Just IT

B2W group

Apprenticeships | Training | Recruitment

Data Technician

Name: Youhem Rouainia

Course Date: 03/02/2025

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

Creating a program that goes from 1 to 100 and test for the number using Python syntax

```
# Generate an integer between 1 and 100 (inclusive)
  # num=int(input("please insert a number , "))
for num in range (1,101,3):
  if num%3==0 and num%5==0:  # testing if the number can be  divided by 3 and 5
    print("fizzbuzz")
  elif num%3==0:             # test if the number can be divided by 3 only
    print("fizz")
  elif num%5==0:             # test if the number can be divided by 5 only
    print("buzz")
else:
  print("number")
```

```
buzz
buzz
buzz
buzz
buzz
buzz
buzz
number
```

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."

1-Read the csv file `df=pd.read_csv('student.csv')`

2-Display the first 5 rows. `df.head()`

```
[18] 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-Get information about dataframe. `df.info()`

```
[21] 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-Get summary statistics for the dataframe `df.describe()`

```
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'."

1-Select the 'name' column. `df['name']`

```
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

2-Select the 'name' and 'mark' columns `df[['name','mark']]`

```
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
13	Bigy	88

3-Select the first 3 rows `df.head(3)`

```
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

4- Select all rows where the 'class' is 'Four'. `class_four = df[df['class'] == 'Four']`
`print(class_four)`

```
class_four = df[df['class'] == 'Four']
print(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."

1-Adding a new column “passed and testing if the student passed (mark>60)
`df['passed'] = df['mark'] >= 60`
`print(df)`

```
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
```

2-Rename the “mark” column to “score”

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

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

3-Dropping the passed column

```
df.drop(columns=['passed'], inplace=True)
print(df)
```

```
df.drop(columns=['passed'], inplace=True)
print(df)
```

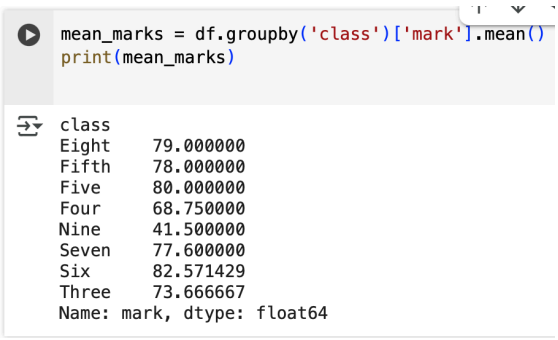
```
    id         name  class  score  gender
0    1     John Deo   Four     75  female
1    2     Max Ruin  Three     85    male
2    3       Arnold  Three     55    male
3    4   Krish Star   Four     60  female
4    5    John Mike   Four     60  female
5    6    Alex John   Four     55    male
6    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
```

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-Group the dataframe by the “class” column and calculating the mean “mark” for each group.

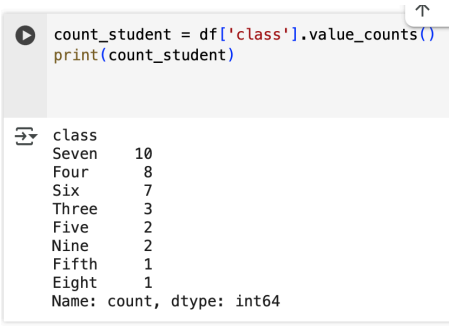
```
mean_marks = df.groupby('class')['mark'].mean()
print(mean_marks)
```



```
mean_marks = df.groupby('class')['mark'].mean()
print(mean_marks)

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
```

2-Counting the number of students in each class.

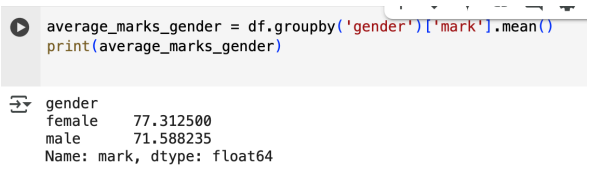
```
count_student = df['class'].value_counts()
print(count_student)
```



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

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

3-Calculating the average mark for each gender.

```
average_marks_gender = df.groupby('gender')['mark'].mean()
print(average_marks_gender)
```



```
average_marks_gender = df.groupby('gender')['mark'].mean()
print(average_marks_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."

1-Creating a pivot table with ‘class’ as row, ‘gender’ as columns and ‘mark’ as values.

```
pivot = df.pivot_table(values='mark', index='class', columns='gender', aggfunc='mean')
print(pivot)
```

```
pivot = df.pivot_table(values='mark', index='class', columns='gender', aggfunc='mean')
print(pivot)
```

```
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
```

2- creating a new column 'grade' where marks >= 85 are 'A', 70-84 are 'B', 60-69 are 'C', and below 60 are 'D'."

```
def grade(mark):
    if mark >= 85:
        return 'A'
    elif mark >= 70:
        return 'B'
    elif mark >= 60:
        return 'C'
    else:
        return 'D'

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

```
def grade(mark):
    if mark >= 85:
        return 'A'
    elif mark >= 70:
        return 'B'
    elif mark >= 60:
        return 'C'
    else:
        return 'D'

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

```
    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
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
19  20       Jackly   Nine    65  female     C
20  21   Babby John   Four    69  female     C
```

3-sorting the DataFrame by 'mark' in descending order."

```
sorted_df = df.sort_values(by='mark', ascending=False)
print(sorted_df)
```

```
sorted_df = df.sort_values(by='mark', ascending=False)
print(sorted_df)
```

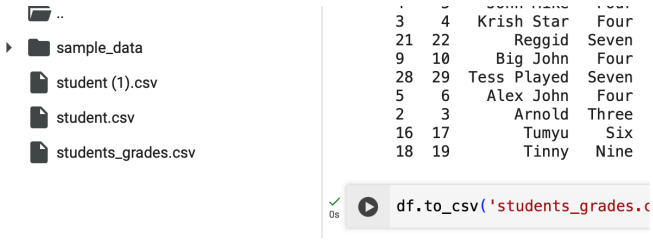
```
    id         name  class  mark  gender grade
32  33    Kenn Rein    Six    96  female     A
11  12        Recky    Six    94  female     A
31  32    Binn Rott  Seven    90  female     A
10  11       Ronald    Six    89  female     A
24  25     Giff Tow  Seven    88    male     A
15  16        Gimmy   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
29  30    Reppy Red    Six    79  female     B
25  26       Crelea  Seven    79    male     B
8    9      Tes Qry    Six    78     NaN     B
6    7  My John Rob  Fifth    78    male     B
23  24    Tiddy Now  Seven    78    male     B
0    1     John Deo   Four    75  female     B
17  18        Honny   Five    75    male     B
20  21   Babby John   Four    69  female     C
33  34     Gain Toe  Seven    69    male     C
19  20       Jackly   Nine    65  female     C
4    5    John Mike   Four    60  female     C
3    4   Krish Star   Four    60  female     C
21  22       Reggid  Seven    55  female     D
9   10     Big John   Four    55  female     D
28  29  Tess Played  Seven    55    male     D
```

Exercise 6: Exporting Data

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

Saving the new dataframe to a new CSV file.

```
df.to_csv('student_grades.csv', index=False)
```

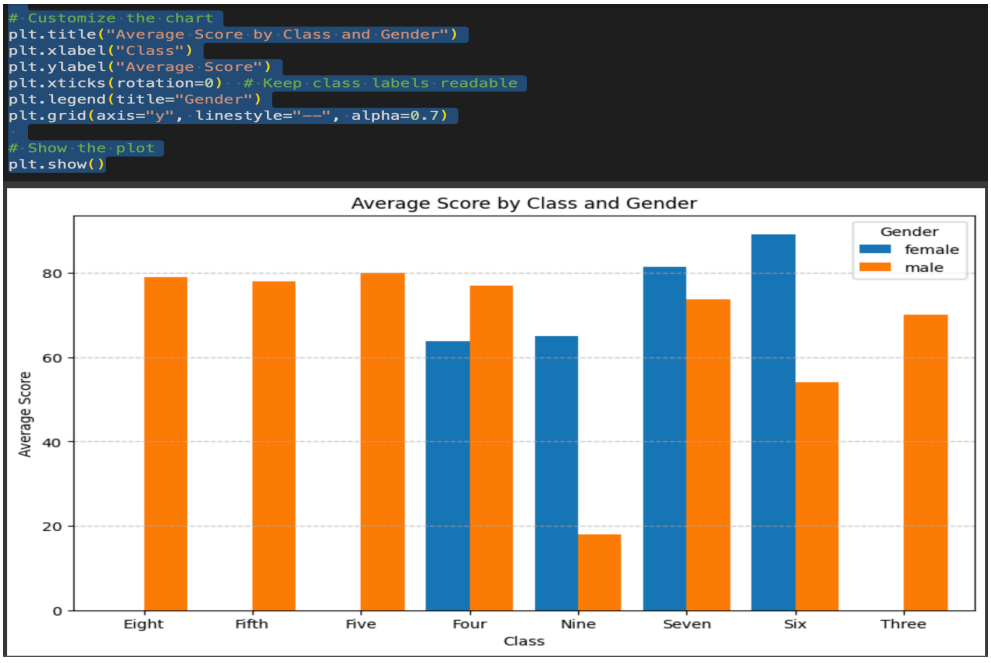



```
df.to_csv('students_grades.csv', index=False)
print(df)
```

```
    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
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
19  20       Jackly   Nine    65  female     C
20  21   Babby John   Four    69  female     C
21  22       Reggid  Seven    55  female     D
22  23        Herod  Eight    79    male     B
23  24    Tiddy Now  Seven    78    male     B
24  25     Giff Tow  Seven    88    male     A
25  26       Crelea  Seven    79    male     B
26  27          NaN  Three    81     NaN     B
27  28     Rojj Base  Seven    86  female     A
28  29  Tess Played  Seven    55    male     D
29  30   Reppy Red     Six    79  female     B
```

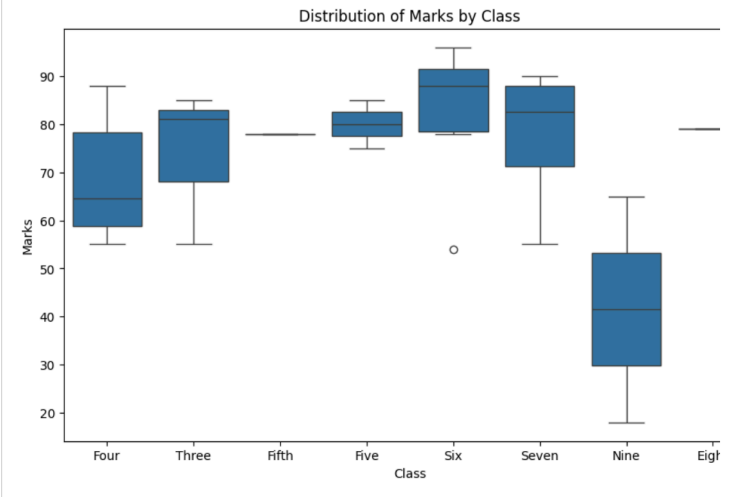
Exercise 7: If finished early try visualising the results

```
import matplotlib.pyplot as plt

# Plot the pivot table as a column chart (bar chart)
pivot.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()
```



Distribution of marks by class

```
#Average marks by class
plt.figure(figsize=(10, 6))
sns.boxplot(x='class', y='mark', data=df)
plt.title('Distribution of Marks by Class')
plt.xlabel('Class')
plt.ylabel('Marks')
plt.show()
```

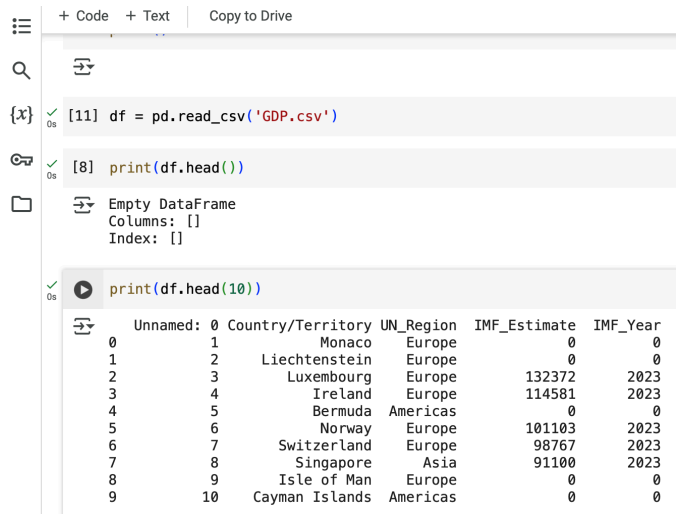


Day 4: Task 1

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.

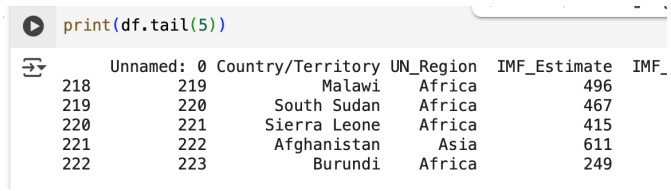
- Read and save the 'GDP (nominal) per Capita' data to a data frame called "df" in Jyputer notebook
- Print the first 10 rows
- Print the last 5 rows
- Print 'Country/Territory' and 'UN_Region' columns

1-reading the CSV file and printing the first 10 rows



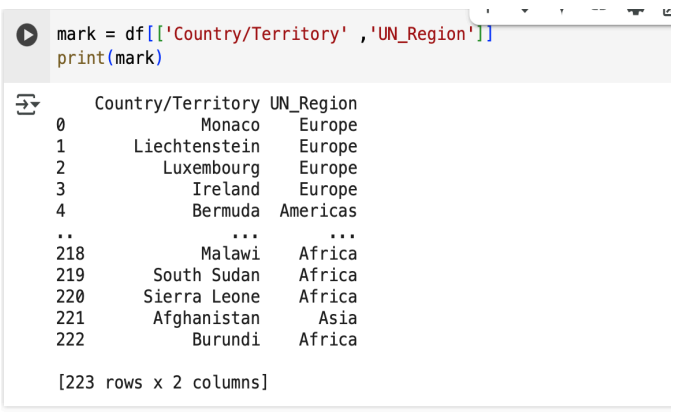
```
+ Code + Text   Copy to Drive

[11] df = pd.read_csv('GDP.csv')

[8] print(df.head())

Empty DataFrame
Columns: []
Index: []

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
```

2-Printing the last 5 rows



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

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

3-Printing the Country/Territory and UN region columns



```
mark = df[['Country/Territory' ,'UN_Region']]
print(mark)

    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]
```

Task1- Detecting null values in the DataFrame

```
import pandas as pd      #importing pandas and numpy library
import numpy as np
data={ "Name":["Alice","Bob",np.nan,"David"],
       "Age": [25,30,np.nan,22],
       "Score":[80,90,np.nan,75]}
df=pd.DataFrame(data)
print(df)
```

```
    Name   Age  Score
0  Alice  25.0   80.0
1    Bob  30.0   90.0
2    NaN   NaN    NaN
3  David  22.0   75.0
```

```
print(df.isnull()) #detecting the null values
```

```
     Name    Age  Score
0  False  False  False
1  False  False  False
2   True   True   True
3  False  False  False
```

2-Counting null values in each column

```
print(df.isnull().sum()) #counting null values in each column
```

```
Name     1
Age      1
Score    1
dtype: int64
```

3-Checking if any missing values exist

```
print(df.isnull().values.any()) # checking if any missing values
                                # exist
```

```
True
```

Task 2 Removing Null Values

1-Remove rows with any Null values

```
df_no_null_rows=df.dropna()  #removing rows with null values
print(df_no_null_rows)
```

```
    Name   Age  Score
0  Alice  25.0   80.0
3  David  22.0   75.0
```

2-Remove rows where all values are NaN

```
df_no_all_null=df.dropna(how="all") #remove rows where all
print(df_no_all_null)                #values are NaN
```

```
    Name   Age  Score
0  Alice  25.0   80.0
1    Bob   NaN   90.0
2    NaN  30.0    NaN
3  David  22.0   75.0
```

3-Remove a specific column if it contains any null values

```
df_no_null_columns=df.dropna(axis=1) # remove a specific colum if it
print(df_no_null_columns)             #a null value
```

```
Empty DataFrame
Columns: []
Index: [0, 1, 2, 3]
```

Task 3 filling Null Values

1-Replace NaN values with a fixed value

```
df_filled=df.fillna(0) # replacing the NaN values with o
print(df_filled)
```

```
    Name   Age  Score
0  Alice  25.0   80.0
1    Bob   0.0   90.0
2      0  30.0    0.0
3  David  22.0   75.0
```

2- Filling NaN values using forward fill

```
df_ffill=df.fillna(method="ffill")  #filling NaN values using
print(df_ffill)                      #forward
```

```
    Name   Age  Score
0  Alice  25.0   80.0
1    Bob  25.0   90.0
2    Bob  30.0   90.0
3  David  22.0   75.0
<ipython-input-39-93cce74016c7>:1: FutureWarning: DataFrame.filln
  df_ffill=df.fillna(method="ffill")  #filling NaN values using
```

3-Filling NaN values using backward fill

```
df_bfill=df.fillna(method="bfill") #filling NaN values using
print(df_bfill)                    #backward
```

```
    Name   Age  Score
0  Alice  25.0   80.0
1    Bob  30.0   90.0
2  David  30.0   75.0
3  David  22.0   75.0
<ipython-input-40-95c0604e2f09>:1: FutureWarning: DataFrame.fillna w
  df_bfill=df.fillna(method="bfill") #filling NaN values using
```

4- Filling NaN values with the column mean

```
#filling NaN values with the column mean
df["Age"].fillna(df["Age"].mean(),inplace=True)
df["Score"].fillna(df["Score"].mean(), inplace=True)
print(df)
```

```
     Name        Age      Score
0   Alice  25.000000  80.000000
1     Bob  25.666667  90.000000
2     NaN  30.000000  81.666667
3   David  22.000000  75.000000
<ipython-input-42-1a2ff946a4fd>:2: FutureWarning: A value is trying t
The behavior will change in pandas 3.0. This inplace method will neve

For example, when doing 'df[col].method(value, inplace=True)', try us


  df["Age"].fillna(df["Age"].mean(),inplace=True)
<ipython-input-42-1a2ff946a4fd>:3: FutureWarning: A value is trying t
The behavior will change in pandas 3.0. This inplace method will neve

For example, when doing 'df[col].method(value, inplace=True)', try us


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

Task 4 Replacing null values conditionally

1- Replacing NaN values in ‘Name’ column with ‘unknown’

```
#Replacing NaN values in 'Name" column with "Unknown"
df["Name"].replace(np.nan, "Unkwnown", inplace=True)
print(df)
```

```
       Name        Age      Score
0     Alice  25.000000  80.000000
1       Bob  25.666667  90.000000
2  Unkwnown  30.000000  81.666667
3     David  22.000000  75.000000
<ipython-input-43-4c7efb4cf2ff>:2: FutureWarning: A value is try
The behavior will change in pandas 3.0. This inplace method will

For example, when doing 'df[col].method(value, inplace=True)', t


  df["Name"].replace(np.nan, "Unkwnown", inplace=True)
```

2- Replace all NaN values in the DataFrame with ‘Missing’

```
df_filled_missing=df.where(pd.notnull(df),"Missing")
print(df_filled_missing)
```

```
       Name        Age      Score
0     Alice  25.000000  80.000000
1       Bob  25.666667  90.000000
2  Unkwnown  30.000000  81.666667
3     David  22.000000  75.000000
```

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.

1-Creating a data frame and printing it

```
df = pd.read_csv("GDP (nominal) per Capita.csv",encoding= 'unicod
print(df)
```

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

     WorldBank_Year  UN_Estimate UN_Year
1              2021       234317    2021
2              2020       169260    2021
3              2021       133745    2021
4              2021       101109    2021
5              2021       112653    2021
..              ...          ...     ...
219            2021          613    2021
220            2015          400    2021
221            2021          505    2021
222            2021          373    2021
223            2021          311    2021

[223 rows x 8 columns]
```

2-Calculating number of countries per region

```
# number of countries per region
data = {
    'Country/Territory': ['Monaco', 'Liechtenstein', 'Luxembourg', 'Ireland', 'Bermuda', 'Malawi', 'South Sudan', 'Sierra Leone', 'Afghanistan', 'Burundi'],
    'UN_Region': ['Europe', 'Europe', 'Europe', 'Europe', 'Americas', 'Africa', 'Africa', 'Africa', 'Asia', 'Africa'],
    'IMF_Estimate': [0, 0, 132372, 114581, 0, 496, 467, 415, 611, 249],
    'IMF_Year': [0, 0, 2023, 2023, 0, 2023, 2023, 2023, 2020, 2023],
    'WorldBank_Estimate': [234316, 157755, 133590, 100172, 114090, 635, 1072, 480, 369, 222],
    'WorldBank_Year': [2021, 2020, 2021, 2021, 2021, 2021, 2015, 2021, 2021, 2021],
    'UN_Estimate': [234317, 169260, 133745, 101109, 112653, 613, 400, 505, 373, 311],
    'UN_Year': [2021, 2021, 2021, 2021, 2021, 2021, 2021, 2021, 2021, 2021]
}

# Group by the 'UN_Region' column and count the number of countries per region
countries_per_region = df.groupby('UN_Region').size().reset_index(name='Number_of_Countries')
print(countries_per_region)
```

```
  UN_Region  Number_of_Countries
0    Africa                   55
1  Americas                   48
2      Asia                   51
3    Europe                   48
4   Oceania                   20
5     World                    1
```

3-Printing the European union countries

```
#What is European Union[n 1]?
# Filter for European countries
europe_df = df[df["UN_Region"] == "Europe"]
print(europe_df)
```

```
           Country/Territory UN_Region  IMF_Estimate  IMF_Year
1                     Monaco    Europe             0         0
2              Liechtenstein    Europe             0         0
3                 Luxembourg    Europe        132372      2023
4                    Ireland    Europe        114581      2023
6                     Norway    Europe        101103      2023
7                Switzerland    Europe         98767      2023
9                Isle of Man    Europe             0         0
13                   Iceland    Europe         75180      2023
14          Channel Islands    Europe             0         0
15            Faroe Islands    Europe             0         0
16                  Denmark    Europe         68827      2023
18              Netherlands    Europe         61098      2023
20                  Austria    Europe         56802      2023
22                   Sweden    Europe         55395      2023
23                  Finland    Europe         54351      2023
24                  Belgium    Europe         53377      2023
25               San Marino    Europe         52949      2023
28                  Germany    Europe         51383      2023
33           United Kingdom    Europe         46371      2023
34                   France    Europe         44408      2023
35                  Andorra    Europe         44387      2023
36                nion[n 1]    Europe         39940      2023
40                    Malta    Europe         36989      2023
```

WhatsApp

4-Countries in Europe below average

```
# Countries in Europe below avarege
europe_df = df[df["UN_Region"] == "Europe"]

# Calculate the average GDP per capita using WorldBank_Estimate
average_gdp = europe_df["WorldBank_Estimate"].mean()
below_average = europe_df[europe_df["WorldBank_Estimate"] < average_gdp]
print("Countries in Europe below average GDP per capita:")
print(below_average[["Country/Territory", "WorldBank_Estimate"]])
```

```
Countries in Europe below average GDP per capita:
          Country/Territory  WorldBank_Estimate
34                   France               43659
35                  Andorra               42137
36      European Union[n 1]               38411
40                    Malta               33487
41                    Italy               35658
51                 Slovenia               29291
52           Czech Republic               26821
53                    Spain               30104
54                  Estonia               27944
57                Lithuania               23723
59                 Portugal               24568
60                   Latvia               21148
62                 Slovakia               21392
63                   Greece               20193
70                  Croatia               17685
72                   Poland               18000
75                  Hungary               18728
78                  Romania               14858
87                 Bulgaria               12222
90                   Russia               12195
103             Montenegro                9466
106                 Serbia                9230
112  Bosnia and Herzegovina                7143
115                 Belarus                7302
118         North Macedonia                6695
```

5-The countries in Europe that has higher GDP than UK

```
## Which countries in Europe has higher GDP than UK?
europe_df = df[df['UN_Region'] == 'Europe']
# Find the GDP of the UK
uk_gdp = europe_df.loc[europe_df['Country/Territory'] == 'United Kingdom', 'IMF_Estimate'].values[0]
# Filter European countries with GDP higher than the UK
higher_gdp_countries = europe_df[europe_df['IMF_Estimate'] > uk_gdp]
print(higher_gdp_countries[['Country/Territory', 'IMF_Estimate']])
```

```
   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
```

6- Grouping by UN_Region by calculating the mean of GDP

```
grouped = df.groupby('UN_Region').agg({
    'IMF_Estimate': 'mean',
    'WorldBank_Estimate': 'mean',
    'UN_Estimate': 'mean'
}).reset_index()

print(grouped)
```

```
   UN_Region  IMF_Estimate  WorldBank_Estimate   UN_Estimate
0     Africa   2802.345455         2470.836364   2417.927273
1   Americas  11871.041667        18565.125000  18703.750000
2       Asia  16665.254902        13921.313725  14069.019608
3     Europe  34446.750000        45193.687500  40610.791667
4    Oceania   9133.150000        15113.650000  12613.750000
5      World  13440.000000        12235.000000  12230.000000
```

7-Grouping by UN_Region and getting the max and min GDP

```
# Group by 'UN_Region' and get the max and min
grouped_stats = df.groupby('UN_Region').agg({
    'IMF_Estimate': ['max', 'min'],
    'WorldBank_Estimate': ['max', 'min'],
    'UN_Estimate': ['max', 'min']
}).reset_index()

print(grouped_stats)
```

```
   UN_Region IMF_Estimate        WorldBank_Estimate        UN_Estimate
                      max    min                max    min         max    min
0     Africa         19536      0              14653      0       12085    311
1   Americas         80034      0             114090      0      112653      0
2       Asia         91100      0              72794      0       66822      0
3     Europe        132372      0             234316   4836      234317      0
4    Oceania         64964      0              60443      0       66916      0
5      World         13440  13440              12235  12235       12230  12230
```

8- Group by UN_Region and count the number of countries

```
# Group by 'UN_Region' and count the number of countries
country_count = df.groupby('UN_Region').size().reset_index(name='Country_Count')

print(country_count)
```

```
   UN_Region  Country_Count
0     Africa             55
1   Americas             48
2       Asia             51
3     Europe             48
4    Oceania             20
5      World              1
```

9-Countries below average IMF world estimates

Which countries below average by IMF world estimate?

```
filtered_df = df[df['IMF_Estimate'] > 0]
average_imf_estimate = filtered_df['IMF_Estimate'].mean()
# countries below the average IMF world estimates
below_average_countries = filtered_df[filtered_df['IMF_Estimate'] < average_imf_estimate]
print("Average IMF Estimate:", average_imf_estimate)
print(below_average_countries[['Country/Territory', 'IMF_Estimate']])
```

```
Average IMF Estimate: 17377.736040609136
    Country/Territory  IMF_Estimate
84             Panama         17350
87           Bulgaria         14893
88              Palau         14804
89         Costa Rica         14733
90             Russia         14403
..                ...           ...
219            Malawi           496
220       South Sudan           467
221      Sierra Leone           415
222       Afghanistan           611
223           Burundi           249

[134 rows x 2 columns]
```

10-IMF estimate 0 values

```
# Replace IMF_Estimate values of 0 with  UN_Estimate
df['IMF_Estimate'] = df.apply(lambda row: row['UN_Estimate'] if row['IMF_Estimate'] == 0 else row['IMF_Estimate'], axis=1)
print(df)
```

```
    Country/Territory UN_Region  IMF_Estimate  IMF_Year  WorldBank_Estimate  \
1              Monaco    Europe        234316         0              234316
2       Liechtenstein    Europe        157755         0              157755
3          Luxembourg    Europe        132372      2023              133590
4             Ireland    Europe        114581      2023              100172
5             Bermuda  Americas        114090         0              114090
..                ...       ...           ...       ...                 ...
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
3              2021       133745    2021
4              2021       101109    2021
5              2021       112653    2021
..              ...          ...     ...
219            2021          613    2021
220            2015          400    2021
221            2021          505    2021
222            2021          373    2021
223            2021          311    2021

[223 rows x 8 columns]
```

11- Countries with the highest UN estimate

```
max_un_estimate_row = df.loc[df['UN_Estimate'].idxmax()] # looking for the row with the maximum UN_Estimate value
country_with_max_un_estimate = max_un_estimate_row['Country/Territory']  # Geting thge name of the country  and the UN Estimate
max_un_estimate_value = max_un_estimate_row['UN_Estimate']
print(f"The country with the highest UN Estimate is {country_with_max_un_estimate} with an estimate of {max_un_estimate_value}.")
```

```
The country with the highest UN Estimate is Monaco with an estimate of 234317.
```

12- The country that has the highest world bank estimate

```
# Findding  the country with the highest WorldBank Estimate
max_worldbank_estimate = df['WorldBank_Estimate'].max()
country_with_max_estimate = df.loc[df['WorldBank_Estimate'].idxmax(), 'Country/Territory']
print(f"The country with the highest WorldBank Estimate is: {country_with_max_estimate}")
print(f"WorldBank Estimate: {max_worldbank_estimate}")
```

```
The country with the highest WorldBank Estimate is: Monaco
WorldBank Estimate: 234316
```

13- The country that has the highest IMF estimates.

```
# Find the country  with the maximum IMF_Estimate
max_imf_estimate = df['IMF_Estimate'].max()
country_with_max_imf = df[df['IMF_Estimate'] == max_imf_estimate]['Country/Territory'].iloc[0]
print(f"The country with the highest IMF Estimate is {country_with_max_imf} with an estimate of {max_imf_estimate}.")
```

```
The country with the highest IMF Estimate is Monaco with an estimate of 234317.0.
```

14- Filling 0 values by average

```
# replace 0 with null values
df.replace(0, np.nan, inplace=True)
print(df)
```

```
  Country/Territory UN_Region  IMF_Estimate  IMF_Year  WorldBank_Estimate  \
0            Monaco    Europe           NaN       NaN              234316
1     Liechtenstein    Europe           NaN       NaN              157755
2        Luxembourg    Europe      132372.0    2023.0              133590
3           Ireland    Europe      114581.0    2023.0              100172
4           Bermuda  Americas           NaN       NaN              114090
5            Malawi    Africa         496.0    2023.0                 635
6       South Sudan    Africa         467.0    2023.0                1072
7      Sierra Leone    Africa         415.0    2023.0                 480
8       Afghanistan      Asia         611.0    2020.0                 369
9           Burundi    Africa         249.0    2023.0                 222

   WorldBank_Year  UN_Estimate  UN_Year
0            2021       234317     2021
1            2020       169260     2021
2            2021       133745     2021
3            2021       101109     2021
4            2021       112653     2021
5            2021          613     2021
6            2015          400     2021
7            2021          505     2021
8            2021          373     2021
9            2021          311     2021
```

15- Calculating the average of ‘Worldbank_Estimate ‘and ‘UN_Estimate’ columns

```
# Calculate the average of 'Worldbank_Estimate' and 'UN_Estimate' columns
worldbank_average = df['WorldBank_Estimate'].mean()# Calculate the average of 'Worldbank_Estimate'
un_average = df['UN_Estimate'].mean()    # Calculate the average of 'UN_Estimate'
print(f"Average of 'WorldBank_Estimate': {worldbank_average}")
print(f"Average of 'UN_Estimate': {un_average}")
```

```
Average of 'WorldBank_Estimate': 74270.1
Average of 'UN_Estimate': 75328.6
```

16- Filling the null values in ‘imf’ column with the calculated average

```
# Fill the null values in 'imf' column with the calculated average
# Calculate the average of non-zero values in the IMF_Estimate column
non_zero_values = df[df['IMF_Estimate'] != 0]['IMF_Estimate']
average_imf_estimate = non_zero_values.mean()
df['IMF_Estimate'] = df['IMF_Estimate'].replace(0, average_imf_estimate)
print(df)
```

```
  Country/Territory UN_Region  IMF_Estimate  IMF_Year  WorldBank_Estimate  \
0            Monaco    Europe           NaN       NaN              234316
1     Liechtenstein    Europe           NaN       NaN              157755
2        Luxembourg    Europe      132372.0    2023.0              133590
3           Ireland    Europe      114581.0    2023.0              100172
4           Bermuda  Americas           NaN       NaN              114090
5            Malawi    Africa         496.0    2023.0                 635
6       South Sudan    Africa         467.0    2023.0                1072
7      Sierra Leone    Africa         415.0    2023.0                 480
8       Afghanistan      Asia         611.0    2020.0                 369
9           Burundi    Africa         249.0    2023.0                 222

   WorldBank_Year  UN_Estimate  UN_Year
0            2021       234317     2021
1            2020       169260     2021
2            2021       133745     2021
3            2021       101109     2021
4            2021       112653     2021
5            2021          613     2021
6            2015          400     2021
7            2021          505     2021
8            2021          373     2021
9            2021          311     2021
```

17-Dropping the temporary ‘avg_woldbank_un’ column if not neede

```
# Drop the temporary 'avg_worldbank_un' column if not needed
if 'avg_worldbank_un' in df.columns:
    df = df.drop(columns=['avg_worldbank_un'])
print(df) #printing the data frame after dropping the column
```

```
  Country/Territory UN_Region  IMF_Estimate  IMF_Year  WorldBank_Estimate  \
0            Monaco    Europe           NaN       NaN              234316
1     Liechtenstein    Europe           NaN       NaN              157755
2        Luxembourg    Europe      132372.0    2023.0              133590
3           Ireland    Europe      114581.0    2023.0              100172
4           Bermuda  Americas           NaN       NaN              114090
5            Malawi    Africa         496.0    2023.0                 635
6       South Sudan    Africa         467.0    2023.0                1072
7      Sierra Leone    Africa         415.0    2023.0                 480
8       Afghanistan      Asia         611.0    2020.0                 369
9           Burundi    Africa         249.0    2023.0                 222

   WorldBank_Year  UN_Estimate  UN_Year
0            2021       234317     2021
1            2020       169260     2021
2            2021       133745     2021
3            2021       101109     2021
4            2021       112653     2021
5            2021          613     2021
6            2015          400     2021
7            2021          505     2021
8            2021          373     2021
9            2021          311     2021
```

18-Checking missing values

Checking Missing Values

```
# Check for missing values in each column
missing_values = df.isnull().sum()
print(missing_values)
```

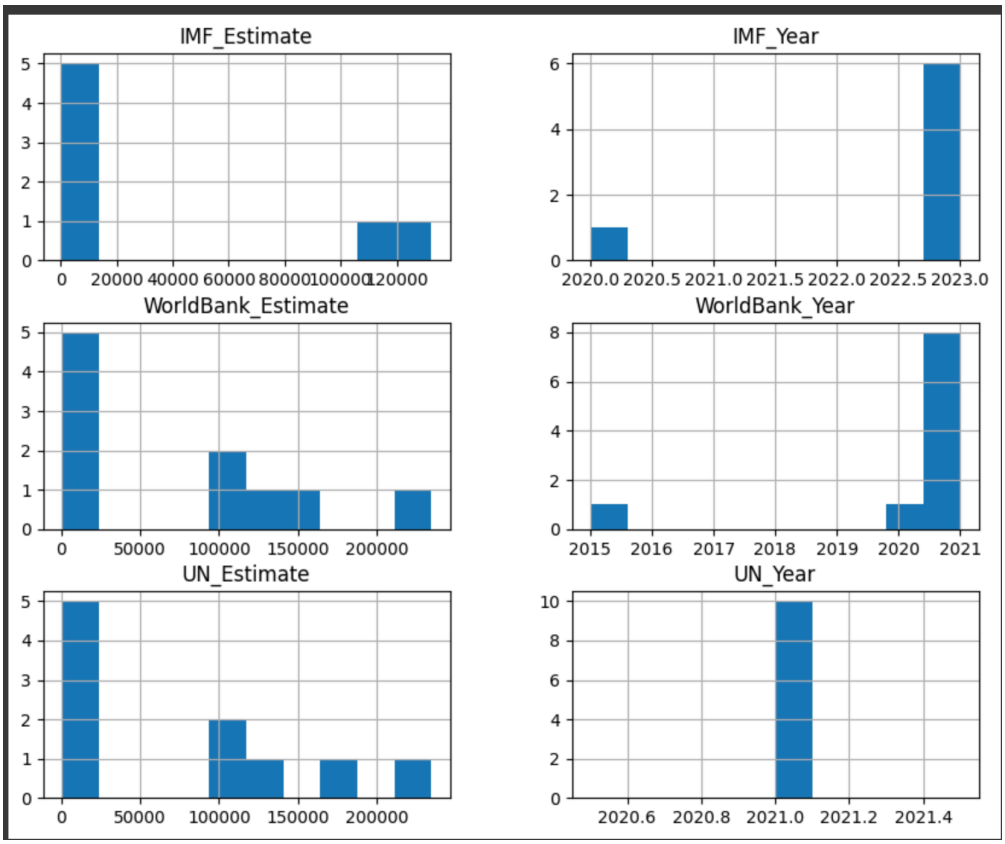
```
Country/Territory     0
UN_Region             0
IMF_Estimate          3
IMF_Year              3
WorldBank_Estimate    0
WorldBank_Year        0
UN_Estimate           0
UN_Year               0
dtype: int64
```

19-Visualisation

*Histogram

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

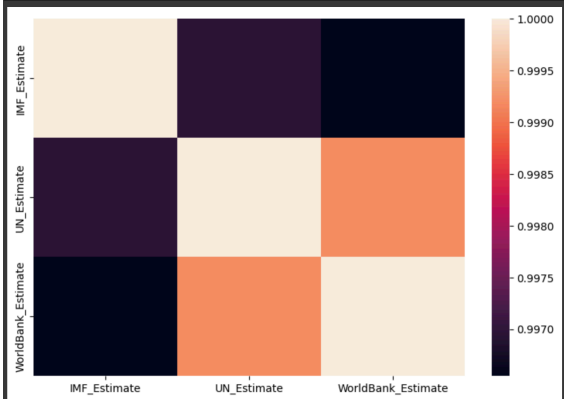
-The histogram (IMF_Estimate) shows the distribution of IMF across countries/territories.
-The histogram (IMF_year) displays the years associated with IMF estimated.
-The histogram (WorldBank_Estimate) shows the distribution of world bank data across countries/territories.
-The histogram (WorldBank_Year) shows the years associated with world bank estimates.
-The histogram (UN_Estimates) displays the distribution of UN estimates.
-The histogram (UN_Year) shows the years associated with UN estimated.



*Correlation Heatmap

The correlation heatmap represents the relationship between the estimated (IMF ,and World bank)

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

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

plt.show()
```



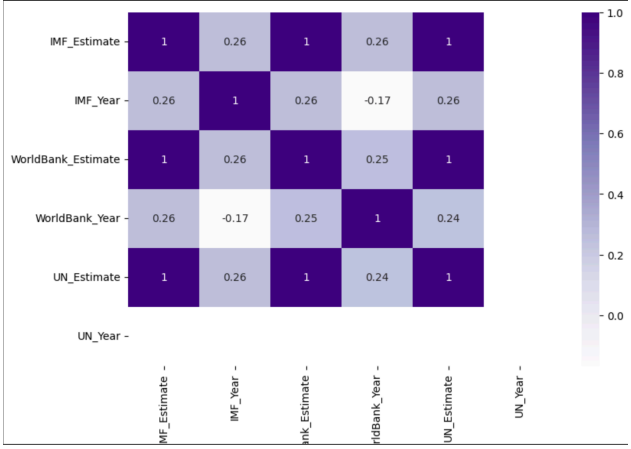
This heatmap shows three types of correlation:

-Positive correlation between IMF_Estimate and WorldBank_estimate when the number is approximately equal to one.

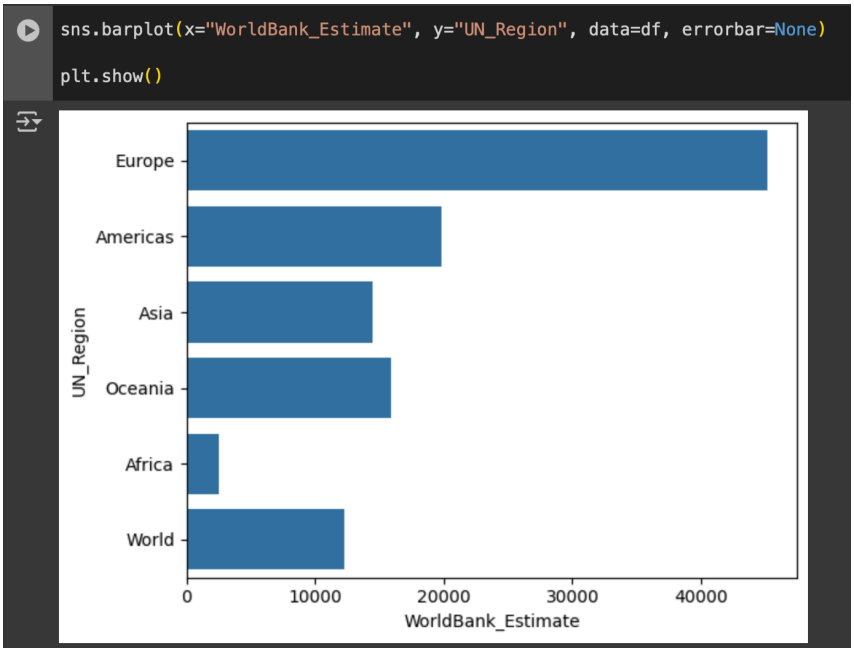
-No correlation between IMF_Year and UN_year when the number is approximately equal to zero.

-Negative correlation between UN_Estimates and WorldBank_Estimate when the number is less than zero(negative).

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

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

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

plt.show()
```



*The horizontal Bar Plot shows that Europe has the tallest bar, that indicates the average world bank estimate for European countries is higher compared to other countries.



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.