Just IT

B2W group

Apprenticeships | Training | Recruitment

Data Technician

Name: Almas Mansuri

Course Date: 12/05/2025

Table of contents

Day 2: Task 1

It is a common software development interview question to create the below with a certain programming language. Create the below using Python syntax, test it and past the completed syntax and output below.

FizzBuzz:

Go through the integers from 1 to 100.
If a number is divisible by 3, print "fizz."
If a number is divisible by 5, print "buzz."
If a number is both divisible by 3 and by 5, print "fizzbuzz."
Otherwise, print just the number.

Paste your completed work to the right

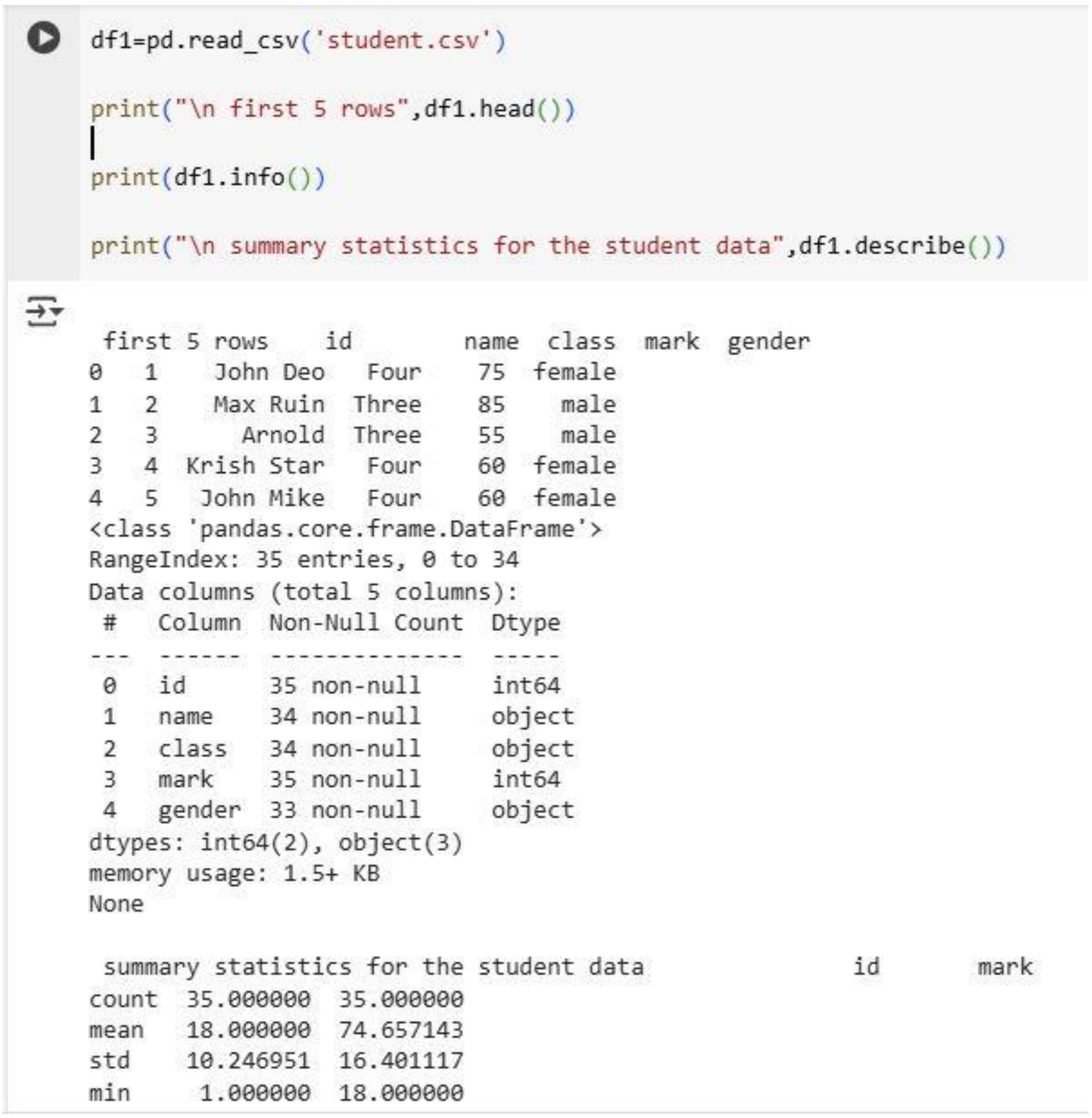
```
for i in range(1, 101):
  if i%3==0 and i%5==0:
  print("FizzBuzz")
  elif i%3 == 0:
    print("Fizz")
    elif i%5 == 0:
    print("Buzz")
    else:
      print(i)
```

Day 3: Task 1

Download the ‘student.csv’, complete the below exercises as a group and paste your input and output. Although this is a group activity, everyone should have the below answered so it supports your portfolio:

Exercise 1: Loading and Exploring the Data

1. Question: "Write the code to read a CSV file into a Pandas DataFrame."
2. Question: "Write the code to display the first 5 rows of the DataFrame."
3. Question: "Write the code to get the information about the DataFrame."
4. Question: "Write the code to get summary statistics for the DataFrame."

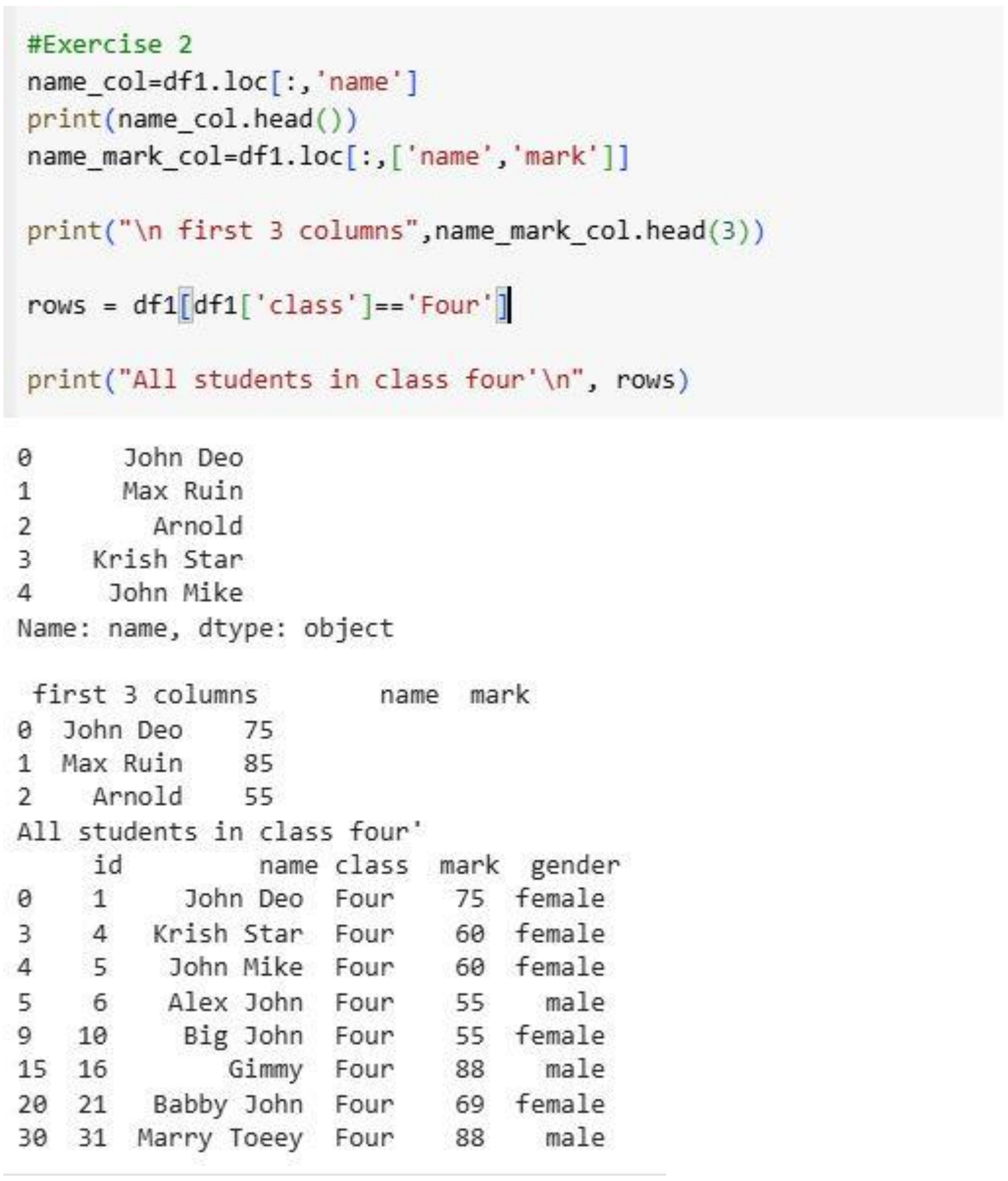


```
df1=pd.read_csv('student.csv')

print("\n first 5 rows",df1.head())

print(df1.info())

print("\n summary statistics for the student data",df1.describe())
```

```
 first 5 rows    id        name  class  mark  gender
0   1    John Deo   Four    75  female
1   2    Max Ruin  Three    85    male
2   3      Arnold  Three    55    male
3   4  Krish Star   Four    60  female
4   5   John Mike   Four    60  female
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35 entries, 0 to 34
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      35 non-null     int64 
 1   name    34 non-null     object
 2   class   34 non-null     object
 3   mark    35 non-null     int64 
 4   gender  33 non-null     object
dtypes: int64(2), object(3)
memory usage: 1.5+ KB
None

 summary statistics for the student data               id        mark
count  35.000000  35.000000
mean   18.000000  74.657143
std    10.246951  16.401117
min     1.000000  18.000000
```

Exercise 2: Indexing and Slicing

1. Question: "Write the code to select the 'name' column."
2. Question: "Write the code to select the 'name' and 'mark' columns."
3. Question: "Write the code to select the first 3 rows."
4. Question: "Write the code to select all rows where the 'class' is 'Four'."

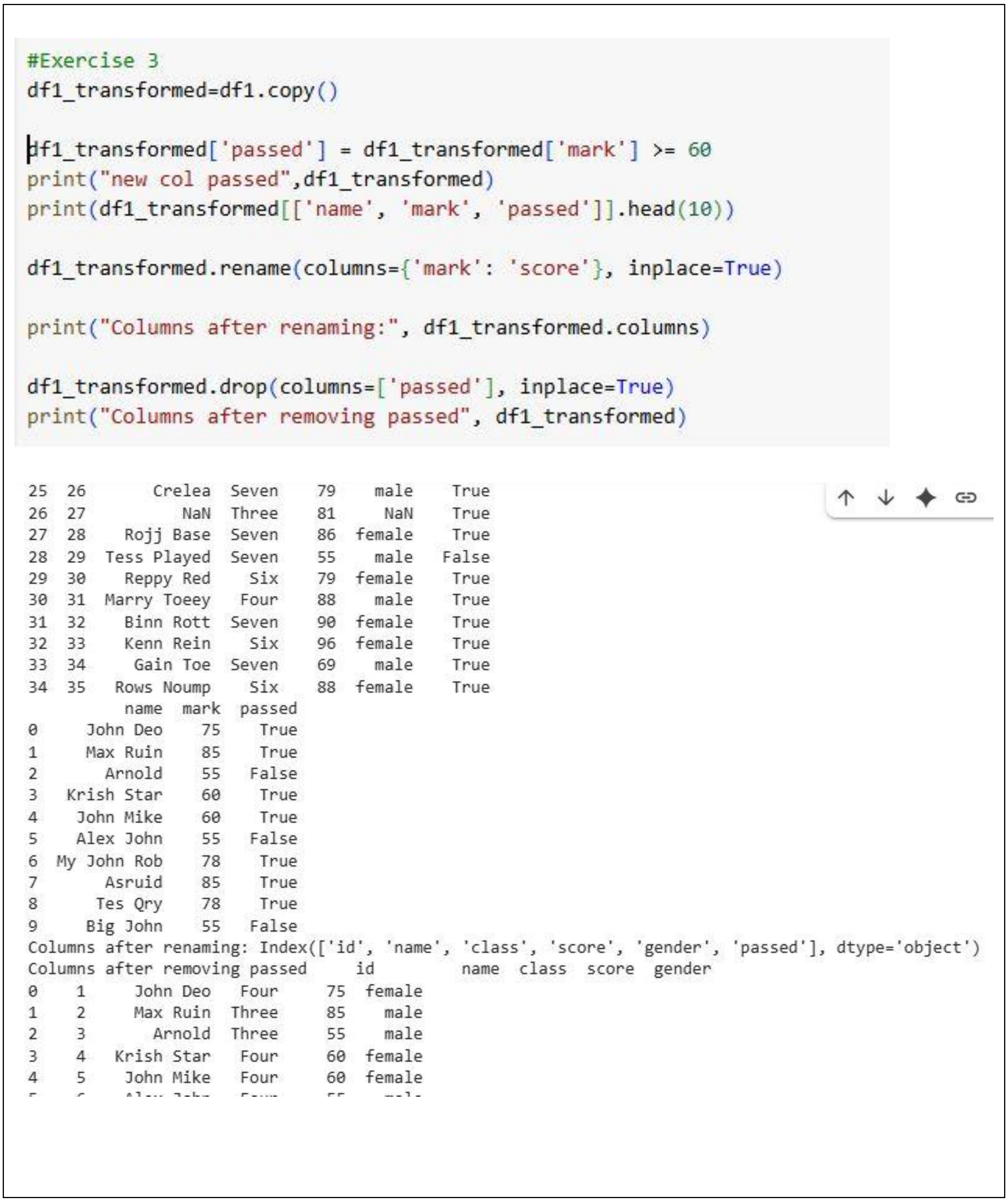


```python
#Exercise 2
name_col=df1.loc[:,'name']
print(name_col.head())
name_mark_col=df1.loc[:,['name','mark']]

print("\n first 3 columns",name_mark_col.head(3))

rows = df1[df1['class']=='Four']

print("All students in class four'\n", rows)
```

```
0      John Deo
1      Max Ruin
2        Arnold
3    Krish Star
4     John Mike
Name: name, dtype: object

 first 3 columns        name  mark
0  John Deo    75
1  Max Ruin    85
2    Arnold    55
All students in class four'
     id          name class  mark  gender
0    1      John Deo  Four    75  female
3    4    Krish Star  Four    60  female
4    5     John Mike  Four    60  female
5    6     Alex John  Four    55    male
9   10      Big John  Four    55  female
15  16         Gimmy  Four    88    male
20  21    Babby John  Four    69  female
30  31  Marry Toeey  Four    88    male
```

Exercise 3: Data Manipulation

1. Question: "Write the code to add a new column 'passed' that indicates whether the student passed (mark >= 60)."
2. Question: "Write the code to rename the 'mark' column to 'score'."
3. Question: "Write the code to drop the 'passed' column."

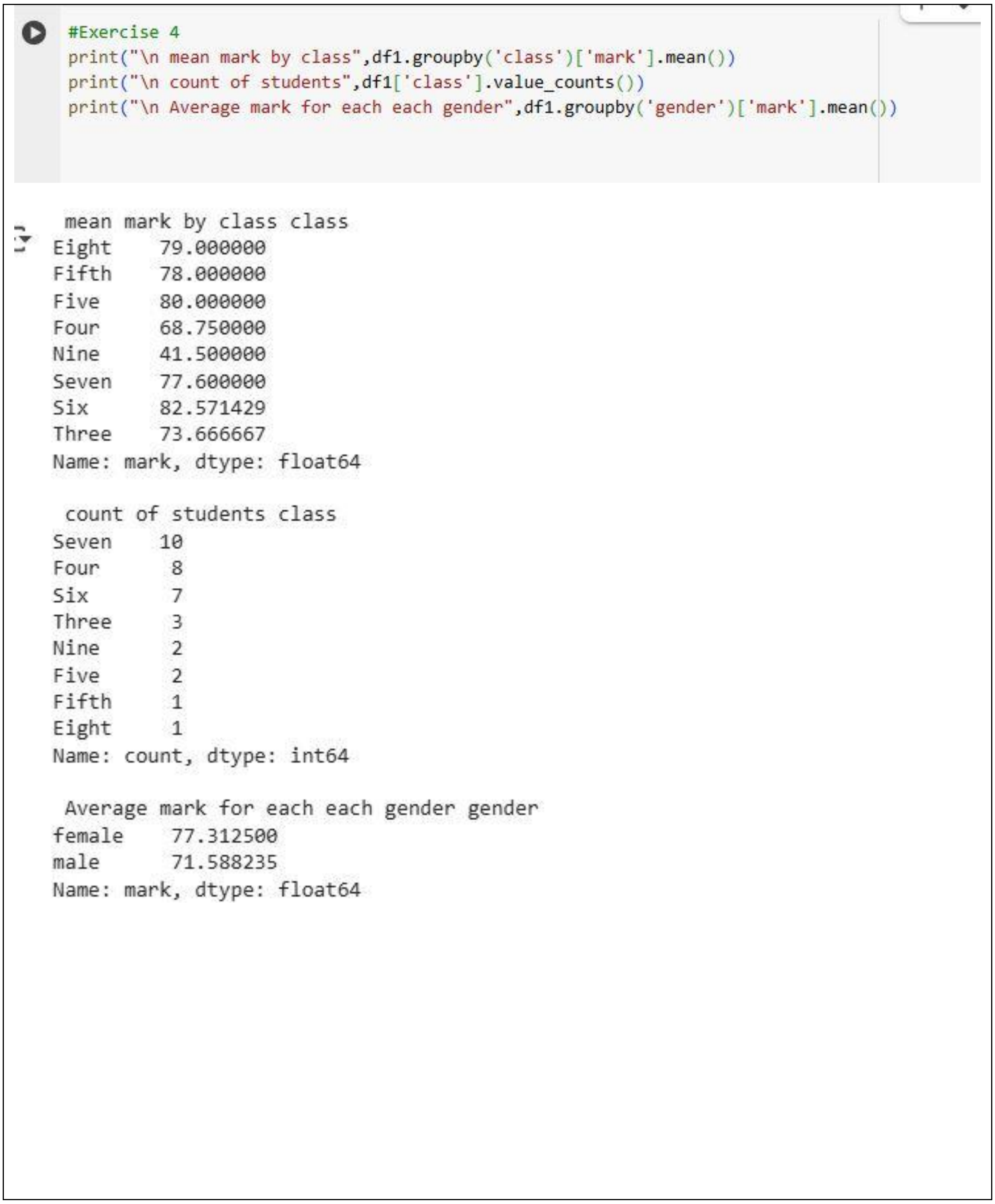


```python
#Exercise 3
df1_transformed=df1.copy()

df1_transformed['passed'] = df1_transformed['mark'] >= 60
print("new col passed",df1_transformed)
print(df1_transformed[['name', 'mark', 'passed']].head(10))

df1_transformed.rename(columns={'mark': 'score'}, inplace=True)

print("Columns after renaming:", df1_transformed.columns)

df1_transformed.drop(columns=['passed'], inplace=True)
print("Columns after removing passed", df1_transformed)
```

```
25  26       Crelea  Seven   79    male    True
26  27          NaN  Three   81     NaN    True
27  28    Rojj Base  Seven   86  female    True
28  29  Tess Played  Seven   55    male   False
29  30    Reppy Red    Six   79  female    True
30  31  Marry Toeey   Four   88    male    True
31  32    Binn Rott  Seven   90  female    True
32  33    Kenn Rein    Six   96  female    True
33  34     Gain Toe  Seven   69    male    True
34  35   Rows Noump    Six   88  female    True
          name  mark  passed
0     John Deo    75    True
1     Max Ruin    85    True
2       Arnold    55   False
3   Krish Star    60    True
4    John Mike    60    True
5    Alex John    55   False
6  My John Rob    78    True
7       Asruid    85    True
8      Tes Qry    78    True
9     Big John    55   False
Columns after renaming: Index(['id', 'name', 'class', 'score', 'gender', 'passed'], dtype='object')
Columns after removing passed     id          name  class  score  gender
0    1     John Deo   Four     75  female
1    2     Max Ruin  Three     85    male
2    3       Arnold  Three     55    male
3    4   Krish Star   Four     60  female
4    5    John Mike   Four     60  female
```

Exercise 4: Aggregation and Grouping

1. Question: "Write the code to group the DataFrame by the 'class' column and calculate the mean 'mark' for each group."
2. Question: "Write the code to count the number of students in each class."
3. Question: "Write the code to calculate the average mark for each gender."

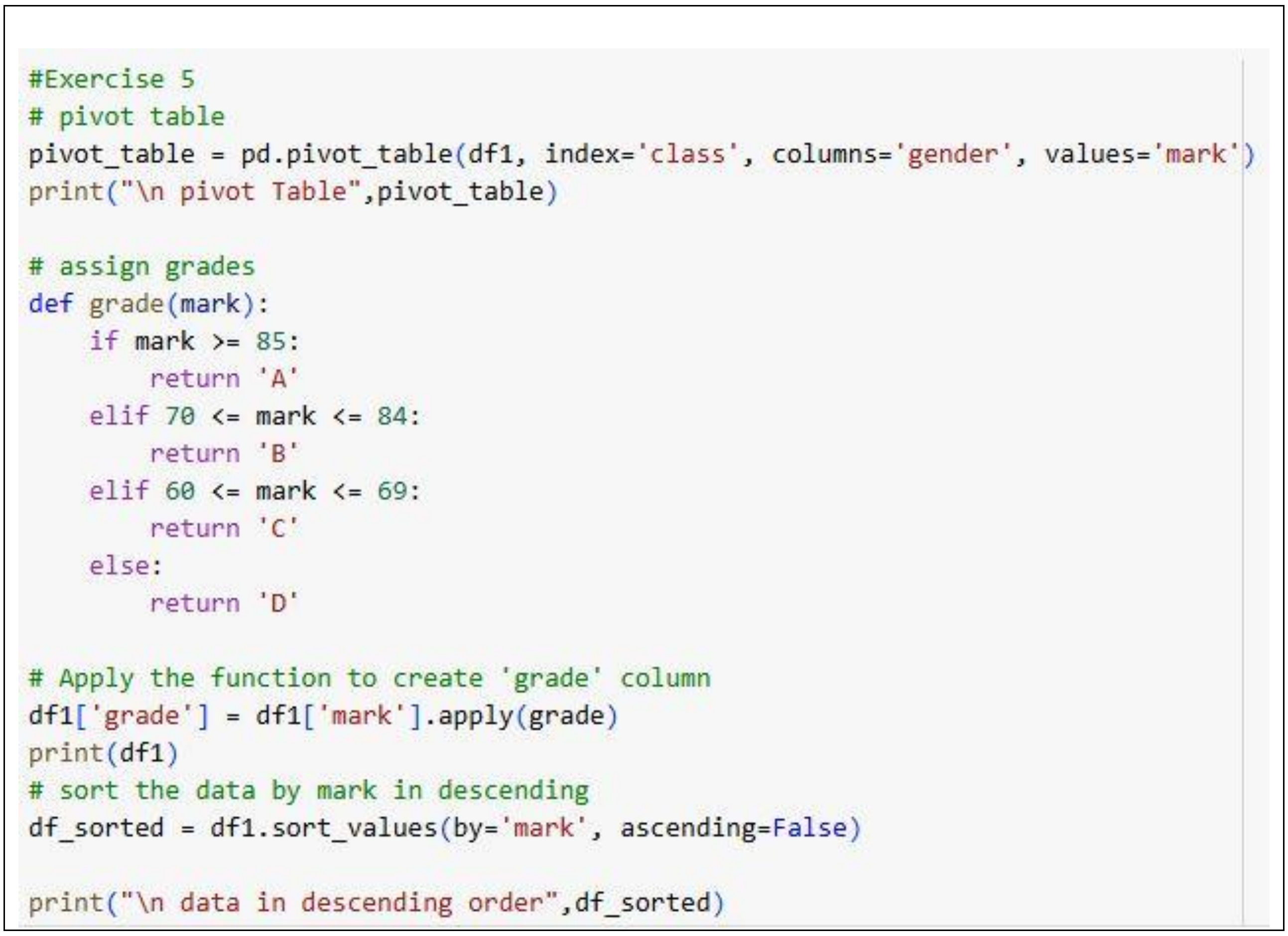


```python
#Exercise 4
print("\n mean mark by class",df1.groupby('class')['mark'].mean())
print("\n count of students",df1['class'].value_counts())
print("\n Average mark for each each gender",df1.groupby('gender')['mark'].mean())
```

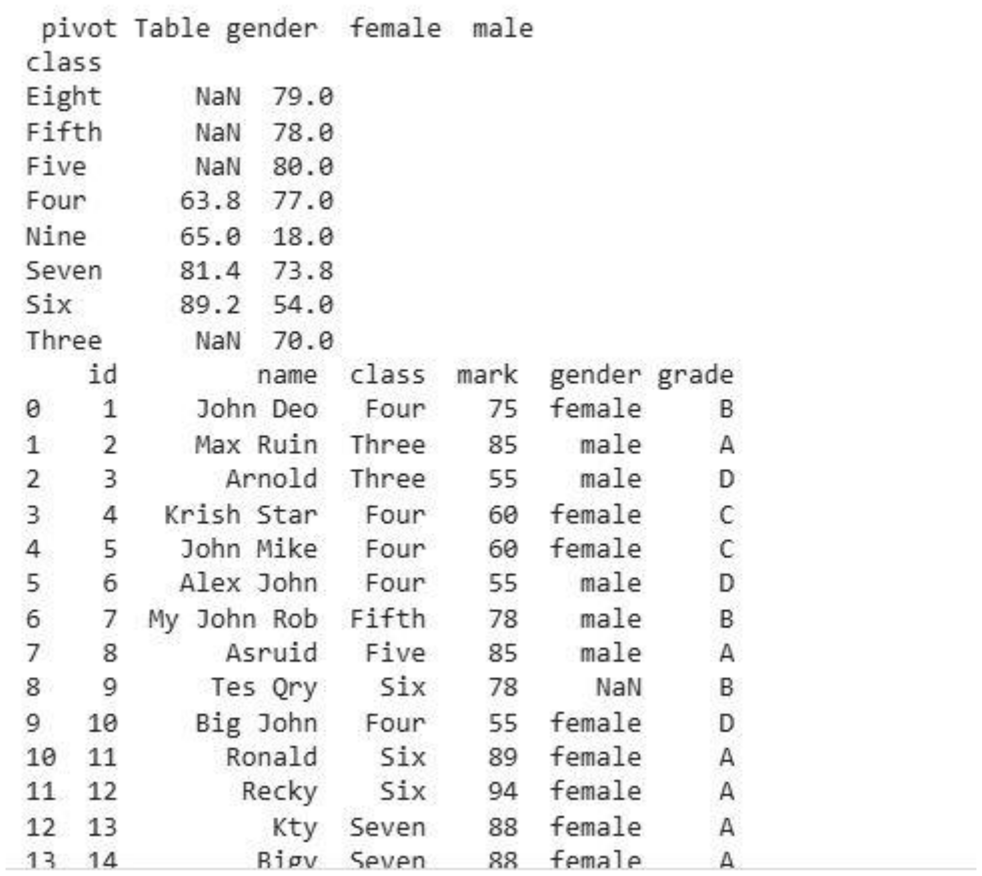
```
 mean mark by class class
Eight    79.000000
Fifth    78.000000
Five     80.000000
Four     68.750000
Nine     41.500000
Seven    77.600000
Six      82.571429
Three    73.666667
Name: mark, dtype: float64

 count of students class
Seven    10
Four      8
Six       7
Three     3
Nine      2
Five      2
Fifth     1
Eight     1
Name: count, dtype: int64

 Average mark for each each gender gender
female    77.312500
male      71.588235
Name: mark, dtype: float64
```

Exercise 5: Advanced Operations

1. Question: "Write the code to create a pivot table with 'class' as rows, 'gender' as columns, and 'mark' as values."
2. Question: "Write the code to create a new column 'grade' where marks >= 85 are 'A', 70-84 are 'B', 60-69 are 'C', and below 60 are 'D'."
3. Question: "Write the code to sort the DataFrame by 'mark' in descending order."



```python
#Exercise 5
# pivot table
pivot_table = pd.pivot_table(df1, index='class', columns='gender', values='mark')
print("\n pivot Table",pivot_table)

# assign grades
def grade(mark):
    if mark >= 85:
        return 'A'
    elif 70 <= mark <= 84:
        return 'B'
    elif 60 <= mark <= 69:
        return 'C'
    else:
        return 'D'

# Apply the function to create 'grade' column
df1['grade'] = df1['mark'].apply(grade)
print(df1)
# sort the data by mark in descending
df_sorted = df1.sort_values(by='mark', ascending=False)

print("\n data in descending order",df_sorted)
```



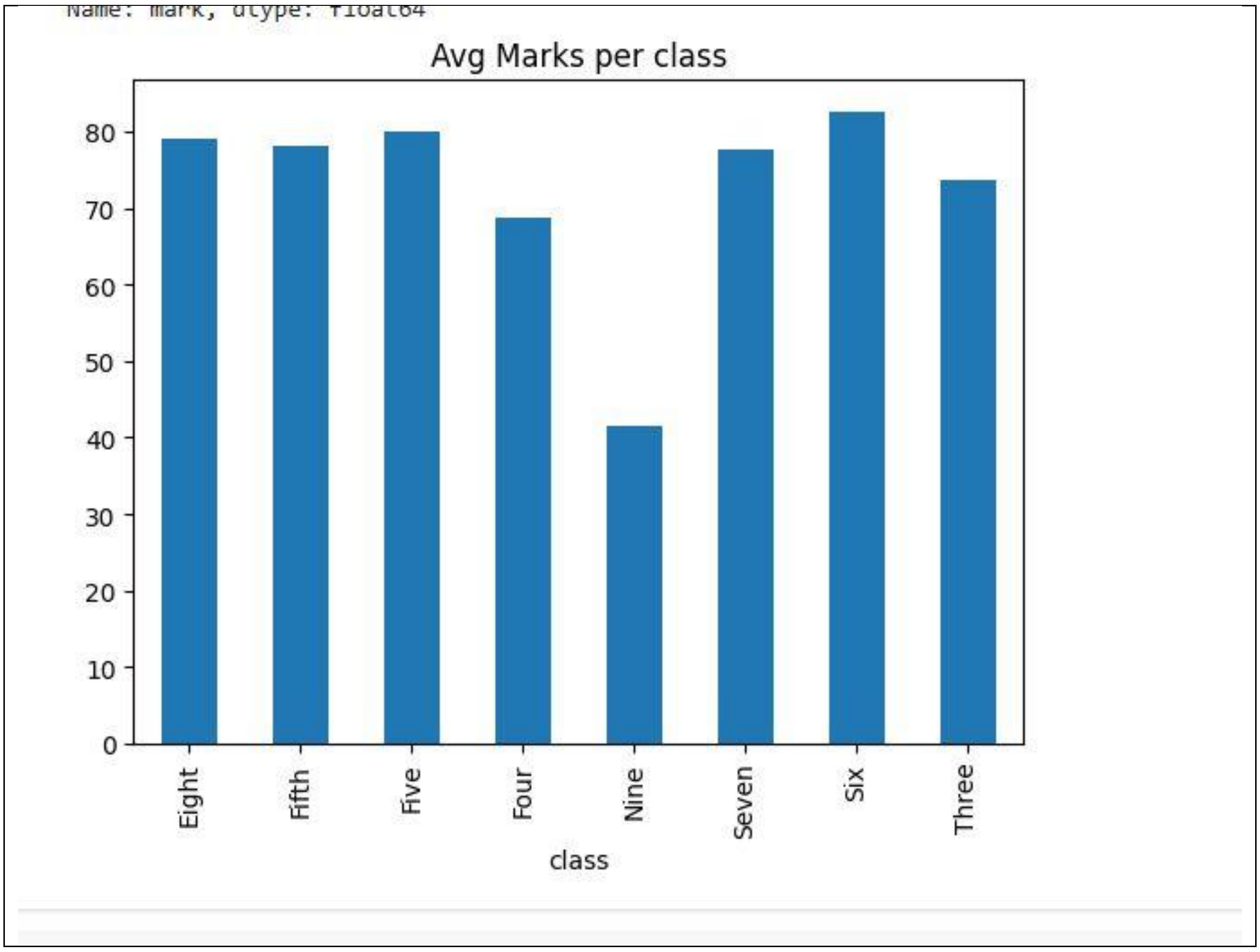
```
 pivot Table gender  female  male
class
Eight          NaN   79.0
Fifth          NaN   78.0
Five           NaN   80.0
Four          63.8   77.0
Nine          65.0   18.0
Seven         81.4   73.8
Six           89.2   54.0
Three          NaN   70.0
    id         name  class  mark  gender grade
0    1     John Deo   Four    75  female     B
1    2     Max Ruin  Three    85    male     A
2    3       Arnold  Three    55    male     D
3    4   Krish Star   Four    60  female     C
4    5    John Mike   Four    60  female     C
5    6    Alex John   Four    55    male     D
6    7  My John Rob  Fifth    78    male     B
7    8       Asruid   Five    85    male     A
8    9      Tes Qry    Six    78     NaN     B
9   10     Big John   Four    55  female     D
10  11       Ronald    Six    89  female     A
11  12        Recky    Six    94  female     A
12  13          Kty  Seven    88  female     A
13  14         Bigy  Seven    88  female     A
```

Exercise 6: Exporting Data

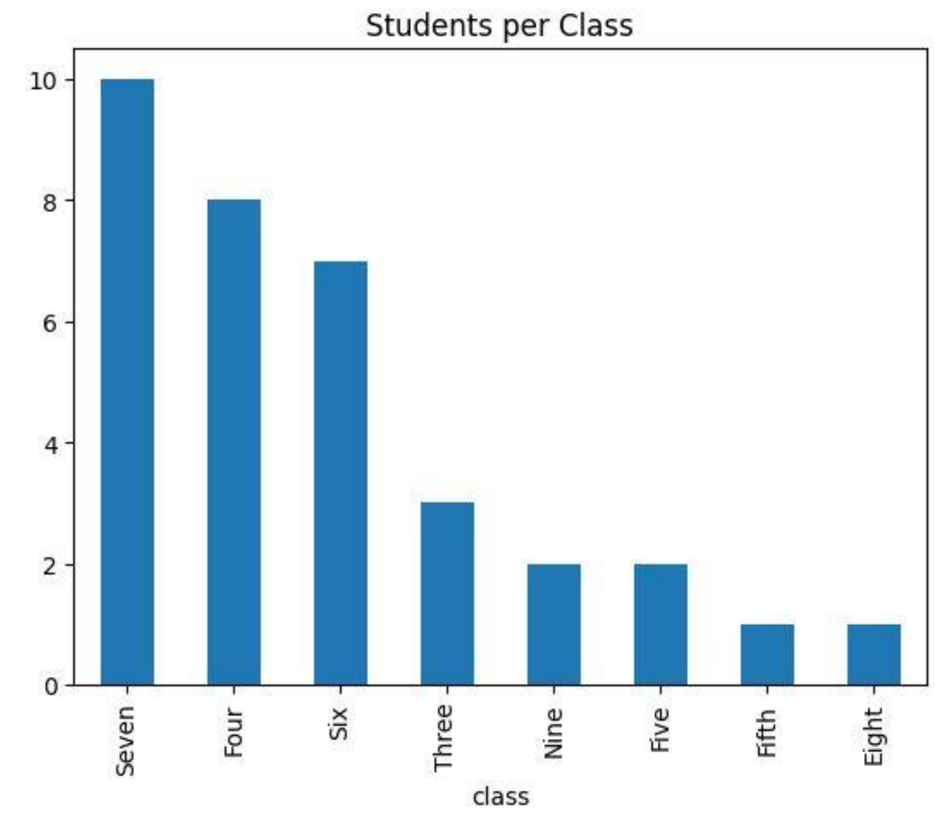
1. Question: "Write the code to save the DataFrame with the new 'grade' column to a new CSV file."

```
df1.to_csv('students_Updated.csv', index=False)
```

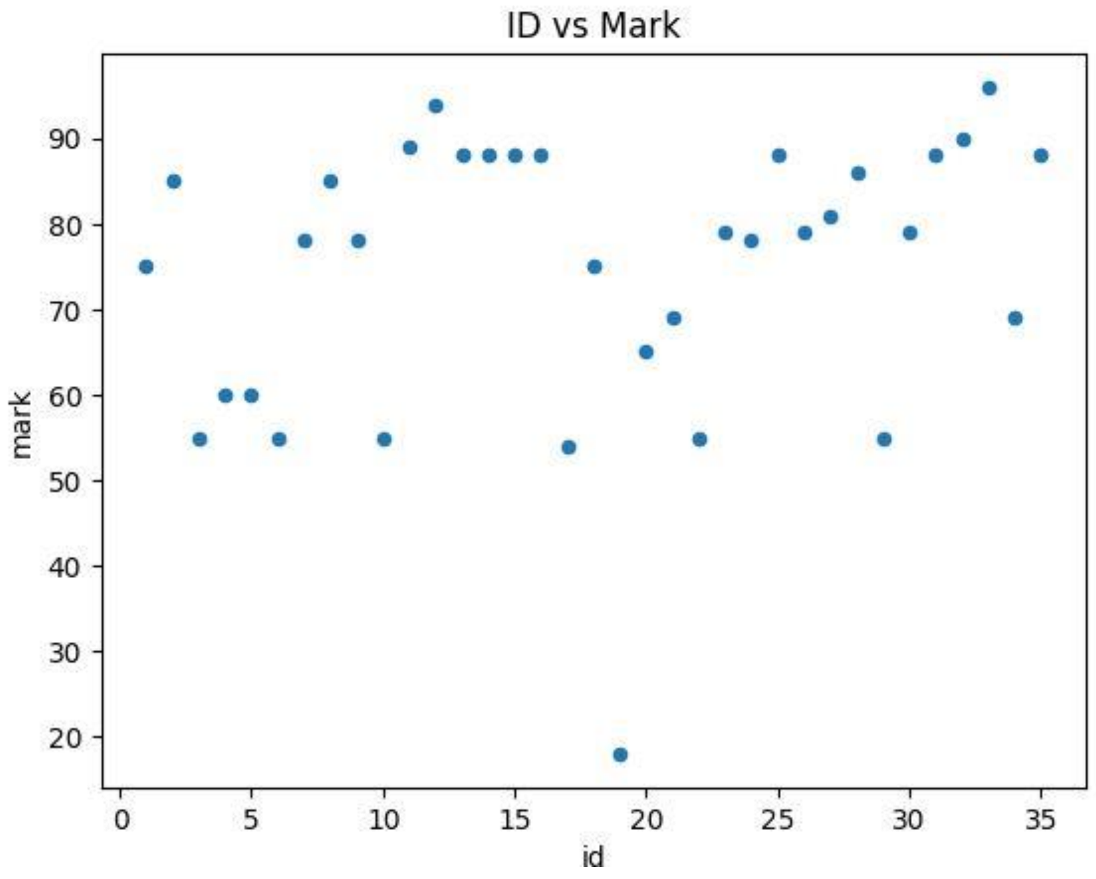
Exercise 7: If finished early try visualising the results



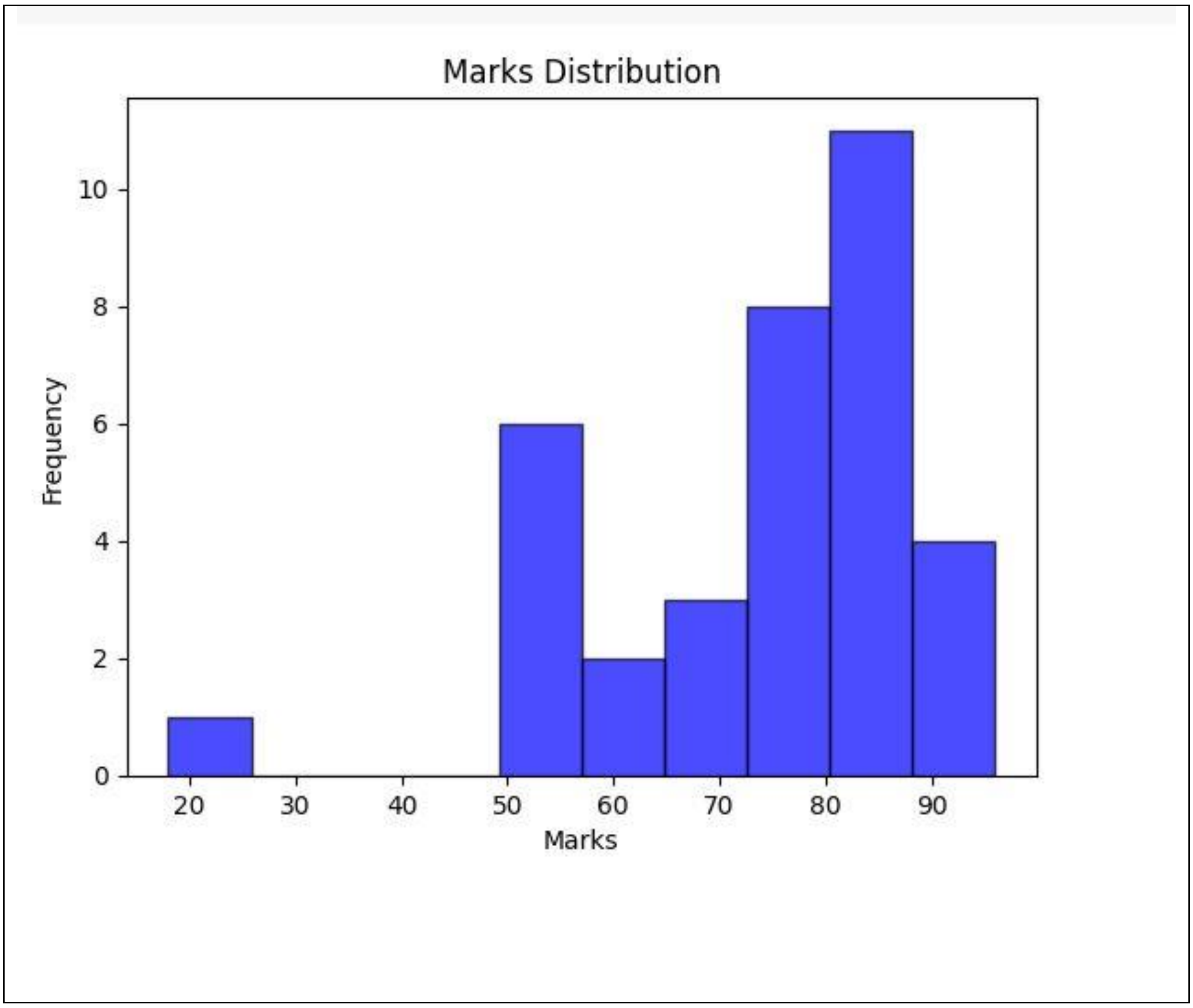
```
Name: mark, dtype: float64
```



Students per Class
10
8
6
4
2
0
Seven
Four
Six
Three
Nine
Five
Fifth
Eight
class



ID vs Mark
mark
90
80
70
60
50
40
30
20
0
5
10
15
20
25
30
35
id



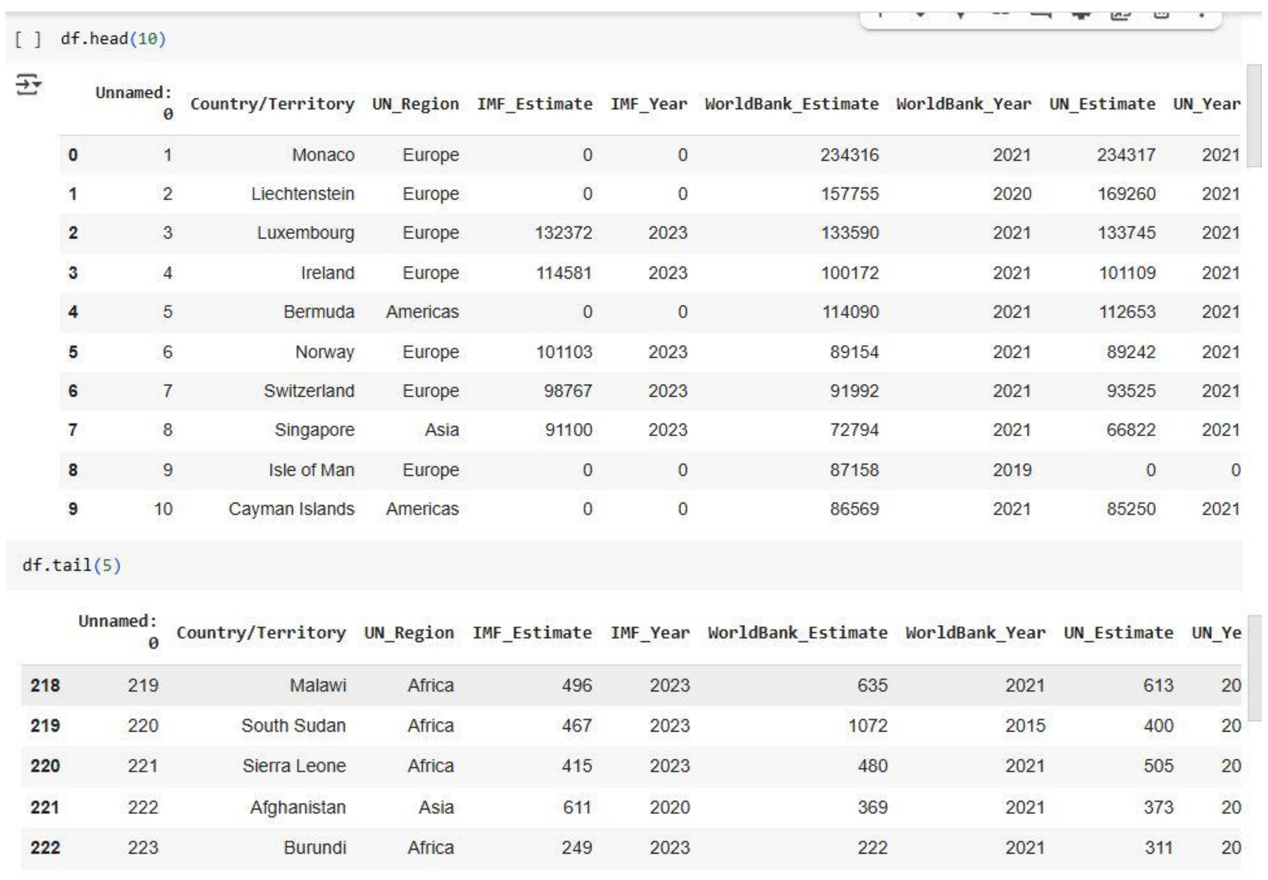
Marks Distribution
Frequency
Marks
0
2
4
6
8
10
20
30
40
50
60
70
80
90

Day 4: Task 1

Using the ‘GDP (nominal) per Capita.csv’ which can be downloaded from the shared Folder, complete the below exercises and paste your input and output. Work individually, but we will work and support each other in the room.

- Read and save the ‘GDP (nominal) per Capita’ data to a data frame called “df” in Jyputer notebook
- Print the first 10 rows
- Print the last 5 rows
- Print ‘Country/Territory’ and ‘UN_Region’ columns

```
df = pd.read_csv('GDP (nominal) per Capita.csv')
```

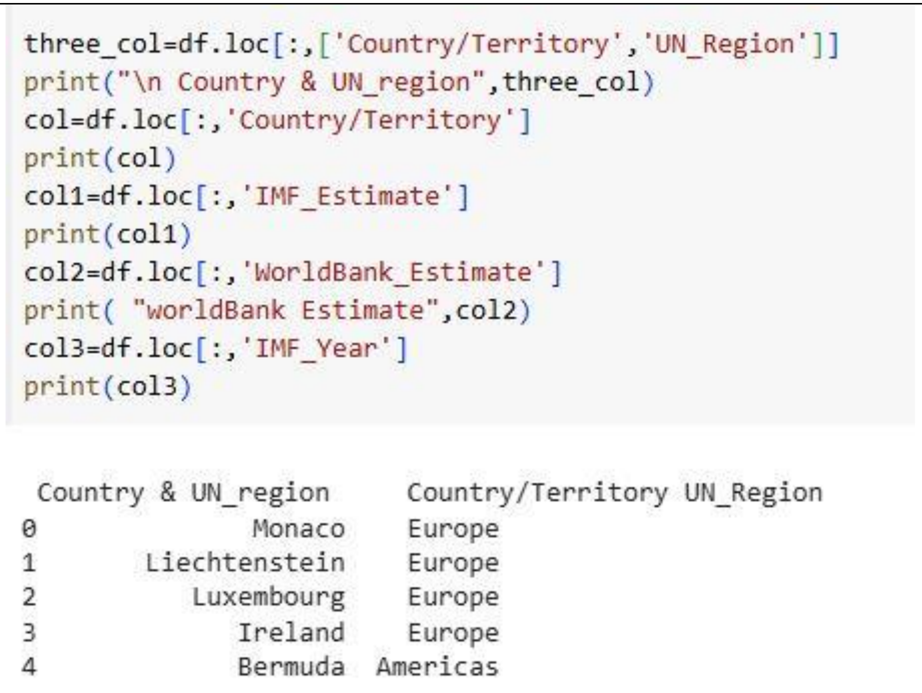


```
[ ] df.head(10)
```

```
   Unnamed:  Country/Territory  UN_Region  IMF_Estimate  IMF_Year  WorldBank_Estimate  WorldBank_Year  UN_Estimate  UN_Year
          0
0         1             Monaco     Europe             0         0              234316            2021       234317     2021
1         2      Liechtenstein     Europe             0         0              157755            2020       169260     2021
2         3         Luxembourg     Europe        132372      2023              133590            2021       133745     2021
3         4            Ireland     Europe        114581      2023              100172            2021       101109     2021
4         5            Bermuda   Americas             0         0              114090            2021       112653     2021
5         6             Norway     Europe        101103      2023               89154            2021        89242     2021
6         7        Switzerland     Europe         98767      2023               91992            2021        93525     2021
7         8          Singapore       Asia         91100      2023               72794            2021        66822     2021
8         9        Isle of Man     Europe             0         0               87158            2019            0        0
9        10     Cayman Islands   Americas             0         0               86569            2021        85250     2021
```

```
df.tail(5)
```

```
     Unnamed:  Country/Territory  UN_Region  IMF_Estimate  IMF_Year  WorldBank_Estimate  WorldBank_Year  UN_Estimate  UN_Ye
            0
218       219             Malawi     Africa           496      2023                 635            2021          613     20
219       220        South Sudan     Africa           467      2023                1072            2015          400     20
220       221       Sierra Leone     Africa           415      2023                 480            2021          505     20
221       222        Afghanistan       Asia           611      2020                 369            2021          373     20
222       223            Burundi     Africa           249      2023                 222            2021          311     20
```



```
three_col=df.loc[:,['Country/Territory','UN_Region']]
print("\n Country & UN_region",three_col)
col=df.loc[:,'Country/Territory']
print(col)
col1=df.loc[:,'IMF_Estimate']
print(col1)
col2=df.loc[:,'WorldBank_Estimate']
print( "worldBank Estimate",col2)
col3=df.loc[:,'IMF_Year']
print(col3)
```

```
 Country & UN_region     Country/Territory UN_Region
0               Monaco    Europe
1        Liechtenstein    Europe
2           Luxembourg    Europe
3              Ireland    Europe
4              Bermuda  Americas
```

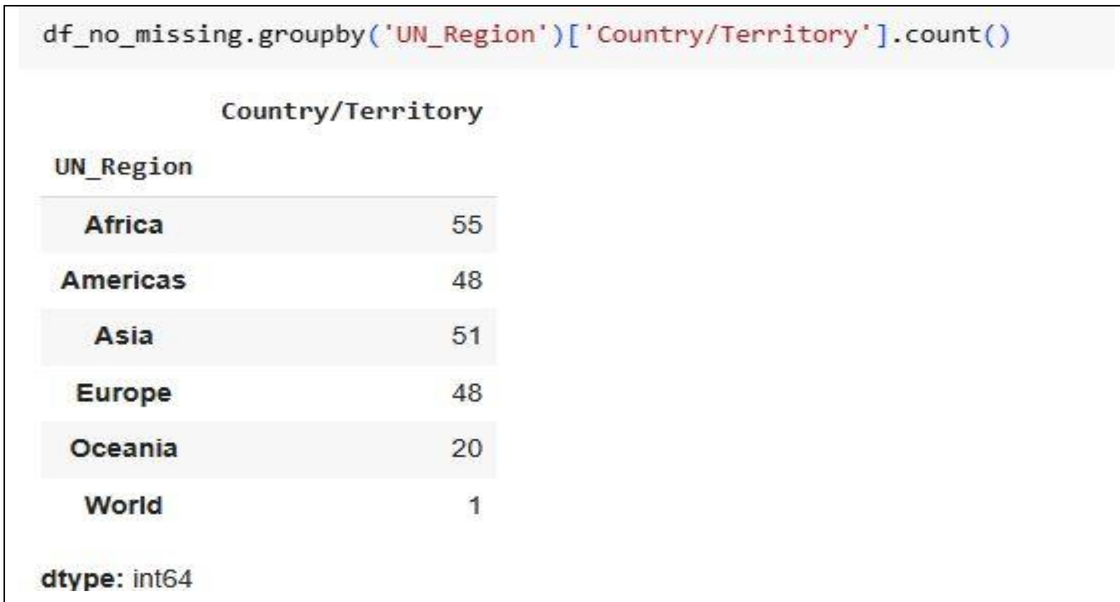
Day 4: Task 2

Back with 'GDP (nominal) per Capita'. As a group, import and work your way through the Day_4_Python_Activity.ipynb notebook which can be found on the shared Folder. There are questions to answer, but also opportunities to have fun with the data – paste your input and output below.

Once complete, and again as a group, work with some more data and have some fun – there is no set agenda for this section, other than to embed the skills developed this week. Paste your input and output below and upon return we'll discuss progress made.

Additional data found here.

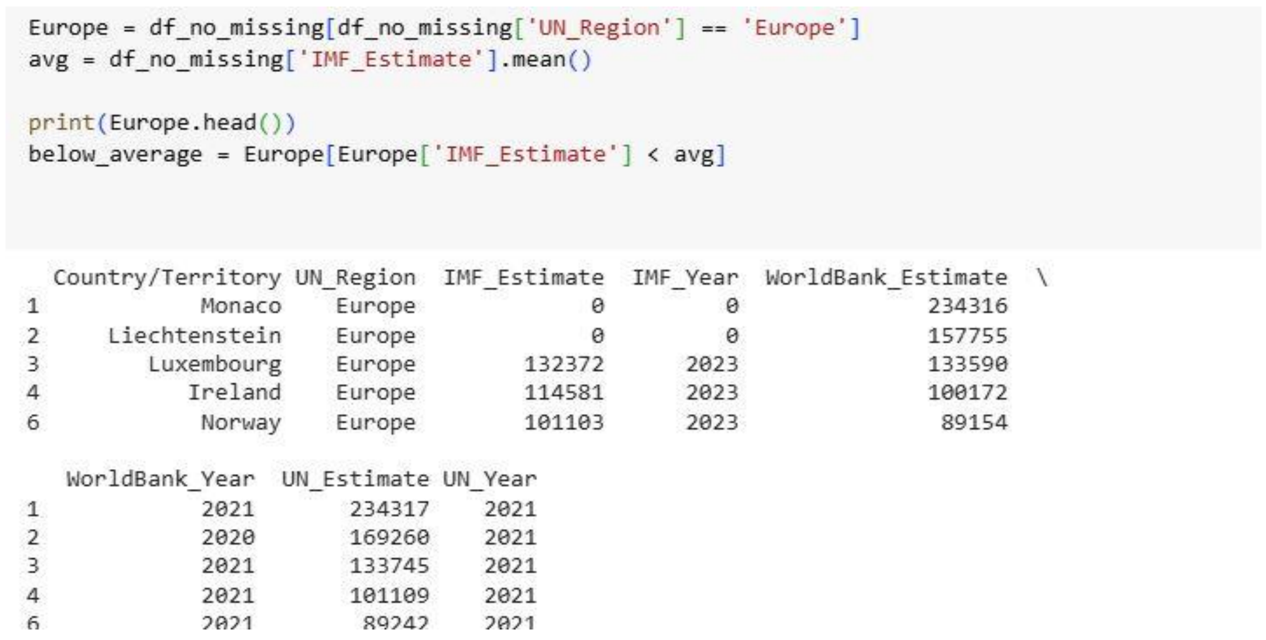
number of countries per region



```
df_no_missing.groupby('UN_Region')['Country/Territory'].count()
```

```
           Country/Territory
UN_Region
  Africa                  55
 Americas                 48
   Asia                   51
  Europe                  48
 Oceania                  20
  World                    1

dtype: int64
```

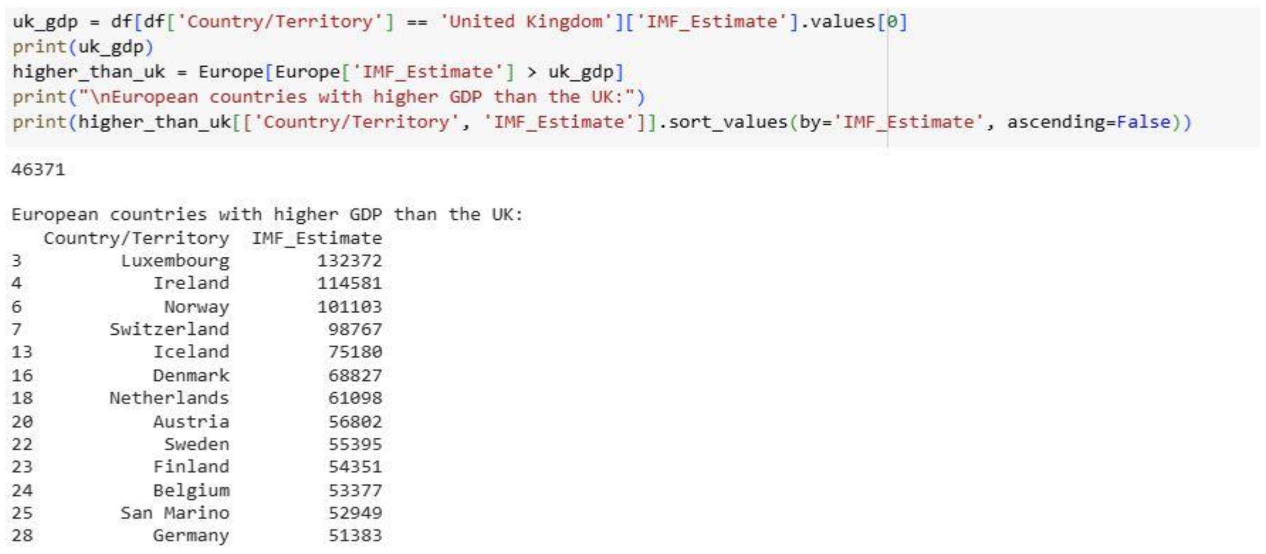
Countries in Europe below average



```
Europe = df_no_missing[df_no_missing['UN_Region'] == 'Europe']
avg = df_no_missing['IMF_Estimate'].mean()

print(Europe.head())
below_average = Europe[Europe['IMF_Estimate'] < avg]
```

```
  Country/Territory UN_Region  IMF_Estimate  IMF_Year  WorldBank_Estimate  \
1            Monaco    Europe             0         0              234316
2     Liechtenstein    Europe             0         0              157755
3        Luxembourg    Europe        132372      2023              133590
4           Ireland    Europe        114581      2023              100172
6            Norway    Europe        101103      2023               89154

   WorldBank_Year  UN_Estimate UN_Year
1            2021       234317    2021
2            2020       169260    2021
3            2021       133745    2021
4            2021       101109    2021
6            2021        89242    2021
```

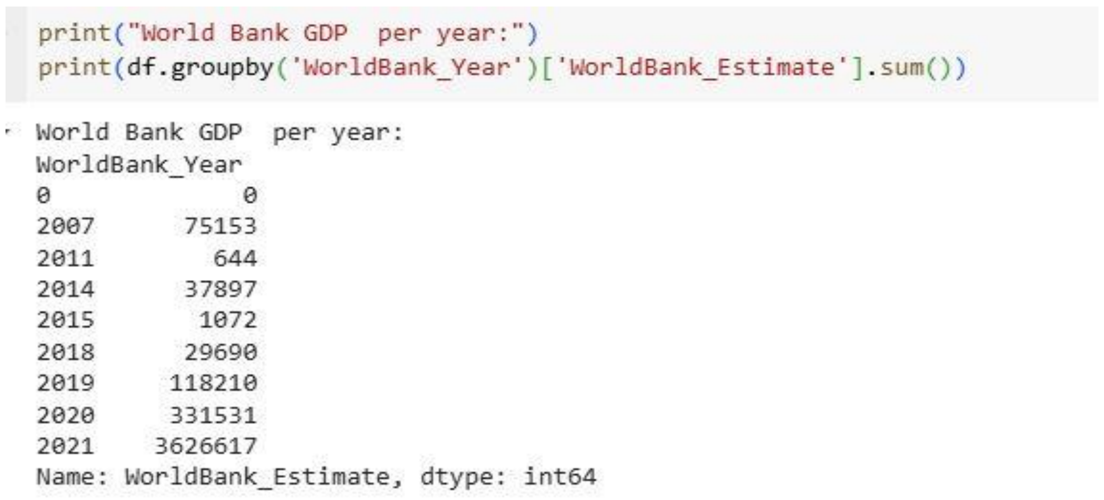
Which countries in Europe has higher GDP than UK?



```
uk_gdp = df[df['Country/Territory'] == 'United Kingdom']['IMF_Estimate'].values[0]
print(uk_gdp)
higher_than_uk = Europe[Europe['IMF_Estimate'] > uk_gdp]
print("\nEuropean countries with higher GDP than the UK:")
print(higher_than_uk[['Country/Territory', 'IMF_Estimate']].sort_values(by='IMF_Estimate', ascending=False))
```

```
46371

European countries with higher GDP than the UK:
   Country/Territory  IMF_Estimate
3         Luxembourg        132372
4            Ireland        114581
6             Norway        101103
7        Switzerland         98767
13           Iceland         75180
16           Denmark         68827
18       Netherlands         61098
20           Austria         56802
22            Sweden         55395
23           Finland         54351
24           Belgium         53377
25        San Marino         52949
28           Germany         51383
```

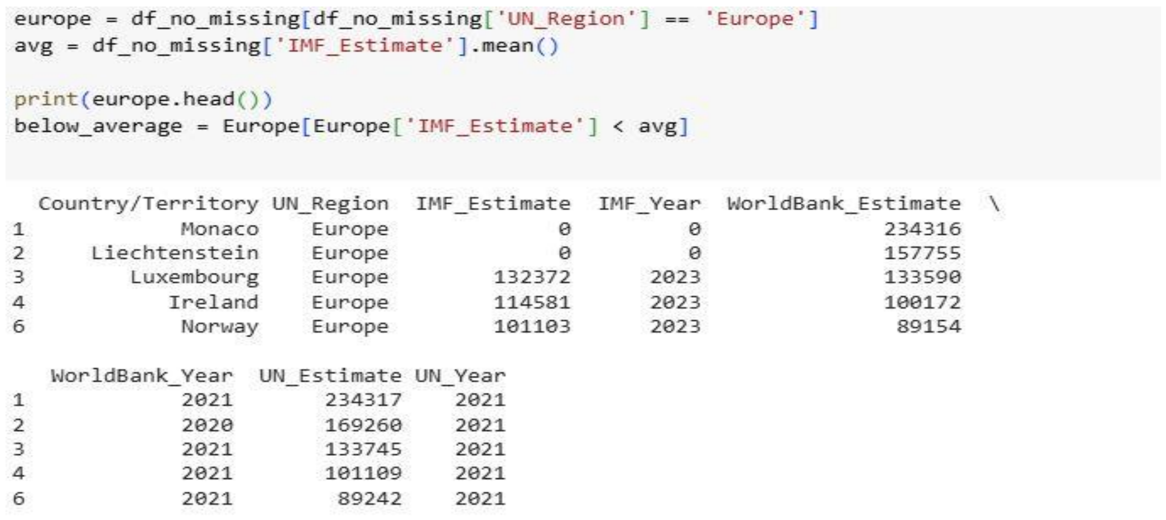
Groupby()



```
print("World Bank GDP  per year:")
print(df.groupby('WorldBank_Year')['WorldBank_Estimate'].sum())
```

```
World Bank GDP  per year:
WorldBank_Year
0             0
2007      75153
2011        644
2014      37897
2015       1072
2018      29690
2019     118210
2020     331531
2021    3626617
Name: WorldBank_Estimate, dtype: int64
```

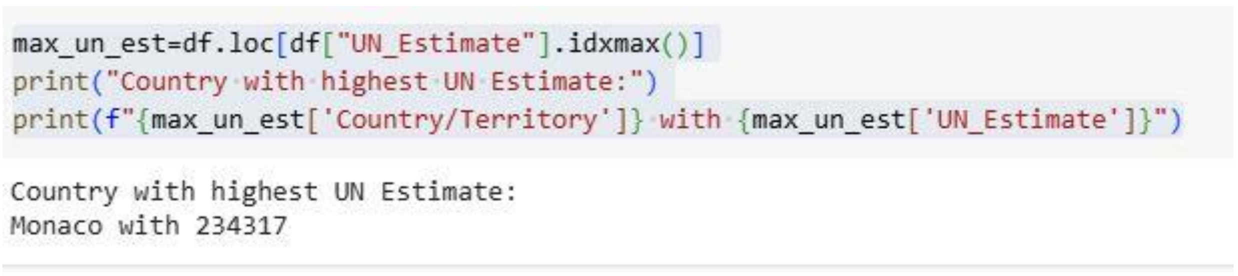
Which countries below average by IMF world estimate?



```
europe = df_no_missing[df_no_missing['UN_Region'] == 'Europe']
avg = df_no_missing['IMF_Estimate'].mean()

print(europe.head())
below_average = Europe[Europe['IMF_Estimate'] < avg]
```

```
  Country/Territory UN_Region  IMF_Estimate  IMF_Year  WorldBank_Estimate  \
1            Monaco    Europe             0         0              234316
2     Liechtenstein    Europe             0         0              157755
3        Luxembourg    Europe        132372      2023              133590
4           Ireland    Europe        114581      2023              100172
6            Norway    Europe        101103      2023               89154

   WorldBank_Year  UN_Estimate  UN_Year
1            2021       234317     2021
2            2020       169260     2021
3            2021       133745     2021
4            2021       101109     2021
6            2021        89242     2021
```

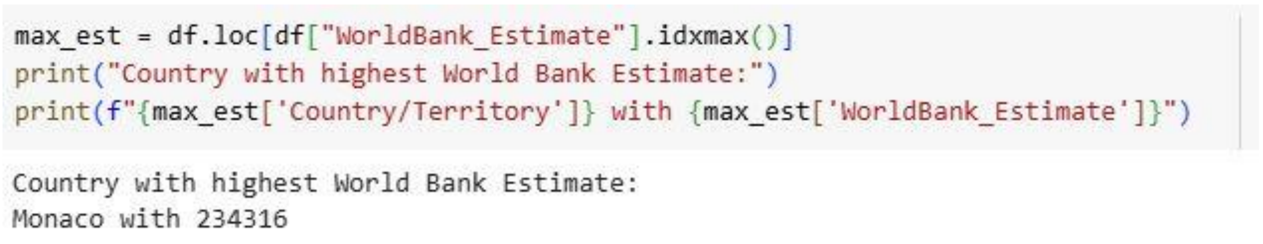
Which country has highest UN Estimate?



```
max_un_est=df.loc[df["UN_Estimate"].idxmax()]
print("Country with highest UN Estimate:")
print(f"{max_un_est['Country/Territory']} with {max_un_est['UN_Estimate']}")
```

```
Country with highest UN Estimate:
Monaco with 234317
```

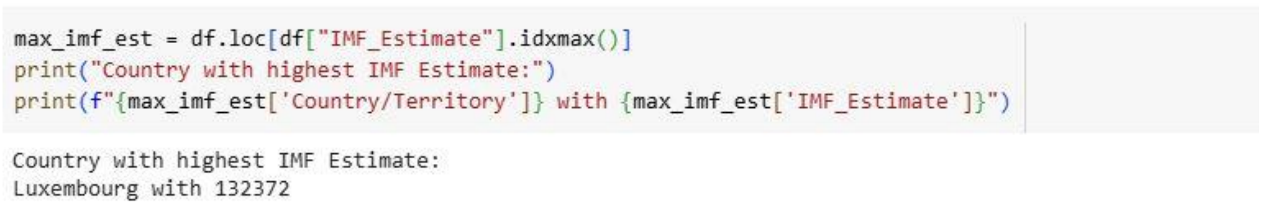
Which country has highest World bank Estimate?



```
max_est = df.loc[df["WorldBank_Estimate"].idxmax()]
print("Country with highest World Bank Estimate:")
print(f"{max_est['Country/Territory']} with {max_est['WorldBank_Estimate']}")
```

```
Country with highest World Bank Estimate:
Monaco with 234316
```

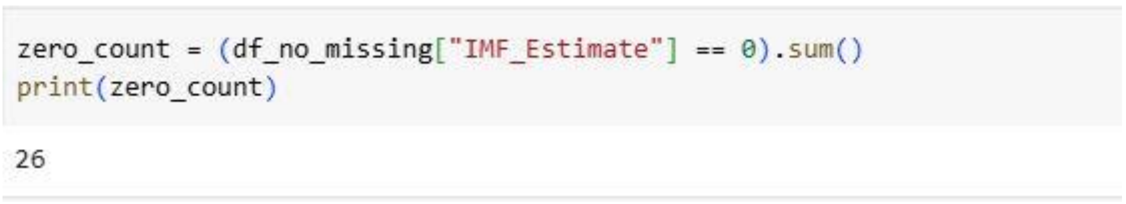
Which country has highest IMF Estimate?



```
max_imf_est = df.loc[df["IMF_Estimate"].idxmax()]
print("Country with highest IMF Estimate:")
print(f"{max_imf_est['Country/Territory']} with {max_imf_est['IMF_Estimate']}")
```

```
Country with highest IMF Estimate:
Luxembourg with 132372
```

IMF estimate 0 values



```
zero_count = (df_no_missing["IMF_Estimate"] == 0).sum()
print(zero_count)
```

```
26
```

Course Notes

It is recommended to take notes from the course, use the space below to do so, or use the revision guide shared with the class:

We have included a range of additional links to further resources and information that you may find useful, these can be found within your revision guide.

END OF WORKBOOK

Please check through your work thoroughly before submitting and update the table of contents if required.

Please send your completed work booklet to your trainer.

Panda youtube link day3
https://www.youtube.com/watch?v=iGFdh6_FePU
https://www.youtube.com/@gilesmcmullen
https://requests.readthedocs.io/en/latest/
https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf
Portfolio project video day 4
https://www.youtube.com/watch?v=-E7nMqPVmyQ
pandas library
https://matplotlib.org/stable/index.html
https://seaborn.pydata.org/
https://numpy.org/doc/
https://plotly.com/python/