Just IT

B2W group

Apprenticeships | Training | Recruitment

Data Technician

Name: Amnah Bibi

Course Date: 03/03/2005

Table of contents

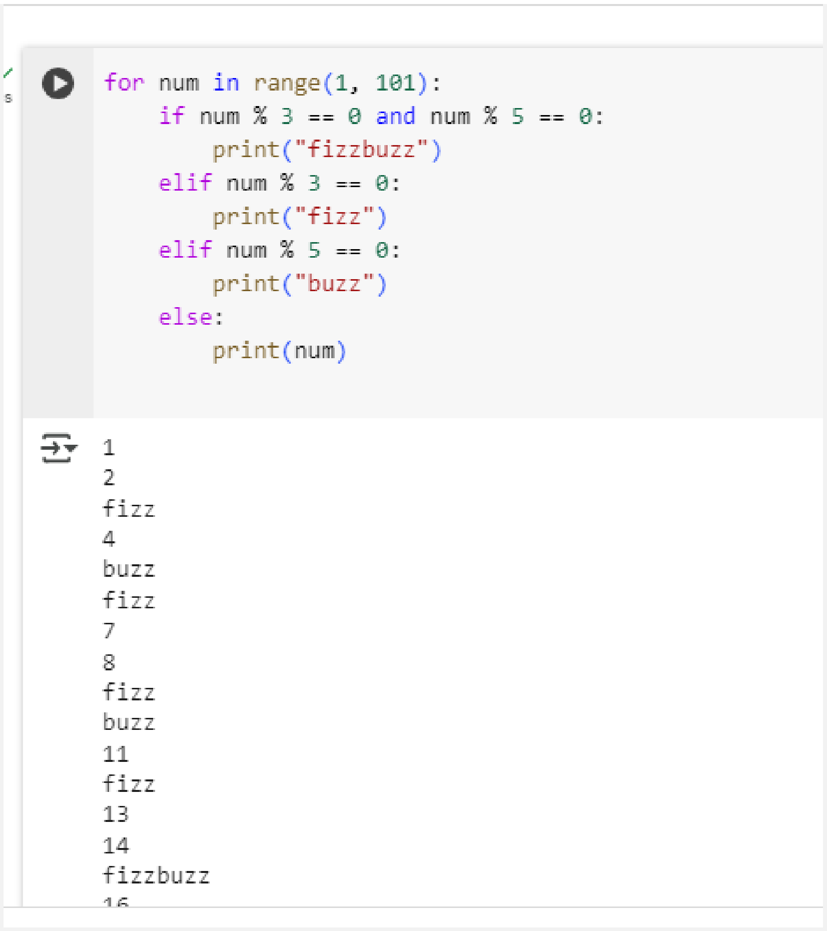
Day 2: Task 1

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 % 3 == 0:
        print("fizz")
    elif num % 5 == 0:
        print("buzz")
    else:
        print(num)
```

```
1
2
fizz
4
buzz
fizz
7
8
fizz
buzz
11
fizz
13
14
fizzbuzz
```

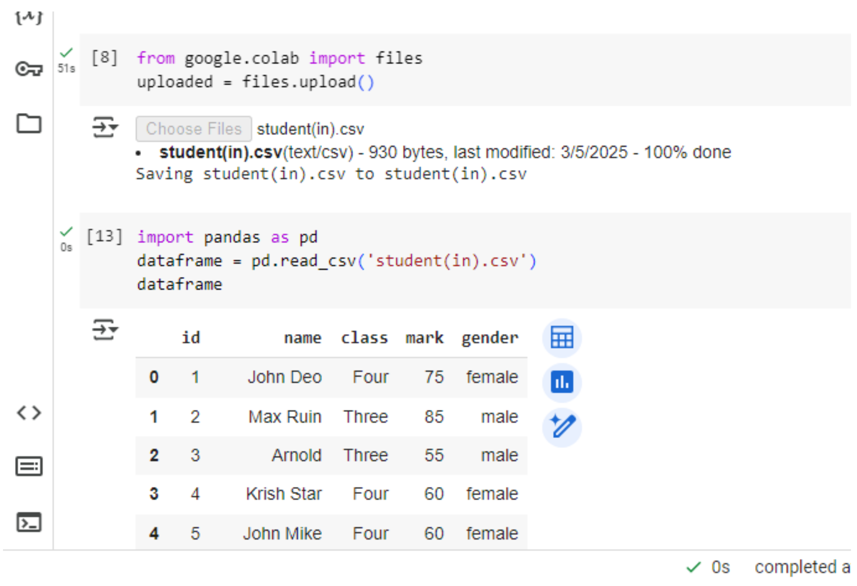
Day 3: Task 1

Using the 'student.csv' which can be downloaded here, 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."

1)



```
[8] from google.colab import files
    uploaded = files.upload()
```

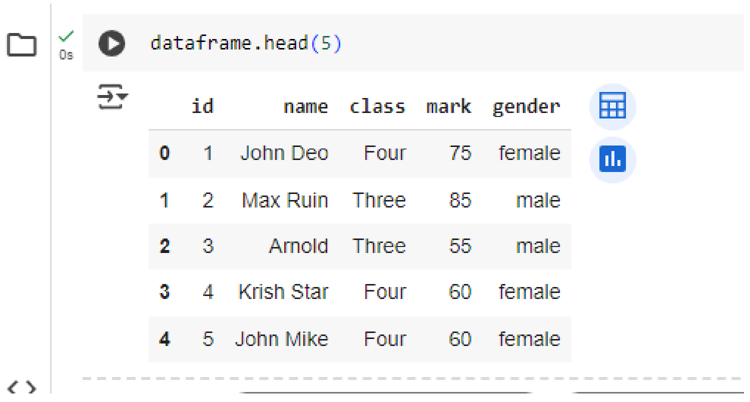
```
Choose Files student(in).csv
• student(in).csv(text/csv) - 930 bytes, last modified: 3/5/2025 - 100% done
Saving student(in).csv to student(in).csv
```

```
[13] import pandas as pd
     dataframe = pd.read_csv('student(in).csv')
     dataframe
```

```
   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
```

✓ 0s completed a

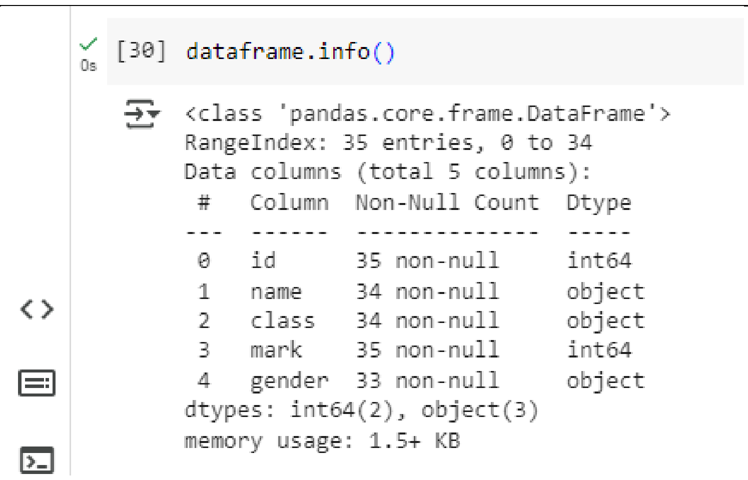
2)



```
dataframe.head(5)
```

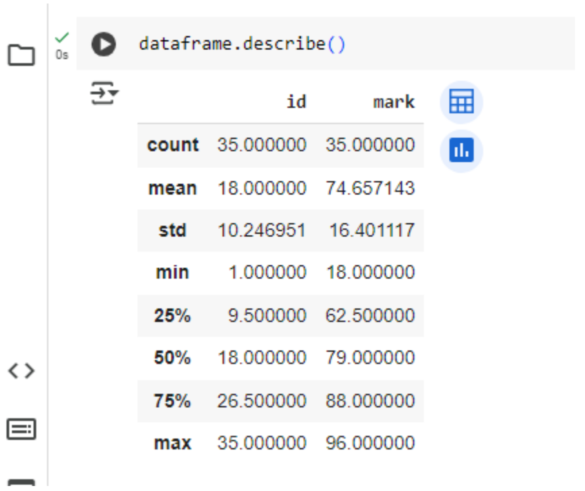
```
   id       name  class  mark  gender
0   1   John Deo   Four    75  female
1   2   Max Ruin  Three    85    male
2   3     Arnold  Three    55    male
3   4 Krish Star   Four    60  female
4   5  John Mike   Four    60  female
```

3)



```
[30] dataframe.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)



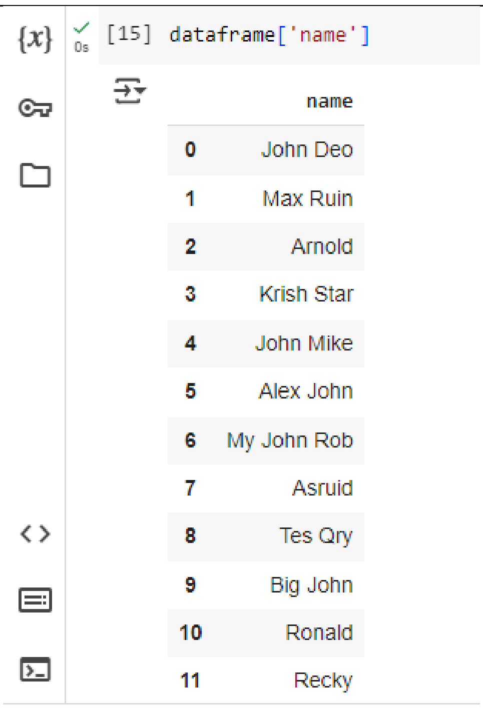
```
dataframe.describe()
```

	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'."

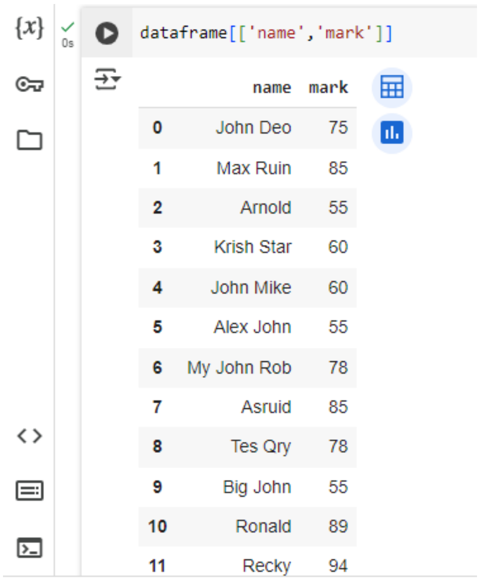
1)



```
[15] dataframe['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
```

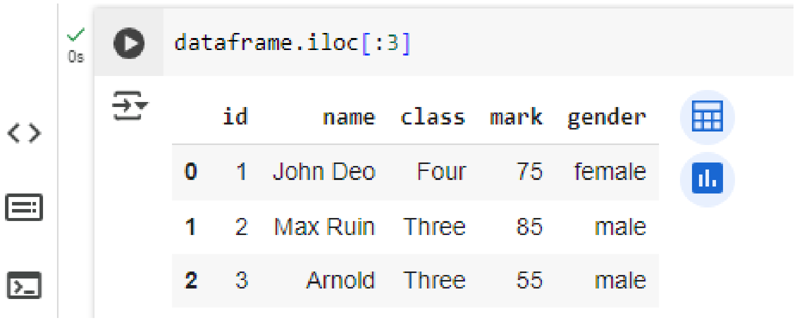
2)



```
dataframe[['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
```

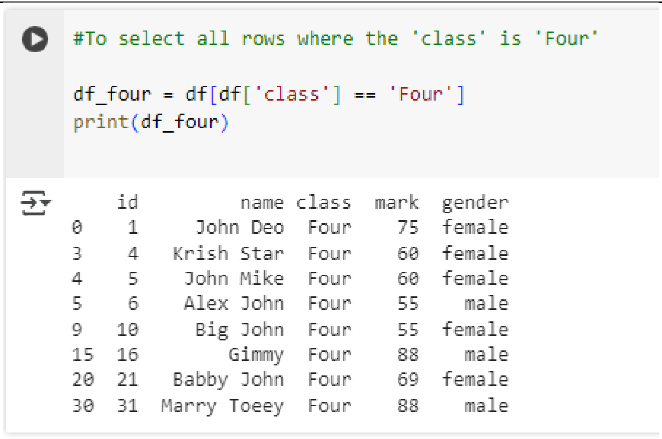
3)



```
dataframe.iloc[: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
```

4)



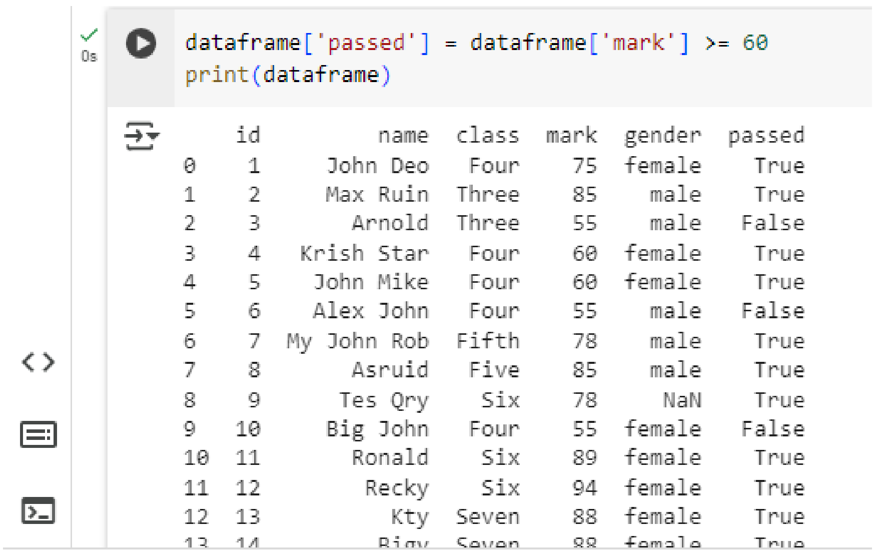
```
#To select all rows where the 'class' is 'Four'

df_four = df[df['class'] == 'Four']
print(df_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."

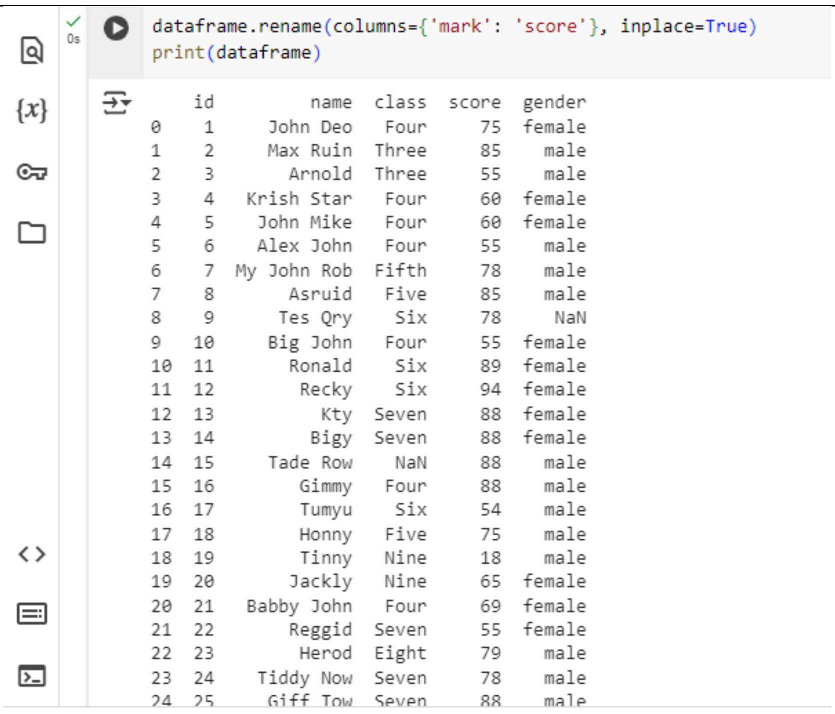
1)



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

```
    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
```

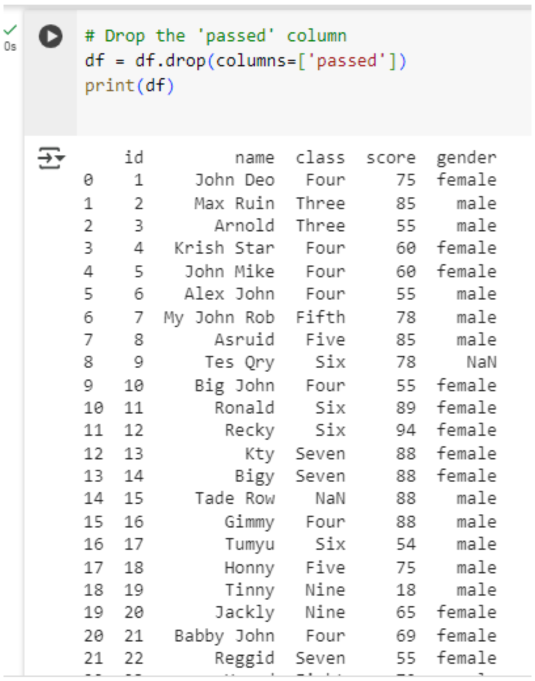
2)



```
dataframe.rename(columns={'mark': 'score'}, inplace=True)
print(dataframe)
```

```
    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
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
23  24     Tiddy Now  Seven     78    male
24  25      Giff Tow  Seven     88    male
```

3)



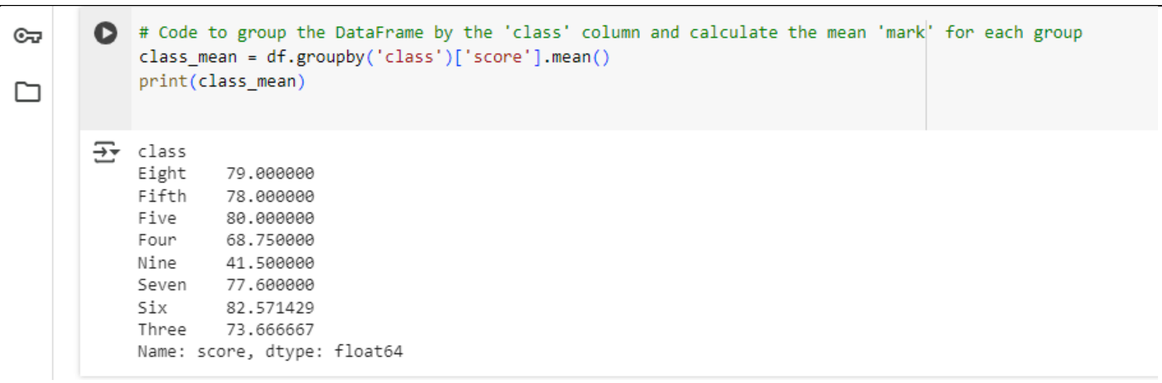
```
# Drop the 'passed' column
df = df.drop(columns=['passed'])
print(df)
```

```
    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
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
```

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."

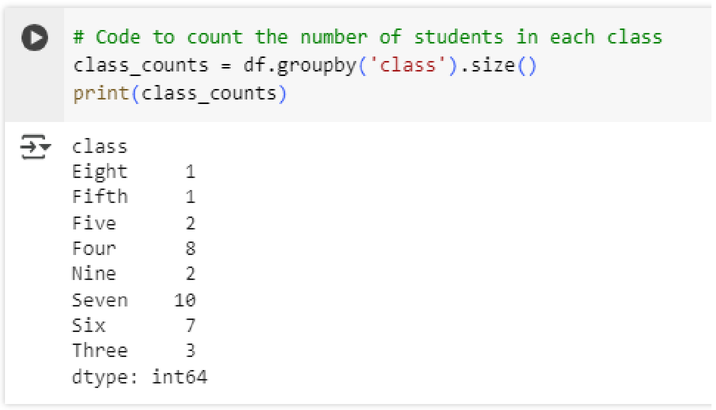
1)



```
# Code to group the DataFrame by the 'class' column and calculate the mean 'mark' for each group
class_mean = df.groupby('class')['score'].mean()
print(class_mean)
```

```
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
```

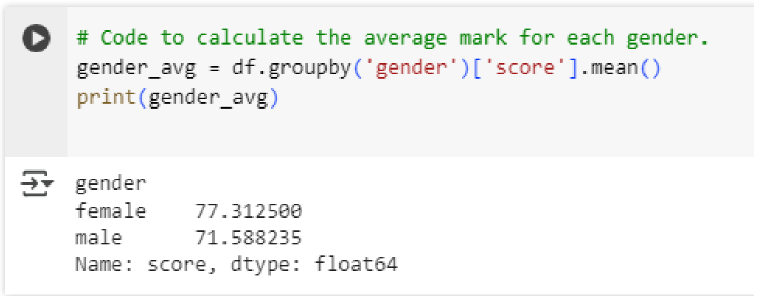
2)



```
# Code to count the number of students in each class
class_counts = df.groupby('class').size()
print(class_counts)
```

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

3)



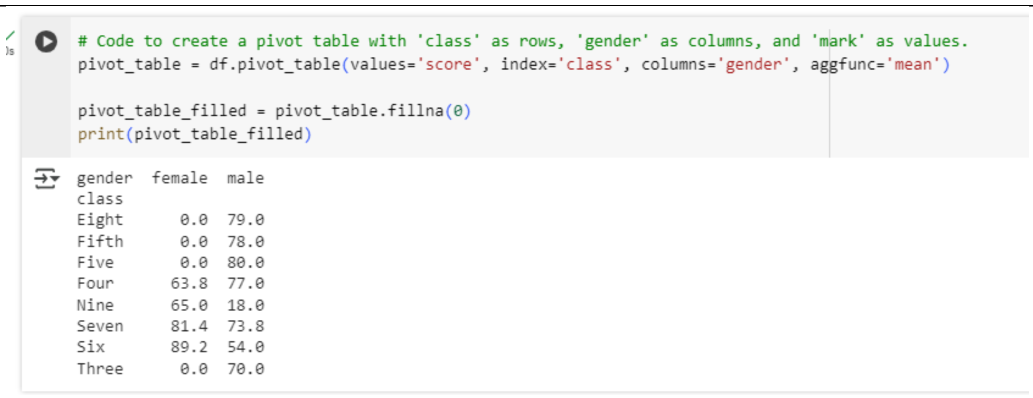
```
# Code to calculate the average mark for each gender.
gender_avg = df.groupby('gender')['score'].mean()
print(gender_avg)
```

```
gender
female    77.312500
male      71.588235
Name: score, 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."

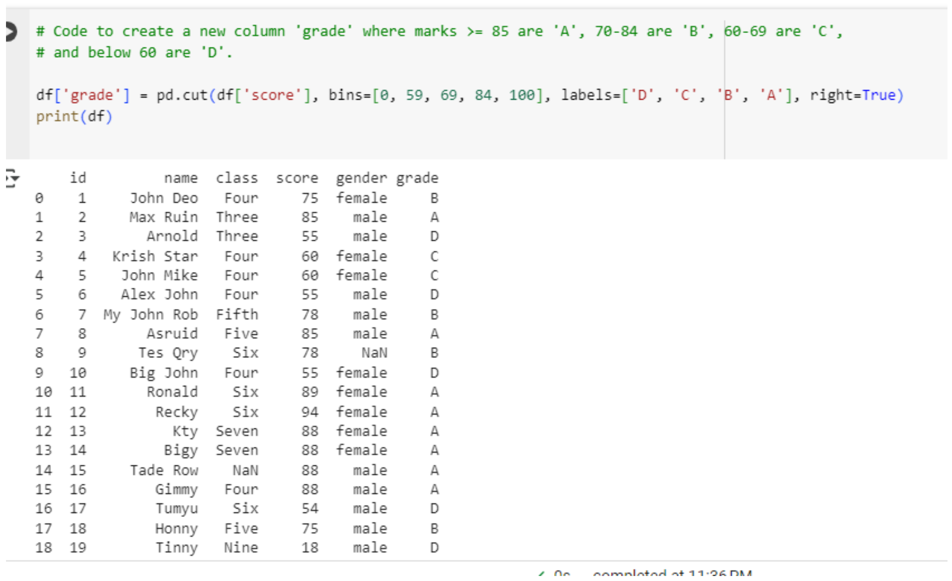
1)



```
# Code to create a pivot table with 'class' as rows, 'gender' as columns, and 'mark' as values.
pivot_table = df.pivot_table(values='score', index='class', columns='gender', aggfunc='mean')

pivot_table_filled = pivot_table.fillna(0)
print(pivot_table_filled)
```

```
gender  female  male
class
Eight      0.0  79.0
Fifth      0.0  78.0
Five       0.0  80.0
Four      63.8  77.0
Nine      65.0  18.0
Seven     81.4  73.8
Six       89.2  54.0
Three      0.0  70.0
```

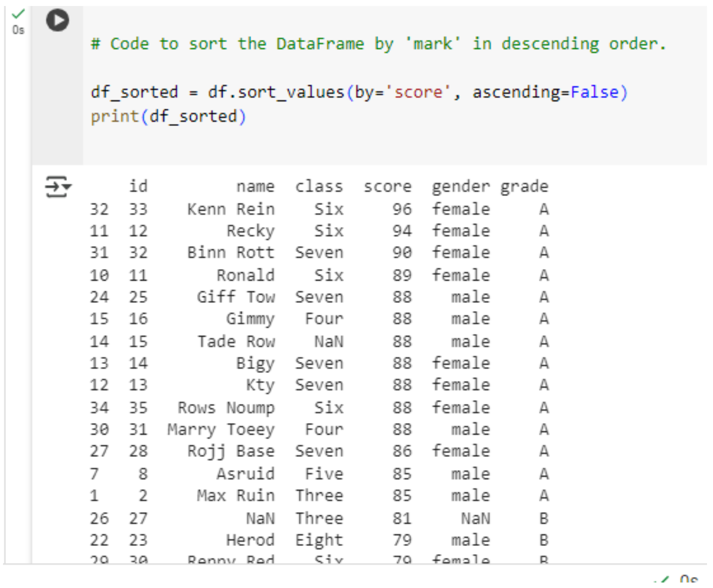
2)



```
# 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'.

df['grade'] = pd.cut(df['score'], bins=[0, 59, 69, 84, 100], labels=['D', 'C', 'B', 'A'], right=True)
print(df)
```

```
    id         name  class  score  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
14  15     Tade Row    NaN     88    male     A
15  16        Gimmy   Four     88    male     A
16  17        Tumyu    Six     54    male     D
17  18        Honny   Five     75    male     B
18  19        Tinny   Nine     18    male     D
```

3)

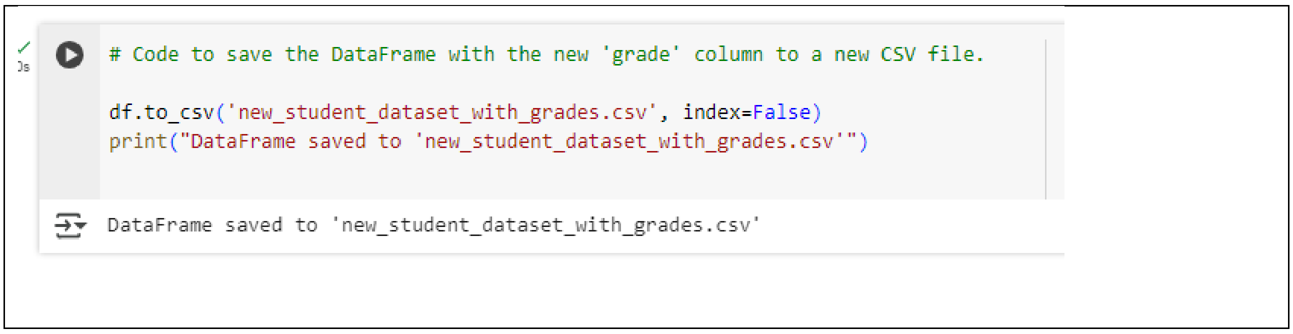


```
# Code to sort the DataFrame by 'mark' in descending order.

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

```
    id         name  class  score  gender grade
32  33    Kenn Rein    Six     96  female     A
11  12        Recky    Six     94  female     A
31  32    Binn Rott  Seven     90  female     A
10  11       Ronald    Six     89  female     A
24  25     Giff Tow  Seven     88    male     A
15  16        Gimmy   Four     88    male     A
14  15     Tade Row    NaN     88    male     A
13  14         Bigy  Seven     88  female     A
12  13          Kty  Seven     88  female     A
34  35   Rows Noump    Six     88  female     A
30  31  Marry Toeey   Four     88    male     A
27  28    Rojj Base  Seven     86  female     A
7    8       Asruid   Five     85    male     A
1    2     Max Ruin  Three     85    male     A
26  27          NaN  Three     81     NaN     B
22  23        Herod  Eight     79    male     B
```

Exercise 6: Exporting Data

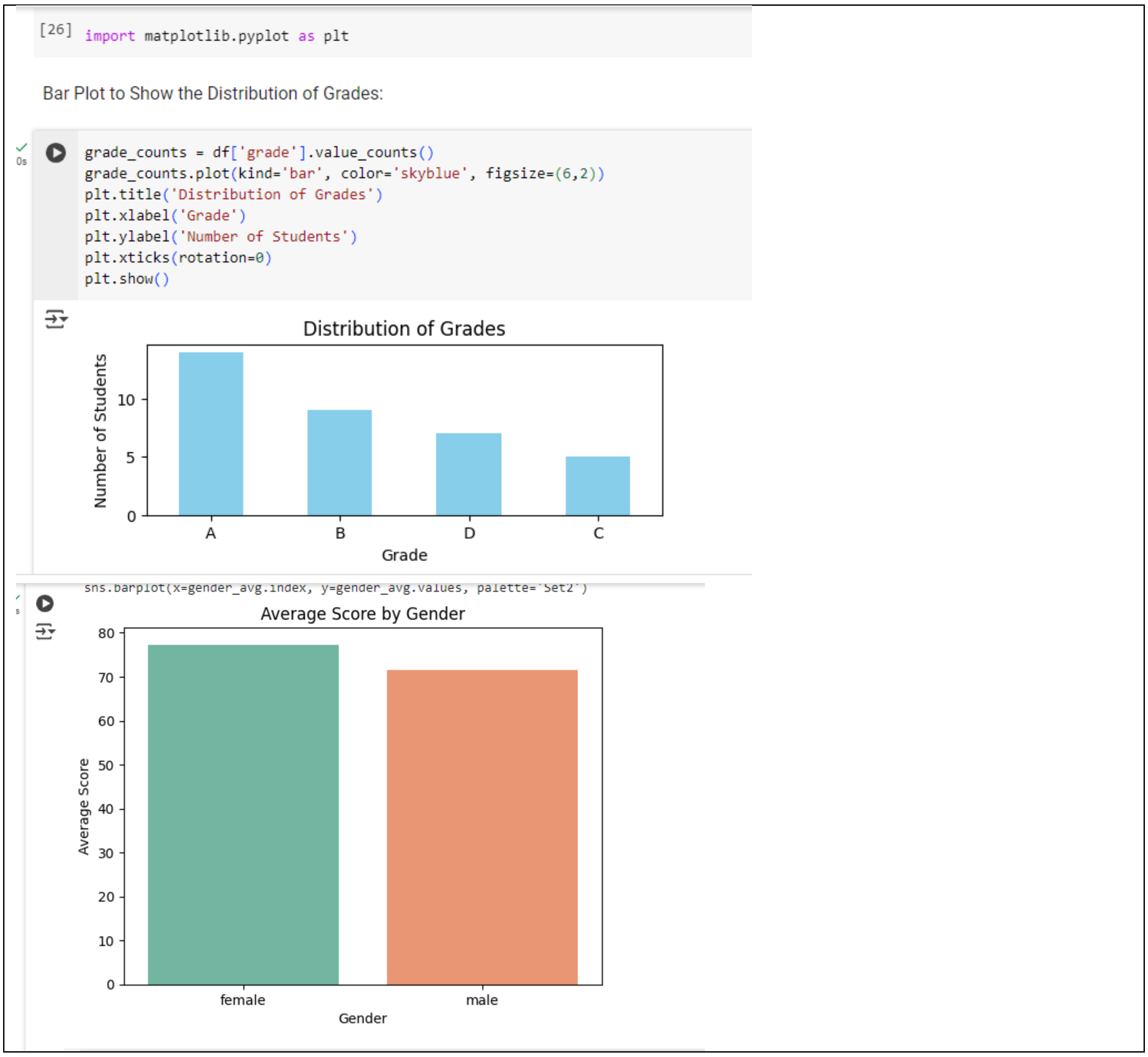
1. Question: "Write the code to save the DataFrame with the new 'grade' column to a new CSV file."



```
# Code to save the DataFrame with the new 'grade' column to a new CSV file.

df.to_csv('new_student_dataset_with_grades.csv', index=False)
print("DataFrame saved to 'new_student_dataset_with_grades.csv'")
```

```
DataFrame saved to 'new_student_dataset_with_grades.csv'
```

Exercise 7: If finished early try visualising the results

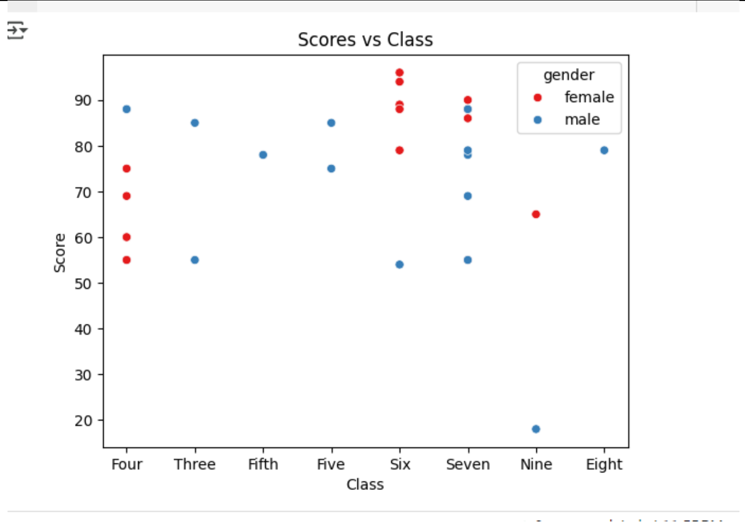


```
[26] import matplotlib.pyplot as plt
```

Bar Plot to Show the Distribution of Grades:

```
grade_counts = df['grade'].value_counts()
grade_counts.plot(kind='bar', color='skyblue', figsize=(6,2))
plt.title('Distribution of Grades')
plt.xlabel('Grade')
plt.ylabel('Number of Students')
plt.xticks(rotation=0)
plt.show()
```

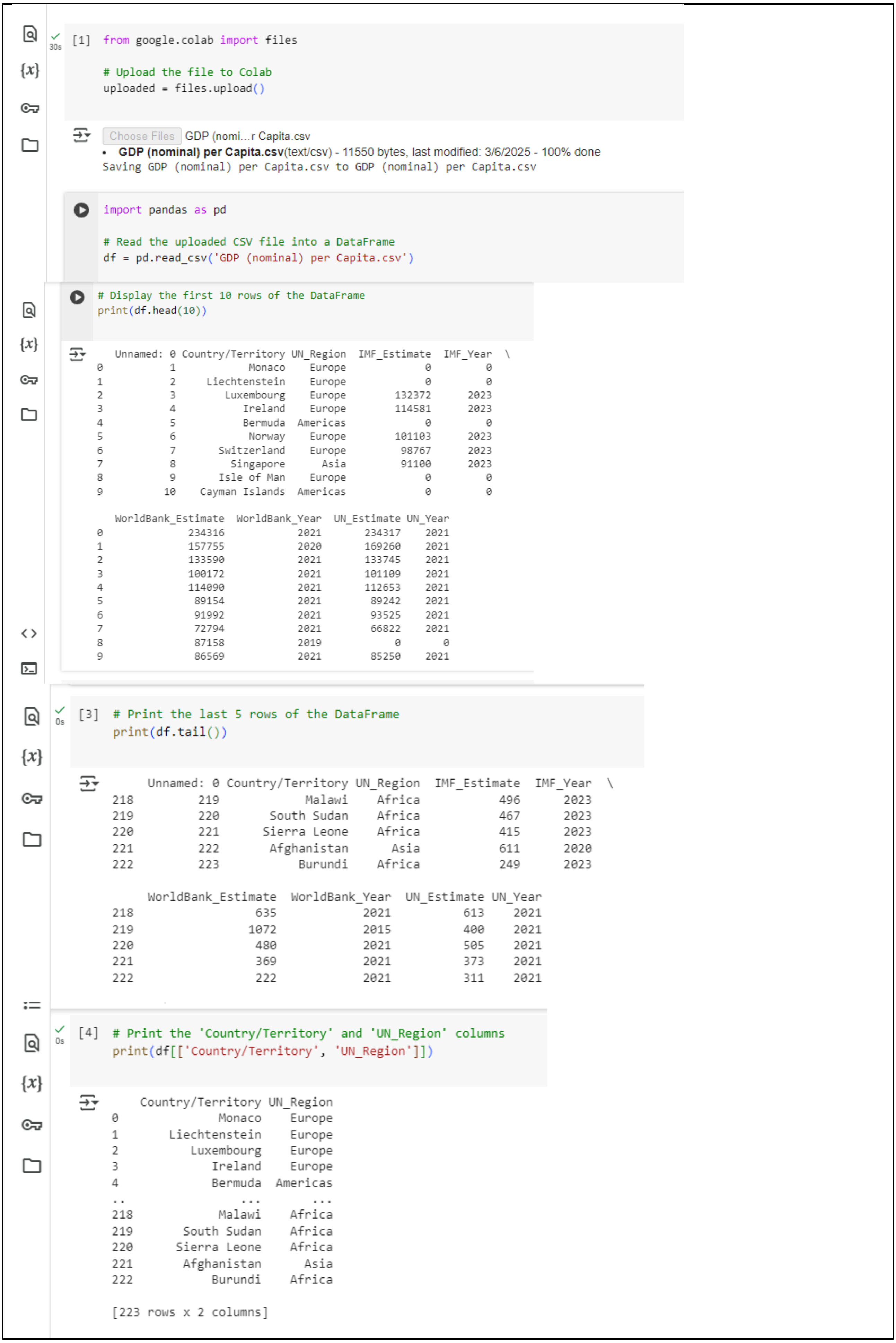
```
sns.barplot(x=gender_avg.index, y=gender_avg.values, palette='Set2')
```



Day 4: Task 1

Using the 'GDP (nominal) per Capita.csv' which can be downloaded here, 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 Jupyter notebook
- Print the first 10 rows
- Print the last 5 rows
- Print 'Country/Territory' and 'UN_Region' columns



```
[1] from google.colab import files

    # Upload the file to Colab
    uploaded = files.upload()
```

```
Choose Files GDP (nomi...r Capita.csv
• GDP (nominal) per Capita.csv(text/csv) - 11550 bytes, last modified: 3/6/2025 - 100% done
Saving GDP (nominal) per Capita.csv to GDP (nominal) per Capita.csv
```

```
import pandas as pd

# Read the uploaded CSV file into a DataFrame
df = pd.read_csv('GDP (nominal) per Capita.csv')
```

```
# Display the first 10 rows of the DataFrame
print(df.head(10))
```

```
   Unnamed: 0 Country/Territory UN_Region  IMF_Estimate  IMF_Year  \
0           1            Monaco    Europe             0         0
1           2     Liechtenstein    Europe             0         0
2           3        Luxembourg    Europe        132372      2023
3           4           Ireland    Europe        114581      2023
4           5           Bermuda  Americas             0         0
5           6            Norway    Europe        101103      2023
6           7       Switzerland    Europe         98767      2023
7           8         Singapore      Asia         91100      2023
8           9       Isle of Man    Europe             0         0
9          10    Cayman Islands  Americas             0         0

   WorldBank_Estimate  WorldBank_Year  UN_Estimate UN_Year
0              234316            2021       234317    2021
1              157755            2020       169260    2021
2              133590            2021       133745    2021
3              100172            2021       101109    2021
4              114090            2021       112653    2021
5               89154            2021        89242    2021
6               91992            2021        93525    2021
7               72794            2021        66822    2021
8               87158            2019            0       0
9               86569            2021        85250    2021
```

```
[3] # Print the last 5 rows of the DataFrame
    print(df.tail())
```

```
     Unnamed: 0 Country/Territory UN_Region  IMF_Estimate  IMF_Year  \
218         219            Malawi    Africa           496      2023
219         220       South Sudan    Africa           467      2023
220         221      Sierra Leone    Africa           415      2023
221         222       Afghanistan      Asia           611      2020
222         223           Burundi    Africa           249      2023

     WorldBank_Estimate  WorldBank_Year  UN_Estimate UN_Year
218                 635            2021          613    2021
219                1072            2015          400    2021
220                 480            2021          505    2021
221                 369            2021          373    2021
222                 222            2021          311    2021
```

```
[4] # Print the 'Country/Territory' and 'UN_Region' columns
    print(df[['Country/Territory', 'UN_Region']])
```

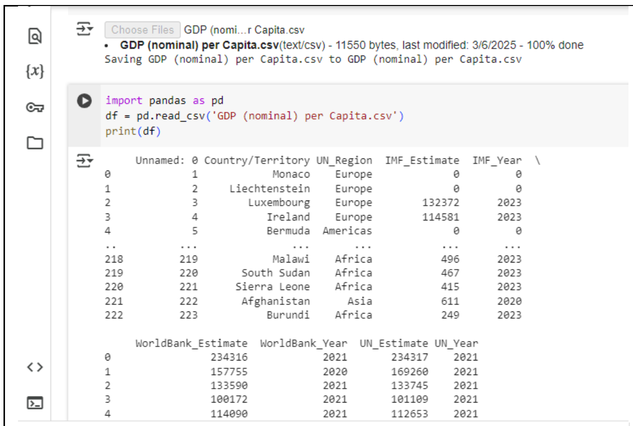
```
    Country/Territory UN_Region
0              Monaco    Europe
1       Liechtenstein    Europe
2          Luxembourg    Europe
3             Ireland    Europe
4             Bermuda  Americas
..                ...       ...
218            Malawi    Africa
219       South Sudan    Africa
220     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

[223 rows x 2 columns]
```

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 here. 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.

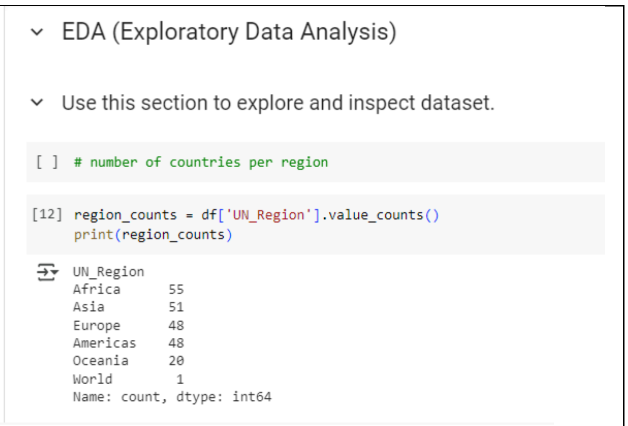


Choose Files GDP (nomi...r Capita.csv
- **GDP (nominal) per Capita.csv**(text/csv) - 11550 bytes, last modified: 3/6/2025 - 100% done

```
Saving GDP (nominal) per Capita.csv to GDP (nominal) per Capita.csv
```

```python
import pandas as pd
df = pd.read_csv('GDP (nominal) per Capita.csv')
print(df)
```

```
     Unnamed: 0 Country/Territory UN_Region  IMF_Estimate  IMF_Year  \
0             1            Monaco    Europe             0         0
1             2     Liechtenstein    Europe             0         0
2             3        Luxembourg    Europe        132372      2023
3             4           Ireland    Europe        114581      2023
4             5           Bermuda  Americas             0         0
..          ...               ...       ...           ...       ...
218         219            Malawi    Africa           496      2023
219         220       South Sudan    Africa           467      2023
220         221      Sierra Leone    Africa           415      2023
221         222       Afghanistan      Asia           611      2020
222         223           Burundi    Africa           249      2023

     WorldBank_Estimate  WorldBank_Year  UN_Estimate UN_Year
0                234316            2021       234317    2021
1                157755            2020       169260    2021
2                133590            2021       133745    2021
3                100172            2021       101109    2021
4                114090            2021       112653    2021
```



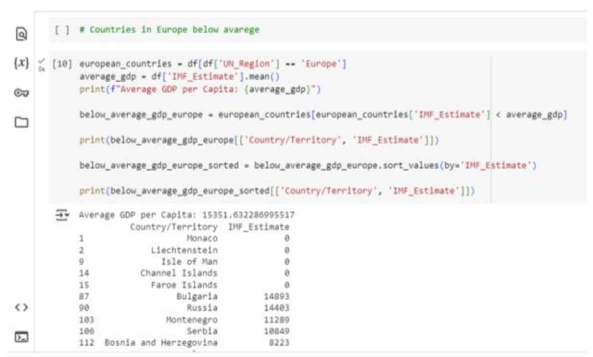
EDA (Exploratory Data Analysis)

Use this section to explore and inspect dataset.

```python
# number of countries per region
```

```python
[12] region_counts = df['UN_Region'].value_counts()
     print(region_counts)
```

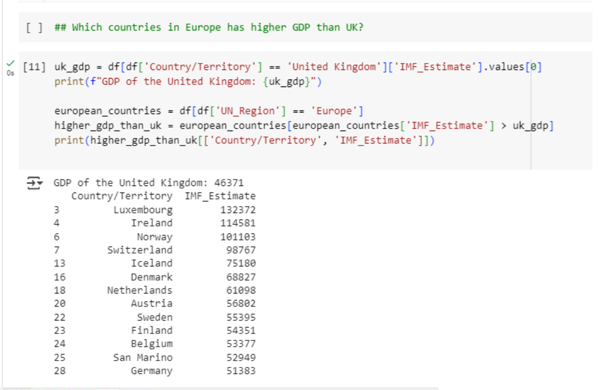
```
UN_Region
Africa      55
Asia        51
Europe      48
Americas    48
Oceania     20
World        1
Name: count, dtype: int64
```



```python
# Countries in Europe below avarege
```

```python
[10] european_countries = df[df['UN_Region'] == 'Europe']
     average_gdp = df['IMF_Estimate'].mean()
     print(f"Average GDP per Capita: {average_gdp}")

     below_average_gdp_europe = european_countries[european_countries['IMF_Estimate'] < average_gdp]

     print(below_average_gdp_europe[['Country/Territory', 'IMF_Estimate']])

     below_average_gdp_europe_sorted = below_average_gdp_europe.sort_values(by='IMF_Estimate')

     print(below_average_gdp_europe_sorted[['Country/Territory', 'IMF_Estimate']])
```

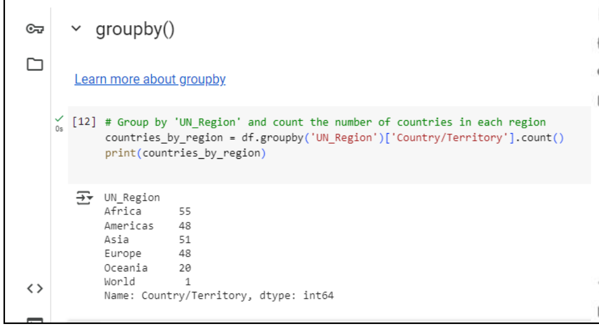
```
Average GDP per Capita: 15351.632286995517
         Country/Territory  IMF_Estimate
1                   Monaco             0
2            Liechtenstein             0
9              Isle of Man             0
14         Channel Islands             0
15           Faroe Islands             0
87                Bulgaria         14893
90                  Russia         14403
103             Montenegro         11289
106                 Serbia         10849
112 Bosnia and Herzegovina          8223
```



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

```python
[11] uk_gdp = df[df['Country/Territory'] == 'United Kingdom']['IMF_Estimate'].values[0]
     print(f"GDP of the United Kingdom: {uk_gdp}")

     european_countries = df[df['UN_Region'] == 'Europe']
     higher_gdp_than_uk = european_countries[european_countries['IMF_Estimate'] > uk_gdp]
     print(higher_gdp_than_uk[['Country/Territory', 'IMF_Estimate']])
```

```
GDP of the United Kingdom: 46371
   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
```

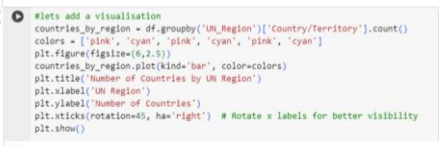


groupby()

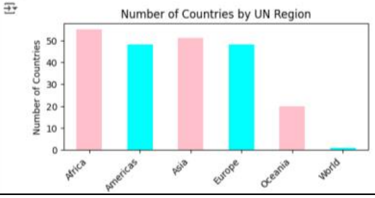
Learn more about groupby

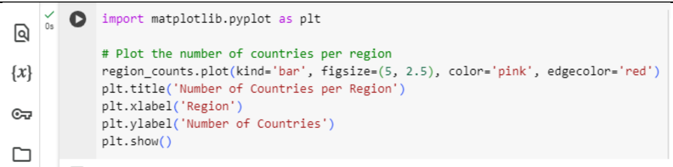
```python
[12] # Group by 'UN_Region' and count the number of countries in each region
     countries_by_region = df.groupby('UN_Region')['Country/Territory'].count()
     print(countries_by_region)
```

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

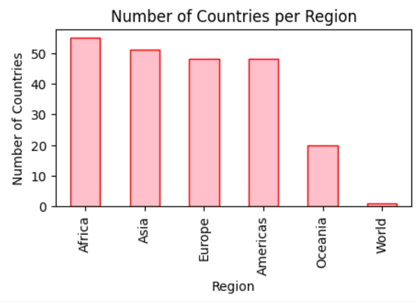


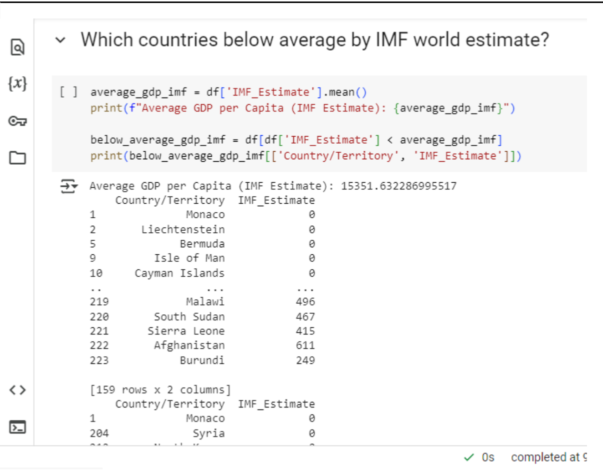
```python
#lets add a visualisation
countries_by_region = df.groupby('UN_Region')['Country/Territory'].count()
colors = ['pink', 'cyan', 'pink', 'cyan', 'pink', 'cyan']
plt.figure(figsize=(6,2.5))
countries_by_region.plot(kind='bar', color=colors)
plt.title('Number of Countries by UN Region')
plt.xlabel('UN Region')
plt.ylabel('Number of Countries')
plt.xticks(rotation=45, ha='right')  # Rotate x labels for better visibility
plt.show()
```





```
import matplotlib.pyplot as plt

# Plot the number of countries per region
region_counts.plot(kind='bar', figsize=(5, 2.5), color='pink', edgecolor='red')
plt.title('Number of Countries per Region')
plt.xlabel('Region')
plt.ylabel('Number of Countries')
plt.show()
```



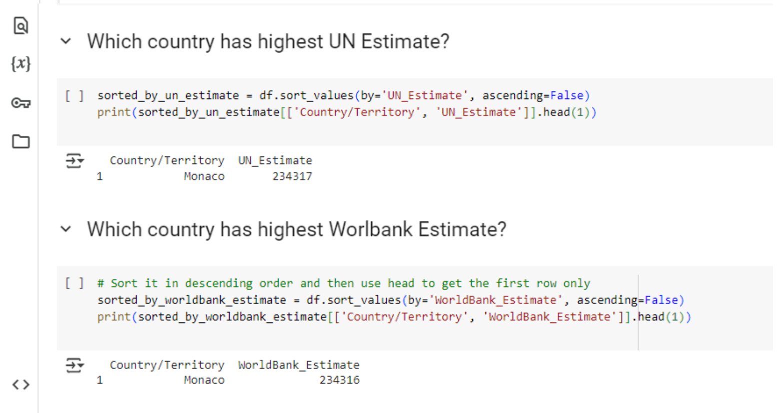


Which countries below average by IMF world estimate?

```
[ ] average_gdp_imf = df['IMF_Estimate'].mean()
    print(f"Average GDP per Capita (IMF Estimate): {average_gdp_imf}")

    below_average_gdp_imf = df[df['IMF_Estimate'] < average_gdp_imf]
    print(below_average_gdp_imf[['Country/Territory', 'IMF_Estimate']])
```

```
Average GDP per Capita (IMF Estimate): 15351.632286995517
    Country/Territory  IMF_Estimate
1              Monaco             0
2       Liechtenstein             0
5             Bermuda             0
9         Isle of Man             0
10     Cayman Islands             0
..                ...           ...
219            Malawi           496
220       South Sudan           467
221      Sierra Leone           415
222       Afghanistan           611
223           Burundi           249

[159 rows x 2 columns]
    Country/Territory  IMF_Estimate
1              Monaco             0
204             Syria             0
```

✓ 0s completed at



Which country has highest UN Estimate?

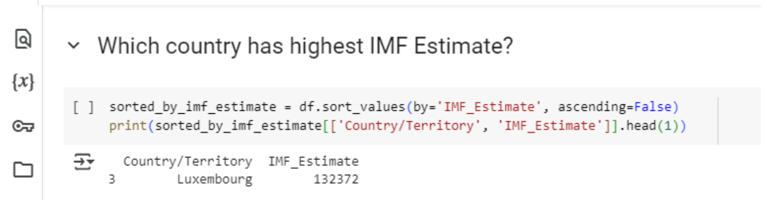
```
[ ] sorted_by_un_estimate = df.sort_values(by='UN_Estimate', ascending=False)
    print(sorted_by_un_estimate[['Country/Territory', 'UN_Estimate']].head(1))
```

```
  Country/Territory  UN_Estimate
1            Monaco       234317
```

Which country has highest Worlbank Estimate?

```
[ ] # Sort it in descending order and then use head to get the first row only
    sorted_by_worldbank_estimate = df.sort_values(by='WorldBank_Estimate', ascending=False)
    print(sorted_by_worldbank_estimate[['Country/Territory', 'WorldBank_Estimate']].head(1))
```

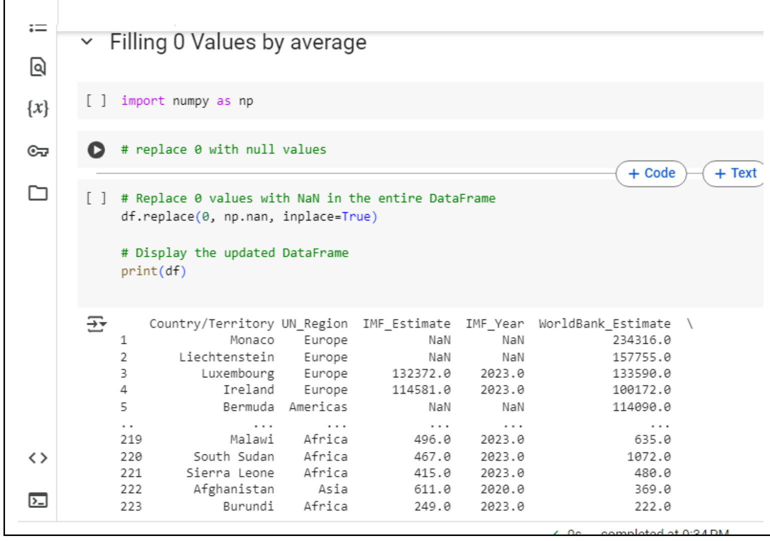
```
  Country/Territory  WorldBank_Estimate
1            Monaco              234316
```



Which country has highest IMF Estimate?

```
[ ] sorted_by_imf_estimate = df.sort_values(by='IMF_Estimate', ascending=False)
    print(sorted_by_imf_estimate[['Country/Territory', 'IMF_Estimate']].head(1))
```

```
  Country/Territory  IMF_Estimate
3        Luxembourg        132372
```



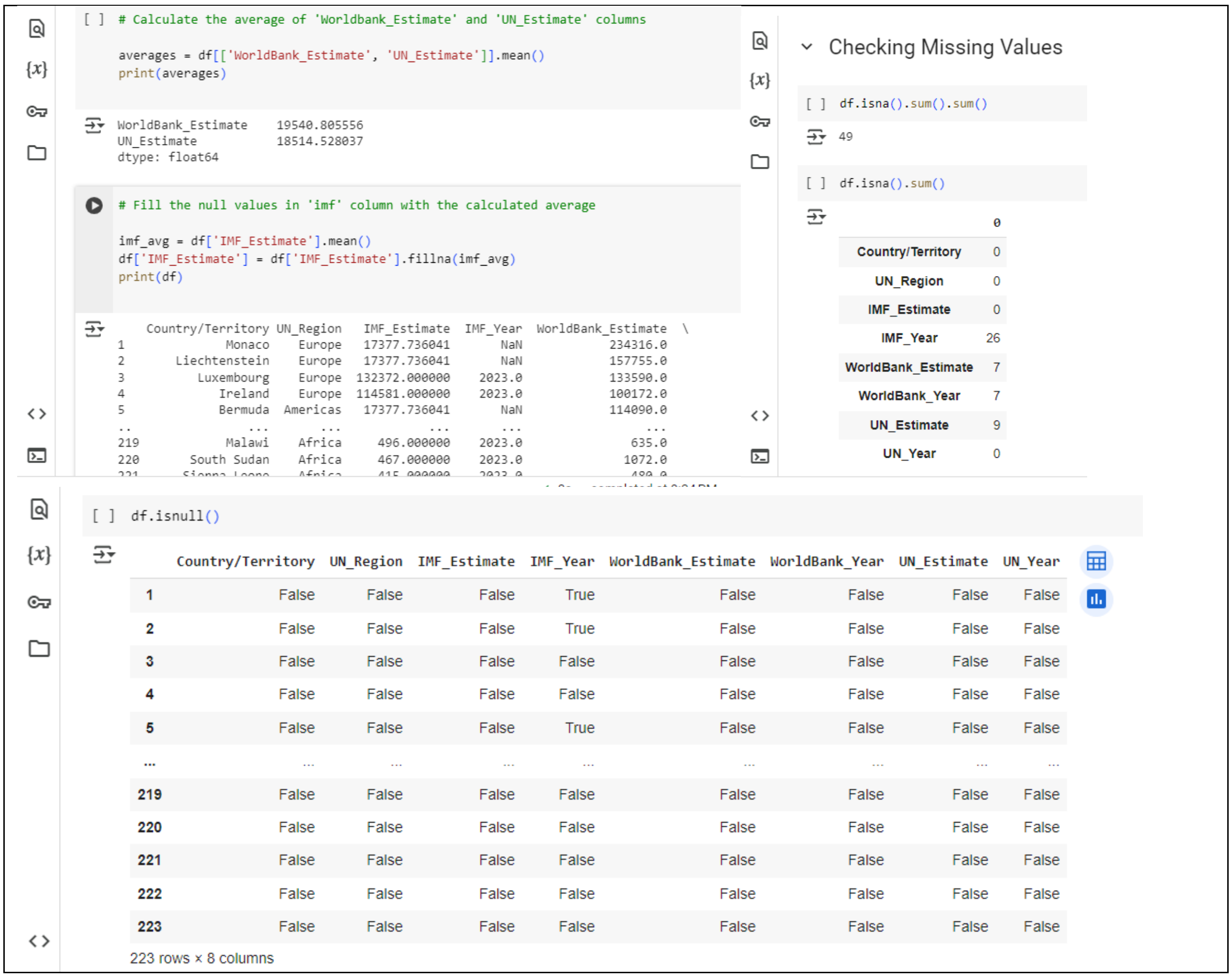
Filling 0 Values by average

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

```
# replace 0 with null values
```

```
[ ] # Replace 0 values with NaN in the entire DataFrame
    df.replace(0, np.nan, inplace=True)

    # Display the updated DataFrame
    print(df)
```

```
    Country/Territory UN_Region  IMF_Estimate  IMF_Year  WorldBank_Estimate  \
1              Monaco    Europe           NaN       NaN            234316.0
2       Liechtenstein    Europe           NaN       NaN            157755.0
3          Luxembourg    Europe      132372.0    2023.0            133590.0
4             Ireland    Europe      114581.0    2023.0            100172.0
5             Bermuda  Americas           NaN       NaN            114090.0
..                ...       ...           ...       ...                 ...
219            Malawi    Africa         496.0    2023.0               635.0
220       South Sudan    Africa         467.0    2023.0              1072.0
221      Sierra Leone    Africa         415.0    2023.0               480.0
222       Afghanistan      Asia         611.0    2020.0               369.0
223           Burundi    Africa         249.0    2023.0               222.0
```



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

    averages = df[['WorldBank_Estimate', 'UN_Estimate']].mean()
    print(averages)
```

```
WorldBank_Estimate    19540.805556
UN_Estimate           18514.528037
dtype: float64
```

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

imf_avg = df['IMF_Estimate'].mean()
df['IMF_Estimate'] = df['IMF_Estimate'].fillna(imf_avg)
print(df)
```

```
    Country/Territory UN_Region   IMF_Estimate  IMF_Year  WorldBank_Estimate  \
1              Monaco    Europe   17377.736041       NaN            234316.0
2       Liechtenstein    Europe   17377.736041       NaN            157755.0
3          Luxembourg    Europe  132372.000000    2023.0            133590.0
4             Ireland    Europe  114581.000000    2023.0            100172.0
5             Bermuda  Americas   17377.736041       NaN            114090.0
..                ...       ...            ...       ...                 ...
219            Malawi    Africa     496.000000    2023.0               635.0
220       South Sudan    Africa     467.000000    2023.0              1072.0
```

Checking Missing Values

```
[ ] df.isna().sum().sum()
```

```
49
```

```
[ ] df.isna().sum()
```

	0
Country/Territory	0
UN_Region	0
IMF_Estimate	0
IMF_Year	26
WorldBank_Estimate	7
WorldBank_Year	7
UN_Estimate	9
UN_Year	0

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

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year
1	False	False	False	True	False	False	False	False
2	False	False	False	True	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
5	False	False	False	True	False	False	False	False
...	...	...	...	...	...	...	...	...
219	False	False	False	False	False	False	False	False
220	False	False	False	False	False	False	False	False
221	False	False	False	False	False	False	False	False
222	False	False	False	False	False	False	False	False
223	False	False	False	False	False	False	False	False

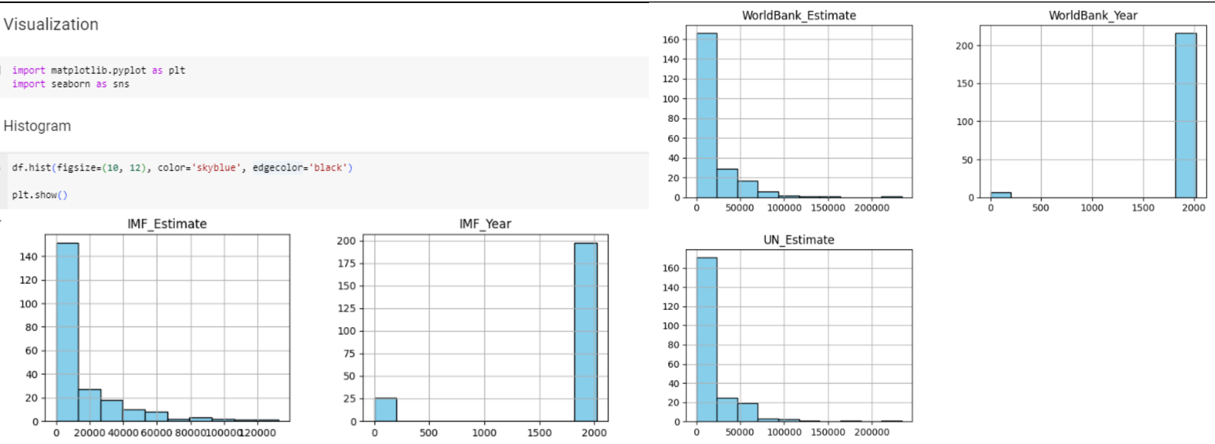
223 rows × 8 columns

Visualization

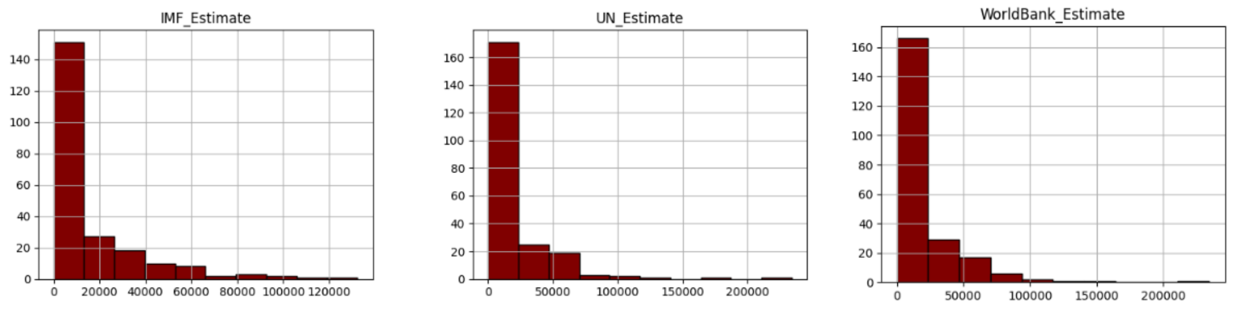
```
[ ] import matplotlib.pyplot as plt
    import seaborn as sns
```

Histogram

```
df.hist(figsize=(10, 12), color='skyblue', edgecolor='black')

plt.show()
```



```
df[["IMF_Estimate", "UN_Estimate", "WorldBank_Estimate"]].hist(figsize=(12,9),color='maroon', edgecolor='black')

plt.show()
```



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

	WorldBank_Estimate
min	222.0
max	234316.0

dtype: float64

```
234316/5
#1 bin size if bins=5
```

```
46863.2
```

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

```
188
```

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

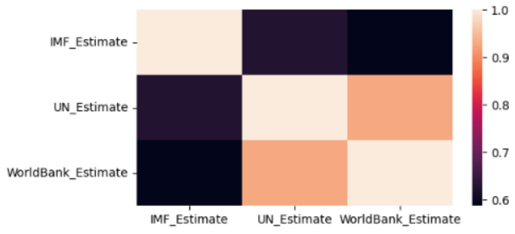
```
23431.6
```

Correlation Heatmap

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

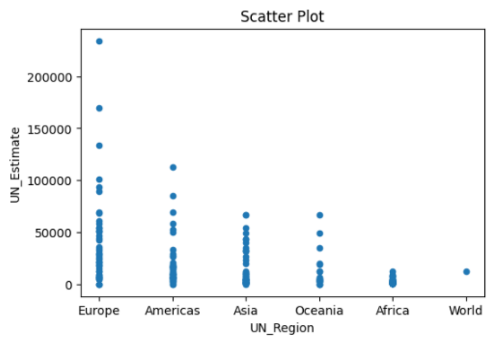
	IMF_Estimate	UN_Estimate	WorldBank_Estimate
IMF_Estimate	1.000000	0.709518	0.695935
UN_Estimate	0.709518	1.000000	0.998438
WorldBank_Estimate	0.695935	0.998438	1.000000

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

plt.figure(figsize=(6,3))
sns.heatmap(corr)
plt.show()
```



Scatter Plot

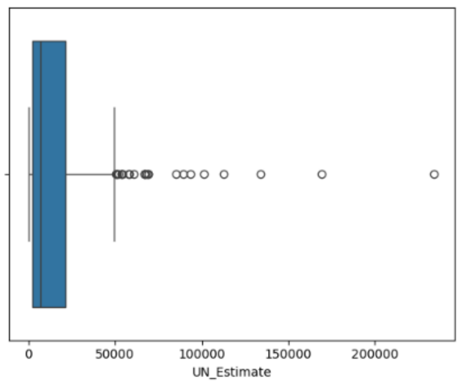
```
df.plot(x='UN_Region', y='UN_Estimate', kind='scatter',
        figsize=(6,4),
        title="Scatter Plot")
plt.show()
```



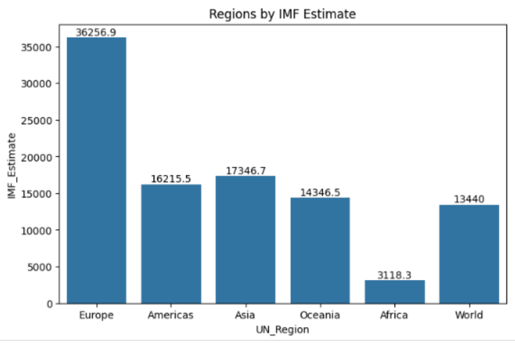
Boxplot and Outliers

image.png

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



```
[ ] 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()
```



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.