Total Sales', 'Operating Profit']].sum()
```

Output:



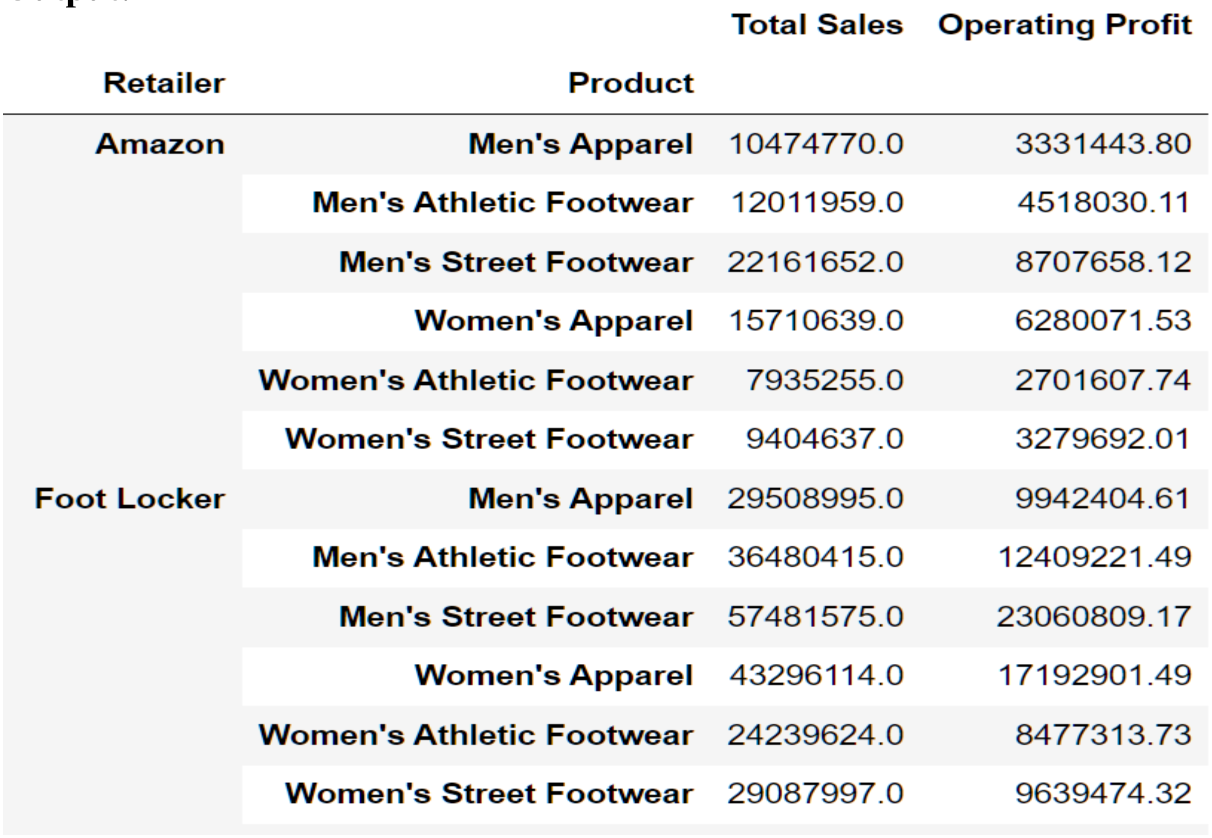
City	Retailer	Total Sales	Operating Profit
Albany	Kohl's	3692639.0	1367451.11
	West Gear	20735165.0	8062399.80
Albuquerque	Kohl's	17065965.0	5783668.15
	Sports Direct	2799051.0	954392.26
Anchorage	Amazon	13365025.0	4143804.75
...	...	...	...
St. Louis	West Gear	1701133.0	681457.25
Wichita	Foot Locker	3520950.0	1230372.67
	Kohl's	6451914.0	2279774.62
Wilmington	Foot Locker	8387568.0	3077352.68
	Kohl's	3910844.0	1446997.38

108 rows × 2 columns

7. Find the total sales &profit of each product sold by the retailer.

file.groupby(['Retailer', 'Product'])[['Total Sales', 'Operating Profit']].sum()

Output:



Retailer	Product	Total Sales	Operating Profit
Amazon	Men's Apparel	10474770.0	3331443.80
	Men's Athletic Footwear	12011959.0	4518030.11
	Men's Street Footwear	22161652.0	8707658.12
	Women's Apparel	15710639.0	6280071.53
	Women's Athletic Footwear	7935255.0	2701607.74
	Women's Street Footwear	9404637.0	3279692.01
Foot Locker	Men's Apparel	29508995.0	9942404.61
	Men's Athletic Footwear	36480415.0	12409221.49
	Men's Street Footwear	57481575.0	23060809.17
	Women's Apparel	43296114.0	17192901.49
	Women's Athletic Footwear	24239624.0	8477313.73
	Women's Street Footwear	29087997.0	9639474.32

8. Find the units sold, total sales &profit of the products sold between the dates 1/1/2020 and 4/15/2020.

```
file[(file['Invoice Date']>'1/1/2020') & (file['Invoice Date']<'4/15/2020')][['Units Sold', 'Total Sales', 'Operating Profit']].sum()
```

Output:

```
Units Sold          137483.00
Total Sales       51549291.00
Operating Profit    17815082.37
dtype: float64
```

9. Find the no of units sold of each product by each retailer in every city.

```
file.groupby(['City', 'Retailer', 'Product'])['Units Sold'].sum()
```

Output:

```
City        Retailer  Product
Albany      Kohl's    Men's Apparel                1375
                      Men's Athletic Footwear      1401
                      Men's Street Footwear        2104
                      Women's Apparel              1613
                      Women's Athletic Footwear    1311
                                                   ...
Wilmington  Kohl's    Men's Athletic Footwear      1490
                      Men's Street Footwear        2638
                      Women's Apparel              1743
                      Women's Athletic Footwear     970
                      Women's Street Footwear      1525
Name: Units Sold, Length: 618, dtype: int64
```

10. Find the products with different price per unit in different cities with proper information.

```
list(file.groupby(['Product', 'City', 'Price per Unit']).groups.keys())
```

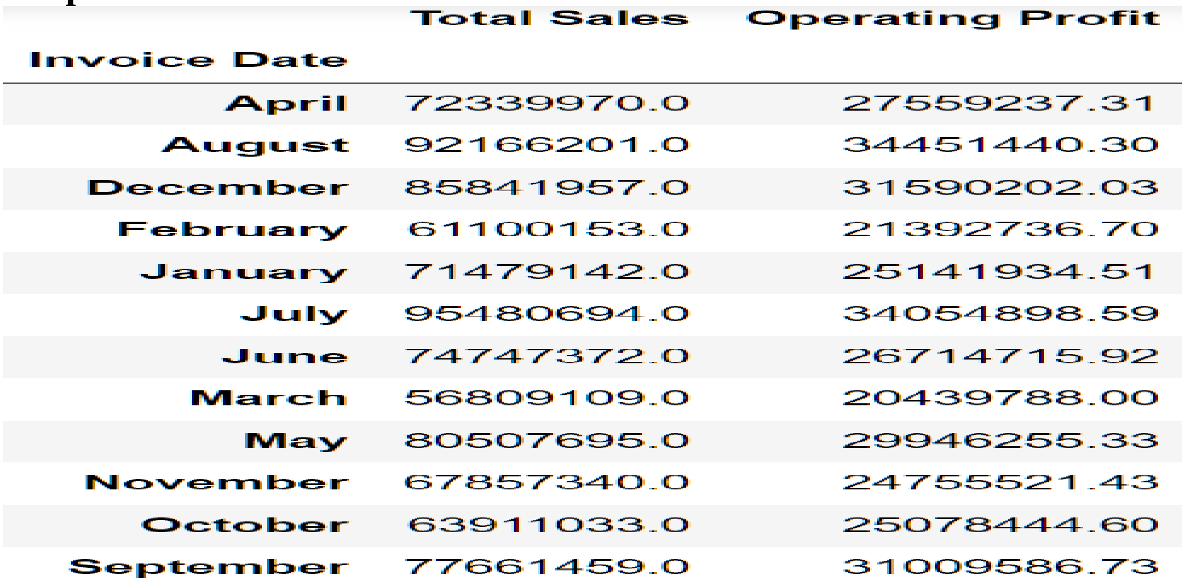
Output:

```
[("Men's Apparel", 'Albany', 49.0),
 ("Men's Apparel", 'Albany', 50.0),
 ("Men's Apparel", 'Albany', 51.0),
 ("Men's Apparel", 'Albany', 52.0),
 ("Men's Apparel", 'Albany', 55.00000000000001),
 ("Men's Apparel", 'Albany', 56.0),
 ("Men's Apparel", 'Albany', 59.0),
 ("Men's Apparel", 'Albany', 60.0),
 ("Men's Apparel", 'Albany', 60.00000000000001),
……..]
```

11. Find the total sales &profits of all products in every month.

file.groupby(file['Invoice Date'].dt.strftime('%B'))[['Total Sales', 'Operating Profit']].sum()

Output:

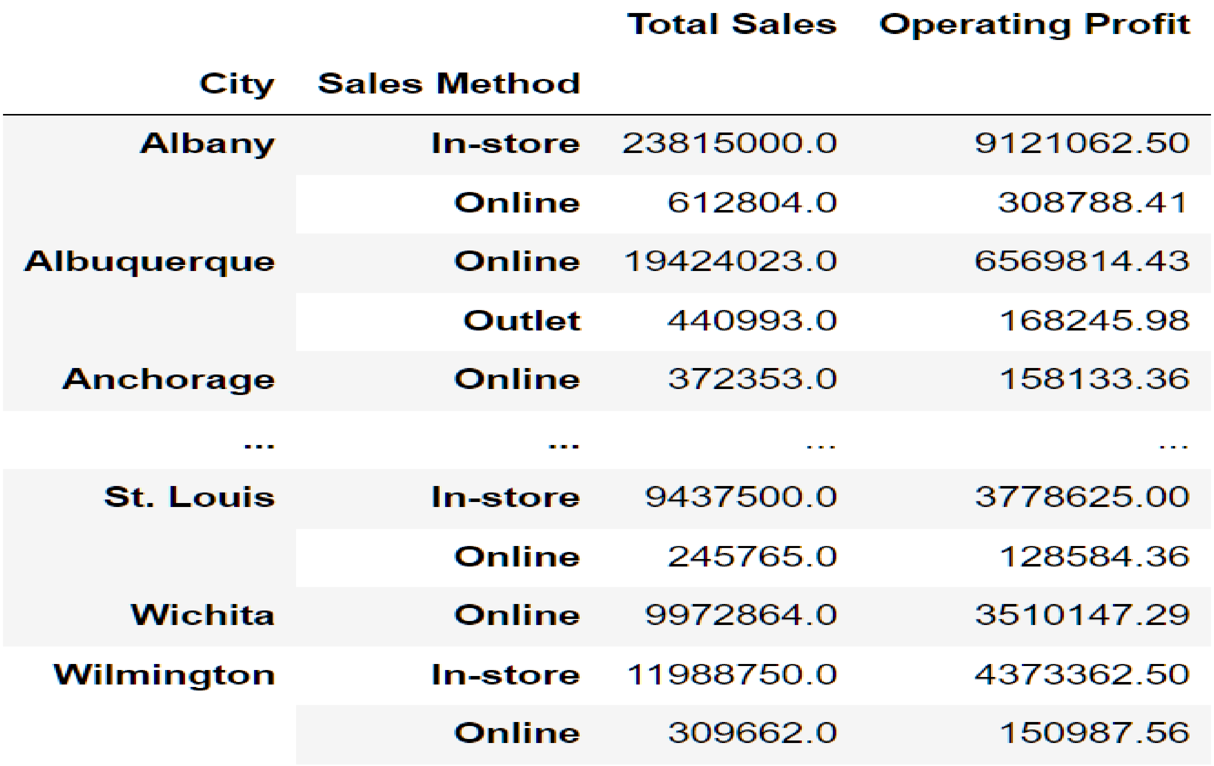


	Total Sales	Operating Profit
Invoice Date		
April	72339970.0	27559237.31
August	92166201.0	34451440.30
December	85841957.0	31590202.03
February	61100153.0	21392736.70
January	71479142.0	25141934.51
July	95480694.0	34054898.59
June	74747372.0	26714715.92
March	56809109.0	20439788.00
May	80507695.0	29946255.33
November	67857340.0	24755521.43
October	63911033.0	25078444.60
September	77661459.0	31009586.73

12. Find the total sales &profit of the products in different sales methods in each city

file.groupby(['City', 'Sales Method'])[['Total Sales', 'Operating Profit']].sum()

Output:



		Total Sales	Operating Profit
City	**Sales Method**		
Albany	**In-store**	23815000.0	9121062.50
	Online	612804.0	308788.41
Albuquerque	**Online**	19424023.0	6569814.43
	Outlet	440993.0	168245.98
Anchorage	**Online**	372353.0	158133.36
...	**...**	...	...
St. Louis	**In-store**	9437500.0	3778625.00
	Online	245765.0	128584.36
Wichita	**Online**	9972864.0	3510147.29
Wilmington	**In-store**	11988750.0	4373362.50
	Online	309662.0	150987.56

13. Find the retailers who sold the same product with different prices in different cities

```
list(file.groupby(['Retailer', 'Product', 'Price per Unit']).groups.keys())
```

Output:

```
[('Amazon', "Men's Apparel", 32.0),
 ('Amazon', "Men's Apparel", 35.0),
 ('Amazon', "Men's Apparel", 36.0),
 ('Amazon', "Men's Apparel", 37.0),
 ('Amazon', "Men's Apparel", 39.0),
 ('Amazon', "Men's Apparel", 40.0),
 ('Amazon', "Men's Apparel", 40.00000000000001),
 ('Amazon', "Men's Apparel", 41.0),
 ('Amazon', "Men's Apparel", 42.0),
 ('Amazon', "Men's Apparel", 43.0),
 ('Amazon', "Men's Apparel", 44.0),
 ('Amazon', "Men's Apparel", 45.0),….]
```

14. Find the products whose sales raises in every month.

```
f=file.groupby([file['Invoice Date'].dt.month,file['Invoice Date'].dt.year,file['Product']])['Total Sales'].sum()
print(f)
l=list(file['Product'].unique())[:]
for i in l:
 print(i,f[:,:,i].is_monotonic_increasing)
```

Output:

```
Invoice Date  Invoice Date  Product
1             2020          Men's Apparel                2288362.0
                            Men's Athletic Footwear      2639958.0
                            Men's Street Footwear        3859495.0
                            Women's Apparel              3066713.0
                            Women's Athletic Footwear    1990181.0
                                                           ...
12            2021          Men's Athletic Footwear      13195038.0
                            Men's Street Footwear        18953848.0
                            Women's Apparel              14910708.0
                            Women's Athletic Footwear    9549962.0
                            Women's Street Footwear      10547148.0
Name: Total Sales, Length: 144, dtype: float64
Men's Street Footwear False
Men's Athletic Footwear False
Women's Street Footwear False
Women's Athletic Footwear False
Men's Apparel False
```

15. Find the retailers whose profit increased every month.

```
f=file.groupby([file['Invoice Date'].dt.month,file['Invoice Date'].dt.year,file['Retailer']])['Operating Profit'].sum()
print(f)
l=list(file['Retailer'].unique())[:]
for i in l:
 print(i,f[:,:,i].is_monotonic_increasing)
```

Output:

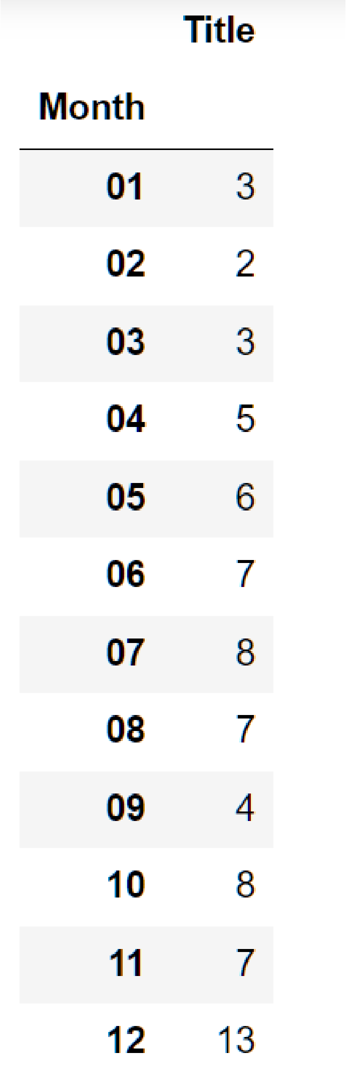
```
Invoice Date  Invoice Date  Retailer
1             2020          Foot Locker      3544899.00
                            West Gear        2285106.41
              2021          Amazon           1510504.30
                            Foot Locker      3761780.79
                            Kohl's           3565743.93
                                                ...
12            2021          Foot Locker     10016105.07
                            Kohl's           2749782.47
                            Sports Direct    5660159.36
                            Walmart           631767.00
                            West Gear        6232275.13
Name: Operating Profit, Length: 104, dtype: float64
Foot Locker False
Walmart False
Sports Direct False
West Gear False
Kohl's False
Amazon False
```

Movies dataset:- Lab cycle-5

1. (i). Find out the no of movies released in every month of the year 1995.

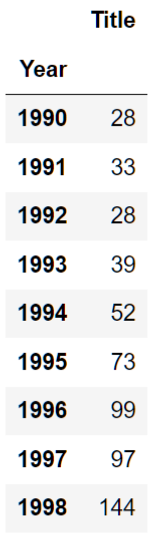
```
import pandas as pd
from datetime import datetime
file = pd.read_csv("E:\Downloads\data-analysis\movies.csv")
file = file[file["Release Date"].str.contains('TBD') == False]
file['Release Date'] = file['Release Date'].apply(pd.to_datetime)
file['Date']=file['Release Date'].dt.strftime('%d')
file['Month']=file['Release Date'].dt.strftime('%m')
file['Year']=file['Release Date'].dt.strftime('%Y')
file[file['Year']=='1995'][['Title', 'Month']].groupby('Month').count()
```

Output:



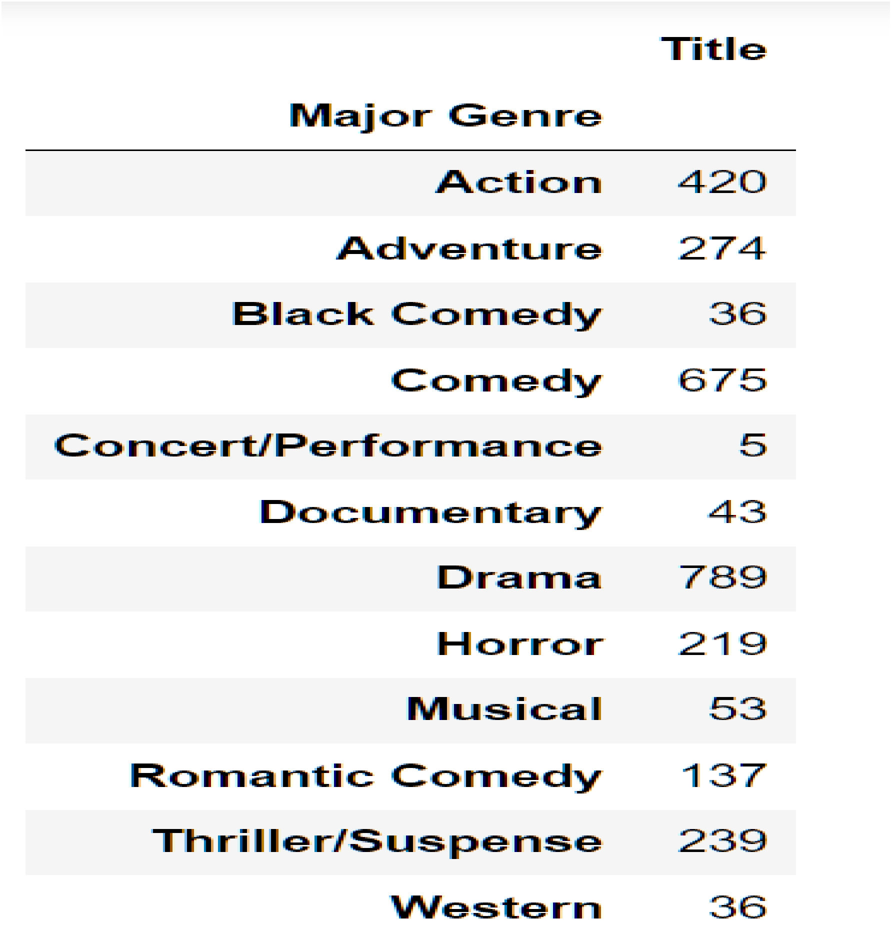
```
       Title
Month
   01      3
   02      2
   03      3
   04      5
   05      6
   06      7
   07      8
   08      7
   09      4
   10      8
   11      7
   12     13
```

(ii). Find out the no. of movies released in every year from 1990 to 1998.
file[(file['Year']>='1990') & (file['Year']<='1998')][['Title', 'Year']].groupby('Year').count()



	Title
Year	
1990	28
1991	33
1992	28
1993	39
1994	52
1995	73
1996	99
1997	97
1998	144

2. (i). Find no. of movies released under each genre given in the database.
file[['Title', 'Major Genre']].dropna().groupby('Major Genre').count()



	Title
Major Genre	
Action	420
Adventure	274
Black Comedy	36
Comedy	675
Concert/Performance	5
Documentary	43
Drama	789
Horror	219
Musical	53
Romantic Comedy	137
Thriller/Suspense	239
Western	36

(ii). Find the movies under each genre with 1MDB rating >7 and rotten tomatoes rating > 60.

```
list(file[(file['IMDB Rating']>7) & (file['Rotten Tomatoes Rating']>60)].dropna().groupby(['Major Genre', 'Title']).groups.keys())
```

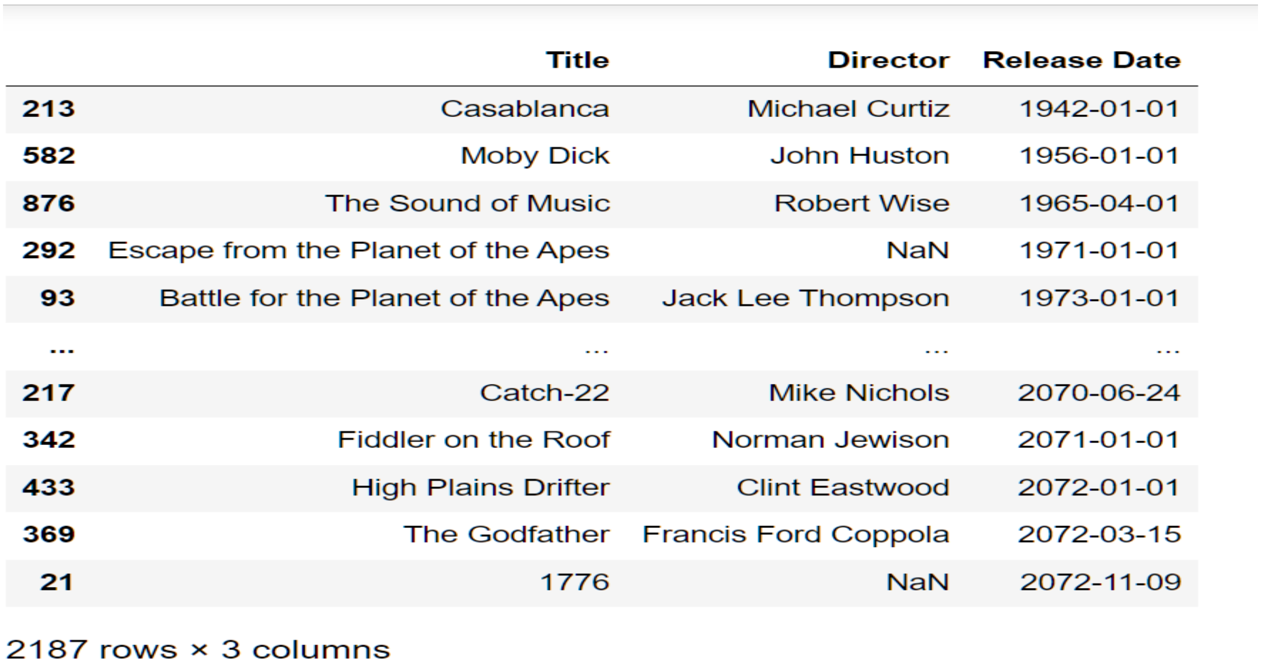
Output:

```
[('Action', 'Black Hawk Down'),
 ('Action', 'Blood Diamond'),
 ('Action', 'Casino Royale'),
 ('Action', 'Inglourious Basterds'),
 ('Action', 'Iron Man'),
 ('Action', 'Live Free or Die Hard'),
 ('Action', 'The Bourne Ultimatum'),
 ('Action', 'The Dark Knight'),
…('Thriller/Suspense', 'Children of Men'),
 ('Thriller/Suspense', 'District 9'),
 ('Thriller/Suspense', 'State of Play'),
 ('Thriller/Suspense', 'Sunshine'),
 ('Thriller/Suspense', 'The Prestige'),
 ('Thriller/Suspense', 'Valkyrie'),
 ('Western', '3:10 to Yuma')]
```

3. (i). Find the movies released under each fiction with each director in the ascending order of release dates.

```
file[file['Creative Type'].fillna('0').str.contains('Fiction')][['Title', 'Director', 'Release Date']].sort_values('Release Date')
```

Output:



	Title	Director	Release Date
213	Casablanca	Michael Curtiz	1942-01-01
582	Moby Dick	John Huston	1956-01-01
876	The Sound of Music	Robert Wise	1965-04-01
292	Escape from the Planet of the Apes	NaN	1971-01-01
93	Battle for the Planet of the Apes	Jack Lee Thompson	1973-01-01
...	...	...	...
217	Catch-22	Mike Nichols	2070-06-24
342	Fiddler on the Roof	Norman Jewison	2071-01-01
433	High Plains Drifter	Clint Eastwood	2072-01-01
369	The Godfather	Francis Ford Coppola	2072-03-15
21	1776	NaN	2072-11-09

2187 rows × 3 columns

(ii). Find movies released under each distributor in the order of genre and director.

```
list(file.dropna().groupby(['Distributor', 'Major Genre', 'Director']).groups.keys()
)
```

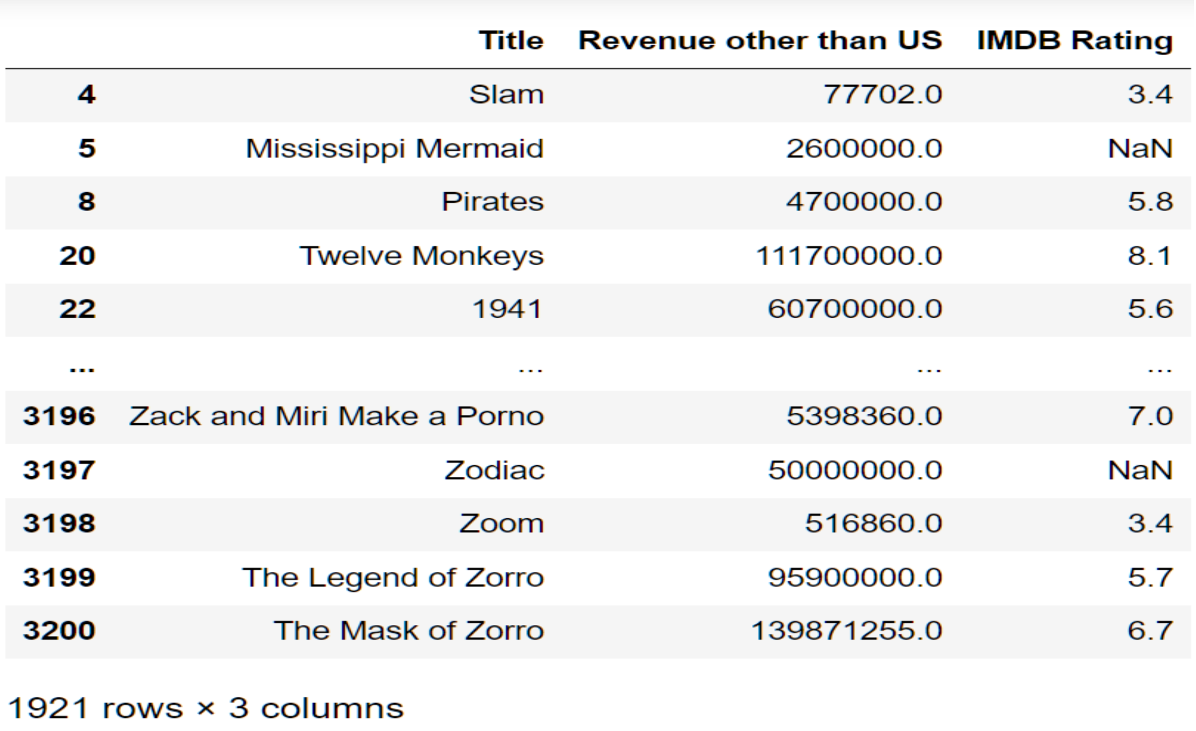
Output:

```
[('20th Century Fox', 'Action', 'Len Wiseman'),
 ('20th Century Fox', 'Action', 'Mathieu Kassovitz'),
 ('20th Century Fox', 'Action', 'Renny Harlin'),
 ('20th Century Fox', 'Action', 'Tim Story'),
 ('20th Century Fox', 'Adventure', 'Gil Kenan'),
 ('20th Century Fox', 'Comedy', 'Adam Shankman'),
 ('20th Century Fox', 'Comedy', 'David Silverman'
…..
 ('Weinstein Co.', 'Comedy', 'Kevin Smith'),
 ('Weinstein Co.', 'Drama', 'Denzel Washington'),
 ('Weinstein Co.', 'Drama', 'Stephen Daldry'),
 ('Weinstein Co.', 'Thriller/Suspense', 'Peter Webber')]
```

4. (i). Find the movies released world-wide and find out the revenue received world-wide other than US with their ratings.

```
file['Revenue other than US'] = file['Worldwide Gross'].replace('Unknown', '0').
astype(float) - file['US Gross'].replace('Unknown', '0').astype(float)
file[file['Revenue other than US'] > 0][['Title', 'Revenue other than US', 'IMDB
Rating']]
```

Output:



	Title	Revenue other than US	IMDB Rating
4	Slam	77702.0	3.4
5	Mississippi Mermaid	2600000.0	NaN
8	Pirates	4700000.0	5.8
20	Twelve Monkeys	111700000.0	8.1
22	1941	60700000.0	5.6
...	...	...	...
3196	Zack and Miri Make a Porno	5398360.0	7.0
3197	Zodiac	50000000.0	NaN
3198	Zoom	516860.0	3.4
3199	The Legend of Zorro	95900000.0	5.7
3200	The Mask of Zorro	139871255.0	6.7

1921 rows × 3 columns

(ii). Find the movies with loss & profit released in each year with genre and ratings.

```
file['Worldwide Gross'] = file['Worldwide Gross'].replace('Unknown', '0').astype(float)
list(file[file['Worldwide Gross']-file['Production Budget']!=0].groupby(['Year', 'Major Genre', 'IMDB Rating']).groups.keys())
```

Output:

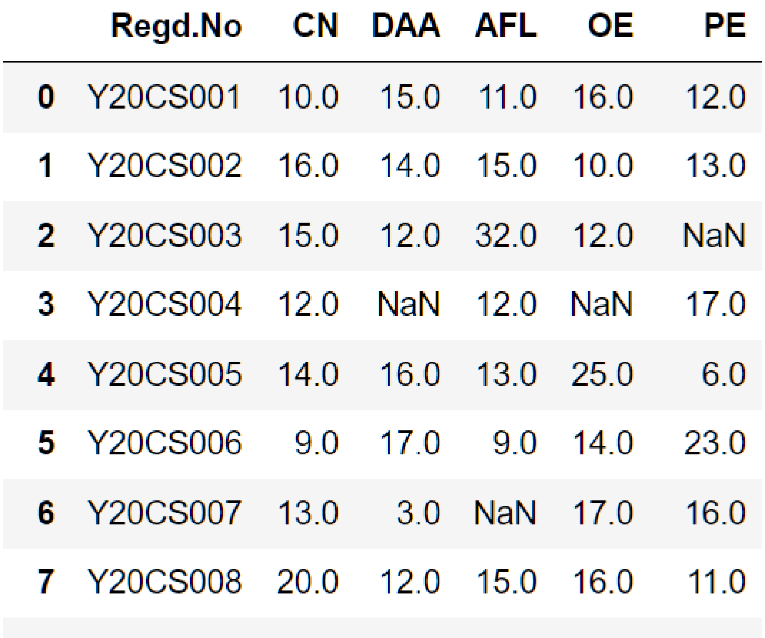
```
[('1929', 'Musical', 6.7),
 ('1930', nan, 7.9),
 ('1931', nan, 2.2),
 ('1934', 'Romantic Comedy', 8.3),
 ('1938', 'Drama', nan),
 ('1938', nan, 8.0),
 ('1939', 'Drama', 8.2),
 ('1940', 'Drama', 8.4),
 ('1940', nan, 7.1),
 ('1942', 'Drama', 8.8),
….]
```

Student dataset:- Lab cycle-6

1. Combine the CSE & IT data and display the data

```
import pandas as pd
import numpy as np
cse = pd.read_csv("E:\Downloads\data-analysis\CSE.csv")
it = pd.read_csv("E:\Downloads\data-analysis\IT.csv")
student = pd.read_csv("E:\Downloads\data-analysis\student.csv")
cse.rename(columns={'Professional Elective':'PE'}, inplace=True)
merged_data = pd.merge(student, cse).append(pd.merge(student, it))
merged_data = merged_data.reset_index()
pd.concat([cse, it])
```

Output:

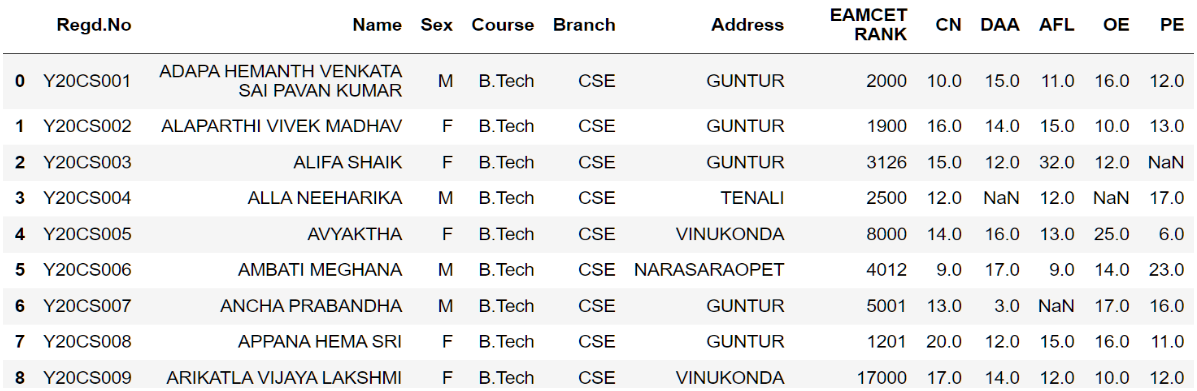


	Regd.No	CN	DAA	AFL	OE	PE
0	Y20CS001	10.0	15.0	11.0	16.0	12.0
1	Y20CS002	16.0	14.0	15.0	10.0	13.0
2	Y20CS003	15.0	12.0	32.0	12.0	NaN
3	Y20CS004	12.0	NaN	12.0	NaN	17.0
4	Y20CS005	14.0	16.0	13.0	25.0	6.0
5	Y20CS006	9.0	17.0	9.0	14.0	23.0
6	Y20CS007	13.0	3.0	NaN	17.0	16.0
7	Y20CS008	20.0	12.0	15.0	16.0	11.0

2. Display all CSE students' marks along with personal information.

```
pd.merge(student, cse, on='Regd.No')
```

Output:

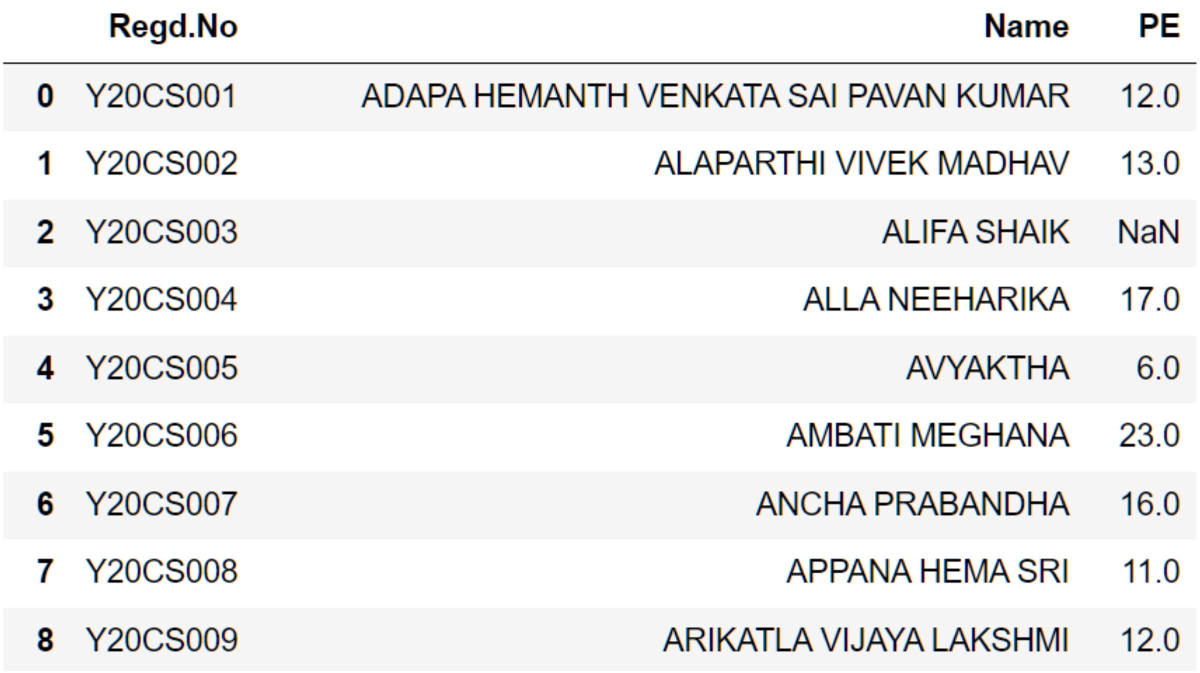


	Regd.No	Name	Sex	Course	Branch	Address	EAMCET RANK	CN	DAA	AFL	OE	PE
0	Y20CS001	ADAPA HEMANTH VENKATA SAI PAVAN KUMAR	M	B.Tech	CSE	GUNTUR	2000	10.0	15.0	11.0	16.0	12.0
1	Y20CS002	ALAPARTHI VIVEK MADHAV	F	B.Tech	CSE	GUNTUR	1900	16.0	14.0	15.0	10.0	13.0
2	Y20CS003	ALIFA SHAIK	F	B.Tech	CSE	GUNTUR	3126	15.0	12.0	32.0	12.0	NaN
3	Y20CS004	ALLA NEEHARIKA	M	B.Tech	CSE	TENALI	2500	12.0	NaN	12.0	NaN	17.0
4	Y20CS005	AVYAKTHA	F	B.Tech	CSE	VINUKONDA	8000	14.0	16.0	13.0	25.0	6.0
5	Y20CS006	AMBATI MEGHANA	M	B.Tech	CSE	NARASARAOPET	4012	9.0	17.0	9.0	14.0	23.0
6	Y20CS007	ANCHA PRABANDHA	M	B.Tech	CSE	GUNTUR	5001	13.0	3.0	NaN	17.0	16.0
7	Y20CS008	APPANA HEMA SRI	F	B.Tech	CSE	GUNTUR	1201	20.0	12.0	15.0	16.0	11.0
8	Y20CS009	ARIKATLA VIJAYA LAKSHMI	F	B.Tech	CSE	VINUKONDA	17000	17.0	14.0	12.0	10.0	12.0

3. Print all students Regd.No, Name and professional elective.

```
merged_data[['Regd.No', 'Name', 'PE']]
```

Output:

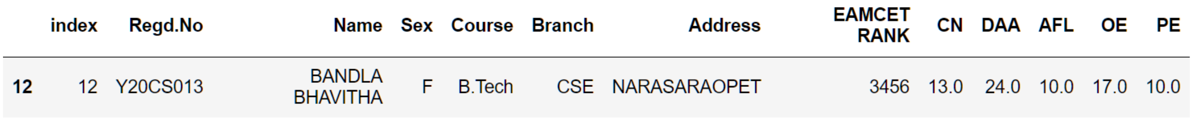


	Regd.No	Name	PE
0	Y20CS001	ADAPA HEMANTH VENKATA SAI PAVAN KUMAR	12.0
1	Y20CS002	ALAPARTHI VIVEK MADHAV	13.0
2	Y20CS003	ALIFA SHAIK	NaN
3	Y20CS004	ALLA NEEHARIKA	17.0
4	Y20CS005	AVYAKTHA	6.0
5	Y20CS006	AMBATI MEGHANA	23.0
6	Y20CS007	ANCHA PRABANDHA	16.0
7	Y20CS008	APPANA HEMA SRI	11.0
8	Y20CS009	ARIKATLA VIJAYA LAKSHMI	12.0

4. Identify the students whose DAA marks are >18.

```
merged_data[merged_data['DAA']>18]
```

Output:

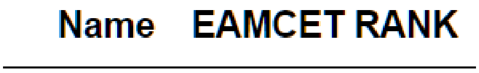


	index	Regd.No	Name	Sex	Course	Branch	Address	EAMCET RANK	CN	DAA	AFL	OE	PE
12	12	Y20CS013	BANDLA BHAVITHA	F	B.Tech	CSE	NARASARAOPET	3456	13.0	24.0	10.0	17.0	10.0

5. Display the names and EAMCET ranks of the students who got minimum 12 marks in all courses.

```
merged_data[(merged_data["CN"]>=12)&(merged_data['OE']>=12)&(merged_data["AFL"]>=12)&(merged_data['PE']>=12)&(merged_data['DAA']>=12)][['Name','EAMCET RANK']]
```

Output:



Name	EAMCET RANK

6. Calculate mean value of all the subject's marks.

```
merged_data.loc[0:,['CN','DAA','AFL','OE','PE']].mean()
```

Output:

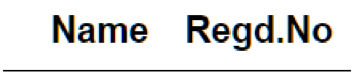
Out[6]:

```
CN     13.607143
DAA    14.000000
AFL    13.214286
OE     14.428571
PE     17.666667
dtype: float64
```

7. Display the names common in both CSE & IT along with Regd.No.

merged_data[merged_data['Name'].duplicated()][['Name', 'Regd.No']]

Output:

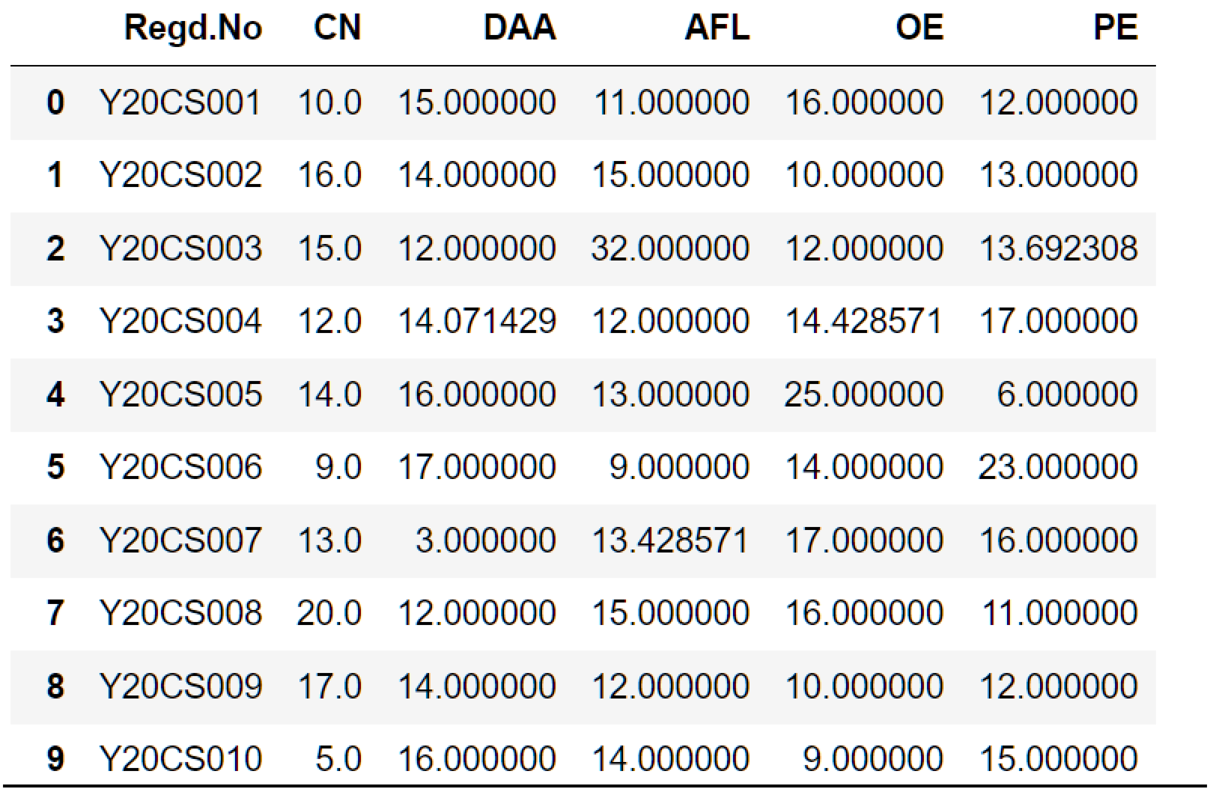


	Name	Regd.No

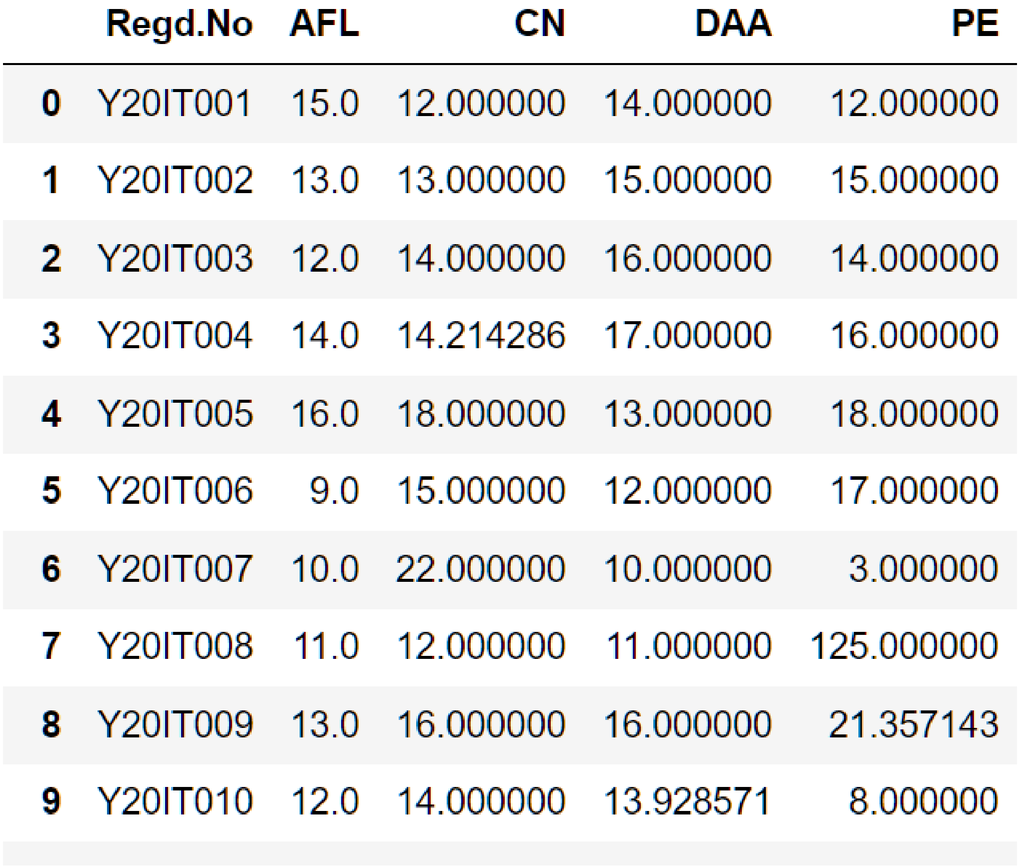
8. Fill the missing values of the data with average marks of the subject of specific group.

print(cse.fillna(cse.mean()))
print(it.fillna(it.mean()))

Output:



	Regd.No	CN	DAA	AFL	OE	PE
0	Y20CS001	10.0	15.000000	11.000000	16.000000	12.000000
1	Y20CS002	16.0	14.000000	15.000000	10.000000	13.000000
2	Y20CS003	15.0	12.000000	32.000000	12.000000	13.692308
3	Y20CS004	12.0	14.071429	12.000000	14.428571	17.000000
4	Y20CS005	14.0	16.000000	13.000000	25.000000	6.000000
5	Y20CS006	9.0	17.000000	9.000000	14.000000	23.000000
6	Y20CS007	13.0	3.000000	13.428571	17.000000	16.000000
7	Y20CS008	20.0	12.000000	15.000000	16.000000	11.000000
8	Y20CS009	17.0	14.000000	12.000000	10.000000	12.000000
9	Y20CS010	5.0	16.000000	14.000000	9.000000	15.000000

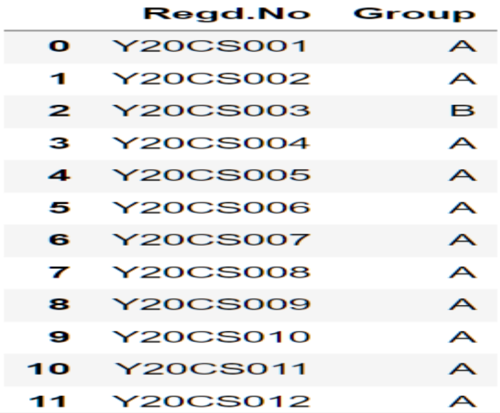


	Regd.No	AFL	CN	DAA	PE
0	Y20IT001	15.0	12.000000	14.000000	12.000000
1	Y20IT002	13.0	13.000000	15.000000	15.000000
2	Y20IT003	12.0	14.000000	16.000000	14.000000
3	Y20IT004	14.0	14.214286	17.000000	16.000000
4	Y20IT005	16.0	18.000000	13.000000	18.000000
5	Y20IT006	9.0	15.000000	12.000000	17.000000
6	Y20IT007	10.0	22.000000	10.000000	3.000000
7	Y20IT008	11.0	12.000000	11.000000	125.000000
8	Y20IT009	13.0	16.000000	16.000000	21.357143
9	Y20IT010	12.0	14.000000	13.928571	8.000000

9. Divide the students into 5 groups based on average marks.

```
merged_data.fillna(merged_data[['CN', 'DAA', 'AFL', 'OE', 'PE']].mean(), inplace=True)
merged_data['Avg_Marks'] = merged_data[['CN', 'DAA', 'AFL', 'OE', 'PE']].mean(axis = 1)
merged_data['Group'] = pd.cut(merged_data['Avg_Marks'], bins=np.arange(10, 36, 5), labels=['A', 'B', 'C', 'D', 'E'])
merged_data[['Regd.No', 'Group']]
```

Output:

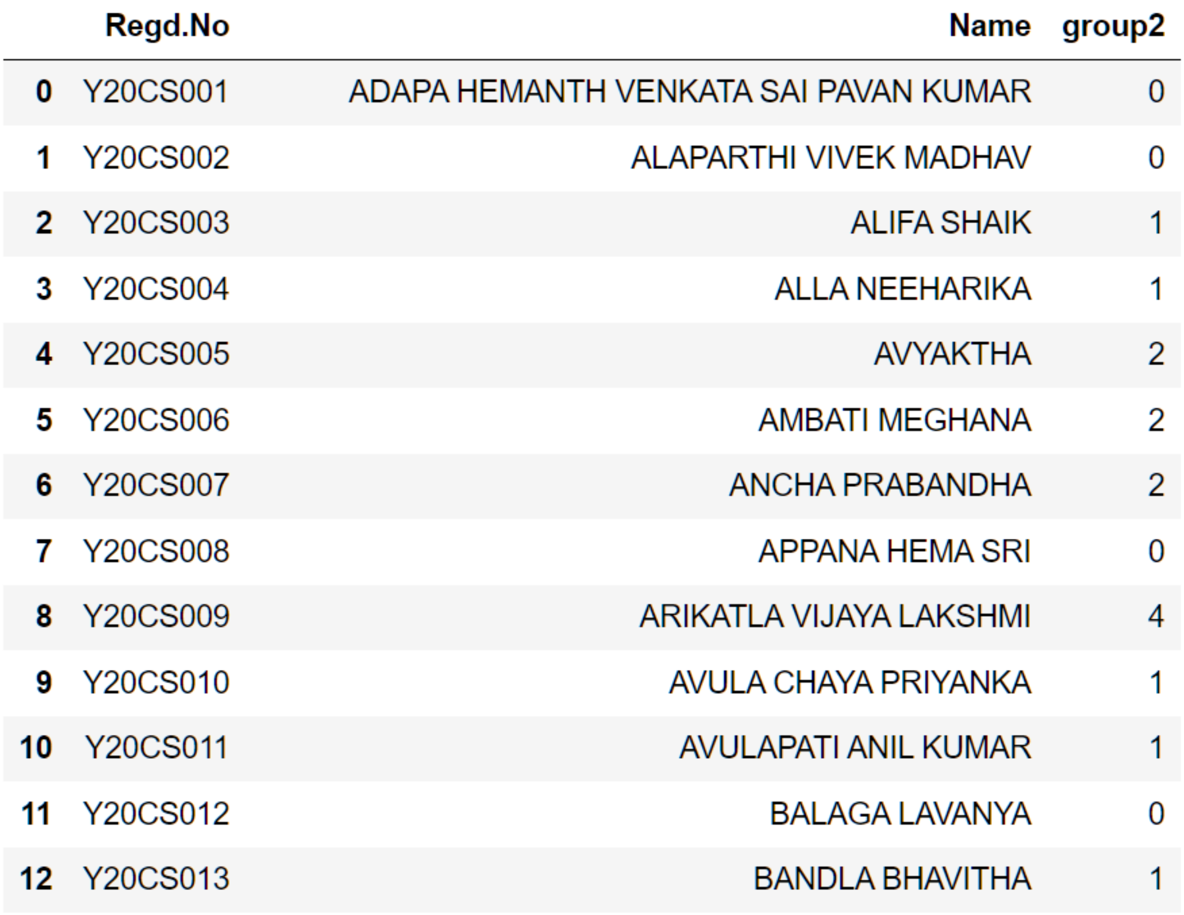


	Regd.No	Group
0	Y20CS001	A
1	Y20CS002	A
2	Y20CS003	B
3	Y20CS004	A
4	Y20CS005	A
5	Y20CS006	A
6	Y20CS007	A
7	Y20CS008	A
8	Y20CS009	A
9	Y20CS010	A
10	Y20CS011	A
11	Y20CS012	A

10.Create equal sized groups of students based on EAMCET Rank.

```
merged_data['group2']=pd.qcut(merged_data['EAMCET RANK'],6,labels=False)
merged_data[['Regd.No','Name','group2']]
```

Output:



	Regd.No	Name	group2
0	Y20CS001	ADAPA HEMANTH VENKATA SAI PAVAN KUMAR	0
1	Y20CS002	ALAPARTHI VIVEK MADHAV	0
2	Y20CS003	ALIFA SHAIK	1
3	Y20CS004	ALLA NEEHARIKA	1
4	Y20CS005	AVYAKTHA	2
5	Y20CS006	AMBATI MEGHANA	2
6	Y20CS007	ANCHA PRABANDHA	2
7	Y20CS008	APPANA HEMA SRI	0
8	Y20CS009	ARIKATLA VIJAYA LAKSHMI	4
9	Y20CS010	AVULA CHAYA PRIYANKA	1
10	Y20CS011	AVULAPATI ANIL KUMAR	1
11	Y20CS012	BALAGA LAVANYA	0
12	Y20CS013	BANDLA BHAVITHA	1

11.Display the electives and the Regd.No of students who opted the elective along with thesubject name.

```
print(cse[cse['OE'].notnull() & cse['PE'].notnull()][['Regd.No', 'OE','PE']])
print(it[it['PE'].notnull()][['Regd.No', 'PE']])
```

Output:

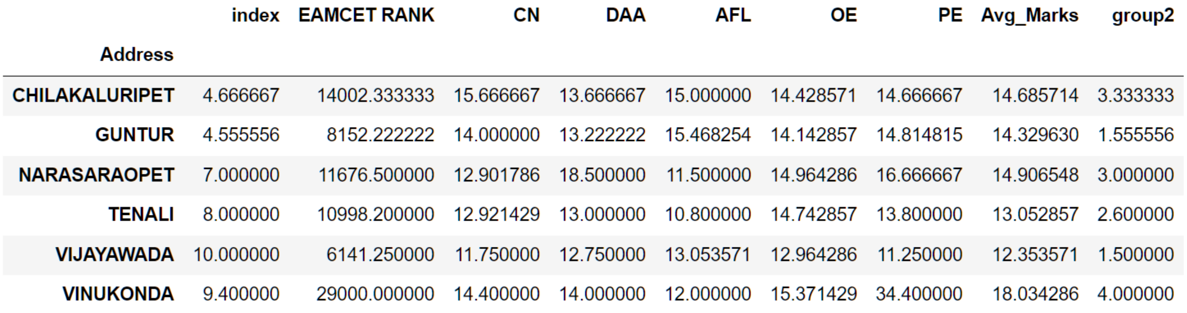
```
   Regd.No    OE    PE
0   Y20CS001  16.0  12.0
1   Y20CS002  10.0  13.0
4   Y20CS005  25.0   6.0
5   Y20CS006  14.0  23.0
6   Y20CS007  17.0  16.0
7   Y20CS008  16.0  11.0
8   Y20CS009  10.0  12.0
9   Y20CS010   9.0  15.0
10  Y20CS011  16.0  14.0
12  Y20CS013  17.0  10.0
```

```
13  Y20CS014  14.0  16.0
14  Y20CS015  13.0  13.0
      Regd.No     PE
0   Y20IT001   12.0
1   Y20IT002   15.0
2   Y20IT003   14.0
3   Y20IT004   16.0
4   Y20IT005   18.0
5   Y20IT006   17.0
6   Y20IT007    3.0
7   Y20IT008  125.0
9   Y20IT010    8.0
10  Y20IT011   14.0
11  Y20IT012   17.0
12  Y20IT013   11.0
13  Y20IT014   13.0
14  Y20IT015   16.0
```

12.Compare the performance of the students from various cities.

```
merged_data.groupby(merged_data['Address']).mean()
```

Output:



	index	EAMCET RANK	CN	DAA	AFL	OE	PE	Avg_Marks	group2
Address									
CHILAKALURIPET	4.666667	14002.333333	15.666667	13.666667	15.000000	14.428571	14.666667	14.685714	3.333333
GUNTUR	4.555556	8152.222222	14.000000	13.222222	15.468254	14.142857	14.814815	14.329630	1.555556
NARASARAOPET	7.000000	11676.500000	12.901786	18.500000	11.500000	14.964286	16.666667	14.906548	3.000000
TENALI	8.000000	10998.200000	12.921429	13.000000	10.800000	14.742857	13.800000	13.052857	2.600000
VIJAYAWADA	10.000000	6141.250000	11.750000	12.750000	13.053571	12.964286	11.250000	12.353571	1.500000
VINUKONDA	9.400000	29000.000000	14.400000	14.000000	12.000000	15.371429	34.400000	18.034286	4.000000

13. Find the correlation between the marks of DS &DAA.

```
df=pd.concat([cse,it])
df['CN'].corr(df['DAA'])
```

Output:

```
-0.2031529941702545
```