Just IT

B2W group

Apprenticeships | Training | Recruitment

Data Technician

Name:

Course Date:

Table of contents

Day 2: Task 1 3

Day 3: Task 1 5

- Exercise 1: Loading and Exploring the Data 5
- Exercise 2: Indexing and Slicing 7
- Exercise 3: Data Manipulation 9
- Exercise 4: Aggregation and Grouping 10
- Exercise 5: Advanced Operations 11
- Exercise 6: Exporting Data 13
- Exercise 7: If finished early try visualising the results 14

Day 4: Task 1 19

Day 4: Task 2 21

Course Notes 29

Additional Information 29

Day 2: Task 1

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

FizzBuzz:

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

Paste your completed work to the right

```
for i in range(1,101):
  if i % 5 == 0 and i % 3 == 0:
    print(f"{i} - fizzbuzz")
  elif i % 3 == 0:
    print(f"{i} - fizz")
  elif i % 5 == 0:
    print (f"{i} - buzz")
  else:
    print(i)
```

```python
for i in range(1,101):
  if i % 5 == 0 and i % 3 == 0:
    print(f"{i} - fizzbuzz")
  elif i % 3 == 0:
    print(f"{i} - fizz")
  elif i % 5 == 0:
    print (f"{i} - buzz")
  else:
    print(i)
```

```
1
2
3 - fizz
4
5 - buzz
6 - fizz
7
8
9 - fizz
10 - buzz
11
12 - fizz
13
14
15 - fizzbuzz
16
17
18 - fizz
19
20 - buzz
21 - fizz
```

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

# 1. Read the file into a table called df
df = pd.read_csv('student.csv')

# 2. Peek at the first 5 rows to check your data
print("----------2. The first 5 rows--------")
print(df.head())

# 3. Print info (counts & data types)
print("\n----------3. Info (counts & data types)--------")
df.info()

# 4. Print summary statistics
print("\n----------4. Summary statistics:--------")
df.describe()
```

```python
import pandas as pd

# 1. Read the file into a table called df
df = pd.read_csv('student.csv')

# 2. Peek at the first 5 rows to check your data
print("----------2. The first 5 rows--------")
print(df.head())

# 3. Print info (counts & data types)
print("\n----------3. Info (counts & data types)--------")
df.info()

# 4. Print summary statistics
print("\n----------4. Summary statistics:--------")
df.describe()
```

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

----------3. Info (counts & data types)--------
<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)
```

```
----------4. Summary statistics:--------
```

	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'."

```
import pandas as pd

# Read the file into a table called df
df = pd.read_csv('student.csv')

# 1. Select the 'name' column
print("----------1. Select the 'name' column--------")
col_name = df['name'].head(10)
print("\n Name \n", col_name)

# 2. Select the 'name' and 'mark' columns.
print("\n----------2. Select the 'name' and 'mark' columns.------
--")
col_name_and_mark = df[['name', 'mark']].head(10)
print(col_name_and_mark)
```

```python
# 3. Select the first 3 rows.
print("\n----------3. First 3 rows --------")
print(df.head(3))

#4. The 'class' is 'Four'.
print("\n----------4. The 'class' is 'Four' --------")
class_four = df[df['class'] == 'Four']
print(class_four)
```

```python
# 1. Select the 'name' column
print("----------1. Select the 'name' column--------")
col_name = df['name'].head(10)
print("\n Name \n", col_name)

# 2. Select the 'name' and 'mark' columns.
print("\n----------2. Select the 'name' and 'mark' columns.--------")
col_name_and_mark = df[['name', 'mark']].head(10)
print(col_name_and_mark)

# 3. Select the first 3 rows.
print("\n----------3. First 3 rows --------")
print(df.head(3))

#4. The 'class' is 'Four'.
print("\n----------4. The 'class' is 'Four' --------")
class_four = df[df['class'] == 'Four']
print(class_four)
```

```
----------1. Select the 'name' column--------

 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
Name: name, dtype: object

----------2. Select the 'name' and 'mark' columns.--------
          name  mark
0     John Deo    75
```

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."

```
import pandas as pd

# Read the file into a table called df
df = pd.read_csv('student.csv')

# 1.  Question: "Write the code to add a new column 'passed' that
indicates whether the student passed (mark >= 60)."
df_student_greater_60 = df.copy()
df_student_greater_60['passed'] = df_student_greater_60['mark']
>= 60
print("\n New column 'passed' that indicates whether the student
passed \n", df_student_greater_60.head(10))

# 2.  Question: "Write the code to rename the 'mark' column to
'score'."
print("\nColumns before renaming:",
df_student_greater_60.columns)
df_renamed = df_student_greater_60.copy()
df_renamed.rename(columns={'mark': 'score'}, inplace=True)
print("Columns after renaming:", df_renamed.columns)

# 3.  Question: "Write the code to drop the 'passed' column."
df_dropped = df_renamed.copy()
df_dropped = df_dropped.drop(columns=['passed'])
print("Columns after dropping:", df_dropped.columns)
```

```python
import pandas as pd

# Read the file into a table called df
df = pd.read_csv('student.csv')

# 1.  Question: "Write the code to add a new column 'passed' that indicates whether the student passed (mark >= 60)."
df_student_greater_60 = df.copy()
df_student_greater_60['passed'] = df_student_greater_60['mark'] >= 60
print("\n New column 'passed' that indicates whether the student passed \n", df_student_greater_60.head(10))

# 2.  Question: "Write the code to rename the 'mark' column to 'score'."
print("\nColumns before renaming:", df_student_greater_60.columns)
df_renamed = df_student_greater_60.copy()
df_renamed.rename(columns={'mark': 'score'}, inplace=True)
print("Columns after renaming:", df_renamed.columns)

# 3.  Question: "Write the code to drop the 'passed' column."
df_dropped = df_renamed.copy()
df_dropped = df_dropped.drop(columns=['passed'])
print("Columns after dropping:", df_dropped.columns)
```

```
 New column 'passed' that indicates whether the student passed
    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

Columns before renaming: Index(['id', 'name', 'class', 'mark', 'gender', 'passed'], dtype='object')
Columns after renaming: Index(['id', 'name', 'class', 'score', 'gender', 'passed'], dtype='object')
Columns after dropping: Index(['id', 'name', 'class', 'score', 'gender'], dtype='object')
```

Exercise 4: Aggregation and Grouping

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

```python
import pandas as pd

# Read the file into a table called df
df = pd.read_csv('student.csv')

# 1.  Group the DataFrame by the 'class' column and calculate the mean 'mark' for each group."
print("\nAverage mark per class:")
print(df.groupby('class')['mark'].mean())

# 2.  Count the number of students in each class."
print("\nNumber of students per class:")
```

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

# 3.  Average mark for each gender.
print("\nAverage mark per gender:")
print(df.groupby('gender')['mark'].mean())
```

```
import pandas as pd

# Read the file into a table called df
df = pd.read_csv('student.csv')

# 1.  Group the DataFrame by the 'class' column and calculate the mean 'mark' for each group."
print("\nAverage mark per class:")
print(df.groupby('class')['mark'].mean())

# 2.  Count the number of students in each class."
print("\nNumber of students per class:")
print(df['class'].value_counts())

# 3.  Average mark for each gender.
print("\nAverage mark per gender:")
print(df.groupby('gender')['mark'].mean())
```

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

Number of students per class:
class
Seven    10
Four      8
Six       7
Three     3
Nine      2
```

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."

```
import pandas as pd
```

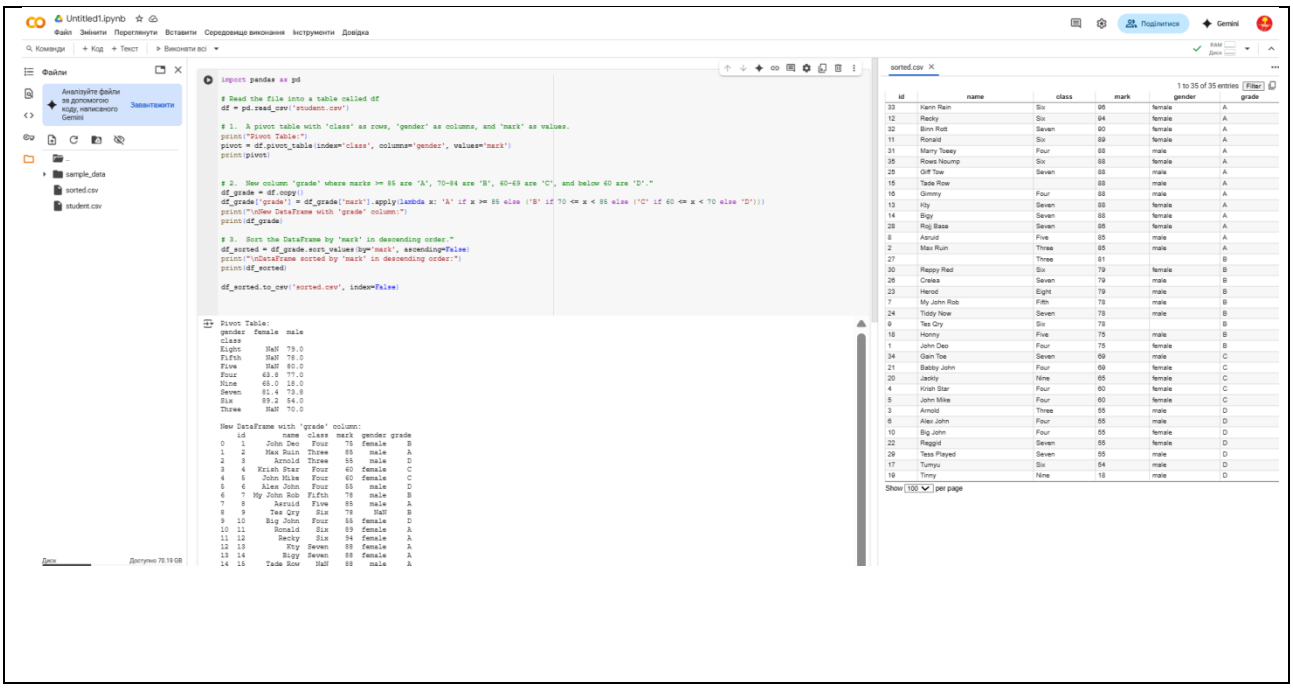
```
# Read the file into a table called df
df = pd.read_csv('student.csv')

# 1.  A pivot table with 'class' as rows, 'gender' as
columns, and 'mark' as values.
print("Pivot Table:")
pivot = df.pivot_table(index='class', columns='gender',
values='mark')
print(pivot)


# 2.  New column 'grade' where marks >= 85 are 'A', 70-
84 are 'B', 60-69 are 'C', and below 60 are 'D'."
df_grade = df.copy()
df_grade['grade'] = df_grade['mark'].apply(lambda x: 'A'
if x >= 85 else ('B' if 70 <= x < 85 else ('C' if 60 <=
x < 70 else 'D')))
print("\nNew DataFrame with 'grade' column:")
print(df_grade)

# 3.  Question: "Write the code to sort the DataFrame by
'mark' in descending order."
df_sorted = df_grade.sort_values(by='mark',
ascending=False)
print("\nDataFrame sorted by 'mark' in descending
order:")
print(df_sorted)
```

```python
import pandas as pd

# Read the file into a table called df
df = pd.read_csv('student.csv')

# 1. A pivot table with 'class' as rows, 'gender' as columns, and 'mark' as values.
print("Pivot Table:")
pivot = df.pivot_table(index='class', columns='gender', values='mark')
print(pivot)


# 2. New column 'grade' where marks >= 85 are 'A', 70-84 are 'B', 60-69 are 'C', and below 60 are 'D'."
df_grade = df.copy()
df_grade['grade'] = df_grade['mark'].apply(lambda x: 'A' if x >= 85 else ('B' if 70 <= x < 85 else ('C' if 60 <= x < 70 else 'D')))
print("\nNew DataFrame with 'grade' column:")
print(df_grade)

# 3. Question: "Write the code to sort the DataFrame by 'mark' in descending order."
df_sorted = df_grade.sort_values(by='mark', ascending=False)
print("\nDataFrame sorted by 'mark' in descending order:")
print(df_sorted)
```

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

New DataFrame with 'grade' column:
    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
```

Exercise 6: Exporting Data

1. Question: "Write the code to save the DataFrame with the new 'grade' column to a new CSV file."
2.



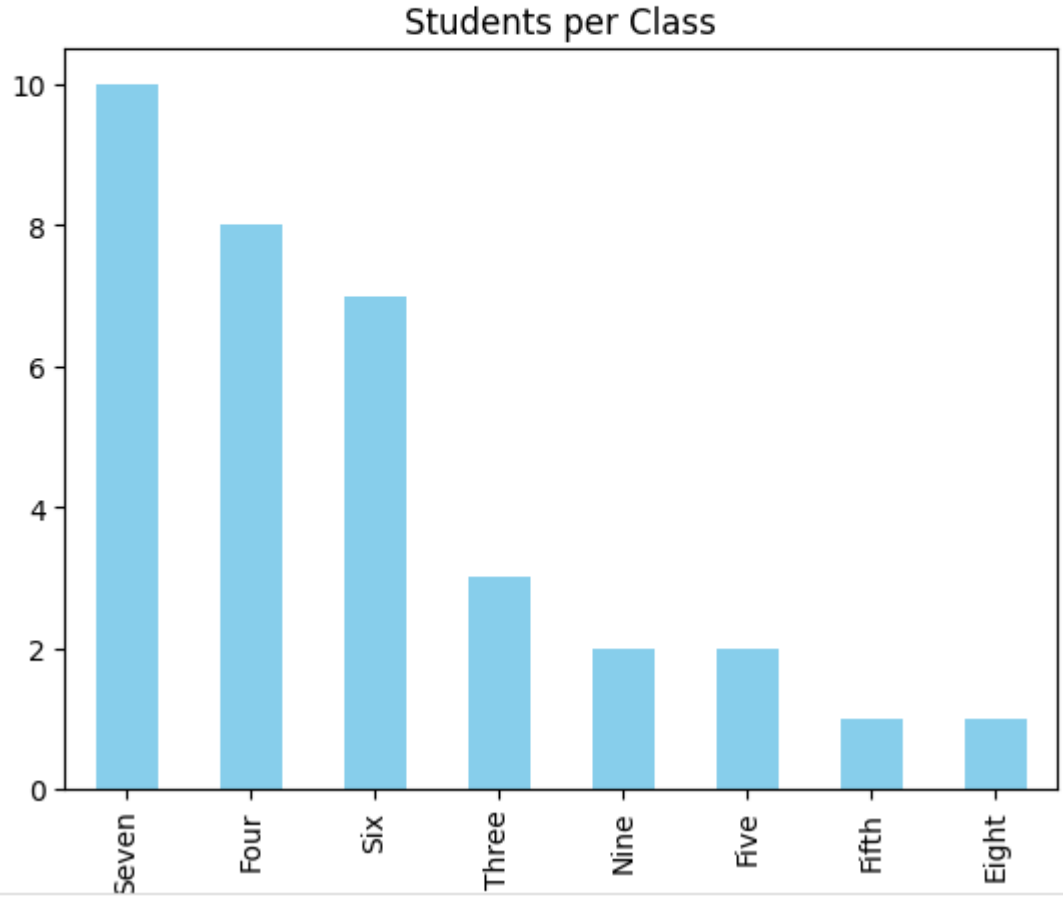
Exercise 7: If finished early try visualising the results

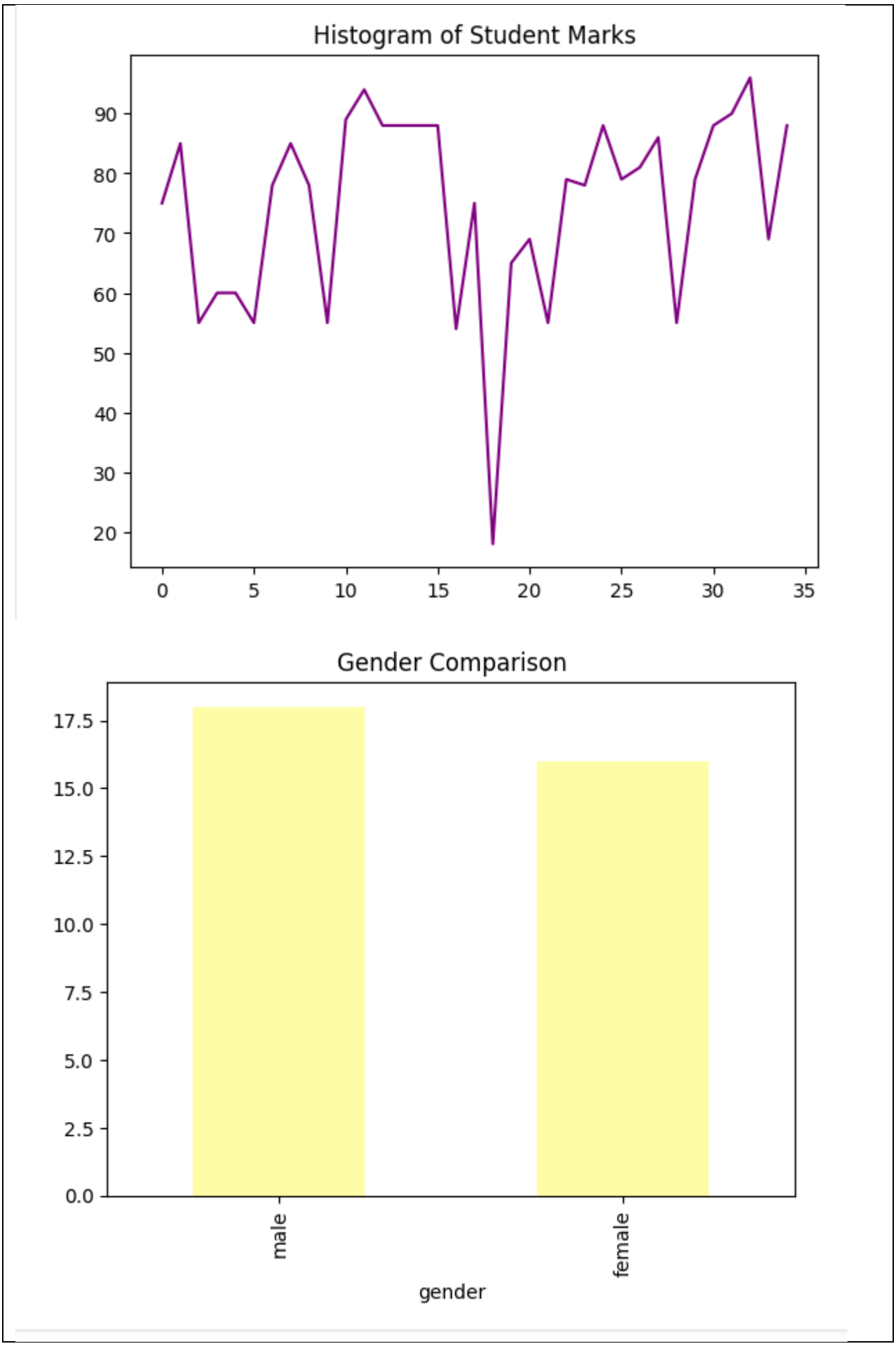
```
import pandas as pd
import matplotlib.pyplot as plt

# 1. Read the file into a table called df
df = pd.read_csv('student.csv')

# --------------------------------
# Step 1: View the original shape and check for missing values
# --------------------------------
print("Original data shape (rows, columns):")
print(df.shape)

print("\nMissing values in each column:")
print(df.isnull().sum())

# --------------------------------
# Step 2: Remove rows with missing data
# --------------------------------
```

```
df_no_missing = df.copy()
df_no_missing = df_no_missing.dropna()

print("\nData shape after dropping rows with missing
values:")
print(df_no_missing.shape)
print("Missing values after dropping:")
print(df_no_missing.isnull().sum())

# --------------------------------
# Step 3: Fill missing values
# --------------------------------
df_filled = df.copy()

# Convert 'mark' to numeric, turn errors into NaN
df_filled['mark'] = pd.to_numeric(df_filled['mark'],
errors='coerce')

# Fill missing marks with the average (mean) value
mean_mark = df_filled['mark'].mean()
df_filled['mark'] = df_filled['mark'].fillna(mean_mark)
print("\nMissing values Mark after filling:")


# Fill missing gender values with most_common_gender
most_common_gender = df_filled['gender'].mode()[0]
df_filled['gender'] =
df_filled['gender'].fillna(most_common_gender)

# Fill missing Name values with 'noName'
df_filled['name'] = df_filled['name'].fillna('noName')

# Fill missing Class values with 'noClass'
df_filled = df_filled.dropna(subset=['class'])


# --------------------------------
# Step 4: Remove duplicate rows
```

```
# --------------------------------
df_no_duplicates = df_filled.copy()

# Drop duplicate rows
df_no_duplicates = df_no_duplicates.drop_duplicates()

print("\nShape after removing duplicate rows:")
print(df_no_duplicates.shape)
print(df_no_duplicates.head())

# --------------------------------
# Step 5: Visualizing the results
# --------------------------------
# How many students are there in each 'class'?
plt.figure()
df_no_duplicates['class'].value_counts().plot(kind='bar'
, title='Students per Class', color='skyblue')
plt.show()

# Line plot of student marks
plt.figure()
pd.to_numeric(df_no_duplicates['mark'],
errors='coerce').plot(title='Histogram of Student
Marks', color='purple')
plt.show()

# The numbers of 'male' and 'female' students
plt.figure()
df_no_duplicates['gender'].value_counts().plot(kind='bar
', title='Gender Comparison', color='#FFFCA8',)
plt.show()
```

```
Shape after removing duplicate rows:
(34, 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
```




Histogram of Student Marks
90
80
70
60
50
40
30
20
0
5
10
15
20
25
30
35
Gender Comparison
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
male
female
gender

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

```
import pandas as pd
# Read and save the 'GDP (nominal) per Capita' data to a
data frame called "df" in Jyputer notebook
df = pd.read_csv('GDP (nominal) per Capita.csv')
print(df.info())

print("==============================")
print("The first 10 rows")
print("==============================")
print(df.head(10))

print("\n==============================")
print("The last 5 rows:")
print("==============================")
print(df.tail(5))

print("\n==============================")
print("'Country/Territory' and 'UN_Region' columns:")
print("==============================")
print(df[['Country/Territory', 'UN_Region']])
```

```python
import pandas as pd
# Read and save the 'GDP (nominal) per Capita' data to a data frame called "df" in Jyputer notebook
df = pd.read_csv('GDP (nominal) per Capita.csv')
print(df.info())

print("============================")
print("The first 10 rows")
print("============================")
print(df.head(10))

print("\n============================")
print("The last 5 rows:")
print("============================")
print(df.tail(5))

print("\n============================")
print("'Country/Territory' and 'UN_Region' columns:")
print("============================")
print(df[['Country/Territory', 'UN_Region']])
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 223 entries, 0 to 222
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Unnamed: 0          223 non-null    int64
 1   Country/Territory   223 non-null    object
 2   UN_Region           223 non-null    object
 3   IMF_Estimate        223 non-null    int64
 4   IMF_Year            223 non-null    int64
 5   WorldBank_Estimate  223 non-null    int64
 6   WorldBank_Year      223 non-null    int64
 7   UN_Estimate         223 non-null    int64
 8   UN_Year             223 non-null    object
dtypes: int64(6), object(3)
memory usage: 15.8+ KB
None
============================
The first 10 rows
============================
   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
```

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.

EDA (Exploratory Data Analysis)

Use this section to explore and inspect dataset.

Додати блок цитати

```
[3] df.shape
```

```
(223, 8)
```

```
[4] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 223 entries, 1 to 223
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Country/Territory   223 non-null    object
 1   UN_Region           223 non-null    object
 2   IMF_Estimate        223 non-null    int64
 3   IMF_Year            223 non-null    int64
 4   WorldBank_Estimate  223 non-null    int64
 5   WorldBank_Year      223 non-null    int64
 6   UN_Estimate         223 non-null    int64
 7   UN_Year             223 non-null    object
dtypes: int64(5), object(3)
memory usage: 15.7+ KB
```

```
df.describe()
```

	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate
count	223.000000	223.000000	223.000000	223.000000	223.000000
mean	15351.632287	1787.098655	18927.417040	1957.278027	17767.304933
std	22550.899445	650.695912	29103.564915	353.145867	28698.104167
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1406.500000	2023.000000	2273.500000	2021.000000	2039.000000
50%	5421.000000	2023.000000	6805.000000	2021.000000	6396.000000
75%	19697.000000	2023.000000	23715.000000	2021.000000	20740.000000
max	132372.000000	2023.000000	234316.000000	2021.000000	234317.000000

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

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year	
1	Monaco	Europe	17377.736041	NaN	234316.0	2021.0	234317.0	2021	-216
2	Liechtenstein	Europe	17377.736041	NaN	157755.0	2020.0	169260.0	2021	-140
3	Luxembourg	Europe	132372.000000	2023.0	133590.0	2021.0	133745.0	2021	-1
4	Ireland	Europe	114581.000000	2023.0	100172.0	2021.0	101109.0	2021	14
5	Bermuda	Americas	17377.736041	NaN	114090.0	2021.0	112653.0	2021	-96

Подальші дії: Створити код зі змінною df | Переглянути рекомендовані графіки | New interactive sheet

```
df.tail()
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year	IM
219	Malawi	Africa	496.0	2023.0	635.0	2021.0	613.0	2021	
220	South Sudan	Africa	467.0	2023.0	1072.0	2015.0	400.0	2021	
221	Sierra Leone	Africa	415.0	2023.0	480.0	2021.0	505.0	2021	
222	Afghanistan	Asia	611.0	2020.0	369.0	2021.0	373.0	2021	
223	Burundi	Africa	249.0	2023.0	222.0	2021.0	311.0	2021	

```
[12] print(df.isnull().sum())
```

```
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
```

```
[112] df_no_missing = df.copy()
      df_no_missing = df_no_missing.dropna()
```

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

```
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
```

```
[112] df_no_missing = df.copy()
      df_no_missing = df_no_missing.dropna()
```

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

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

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

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

```
europe = df_no_missing[df_no_missing['UN_Region'] == 'Europe']
print(europe.head())
```

```
   Country/Territory UN_Region  IMF_Estimate  IMF_Year  WorldBank_Estimate  \
3         Luxembourg    Europe      132372.0    2023.0            133590.0
4            Ireland    Europe      114581.0    2023.0            100172.0
6             Norway    Europe      101103.0    2023.0             89154.0
7        Switzerland    Europe       98767.0    2023.0             91992.0
13           Iceland    Europe       75180.0    2023.0             68728.0

    WorldBank_Year  UN_Estimate UN_Year  IMF_vs_WB
3           2021.0     133745.0    2021    -1218.0
4           2021.0     101109.0    2021    14409.0
6           2021.0      89242.0    2021    11949.0
7           2021.0      93525.0    2021     6775.0
13          2021.0      69133.0    2021     6452.0
```

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

```
[27] average_imf = df_no_missing['IMF_Estimate'].mean()
     below_average = europe[europe['IMF_Estimate'] < average_imf]
     below_average
```

	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
9	Isle of Man	Europe	0	0	87158	2019	0	0
14	Channel Islands	Europe	0	0	75153	2007	0	0

```
print("Average IMF GDP estimate per UN region:")
print(df.groupby('UN_Region')['IMF_Estimate'].mean())
```

```
Average IMF GDP estimate per UN region:
UN_Region
Africa        3118.304292
Americas     16215.475677
Asia         17346.734747
Europe       36256.930838
Oceania      14346.470812
World        13440.000000
Name: IMF_Estimate, dtype: float64
```

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

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

```
[37] print("Maximum UN GDP estimate in each UN region:")
     print(df.groupby('UN_Region')['UN_Estimate'].max())
```

```
Maximum UN GDP estimate in each UN region:
UN_Region
Africa        12085
Americas     112653
Asia          66822
Europe       234317
Oceania       66916
World         12230
```

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

```
[34] uk_gdp = df[df['Country/Territory'] == 'United Kingdom']['IMF_Estimate'].values[0]
     print(uk_gdp)

     higher_than_uk = europe[europe['IMF_Estimate'] > uk_gdp]

     print("\nEuropean countries with higher GDP than the UK:")
     print(higher_than_uk[['Country/Territory', 'IMF_Estimate']].sort_values(by='IMF_Estimate', ascending=False)
```

```
46371

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

```
[106] df['IMF_vs_WB'] = df['IMF_Estimate'] - df['WorldBank_Estimate']
      print("Average difference between IMF and World Bank estimates per region:")
      print(df.groupby('UN_Region')['IMF_vs_WB'].mean())
```

```
Average difference between IMF and World Bank estimates per region:
UN_Region
Africa        625.449746
Americas    -4117.545157
Asia         2802.709442
Europe      -8936.756662
Oceania     -1730.379188
World        1205.000000
Name: IMF_vs_WB, dtype: float64
```

Which country has highest UN Estimate?

```
[39] # Locate the row where UN_Estimate is maximal
     idx_max = df['UN_Estimate'].idxmax()

     # Extract country name and its estimate
     top_country = df.loc[idx_max, 'Country/Territory']
     top_value   = df.loc[idx_max, 'UN_Estimate']

     print(f"The country with the highest UN estimate is {top_country} ({top_value}).")
```

```
The country with the highest UN estimate is Monaco (234317).
```

[ ] Почніть кодувати або генерувати код за допомогою ШІ.

Which country has highest Worlbank Estimate?

```
[108] idx_max_world = df['WorldBank_Estimate'].idxmax()

      # Get the country and the estimate
      top_country = df.loc[idx_max_world , 'Country/Territory']
      top_value = df.loc[idx_max_world, 'WorldBank_Estimate']

      print(f"The country with the highest World Bank estimate is {top_country} ({top_value}).")
```

```
The country with the highest World Bank estimate is Monaco (234316.0).
```

[ ] Почніть кодувати або генерувати код за допомогою ШІ.

Which country has highest IMF Estimate?

```
[109] idx_max_imf = df['IMF_Estimate'].idxmax()

      top_country = df.loc[idx_max_imf , 'Country/Territory']
      top_value = df.loc[idx_max_imf, 'IMF_Estimate']

      print(f"The country with the highest IMF estimate is {top_country} ({top_value}).")
```

```
The country with the highest IMF estimate is Luxembourg (132372.0).
```

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

      average_imf_fill = df['IMF_Estimate'].mean()
      df['IMF_Estimate'].fillna(average_imf_fill, inplace=True)

      average_worldbank_fill = df['WorldBank_Estimate'].mean()
      df['WorldBank_Estimate'].fillna(average_worldbank_fill, inplace=True)

      average_UN_fill = df['UN_Estimate'].mean()
      df['UN_Estimate'].fillna(average_UN_fill, inplace=True)

      print("\nNull values after filling 'IMF_Estimate':")
      print(df.isnull().sum())
```

```
Null values after filling 'IMF_Estimate':
Country/Territory      0
UN_Region              0
IMF_Estimate           0
IMF_Year              26
WorldBank_Estimate     0
WorldBank_Year         7
UN_Estimate            0
UN_Year                0
IMF_vs_WB              1
```

Filling 0 Values by average

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

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

      df = df.replace(0, np.nan)
      df.head()
```

	Country/Territory	UN_Region	IMF_Estimate	IMF_Year	WorldBank_Estimate	WorldBank_Year	UN_Estimate	UN_Year	
1	Monaco	Europe	17377.736041	NaN	234316.0	2021.0	234317.0	2021	-216
2	Liechtenstein	Europe	17377.736041	NaN	157755.0	2020.0	169260.0	2021	-140
3	Luxembourg	Europe	132372.000000	2023.0	133590.0	2021.0	133745.0	2021	-1
4	Ireland	Europe	114581.000000	2023.0	100172.0	2021.0	101109.0	2021	14
5	Bermuda	Americas	17377.736041	NaN	114090.0	2021.0	112653.0	2021	-96

Подальші дії: Створити код зі змінною df | Переглянути рекомендовані графіки | New interactive sheet

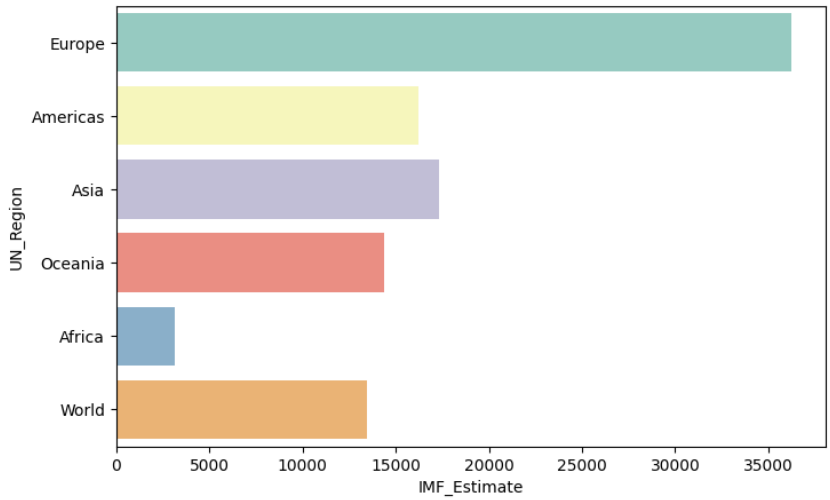
```
[49] # Calculate the average of 'Worldbank_Estimate' and 'UN_Estimate' columns
     average_worldbank = df['WorldBank_Estimate'].mean()
     average_un = df['UN_Estimate'].mean()

     print(f"Average Worldbank Estimate: {average_worldbank}")
     print(f"Average UN Estimate: {average_un}")
```

```
Average Worldbank Estimate: 19540.805555555555
Average UN Estimate: 18514.528037383177
```

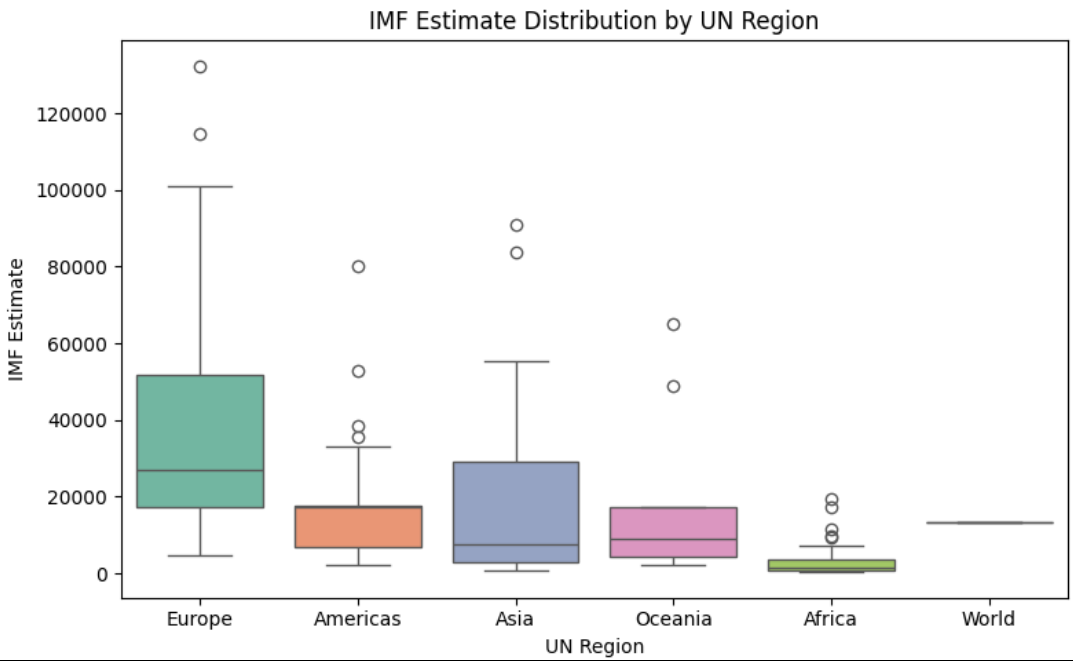
```python
# Calculate the average of 'Worldbank_Estimate' and 'UN_Estimate' and store it in a temporary column
df['avg_worldbank_un'] = df[['WorldBank_Estimate', 'UN_Estimate']].mean(axis=1)

# Fill null values in 'WorldBank_Estimate' and 'UN_Estimate' with the average from the temporary column
df['WorldBank_Estimate'].fillna(df['avg_worldbank_un'], inplace=True)
df['UN_Estimate'].fillna(df['avg_worldbank_un'], inplace=True)

# Drop the temporary 'avg_worldbank_un' column
df.drop('avg_worldbank_un', axis=1, inplace=True)

print("\nNull values after filling 'WorldBank_Estimate' and 'UN_Estimate' with combined average:")
print(df.isnull().sum())
print("\nDataFrame head after filling null values:")
print(df.head())
```

```
Null values after filling 'WorldBank_Estimate' and 'UN_Estimate' with combined average:
Country/Territory      0
UN_Region              0
IMF_Estimate           0
IMF_Year              26
WorldBank_Estimate     1
WorldBank_Year         7
UN_Estimate            1
UN_Year                0
IMF_vs_WB              6
dtype: int64

DataFrame head after filling null values:
  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
```

```python
fig = plt.figure(figsize = (8,5))
sns.barplot(x = "IMF_Estimate", y = "UN_Region", hue="UN_Region", data = df,  palette='Set3', errorbar=None, legend=False)
plt.show()
```



```
plt.figure(figsize=(8, 5))
sns.boxplot(x='UN_Region', y='IMF_Estimate', hue='UN_Region', data=df, palette='Set2', legend=False)
plt.title("IMF Estimate Distribution by UN Region")
plt.xlabel("UN Region")
plt.ylabel("IMF Estimate")
plt.tight_layout()
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:

```
name = input("What is your name? ")
glasses_of_water = int(input("How many glasses of water you drank today? "))
exercise_hours = int(input("How many hours you exercised today? "))

healthy_score = (glasses_of_water * 2) + (exercise_hours * 5)

print(f"\nHello, {name}! Hope you are well! ")
print(f"Your healthy score is: {healthy_score}.")
print("Good job!")
```

Concept	What it does	Example Use
Loop	Repeats code	Go through numbers 1–10
Function	Groups code into **reusable** blocks	Create a "greet" tool

https://www.youtube.com/watch?v=iGFdh6_FePU

Problem with Single Backslash

In strings, a backslash is used to **escape** characters like:

- `\n` = newline
- `\t` = tab
- `\\` = literal backslash
- `\"` = double quote
- `\'` = single quote

Example of a mistake:

```
python
КопіюватиРедагувати
path = "C:\newfolder\test"
print(path)
```

□ **Problem:**

- `\n` becomes a **newline**
- `\t` becomes a **tab**

https://pandas.pydata.org/

💥 Most Powerful Features of Pandas (for Data Analysis)

Feature	Why It's Powerful	Example / Use Case
📊 **DataFrame & Series**	Core data structures for 1D & 2D labeled data	Like tables in Excel or SQL
📥 **Read/write many file types**	Easy loading/saving: CSV, Excel, JSON, SQL, etc.	`pd.read_csv()`, `df.to_excel()`
🧹 **Data cleaning**	Handle missing values, duplicates, string formatting	`df.dropna()`, `df.fillna()`, `df.duplicated()`
🔍 **Filtering & slicing**	Select rows/columns using labels or conditions	`df[df['score'] > 90]`
🧠 **GroupBy & aggregation**	Summarize data easily	`df.groupby('city')['sales'].mean()`
🔁 **Merge & join**	Combine datasets like in SQL	`pd.merge()`, `df.join()`
🔄 **Pivot tables**	Reshape & summarize data flexibly	`df.pivot_table()`
📈 **Time series tools**	Work with dates, resampling, rolling stats	`df.resample('M')`, `df.rolling(7).mean()`
⚡ **Vectorized operations**	Fast processing on entire ↓ umns (no	`df['price'] * 1.2`

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.

🧠 In General:

Term	Meaning	Common Use
unique	All different values in a **single column**	Python (Pandas)
distinct	All different rows or values across **multiple columns**	SQL and sometimes Pandas

✅ Summary

Task	Pandas Code	SQL Equivalent
Unique values in 1 column	`df['col'].unique()`	`SELECT DISTINCT col FROM ...`
Unique rows based on some columns	`df.drop_duplicates(['col1', 'col2'])`	`SELECT DISTINCT col1, col2 FROM ...`
Unique rows (all columns)	`df.drop_duplicates()`	`SELECT DISTINCT * FROM ...`

https://www.youtube.com/watch?v=-E7nMqPVmyQ

https://data.gov/

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.