Just IT

B2W group

Apprenticeships | Training | Recruitment

Data Technician

Name:

Course Date:

Table of contents

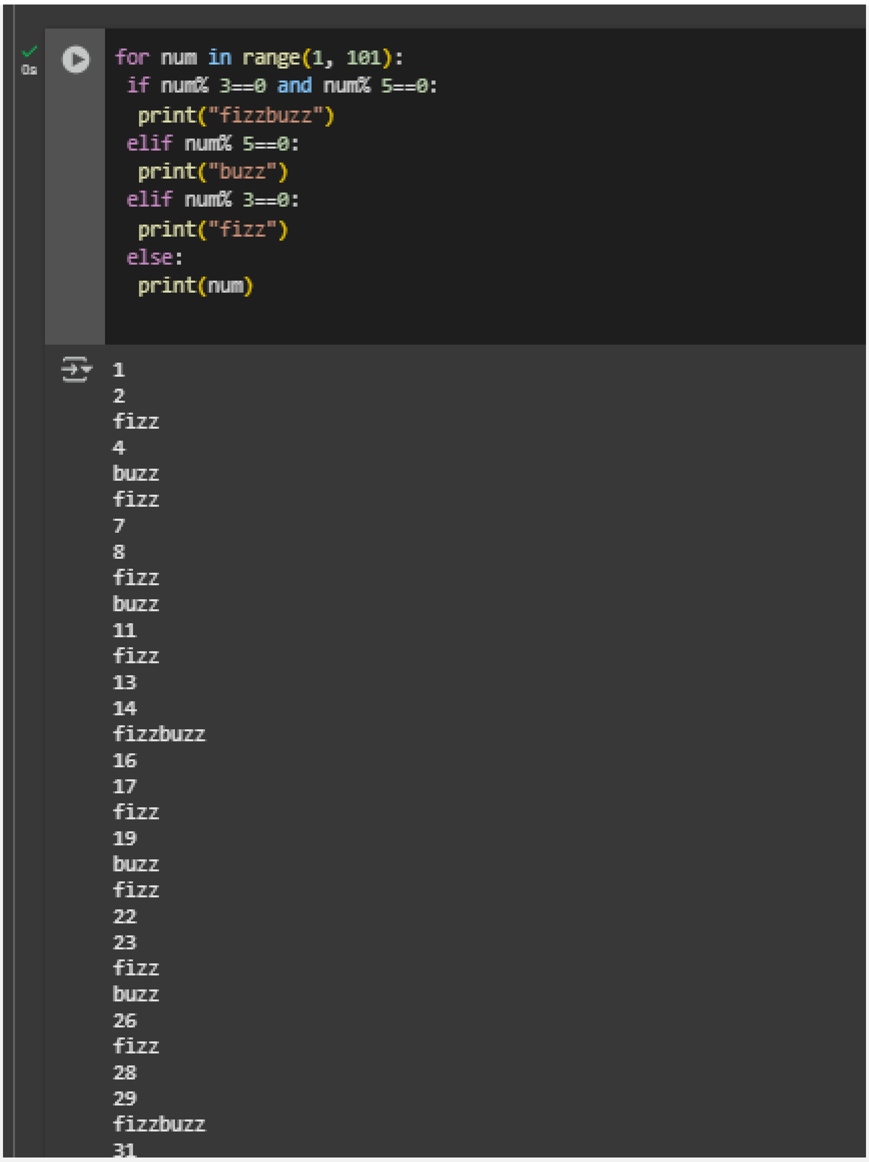
Day 2: Task 1

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 num in range(1, 101):
 if num% 3==0 and num% 5==0:
  print("fizzbuzz")
 elif num% 5==0:
  print("buzz")
 elif num% 3==0:
  print("fizz")
 else:
  print(num)
```

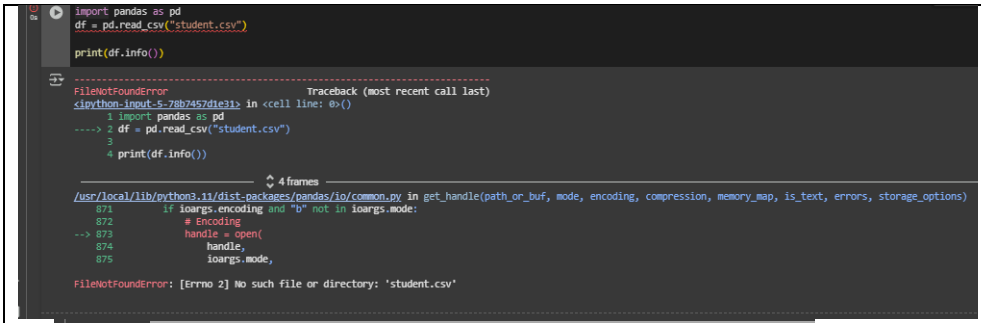
```
1
2
fizz
4
buzz
fizz
7
8
fizz
buzz
11
fizz
13
14
fizzbuzz
16
17
fizz
19
buzz
fizz
22
23
fizz
buzz
26
fizz
28
29
fizzbuzz
31
```

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

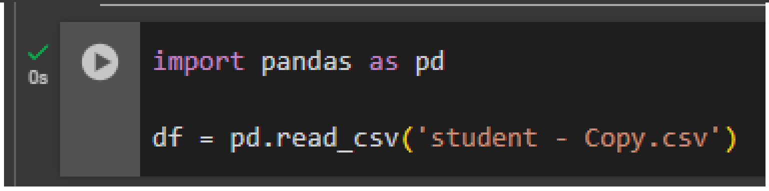
1. Question: "Write the code to read a CSV file into a Pandas DataFrame."
2. Question: "Write the code to display the first 5 rows of the DataFrame."
3. Question: "Write the code to get the information about the DataFrame."
4. Question: "Write the code to get summary statistics for the DataFrame."



```
import pandas as pd
df = pd.read_csv("student.csv")

print(df.info())
```

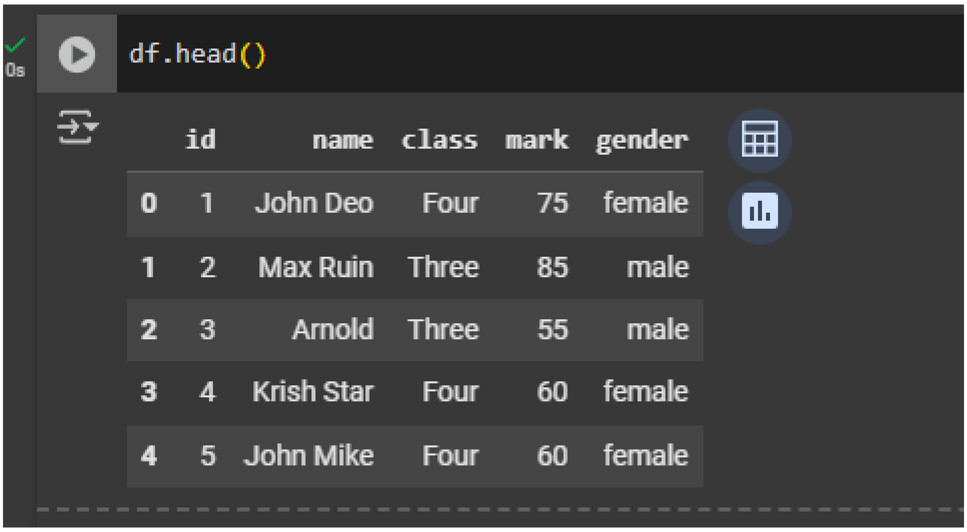
```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-5-78b7457d1e31> in <cell line: 0>()
      1 import pandas as pd
----> 2 df = pd.read_csv("student.csv")
      3 
      4 print(df.info())

4 frames
/usr/local/lib/python3.11/dist-packages/pandas/io/common.py in get_handle(path_or_buf, mode, encoding, compression, memory_map, is_text, errors, storage_options)
    871         if ioargs.encoding and "b" not in ioargs.mode:
    872             # Encoding
--> 873             handle = open(
    874                 handle,
    875                 ioargs.mode,

FileNotFoundError: [Errno 2] No such file or directory: 'student.csv'
```

1)



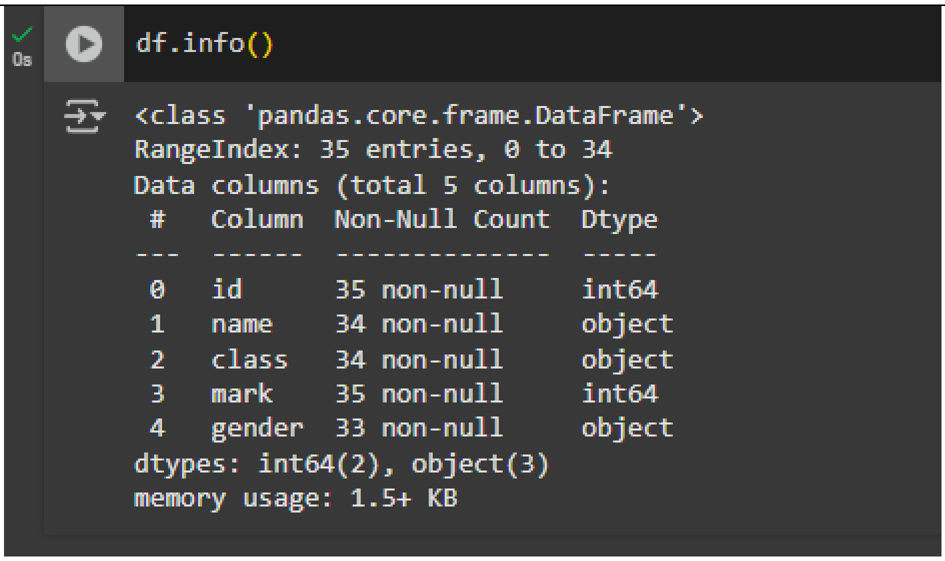
```
import pandas as pd

df = pd.read_csv('student - Copy.csv')
```

2)



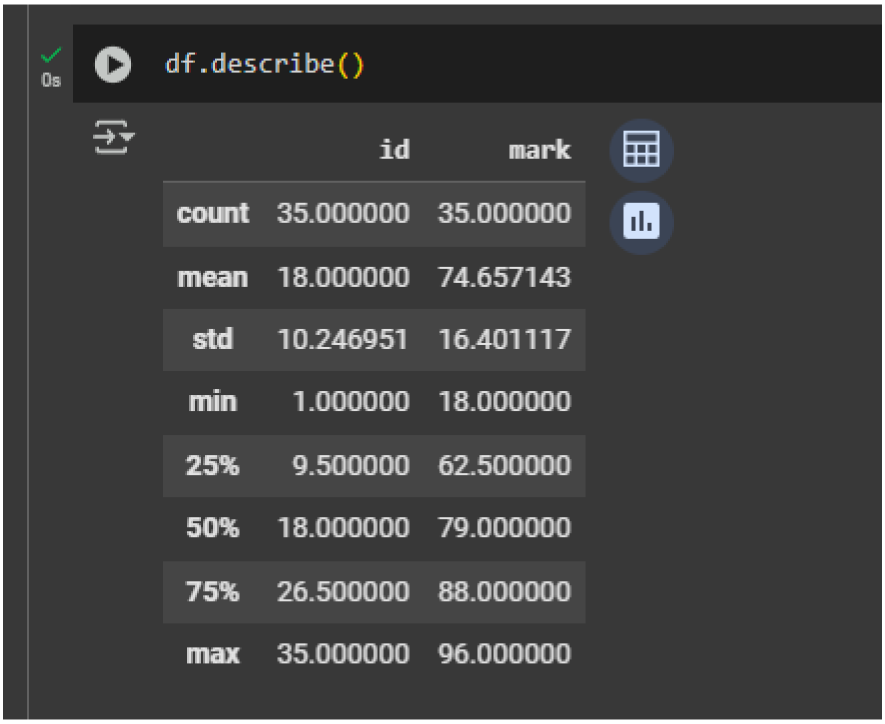
```
df.head()
```

```
   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
```

3) 

```
df.info()
```

```
<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
```

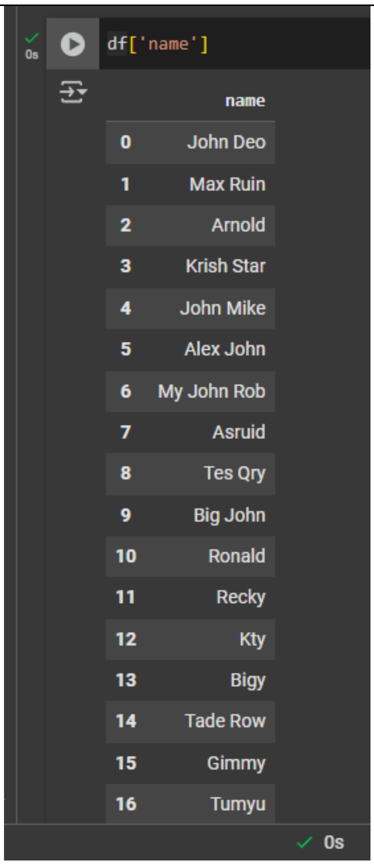
4) 

```
df.describe()
```

```
              id       mark
count  35.000000  35.000000
mean   18.000000  74.657143
std    10.246951  16.401117
min     1.000000  18.000000
25%     9.500000  62.500000
50%    18.000000  79.000000
75%    26.500000  88.000000
max    35.000000  96.000000
```

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'."

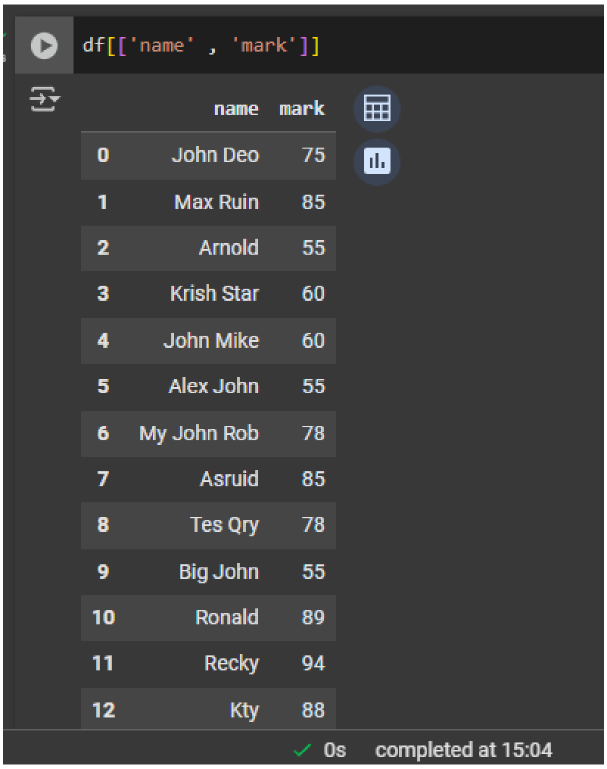


```
df['name']
```

```
          name
0     John Deo
1     Max Ruin
2       Arnold
3   Krish Star
4    John Mike
5    Alex John
6  My John Rob
7       Asruid
8      Tes Qry
9     Big John
10      Ronald
11       Recky
12         Kty
13        Bigy
14    Tade Row
15       Gimmy
16       Tumyu
```

✓ 0s

1)

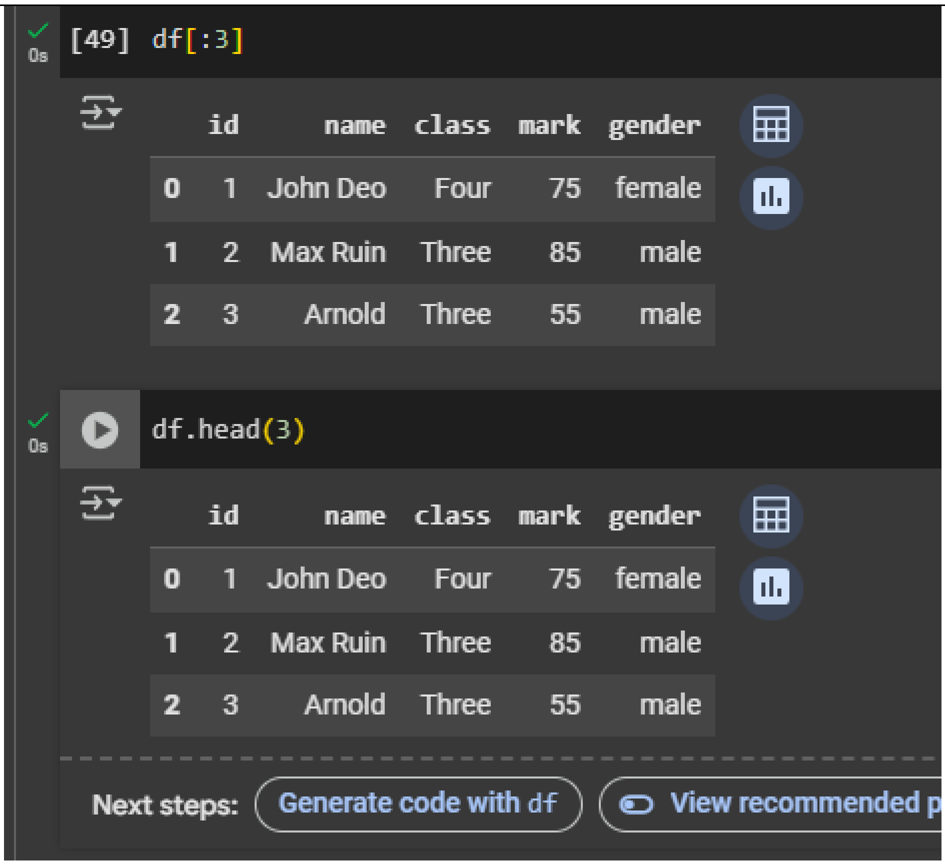


```
df[['name' , 'mark']]
```

```
          name  mark
0     John Deo    75
1     Max Ruin    85
2       Arnold    55
3   Krish Star    60
4    John Mike    60
5    Alex John    55
6  My John Rob    78
7       Asruid    85
8      Tes Qry    78
9     Big John    55
10      Ronald    89
11       Recky    94
12         Kty    88
```

✓ 0s completed at 15:04

2)



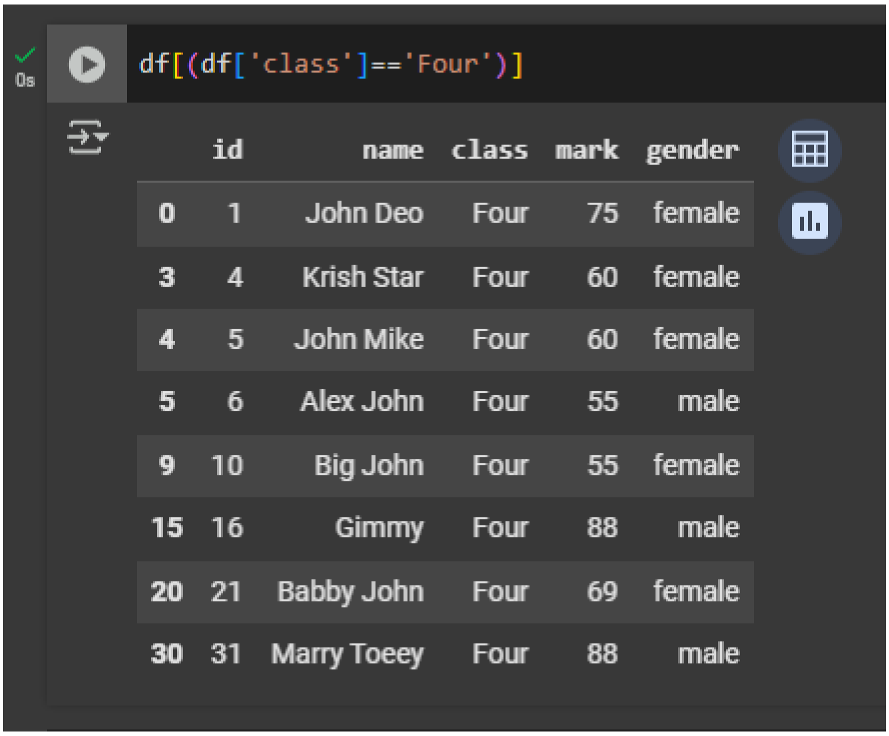
```
[49] df[:3]
```

```
   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
```

```
df.head(3)
```

```
   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)



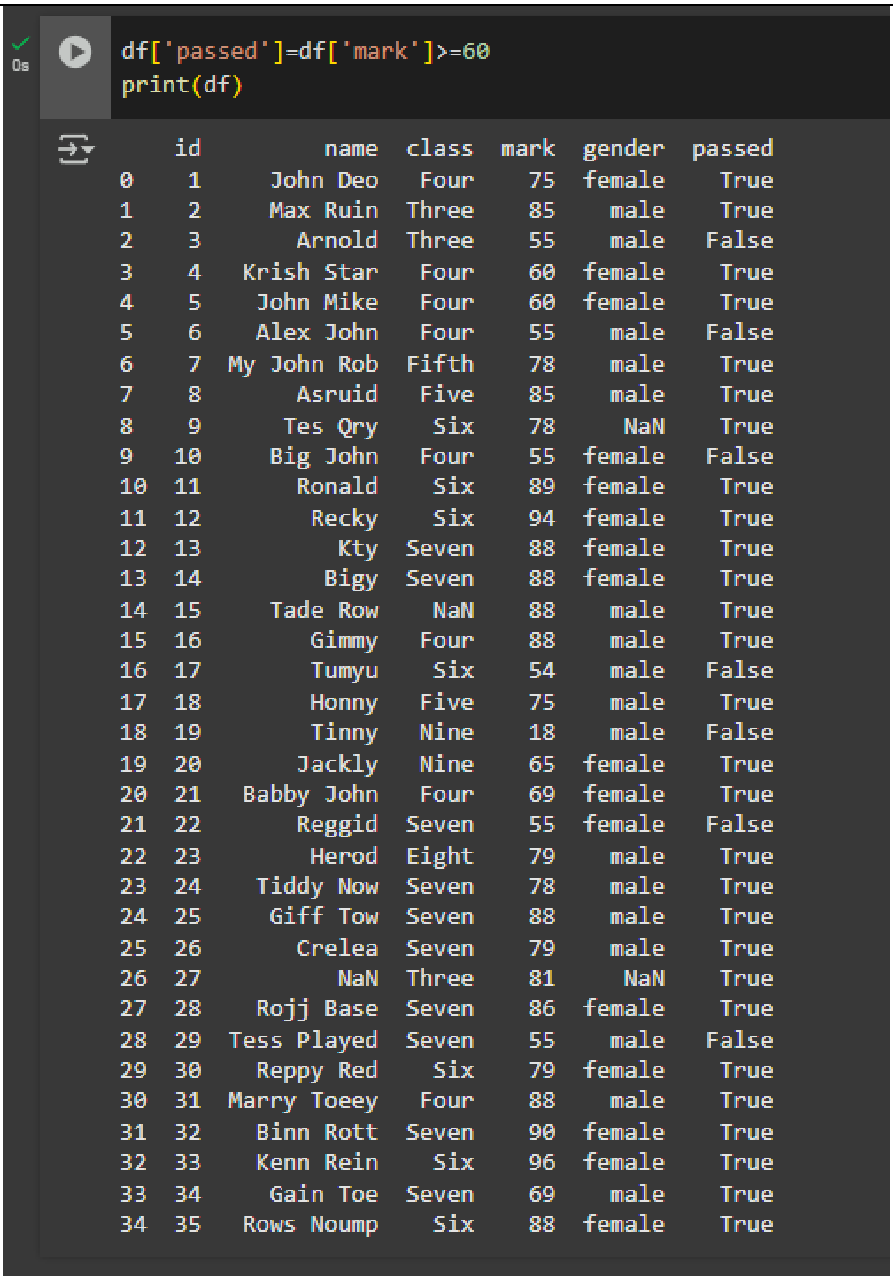
```
df[(df['class']=='Four')]
```

```
    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
```

4)

Exercise 3: Data Manipulation

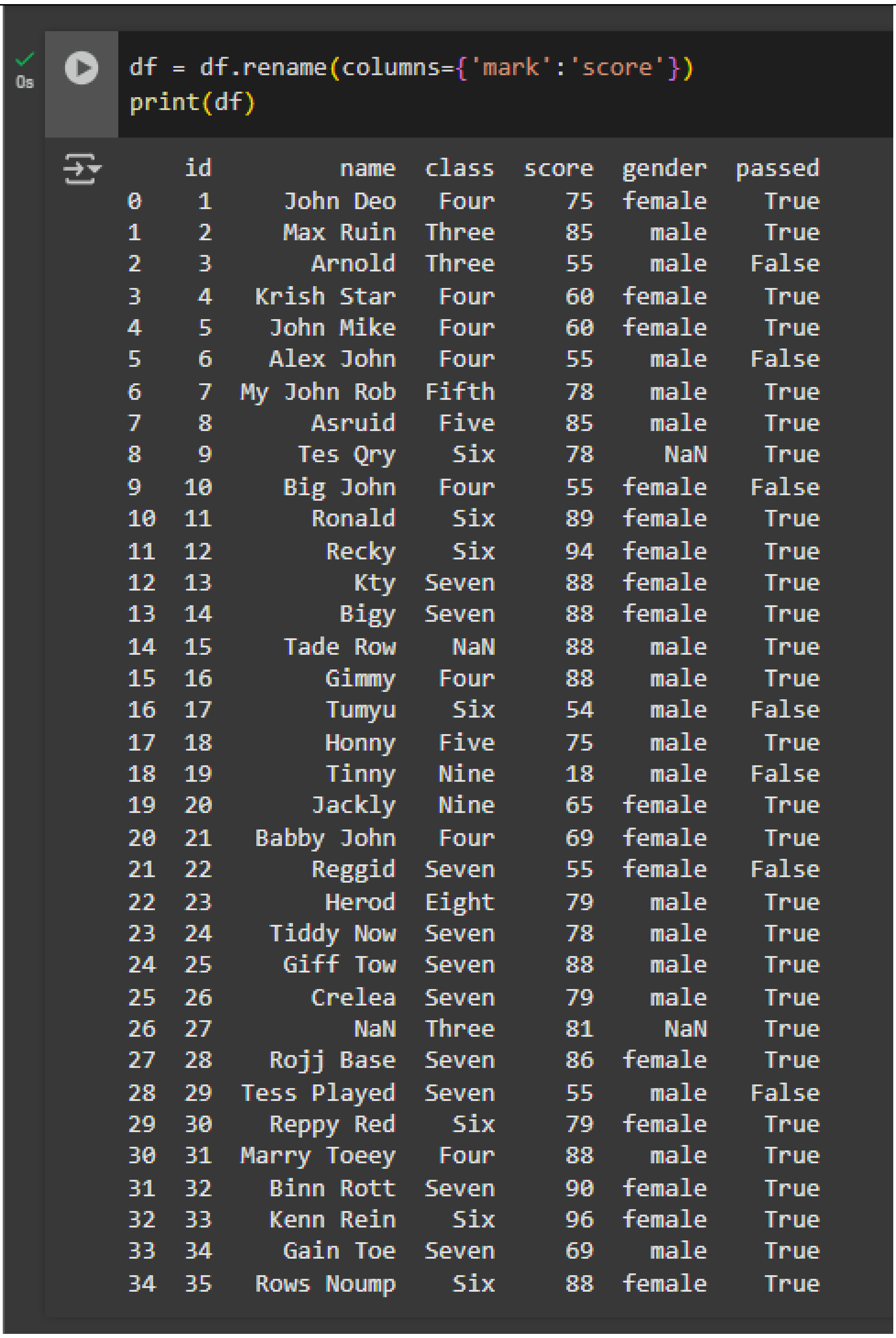
1. Question: "Write the code to add a new column 'passed' that indicates whether the student passed (mark >= 60)."
2. Question: "Write the code to rename the 'mark' column to 'score'."
3. Question: "Write the code to drop the 'passed' column."



```
df['passed']=df['mark']>=60
print(df)
```

```
    id         name  class  mark  gender  passed
0    1     John Deo   Four    75  female    True
1    2     Max Ruin  Three    85    male    True
2    3       Arnold  Three    55    male   False
3    4   Krish Star   Four    60  female    True
4    5    John Mike   Four    60  female    True
5    6    Alex John   Four    55    male   False
6    7  My John Rob  Fifth    78    male    True
7    8       Asruid   Five    85    male    True
8    9      Tes Qry    Six    78     NaN    True
9   10     Big John   Four    55  female   False
10  11       Ronald    Six    89  female    True
11  12        Recky    Six    94  female    True
12  13          Kty  Seven    88  female    True
13  14         Bigy  Seven    88  female    True
14  15     Tade Row    NaN    88    male    True
15  16        Gimmy   Four    88    male    True
16  17        Tumyu    Six    54    male   False
17  18        Honny   Five    75    male    True
18  19        Tinny   Nine    18    male   False
19  20       Jackly   Nine    65  female    True
20  21   Babby John   Four    69  female    True
21  22       Reggid  Seven    55  female   False
22  23        Herod  Eight    79    male    True
23  24    Tiddy Now  Seven    78    male    True
24  25     Giff Tow  Seven    88    male    True
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
```

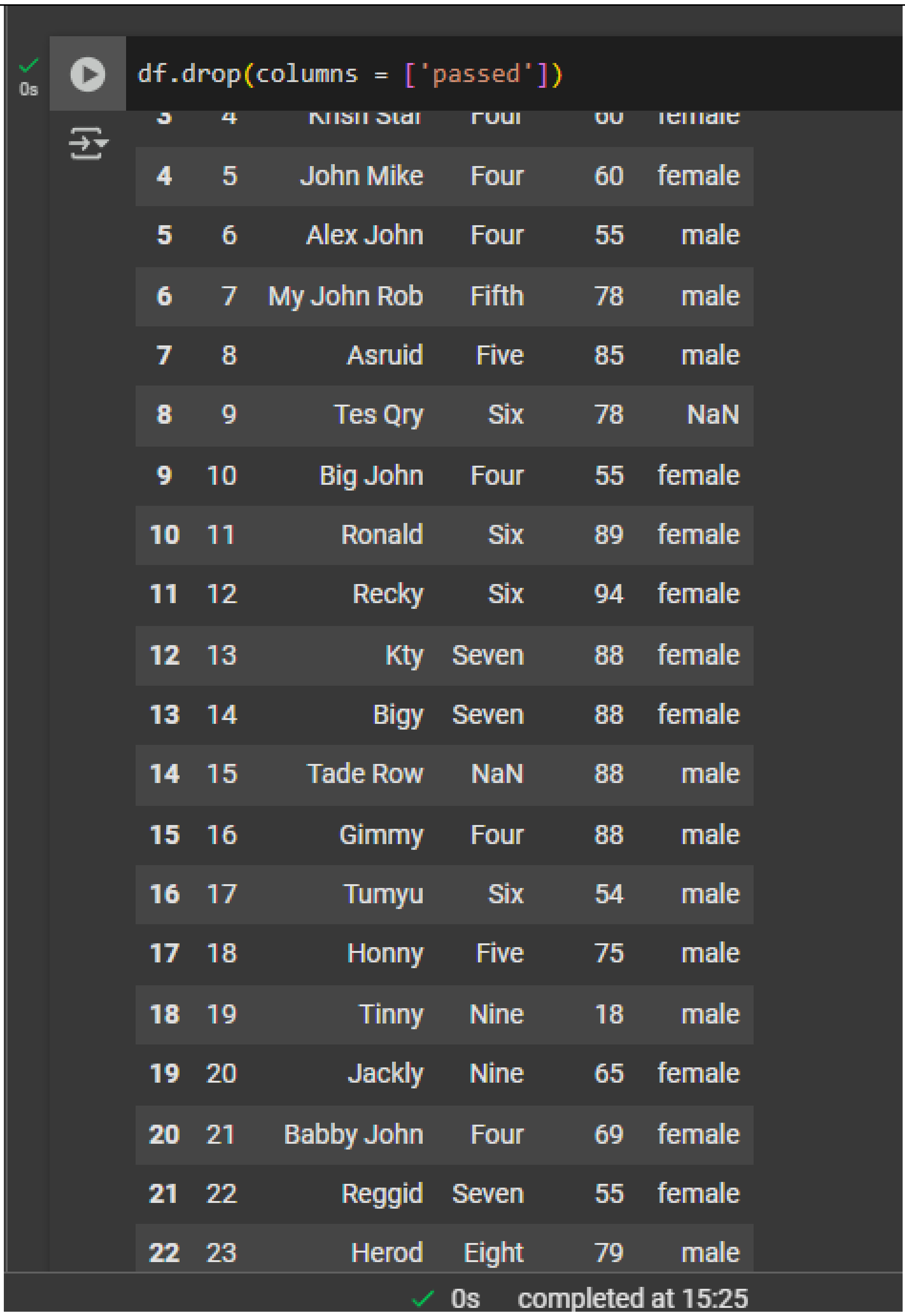
1)



```
df = df.rename(columns={'mark':'score'})
print(df)
```

```
    id         name  class  score  gender  passed
0    1     John Deo   Four     75  female    True
1    2     Max Ruin  Three     85    male    True
2    3       Arnold  Three     55    male   False
3    4   Krish Star   Four     60  female    True
4    5    John Mike   Four     60  female    True
5    6    Alex John   Four     55    male   False
6    7  My John Rob  Fifth     78    male    True
7    8       Asruid   Five     85    male    True
8    9      Tes Qry    Six     78     NaN    True
9   10     Big John   Four     55  female   False
10  11       Ronald    Six     89  female    True
11  12        Recky    Six     94  female    True
12  13          Kty  Seven     88  female    True
13  14         Bigy  Seven     88  female    True
14  15     Tade Row    NaN     88    male    True
15  16        Gimmy   Four     88    male    True
16  17        Tumyu    Six     54    male   False
17  18        Honny   Five     75    male    True
18  19        Tinny   Nine     18    male   False
19  20       Jackly   Nine     65  female    True
20  21   Babby John   Four     69  female    True
21  22       Reggid  Seven     55  female   False
22  23        Herod  Eight     79    male    True
23  24    Tiddy Now  Seven     78    male    True
24  25     Giff Tow  Seven     88    male    True
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
```

2)

3) 

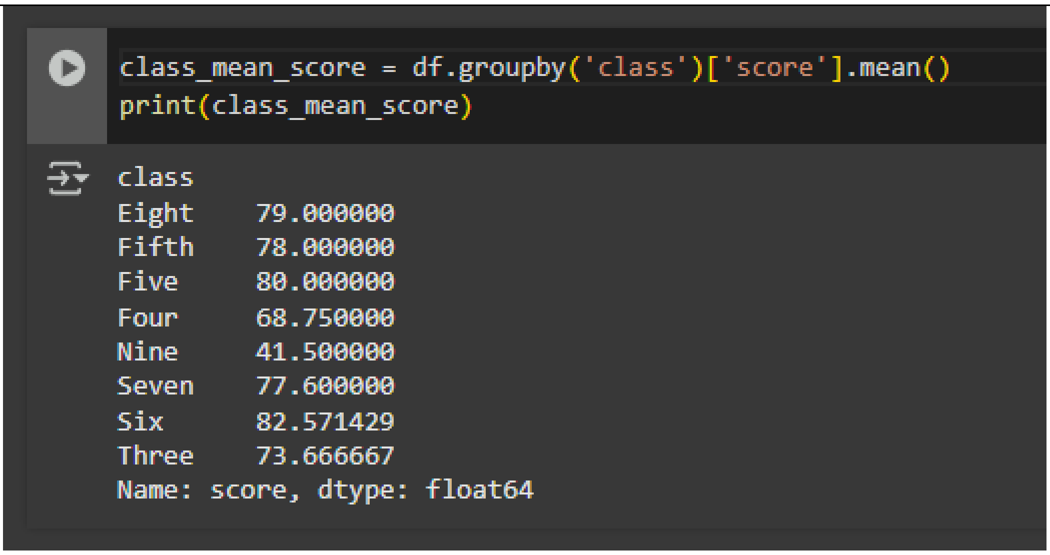
```
df.drop(columns = ['passed'])
```

```
3    4     Krish Star   Four   60  female
4    5      John Mike   Four   60  female
5    6      Alex John   Four   55    male
6    7  My John Rob    Fifth   78    male
7    8         Asruid   Five   85    male
8    9        Tes Qry    Six   78     NaN
9   10       Big John   Four   55  female
10  11         Ronald    Six   89  female
11  12          Recky    Six   94  female
12  13            Kty  Seven   88  female
13  14           Bigy  Seven   88  female
14  15       Tade Row    NaN   88    male
15  16         Gimmy    Four   88    male
16  17         Tumyu     Six   54    male
17  18         Honny    Five   75    male
18  19         Tinny    Nine   18    male
19  20        Jackly    Nine   65  female
20  21    Babby John    Four   69  female
21  22        Reggid   Seven   55  female
22  23         Herod   Eight   79    male
```

✓ 0s completed at 15:25

Exercise 4: Aggregation and Grouping

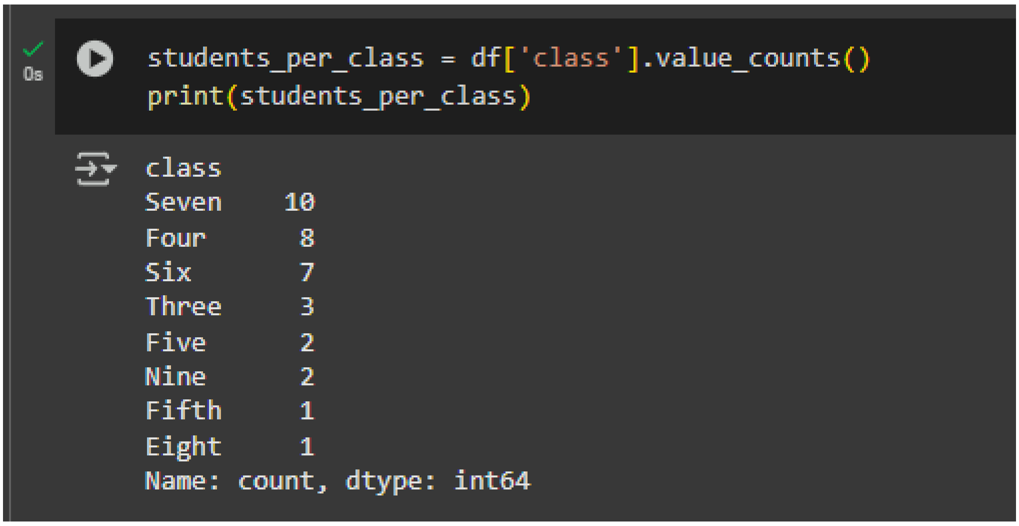
1. Question: "Write the code to group the DataFrame by the 'class' column and calculate the mean 'mark' for each group."
2. Question: "Write the code to count the number of students in each class."
3. Question: "Write the code to calculate the average mark for each gender."



```
class_mean_score = df.groupby('class')['score'].mean()
print(class_mean_score)
```

```
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: score, dtype: float64
```

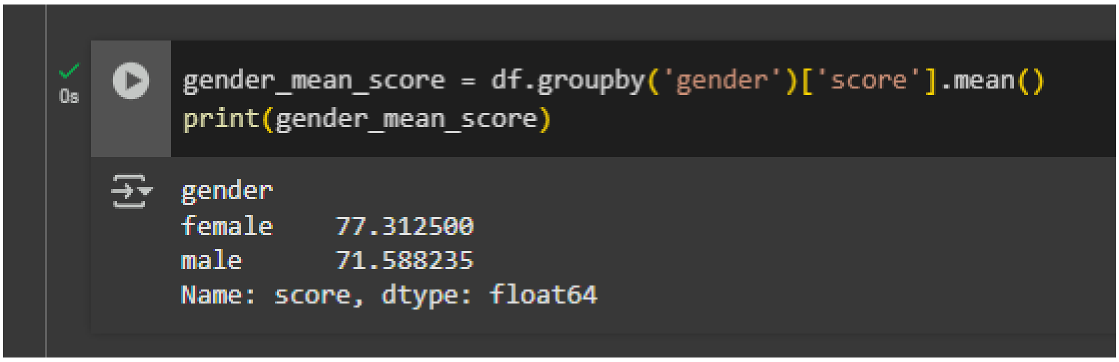
1)



```
students_per_class = df['class'].value_counts()
print(students_per_class)
```

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

2)



```
gender_mean_score = df.groupby('gender')['score'].mean()
print(gender_mean_score)
```

```
gender
female    77.312500
male      71.588235
Name: score, dtype: float64
```

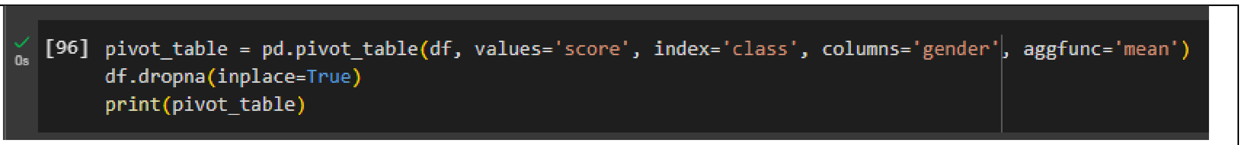
3)

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'."

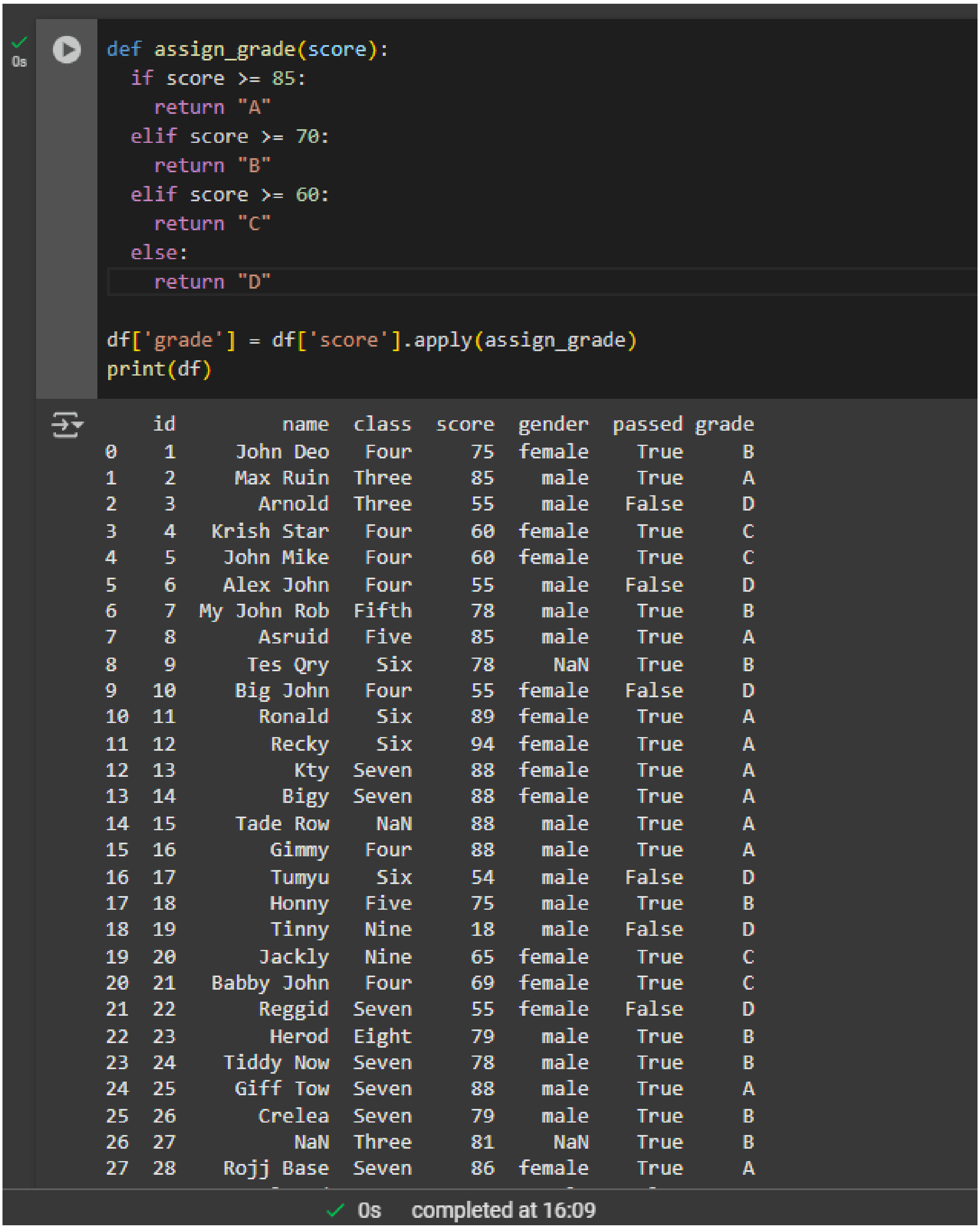
3. Question: "Write the code to sort the DataFrame by 'mark' in descending order."

1)



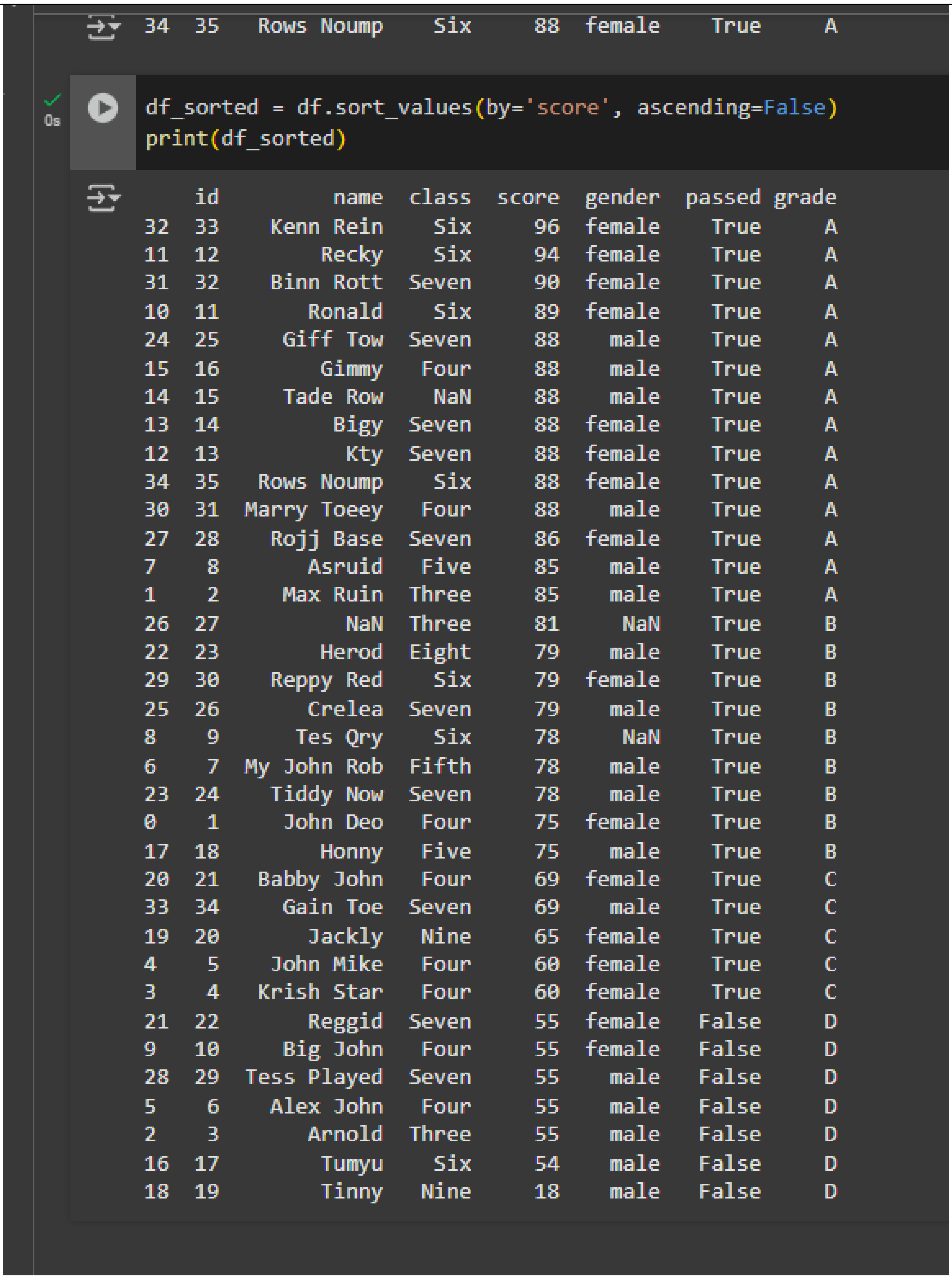
```
[96] pivot_table = pd.pivot_table(df, values='score', index='class', columns='gender', aggfunc='mean')
     df.dropna(inplace=True)
     print(pivot_table)
```

2)



```
def assign_grade(score):
  if score >= 85:
    return "A"
  elif score >= 70:
    return "B"
  elif score >= 60:
    return "C"
  else:
    return "D"

df['grade'] = df['score'].apply(assign_grade)
print(df)
```

```
    id          name   class  score  gender  passed grade
0    1      John Deo    Four     75  female    True     B
1    2      Max Ruin   Three     85    male    True     A
2    3        Arnold   Three     55    male   False     D
3    4    Krish Star    Four     60  female    True     C
4    5     John Mike    Four     60  female    True     C
5    6     Alex John    Four     55    male   False     D
6    7   My John Rob   Fifth     78    male    True     B
7    8        Asruid    Five     85    male    True     A
8    9       Tes Qry     Six     78     NaN    True     B
9   10      Big John    Four     55  female   False     D
10  11        Ronald     Six     89  female    True     A
11  12         Recky     Six     94  female    True     A
12  13           Kty   Seven     88  female    True     A
13  14          Bigy   Seven     88  female    True     A
14  15     Tade Row      NaN     88    male    True     A
15  16         Gimmy    Four     88    male    True     A
16  17        Tumyu      Six     54    male   False     D
17  18         Honny    Five     75    male    True     B
18  19         Tinny    Nine     18    male   False     D
19  20        Jackly    Nine     65  female    True     C
20  21   Babby John    Four     69  female    True     C
21  22        Reggid   Seven     55  female   False     D
22  23         Herod   Eight     79    male    True     B
23  24     Tiddy Now   Seven     78    male    True     B
24  25      Giff Tow   Seven     88    male    True     A
25  26        Crelea   Seven     79    male    True     B
26  27           NaN   Three     81     NaN    True     B
27  28     Rojj Base   Seven     86  female    True     A
```



```
34  35   Rows Noump     Six     88  female    True     A
```

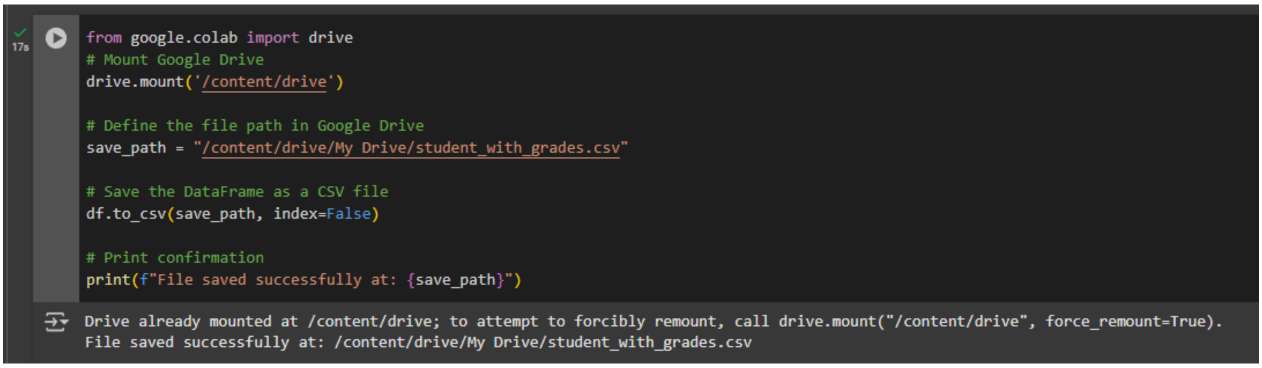
```
df_sorted = df.sort_values(by='score', ascending=False)
print(df_sorted)
```

```
    id         name   class  score  gender  passed grade
32  33    Kenn Rein     Six     96  female    True     A
11  12        Recky     Six     94  female    True     A
31  32    Binn Rott   Seven     90  female    True     A
10  11       Ronald     Six     89  female    True     A
24  25     Giff Tow   Seven     88    male    True     A
15  16        Gimmy    Four     88    male    True     A
14  15     Tade Row     NaN     88    male    True     A
13  14         Bigy   Seven     88  female    True     A
12  13          Kty   Seven     88  female    True     A
34  35   Rows Noump     Six     88  female    True     A
30  31  Marry Toeey    Four     88    male    True     A
27  28    Rojj Base   Seven     86  female    True     A
7    8       Asruid    Five     85    male    True     A
1    2      Max Ruin   Three     85    male    True     A
26  27          NaN   Three     81     NaN    True     B
22  23        Herod   Eight     79    male    True     B
29  30    Reppy Red     Six     79  female    True     B
25  26       Crelea   Seven     79    male    True     B
8    9      Tes Qry     Six     78     NaN    True     B
6    7  My John Rob   Fifth     78    male    True     B
23  24    Tiddy Now   Seven     78    male    True     B
0    1     John Deo    Four     75  female    True     B
17  18        Honny    Five     75    male    True     B
20  21   Babby John    Four     69  female    True     C
33  34     Gain Toe   Seven     69    male    True     C
19  20       Jackly    Nine     65  female    True     C
4    5    John Mike    Four     60  female    True     C
3    4   Krish Star    Four     60  female    True     C
21  22       Reggid   Seven     55  female   False     D
9   10     Big John    Four     55  female   False     D
28  29  Tess Played   Seven     55    male   False     D
5    6   Alex John    Four     55    male   False     D
2    3       Arnold   Three     55    male   False     D
16  17        Tumyu     Six     54    male   False     D
18  19        Tinny    Nine     18    male   False     D
```

3)

Exercise 6: Exporting Data

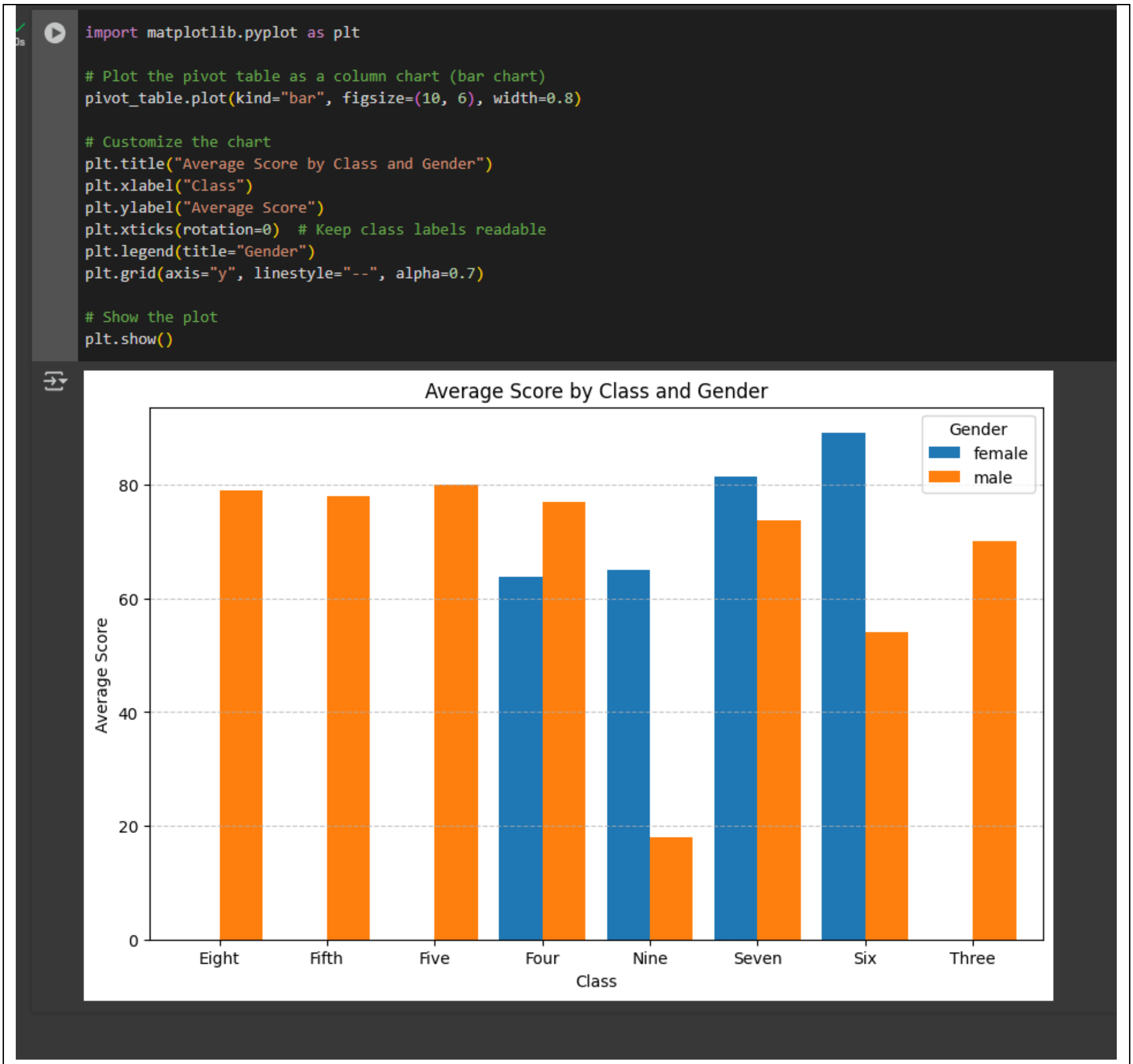
1. Question: "Write the code to save the DataFrame with the new 'grade' column to a new CSV file."



```
from google.colab import drive
# Mount Google Drive
drive.mount('/content/drive')

# Define the file path in Google Drive
save_path = "/content/drive/My Drive/student_with_grades.csv"

# Save the DataFrame as a CSV file
df.to_csv(save_path, index=False)

# Print confirmation
print(f"File saved successfully at: {save_path}")
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
File saved successfully at: /content/drive/My Drive/student_with_grades.csv
```

Exercise 7: If finished early try visualising the results

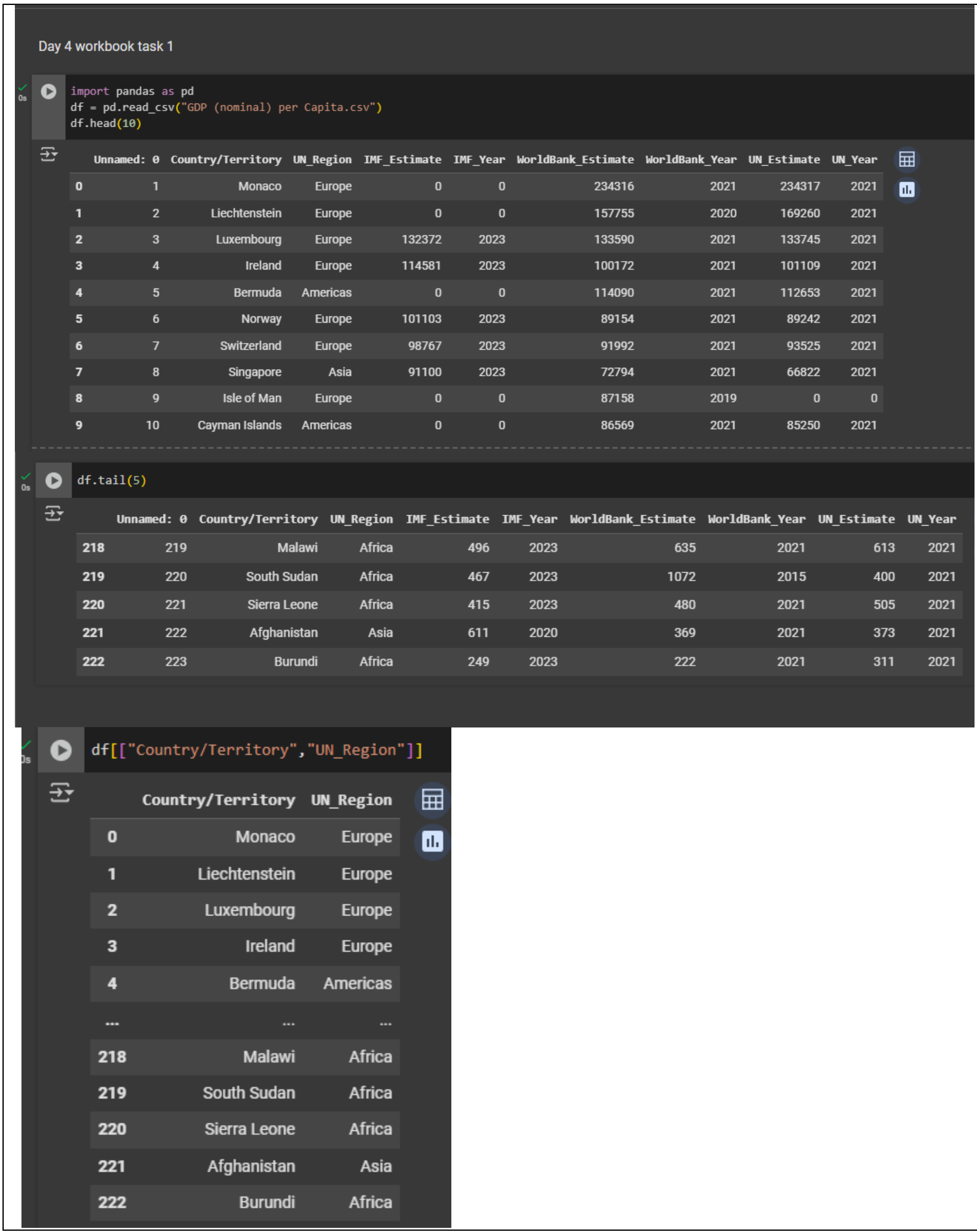


```python
import matplotlib.pyplot as plt

# Plot the pivot table as a column chart (bar chart)
pivot_table.plot(kind="bar", figsize=(10, 6), width=0.8)

# Customize the chart
plt.title("Average Score by Class and Gender")
plt.xlabel("Class")
plt.ylabel("Average Score")
plt.xticks(rotation=0)  # Keep class labels readable
plt.legend(title="Gender")
plt.grid(axis="y", linestyle="--", alpha=0.7)

# Show the plot
plt.show()
```

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



Day 4 workbook task 1

```
import pandas as pd
df = pd.read_csv("GDP (nominal) per Capita.csv")
df.head(10)
```

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

```
df[["Country/Territory","UN_Region"]]
```

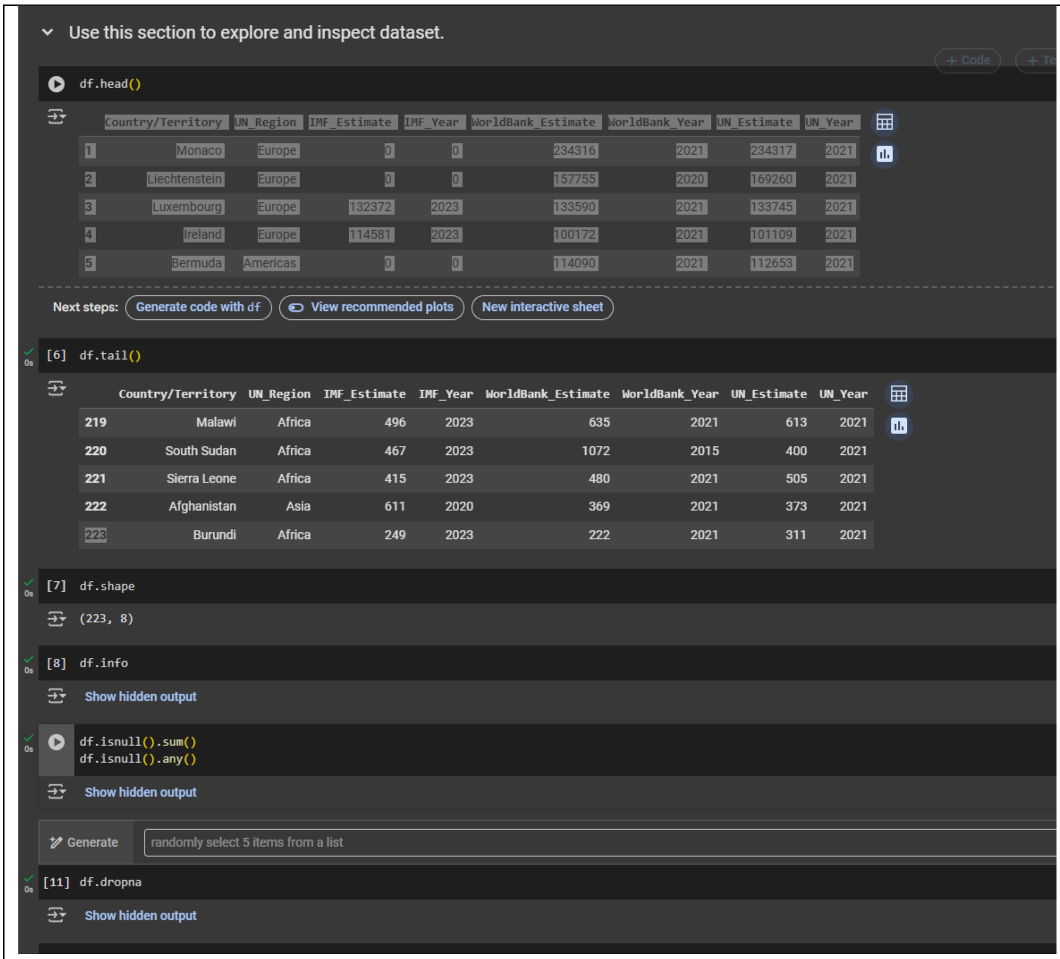
```
     Country/Territory UN_Region
0               Monaco    Europe
1        Liechtenstein    Europe
2           Luxembourg    Europe
3              Ireland    Europe
4              Bermuda  Americas
...                ...       ...
218             Malawi    Africa
219        South Sudan    Africa
220       Sierra Leone    Africa
221        Afghanistan      Asia
222            Burundi    Africa
```

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.



Use this section to explore and inspect dataset.

```
df.head()
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year
1	Monaco	Europe	0	0	234316	2021	234317	2021
2	Liechtenstein	Europe	0	0	157755	2020	169260	2021
3	Luxembourg	Europe	132372	2023	133590	2021	133745	2021
4	Ireland	Europe	114581	2023	100172	2021	101109	2021
5	Bermuda	Americas	0	0	114090	2021	112653	2021

```
[6] df.tail()
```

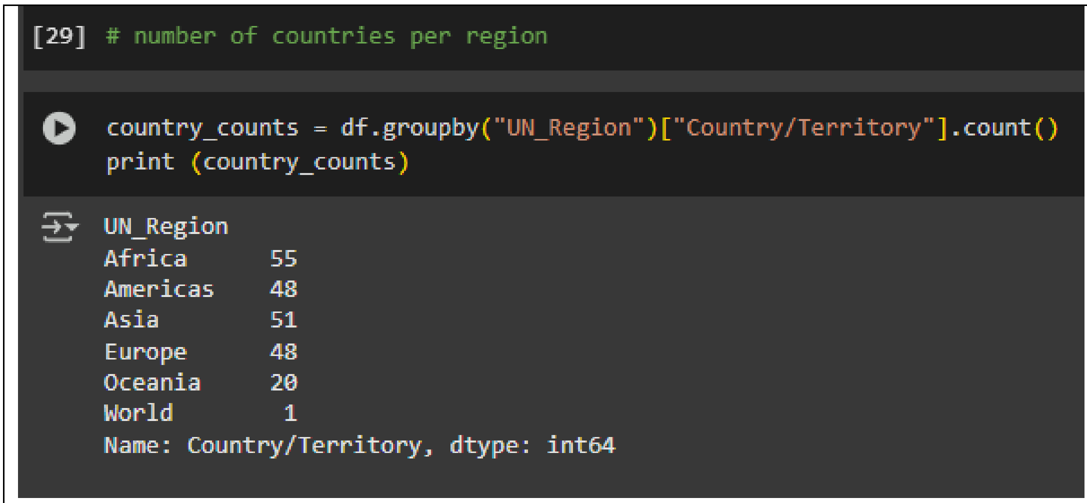
	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year
219	Malawi	Africa	496	2023	635	2021	613	2021
220	South Sudan	Africa	467	2023	1072	2015	400	2021
221	Sierra Leone	Africa	415	2023	480	2021	505	2021
222	Afghanistan	Asia	611	2020	369	2021	373	2021
223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

```
[7] df.shape
(223, 8)
```

```
[8] df.info
```

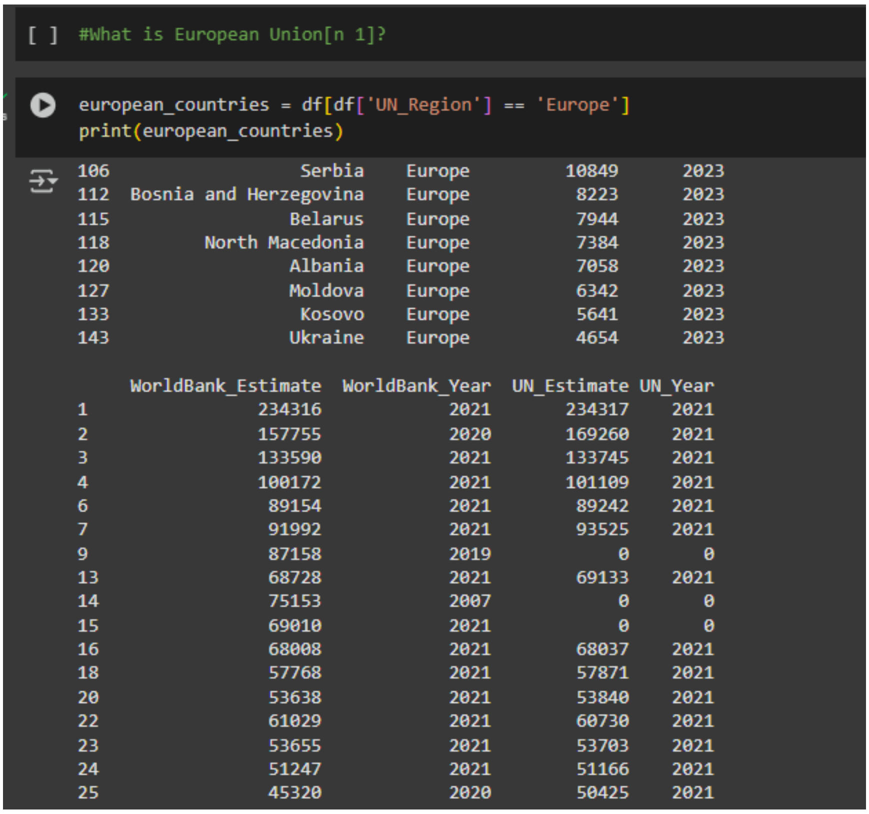
```
df.isnull().sum()
df.isnull().any()
```

```
[11] df.dropna
```



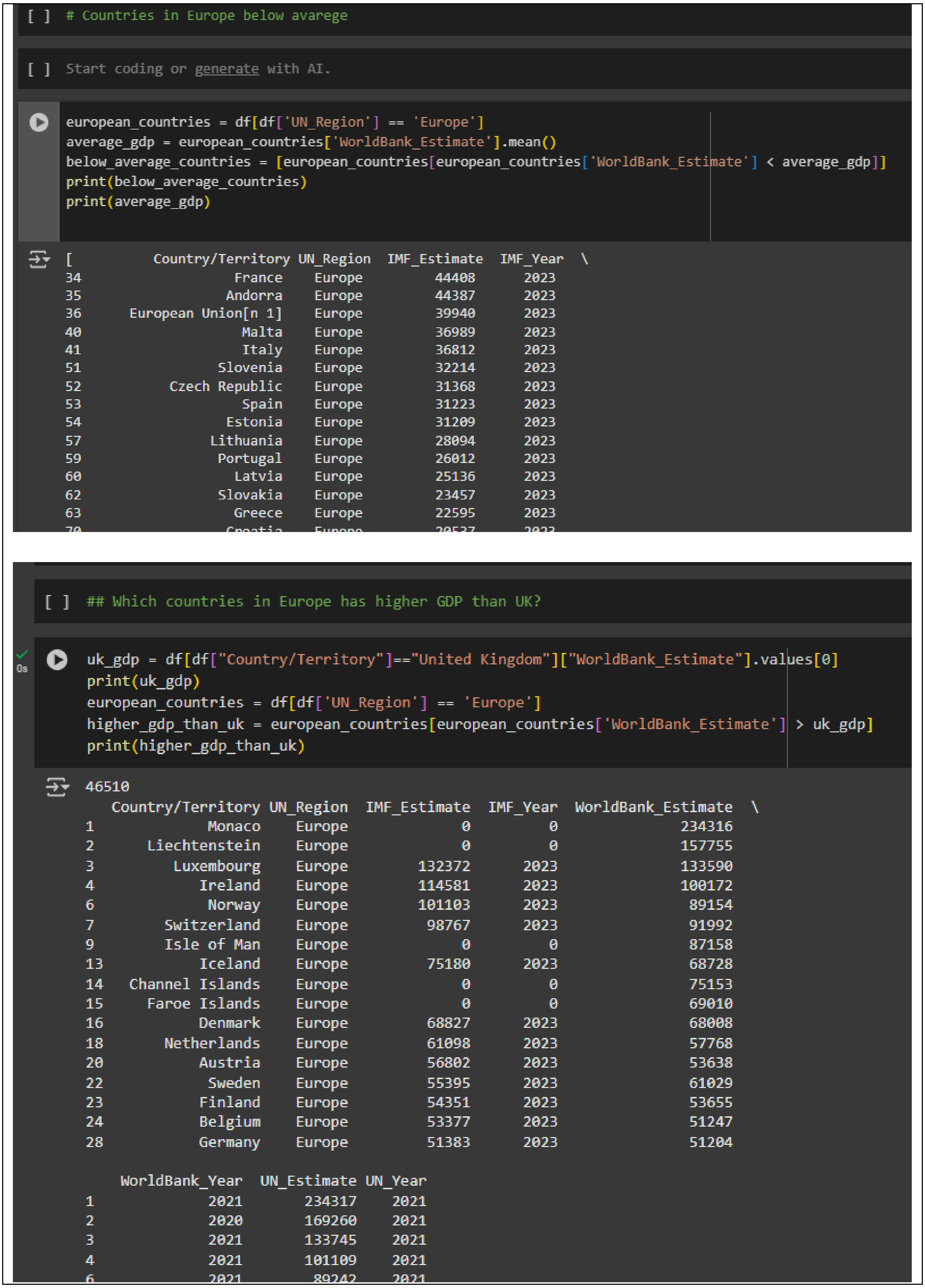
```
[29] # number of countries per region

country_counts = df.groupby("UN_Region")["Country/Territory"].count()
print (country_counts)
```

```
UN_Region
Africa      55
Americas    48
Asia        51
Europe      48
Oceania     20
World        1
Name: Country/Territory, dtype: int64
```



```
[ ] #What is European Union[n 1]?

european_countries = df[df['UN_Region'] == 'Europe']
print(european_countries)
```

```
106                 Serbia   Europe        10849      2023
112 Bosnia and Herzegovina   Europe         8223      2023
115                Belarus   Europe         7944      2023
118        North Macedonia   Europe         7384      2023
120                Albania   Europe         7058      2023
127                Moldova   Europe         6342      2023
133                 Kosovo   Europe         5641      2023
143                Ukraine   Europe         4654      2023

    WorldBank_Estimate  WorldBank_Year  UN_Estimate UN_Year
1               234316            2021       234317    2021
2               157755            2020       169260    2021
3               133590            2021       133745    2021
4               100172            2021       101109    2021
6                89154            2021        89242    2021
7                91992            2021        93525    2021
9                87158            2019            0       0
13               68728            2021        69133    2021
14               75153            2007            0       0
15               69010            2021            0       0
16               68008            2021        68037    2021
18               57768            2021        57871    2021
20               53638            2021        53840    2021
22               61029            2021        60730    2021
23               53655            2021        53703    2021
24               51247            2021        51166    2021
25               45320            2020        50425    2021
```



```
[ ] # Countries in Europe below avarege
```

```
[ ] Start coding or generate with AI.
```

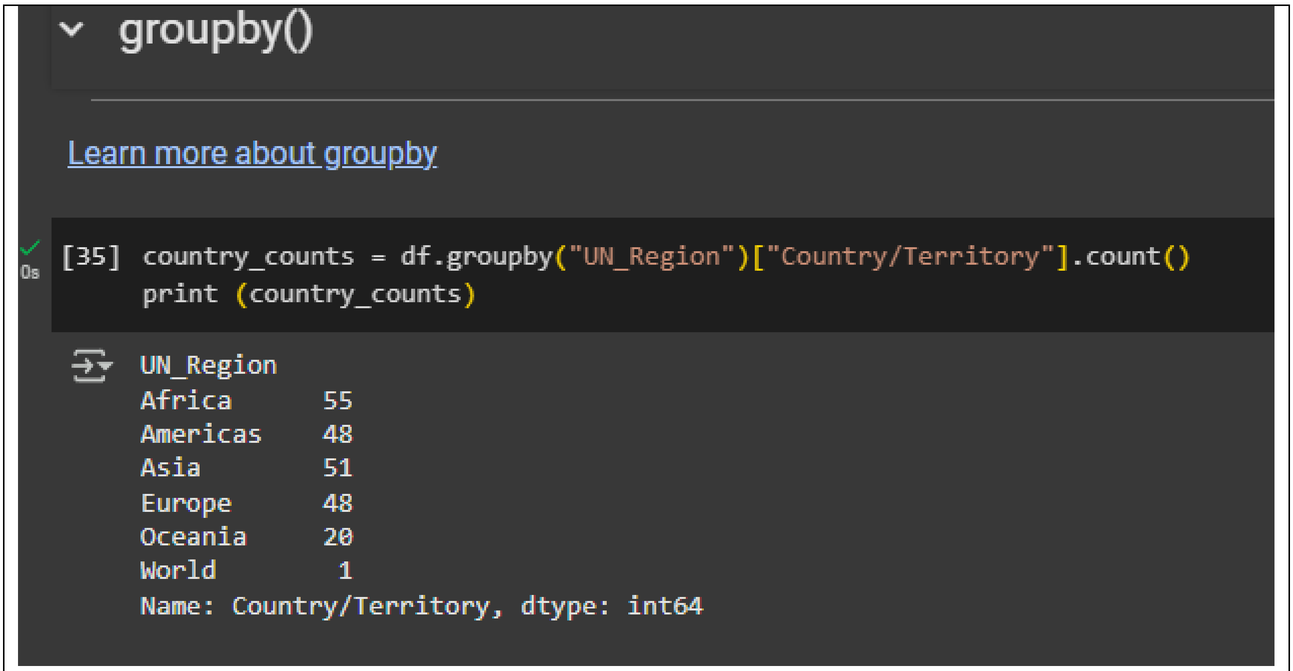
```
european_countries = df[df['UN_Region'] == 'Europe']
average_gdp = european_countries['WorldBank_Estimate'].mean()
below_average_countries = [european_countries[european_countries['WorldBank_Estimate'] < average_gdp]]
print(below_average_countries)
print(average_gdp)
```

```
[          Country/Territory UN_Region  IMF_Estimate  IMF_Year  \
34                 France    Europe         44408      2023
35                Andorra    Europe         44387      2023
36     European Union[n 1]    Europe         39940      2023
40                  Malta    Europe         36989      2023
41                  Italy    Europe         36812      2023
51               Slovenia    Europe         32214      2023
52         Czech Republic    Europe         31368      2023
53                  Spain    Europe         31223      2023
54                Estonia    Europe         31209      2023
57              Lithuania    Europe         28094      2023
59               Portugal    Europe         26012      2023
60                 Latvia    Europe         25136      2023
62               Slovakia    Europe         23457      2023
63                 Greece    Europe         22595      2023
70                Croatia    Europe         20537      2023
```

```
[ ] ## Which countries in Europe has higher GDP than UK?
```

```
uk_gdp = df[df["Country/Territory"]=="United Kingdom"]["WorldBank_Estimate"].values[0]
print(uk_gdp)
european_countries = df[df['UN_Region'] == 'Europe']
higher_gdp_than_uk = european_countries[european_countries['WorldBank_Estimate'] > uk_gdp]
print(higher_gdp_than_uk)
```

```
46510
   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
7        Switzerland    Europe         98767      2023               91992
9        Isle of Man    Europe             0         0               87158
13           Iceland    Europe         75180      2023               68728
14   Channel Islands    Europe             0         0               75153
15     Faroe Islands    Europe             0         0               69010
16           Denmark    Europe         68827      2023               68008
18       Netherlands    Europe         61098      2023               57768
20           Austria    Europe         56802      2023               53638
22            Sweden    Europe         55395      2023               61029
23           Finland    Europe         54351      2023               53655
24           Belgium    Europe         53377      2023               51247
28           Germany    Europe         51383      2023               51204

    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
```

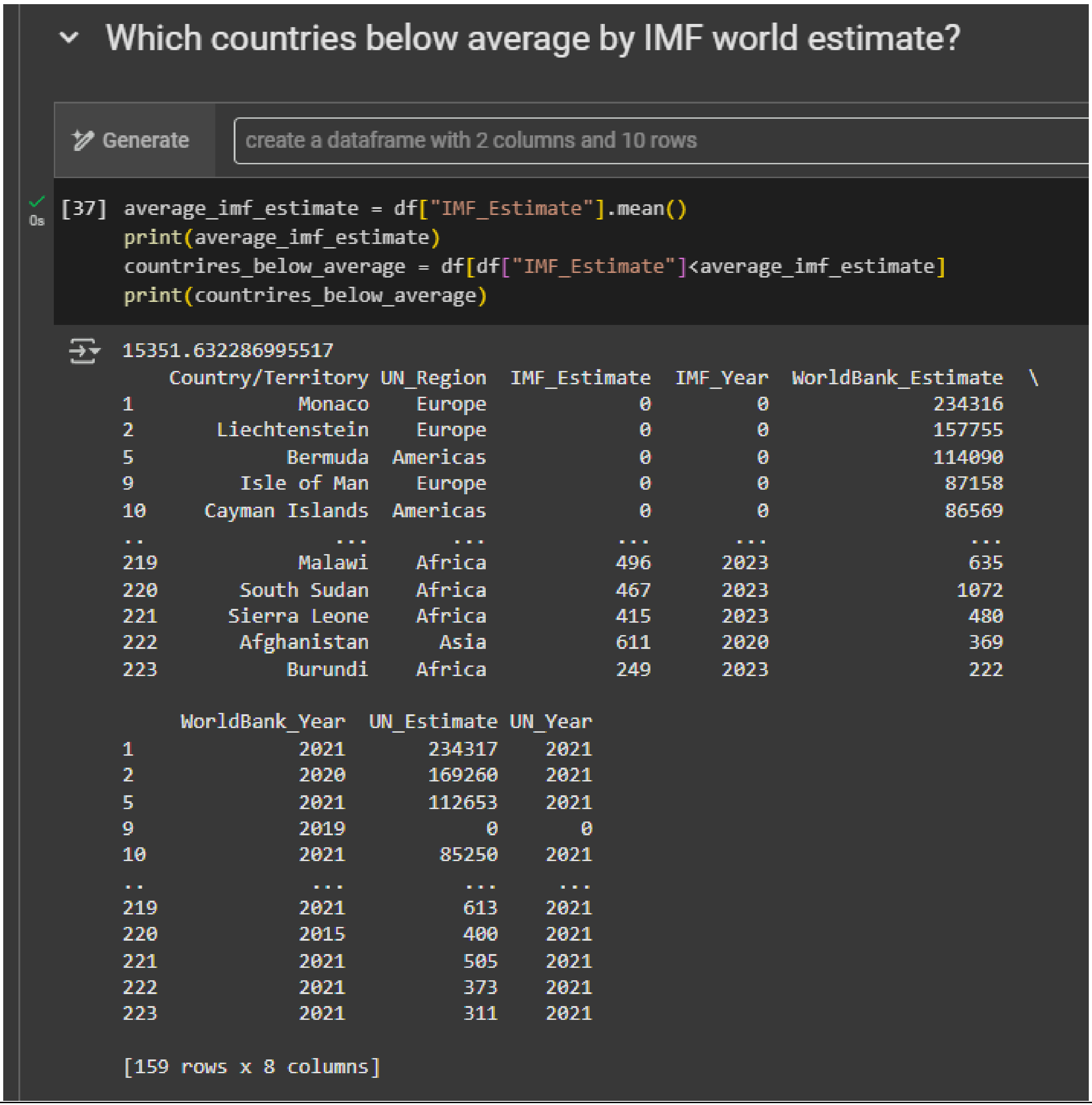


groupby()

Learn more about groupby

```
[35] country_counts = df.groupby("UN_Region")["Country/Territory"].count()
     print (country_counts)
```

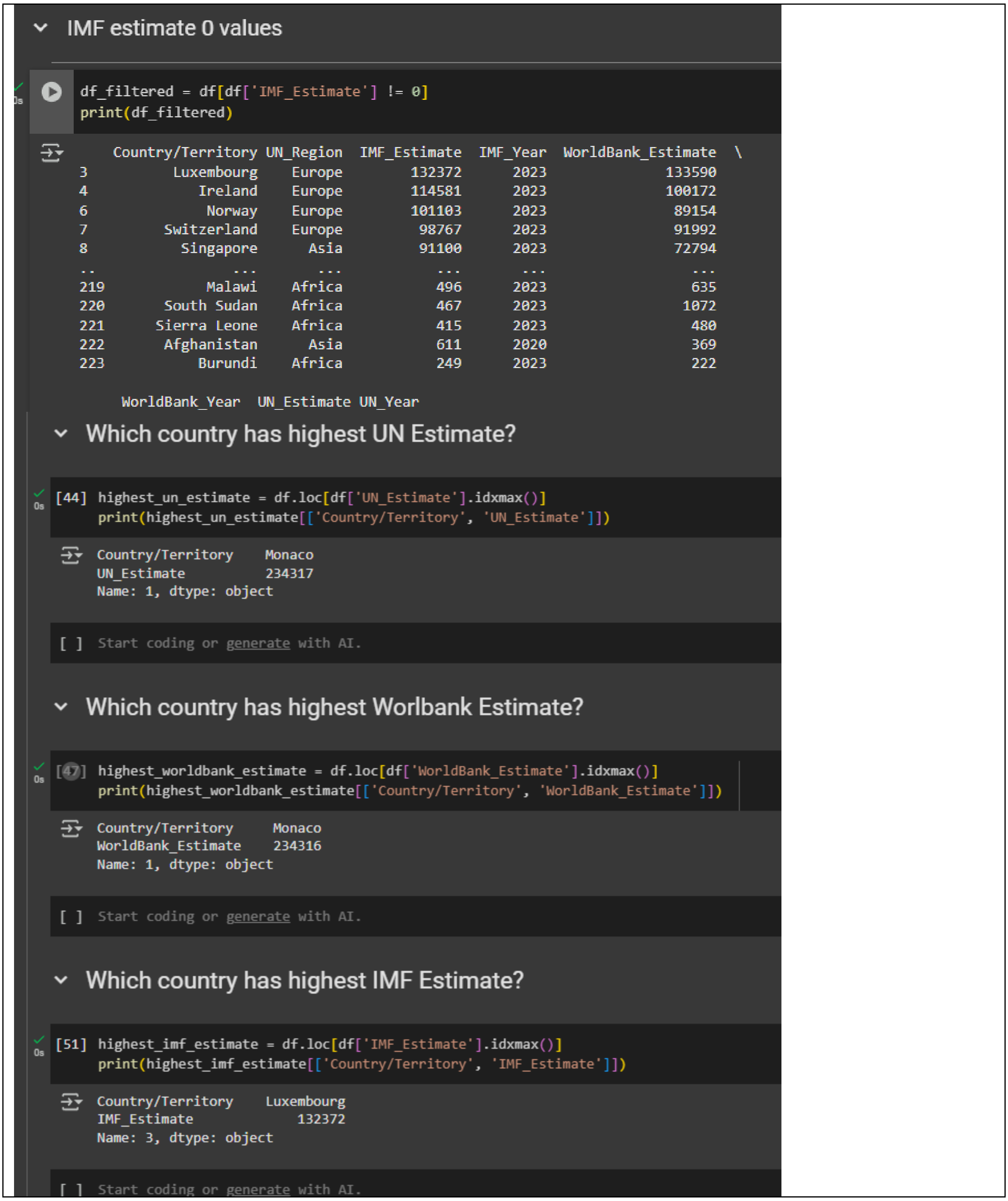
```
UN_Region
Africa      55
Americas    48
Asia        51
Europe      48
Oceania     20
World        1
Name: Country/Territory, dtype: int64
```



Which countries below average by IMF world estimate?

```
[37] average_imf_estimate = df["IMF_Estimate"].mean()
     print(average_imf_estimate)
     countrires_below_average = df[df["IMF_Estimate"]<average_imf_estimate]
     print(countrires_below_average)
```

```
15351.632286995517
    Country/Territory UN_Region  IMF_Estimate  IMF_Year  WorldBank_Estimate  \
1              Monaco    Europe             0         0              234316
2       Liechtenstein    Europe             0         0              157755
5             Bermuda  Americas             0         0              114090
9         Isle of Man    Europe             0         0               87158
10     Cayman Islands  Americas             0         0               86569
..                ...       ...           ...       ...                 ...
219            Malawi    Africa           496      2023                 635
220       South Sudan    Africa           467      2023                1072
221      Sierra Leone    Africa           415      2023                 480
222       Afghanistan      Asia           611      2020                 369
223           Burundi    Africa           249      2023                 222

     WorldBank_Year  UN_Estimate  UN_Year
1              2021       234317     2021
2              2020       169260     2021
5              2021       112653     2021
9              2019            0        0
10             2021        85250     2021
..              ...          ...      ...
219            2021          613     2021
220            2015          400     2021
221            2021          505     2021
222            2021          373     2021
223            2021          311     2021

[159 rows x 8 columns]
```



IMF estimate 0 values

```
df_filtered = df[df['IMF_Estimate'] != 0]
print(df_filtered)
```

```
    Country/Territory UN_Region  IMF_Estimate  IMF_Year  WorldBank_Estimate  \
3          Luxembourg    Europe        132372      2023              133590
4             Ireland    Europe        114581      2023              100172
6              Norway    Europe        101103      2023               89154
7         Switzerland    Europe         98767      2023               91992
8           Singapore      Asia         91100      2023               72794
..                ...       ...           ...       ...                 ...
219            Malawi    Africa           496      2023                 635
220       South Sudan    Africa           467      2023                1072
221      Sierra Leone    Africa           415      2023                 480
222       Afghanistan      Asia           611      2020                 369
223           Burundi    Africa           249      2023                 222

     WorldBank_Year  UN_Estimate UN_Year
```

Which country has highest UN Estimate?

```
[44] highest_un_estimate = df.loc[df['UN_Estimate'].idxmax()]
     print(highest_un_estimate[['Country/Territory', 'UN_Estimate']])
```

```
Country/Territory    Monaco
UN_Estimate          234317
Name: 1, dtype: object
```

```
[ ] Start coding or generate with AI.
```

Which country has highest Worlbank Estimate?

```
[47] highest_worldbank_estimate = df.loc[df['WorldBank_Estimate'].idxmax()]
     print(highest_worldbank_estimate[['Country/Territory', 'WorldBank_Estimate']])
```

```
Country/Territory     Monaco
WorldBank_Estimate    234316
Name: 1, dtype: object
```

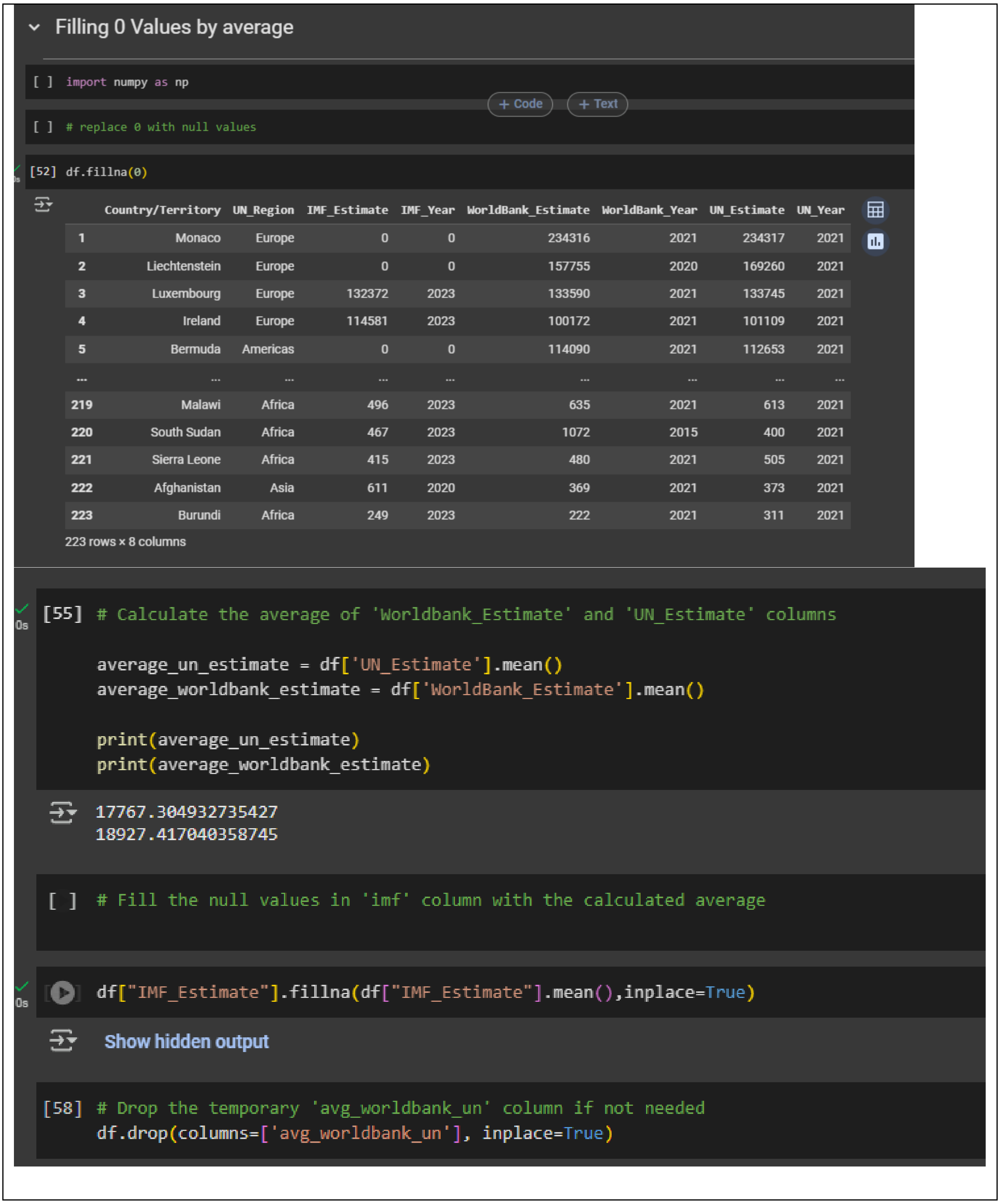
```
[ ] Start coding or generate with AI.
```

Which country has highest IMF Estimate?

```
[51] highest_imf_estimate = df.loc[df['IMF_Estimate'].idxmax()]
     print(highest_imf_estimate[['Country/Territory', 'IMF_Estimate']])
```

```
Country/Territory    Luxembourg
IMF_Estimate             132372
Name: 3, dtype: object
```

```
[ ] Start coding or generate with AI.
```



Filling 0 Values by average

```
[ ] import numpy as np
```

```
[ ] # replace 0 with null values
```

```
[52] df.fillna(0)
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year
1	Monaco	Europe	0	0	234316	2021	234317	2021
2	Liechtenstein	Europe	0	0	157755	2020	169260	2021
3	Luxembourg	Europe	132372	2023	133590	2021	133745	2021
4	Ireland	Europe	114581	2023	100172	2021	101109	2021
5	Bermuda	Americas	0	0	114090	2021	112653	2021
...	...	...	...	...	...	...	...	...
219	Malawi	Africa	496	2023	635	2021	613	2021
220	South Sudan	Africa	467	2023	1072	2015	400	2021
221	Sierra Leone	Africa	415	2023	480	2021	505	2021
222	Afghanistan	Asia	611	2020	369	2021	373	2021
223	Burundi	Africa	249	2023	222	2021	311	2021

223 rows × 8 columns

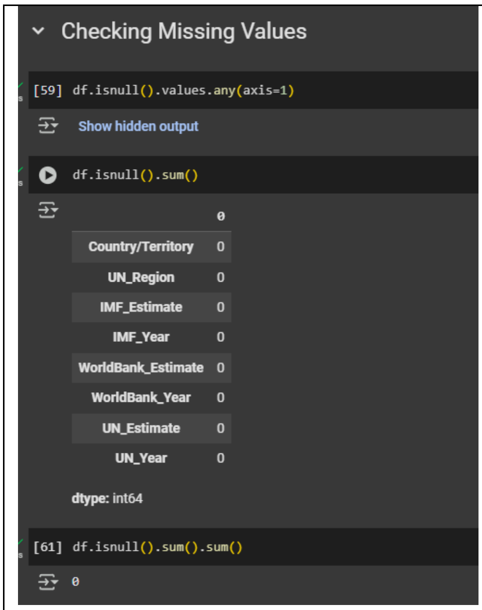
```
[55] # Calculate the average of 'Worldbank_Estimate' and 'UN_Estimate' columns

     average_un_estimate = df['UN_Estimate'].mean()
     average_worldbank_estimate = df['WorldBank_Estimate'].mean()

     print(average_un_estimate)
     print(average_worldbank_estimate)
```

```
17767.304932735427
18927.417040358745
```

```
[ ] # Fill the null values in 'imf' column with the calculated average
```

```
df["IMF_Estimate"].fillna(df["IMF_Estimate"].mean(),inplace=True)
```

```
[58] # Drop the temporary 'avg_worldbank_un' column if not needed
     df.drop(columns=['avg_worldbank_un'], inplace=True)
```



```
[59] df.isnull().values.any(axis=1)
```

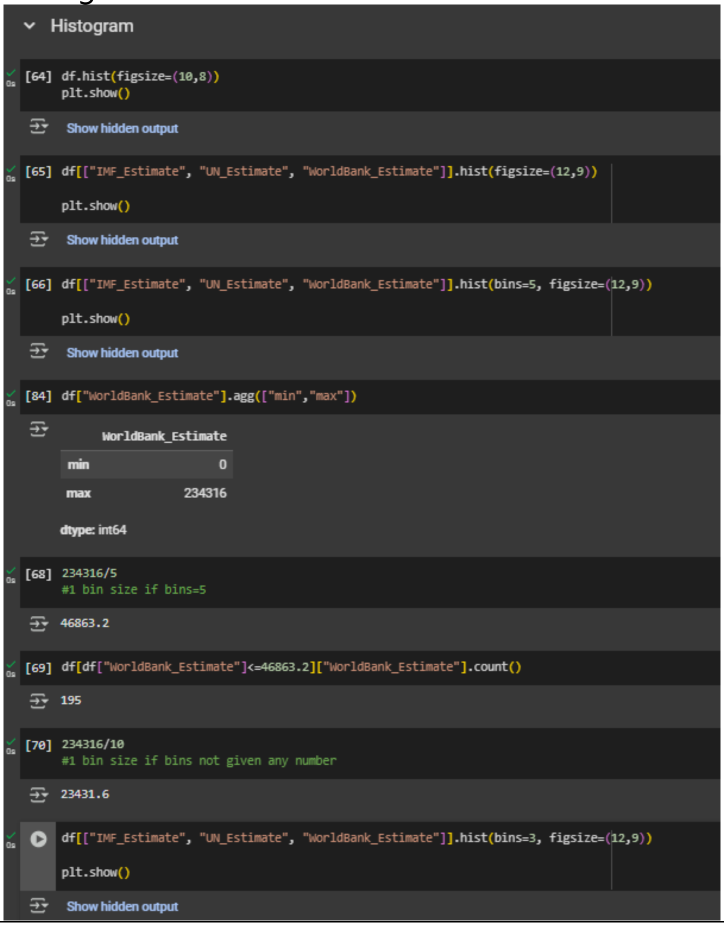
```
df.isnull().sum()
```

```
                     0
Country/Territory    0
UN_Region            0
IMF_Estimate         0
IMF_Year             0
WorldBank_Estimate   0
WorldBank_Year       0
UN_Estimate          0
UN_Year              0

dtype: int64
```

```
[61] df.isnull().sum().sum()
```

```
0
```

Histogram:



```
[64] df.hist(figsize=(10,8))
     plt.show()
```

```
[65] df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].hist(figsize=(12,9))

     plt.show()
```

```
[66] df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].hist(bins=5, figsize=(12,9))

     plt.show()
```

```
[84] df["WorldBank_Estimate"].agg(["min","max"])
```

```
      WorldBank_Estimate
min                    0
max               234316

dtype: int64
```

```
[68] 234316/5
     #1 bin size if bins=5
```

```
46863.2
```

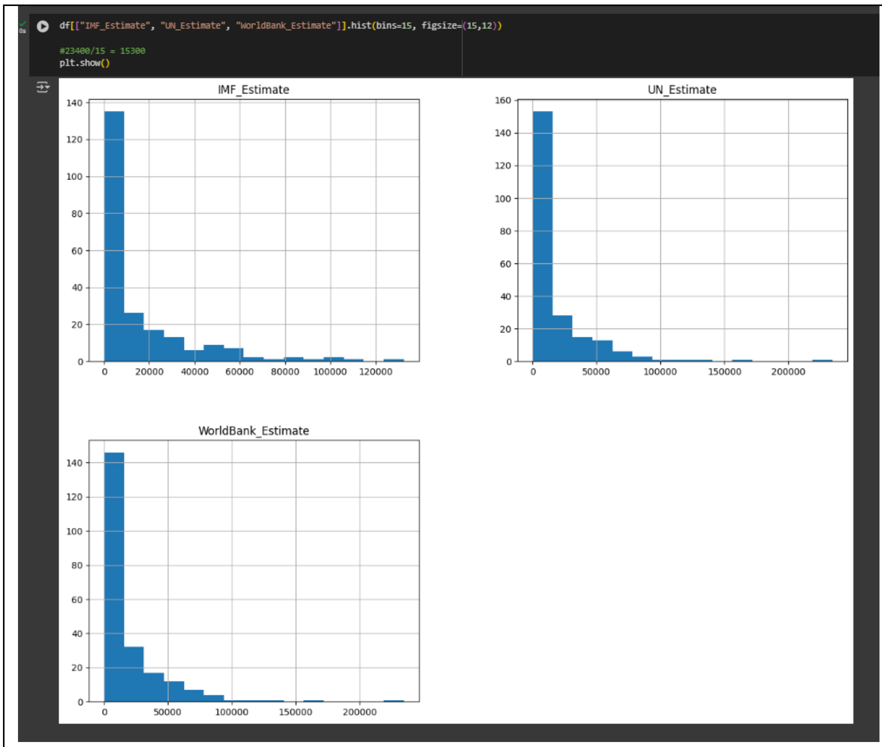
```
[69] df[df["WorldBank_Estimate"]<=46863.2]["WorldBank_Estimate"].count()
```

```
195
```

```
[70] 234316/10
     #1 bin size if bins not given any number
```

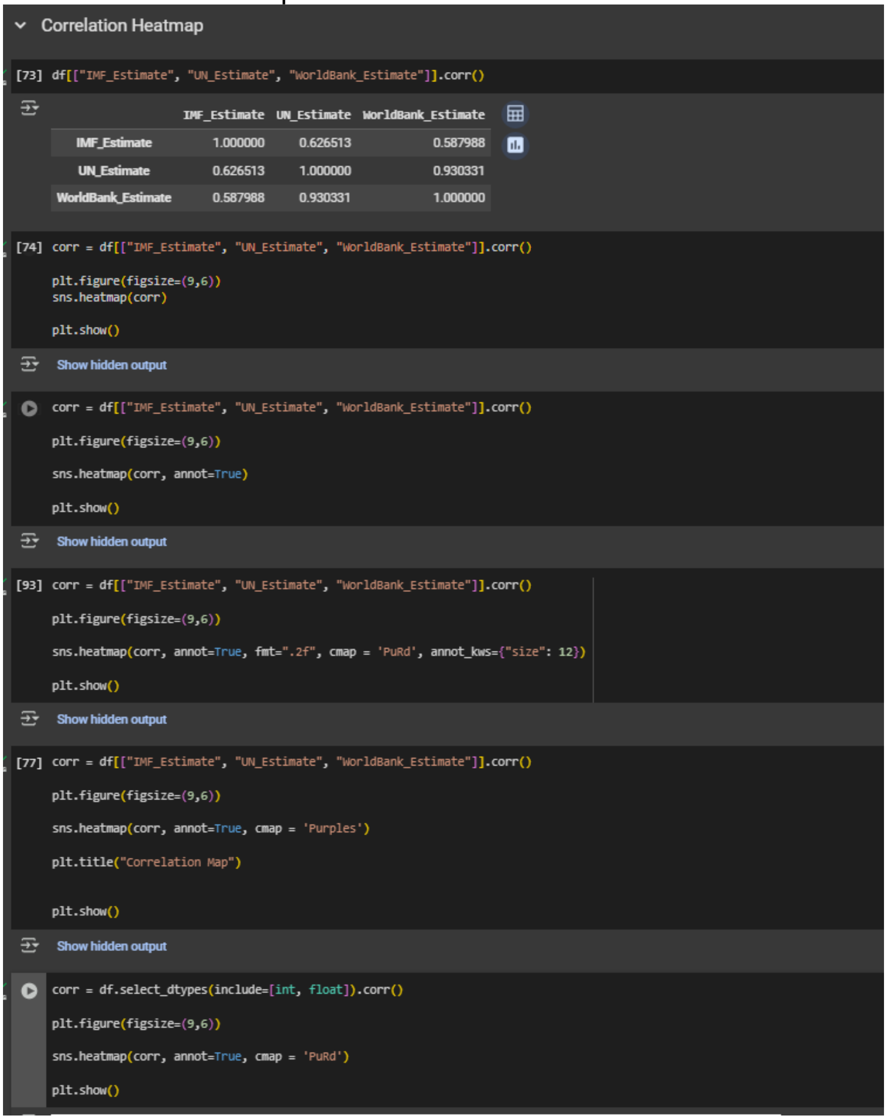
```
23431.6
```

```
df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].hist(bins=3, figsize=(12,9))

plt.show()
```



```
df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].hist(bins=15, figsize=(15,12))

#23400/15 = 15300
plt.show()
```

Correlation Heatmap:



Correlation Heatmap

```
[73] df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].corr()
```

	IMF_Estimate	UN_Estimate	WorldBank_Estimate
IMF_Estimate	1.000000	0.626513	0.587988
UN_Estimate	0.626513	1.000000	0.930331
WorldBank_Estimate	0.587988	0.930331	1.000000

```
[74] corr = df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].corr()

     plt.figure(figsize=(9,6))
     sns.heatmap(corr)

     plt.show()
```

Show hidden output

```
corr = df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].corr()

plt.figure(figsize=(9,6))

sns.heatmap(corr, annot=True)

plt.show()
```

Show hidden output

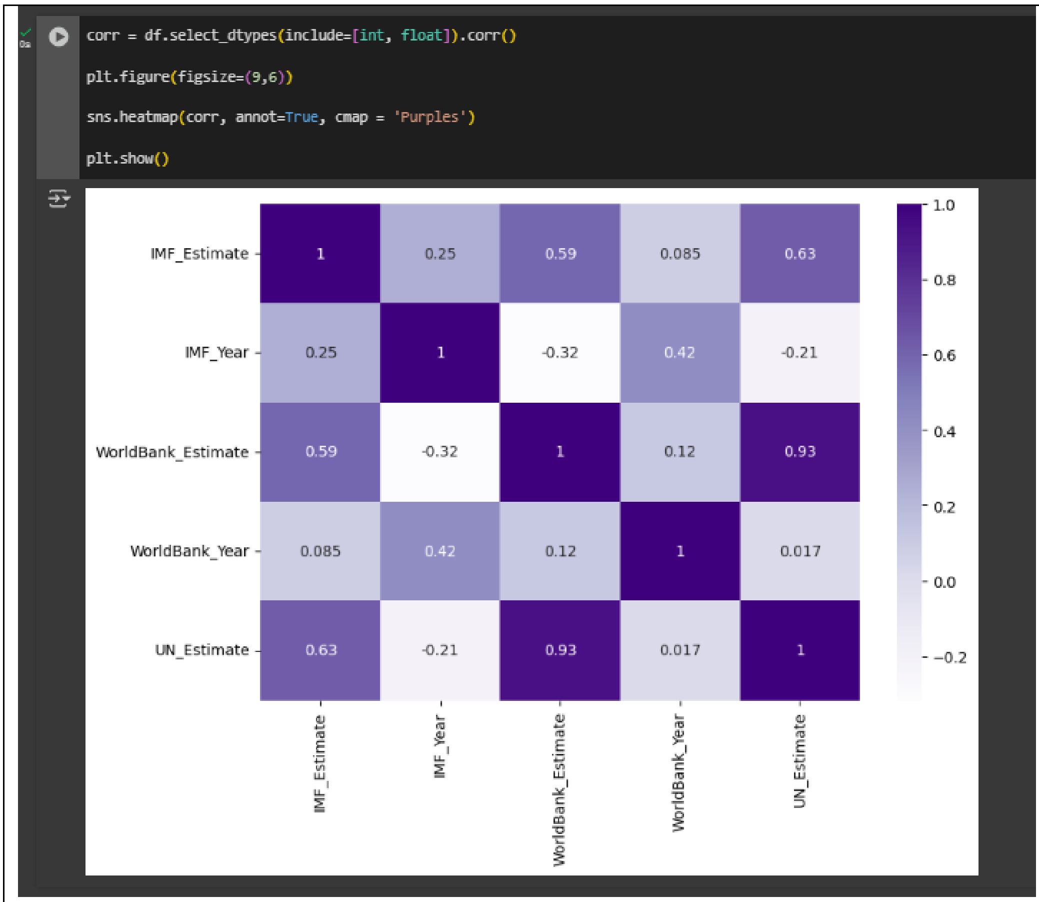
```
[93] corr = df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].corr()

     plt.figure(figsize=(9,6))

     sns.heatmap(corr, annot=True, fmt=".2f", cmap = 'PuRd', annot_kws={"size": 12})

     plt.show()
```

Show hidden output

```
[77] corr = df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].corr()

     plt.figure(figsize=(9,6))

     sns.heatmap(corr, annot=True, cmap = 'Purples')

     plt.title("Correlation Map")


     plt.show()
```

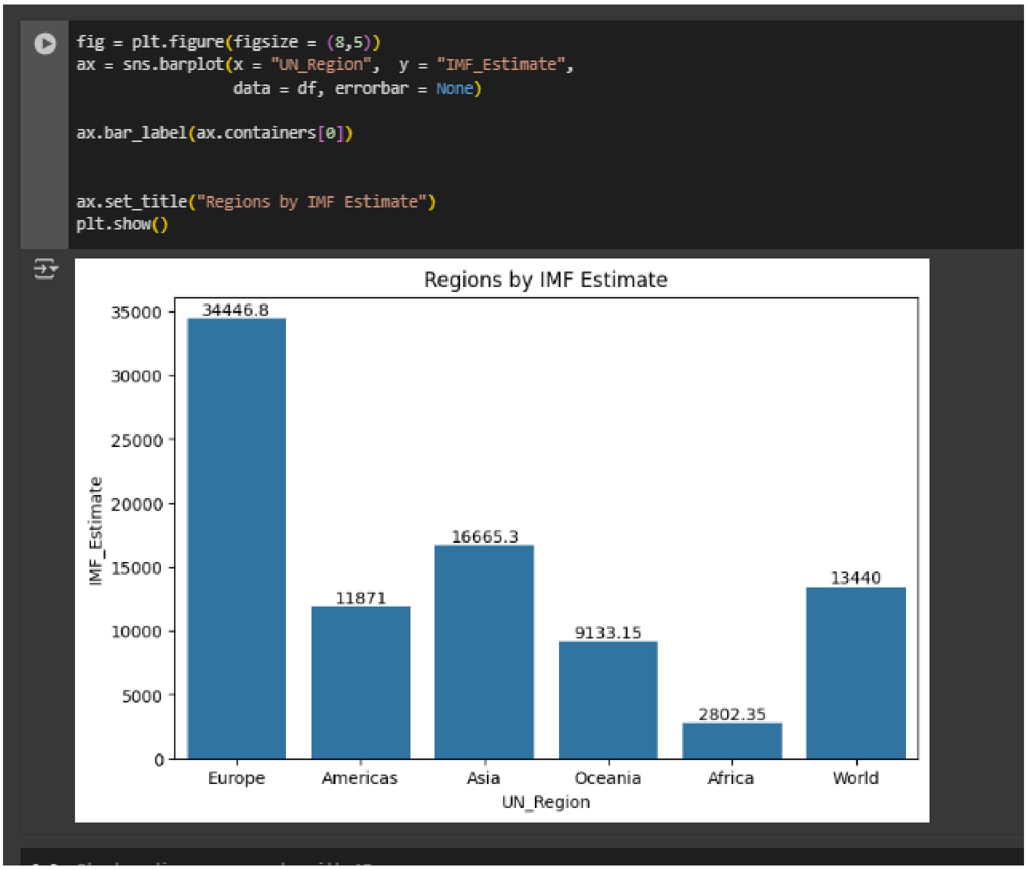
Show hidden output

```
corr = df.select_dtypes(include=[int, float]).corr()

plt.figure(figsize=(9,6))

sns.heatmap(corr, annot=True, cmap = 'PuRd')

plt.show()
```



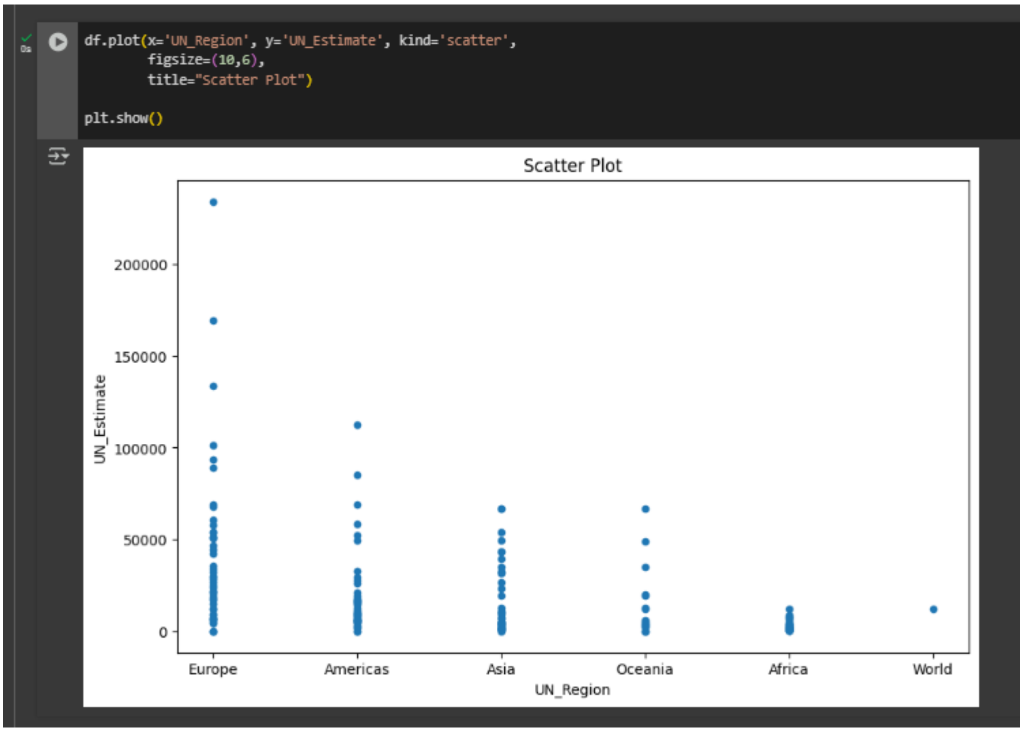
```
corr = df.select_dtypes(include=[int, float]).corr()

plt.figure(figsize=(9,6))

sns.heatmap(corr, annot=True, cmap = 'Purples')

plt.show()
```

Bar Plot:



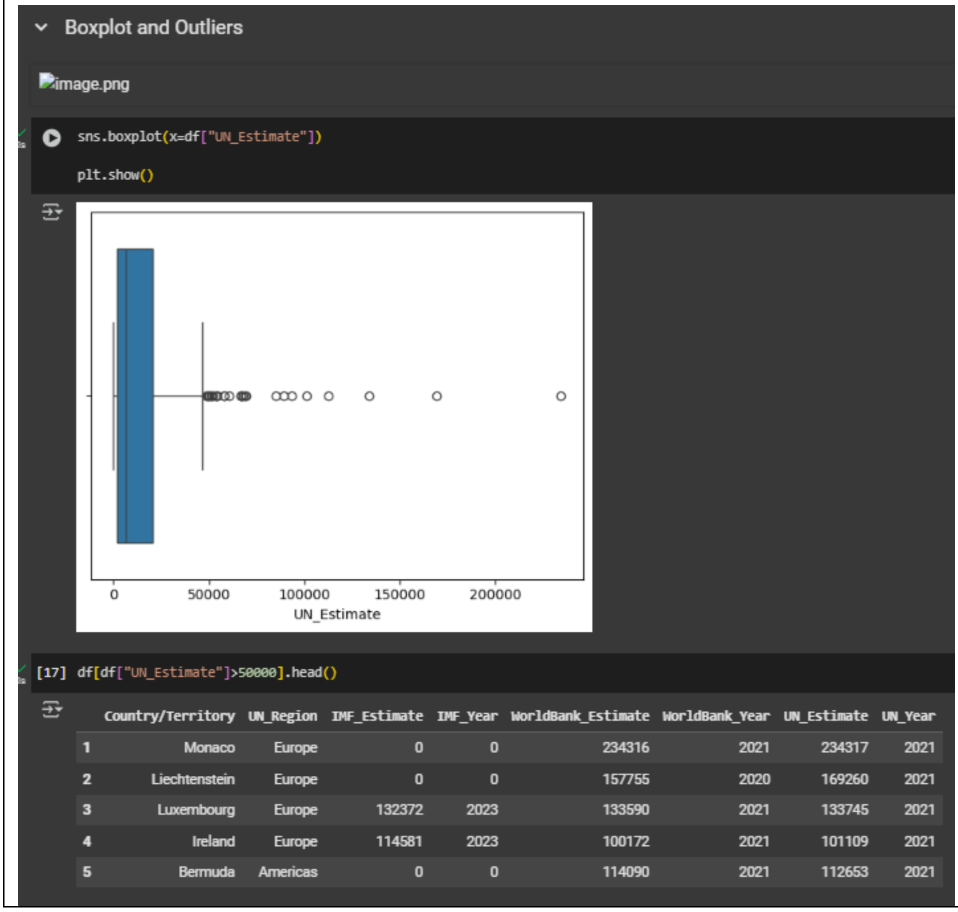
```
fig = plt.figure(figsize = (8,5))
ax = sns.barplot(x = "UN_Region",  y = "IMF_Estimate",
                 data = df, errorbar = None)

ax.bar_label(ax.containers[0])


ax.set_title("Regions by IMF Estimate")
plt.show()
```

Scatter Plot:



```
df.plot(x='UN_Region', y='UN_Estimate', kind='scatter',
        figsize=(10,6),
        title="Scatter Plot")

plt.show()
```

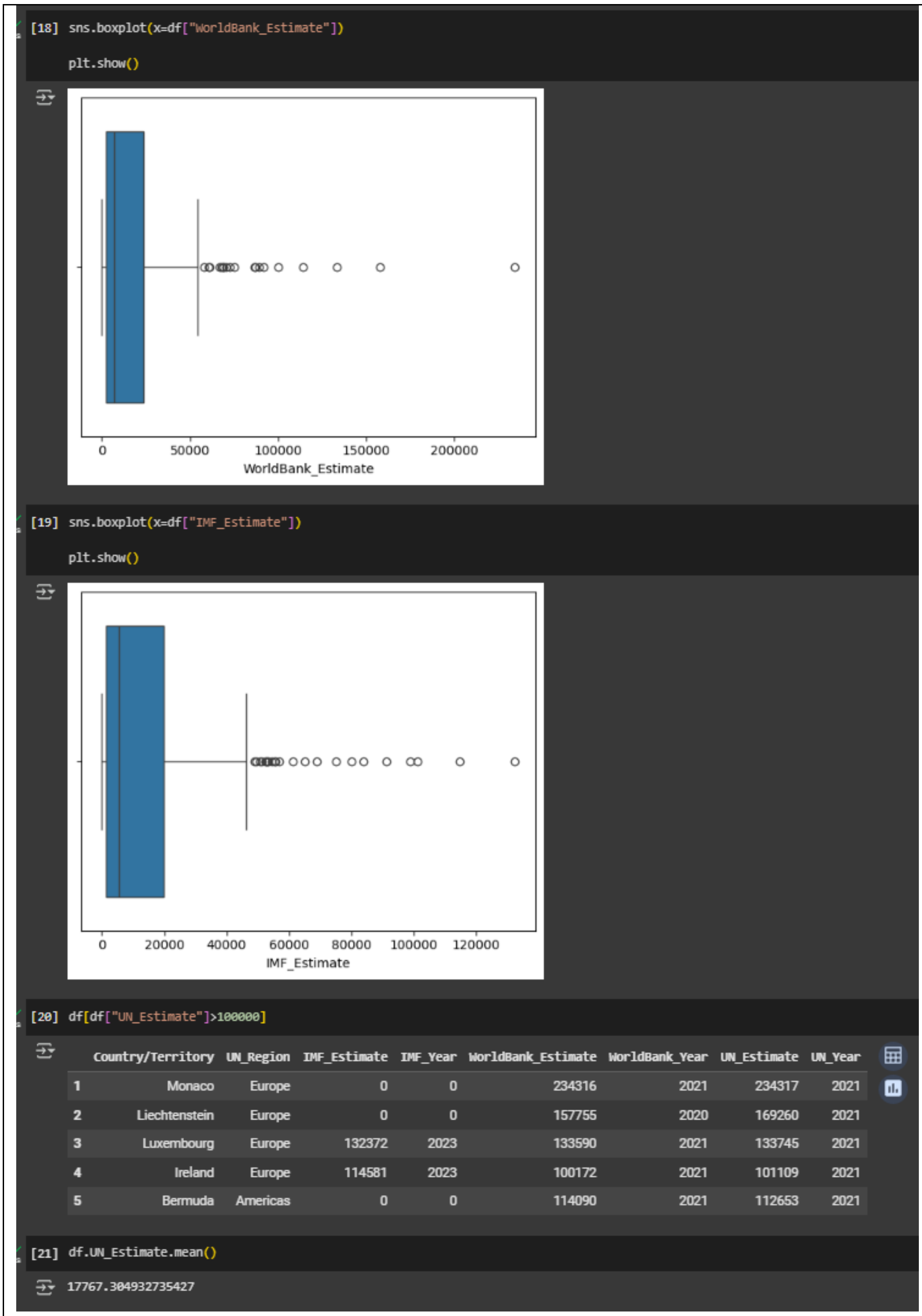
Boxplot and Outliers:



```
sns.boxplot(x=df["UN_Estimate"])

plt.show()
```

```
[17] df[df["UN_Estimate"]>50000].head()
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year
1	Monaco	Europe	0	0	234316	2021	234317	2021
2	Liechtenstein	Europe	0	0	157755	2020	169260	2021
3	Luxembourg	Europe	132372	2023	133590	2021	133745	2021
4	Ireland	Europe	114581	2023	100172	2021	101109	2021
5	Bermuda	Americas	0	0	114090	2021	112653	2021



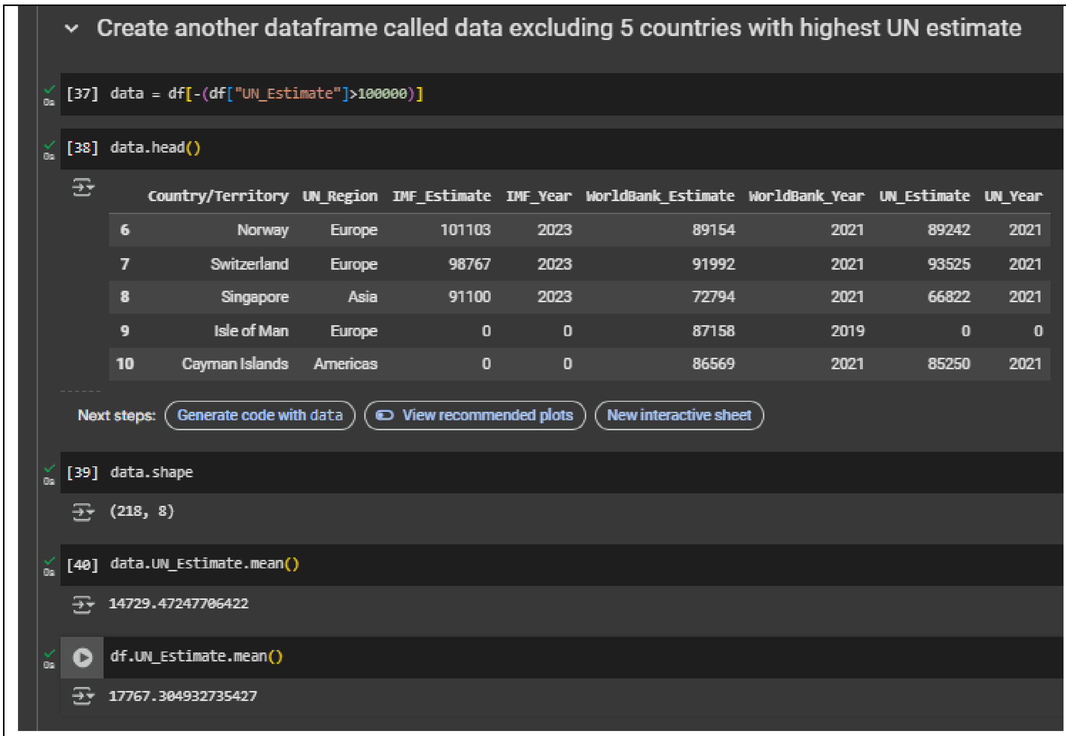
```
[18] sns.boxplot(x=df["WorldBank_Estimate"])

     plt.show()
```

```
[19] sns.boxplot(x=df["IMF_Estimate"])

     plt.show()
```

```
[20] df[df["UN_Estimate"]>100000]
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year
1	Monaco	Europe	0	0	234316	2021	234317	2021
2	Liechtenstein	Europe	0	0	157755	2020	169260	2021
3	Luxembourg	Europe	132372	2023	133590	2021	133745	2021
4	Ireland	Europe	114581	2023	100172	2021	101109	2021
5	Bermuda	Americas	0	0	114090	2021	112653	2021

```
[21] df.UN_Estimate.mean()

     17767.304932735427
```



Create another dataframe called data excluding 5 countries with highest UN estimate

```
[37] data = df[-(df["UN_Estimate"]>100000)]
```

```
[38] data.head()
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year
6	Norway	Europe	101103	2023	89154	2021	89242	2021
7	Switzerland	Europe	98767	2023	91992	2021	93525	2021
8	Singapore	Asia	91100	2023	72794	2021	66822	2021
9	Isle of Man	Europe	0	0	87158	2019	0	0
10	Cayman Islands	Americas	0	0	86569	2021	85250	2021

```
[39] data.shape
```

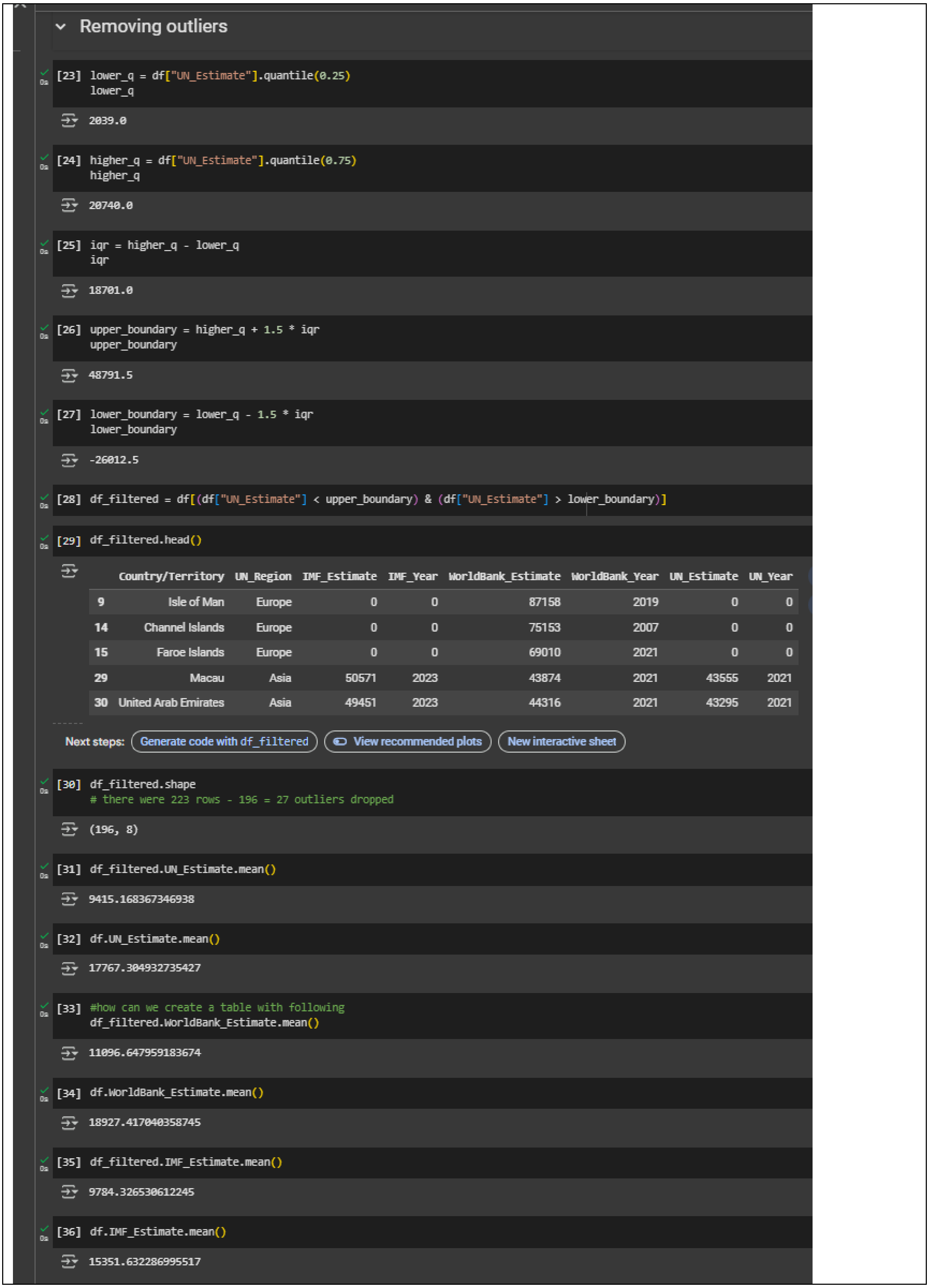
```
(218, 8)
```

```
[40] data.UN_Estimate.mean()
```

```
14729.47247706422
```

```
df.UN_Estimate.mean()
```

```
17767.304932735427
```



Removing outliers

```
[23] lower_q = df["UN_Estimate"].quantile(0.25)
     lower_q
```

```
2039.0
```

```
[24] higher_q = df["UN_Estimate"].quantile(0.75)
     higher_q
```

```
20740.0
```

```
[25] iqr = higher_q - lower_q
     iqr
```

```
18701.0
```

```
[26] upper_boundary = higher_q + 1.5 * iqr
     upper_boundary
```

```
48791.5
```

```
[27] lower_boundary = lower_q - 1.5 * iqr
     lower_boundary
```

```
-26012.5
```

```
[28] df_filtered = df[(df["UN_Estimate"] < upper_boundary) & (df["UN_Estimate"] > lower_boundary)]
```

```
[29] df_filtered.head()
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year
9	Isle of Man	Europe	0	0	87158	2019	0	0
14	Channel Islands	Europe	0	0	75153	2007	0	0
15	Faroe Islands	Europe	0	0	69010	2021	0	0
29	Macau	Asia	50571	2023	43874	2021	43555	2021
30	United Arab Emirates	Asia	49451	2023	44316	2021	43295	2021

```
[30] df_filtered.shape
     # there were 223 rows - 196 = 27 outliers dropped
```

```
(196, 8)
```

```
[31] df_filtered.UN_Estimate.mean()
```

```
9415.168367346938
```

```
[32] df.UN_Estimate.mean()
```

```
17767.304932735427
```

```
[33] #how can we create a table with following
     df_filtered.WorldBank_Estimate.mean()
```

```
11096.647959183674
```

```
[34] df.WorldBank_Estimate.mean()
```

```
18927.417040358745
```

```
[35] df_filtered.IMF_Estimate.mean()
```

```
9784.326530612245
```

```
[36] df.IMF_Estimate.mean()
```

```
15351.632286995517
```

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.