Coursework 1 - Report

Preparation

`Panadas` is a library dedicated to DataFrame exploration and data wrangling. Operating `import pandas as pd` to access the library. Reading the dataset using the `pd.read_csv()` function and assigning it to variable `data`. `data.head()` and `data.info()` functions were performed for a glimpse of data structures (datatypes/columns which contain NULL values). The `describe()` function was used to provide summary statistics for all non-null values in the DataFrame.

Question 1

A `dropna()` function was performed to delete all rows that show NULL values, this is to prepare for calculations later. The new column death_excess was calculated by subtraction between columns deaths_2020_all_ages and deaths_2019_all_ages.

From the output of `data.info()`, we can tell the deaths_2020_all_ages and the deaths_2019_all_ages columns have 1837 and 1872 non-null values respectively. After the `dropna()`function was performed, there were 1817 rows left. Why wasn't 1837? This is because the distribution of the null values in both columns varies. Given the example (See table below): suppose there are 4 non-null rows in 2019 and 6 non-null rows in 2020. When deleting rows that contain NULL values, we can see there are 3 rows left instead of 4.

2019 (4 non-null rows)	2020 (6 non-null rows)
3	2
5	NULL
6	8
NULL	9
NULL	NULL
NULL	10
1	11
NULL	12

Question 2

A conditional code using `.loc[]` was performed to return all rows where `death_excess > 0`. `unique()` and `nuique()` were used for the column ‘location’ to get all countries' names and the numbers of countries that have excess deaths greater than 0.

Question 3

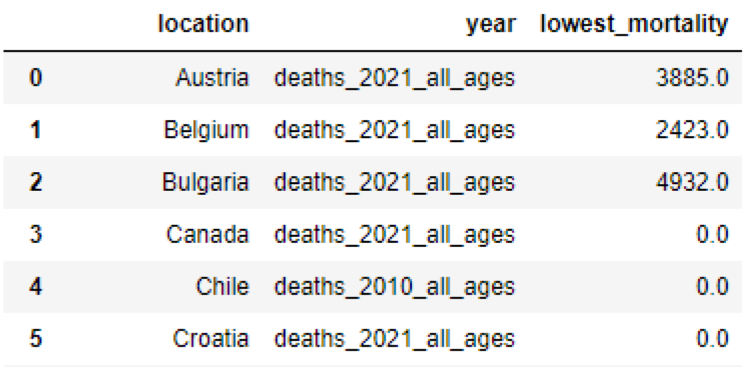
By grouping the column ‘location’ `groupby()` we can get information based on each country. Using `apply()` function by applying a `lambda` function to sort every country’s death_excess in a descending order and extracting the top 5 values using `head(5)`. `reset_index(drop=True)` Reset the index, discarding the current index to get a clear view. The table sometimes is truncated in jupyterNotebook, so `pd.set_option('display.max_rows', None)` was used to show all rows.

Question 4

In the dataset, there are missing values in each year's death column, thus the method of filling all NULL with 0 using `fillna(0)` was performed. Later, I summed up the mortality of each year of each country using `df2.groupby('location').sum()`.

To get the year with the lowest mortality for each location, the solution was creating 3 lists/series for location, year and lowest mortality columns and writing them into `result_df` using `pd.DataFrame()` function (Part of the output see Figure 1).

- The location list was extracted using a for loop by appending each location from the location column into an empty list.
- To get the lowest value of each year is to get the minimum value of each row and its corresponding column name. `df_data.min(axis=1)` and `df_data.idxmin(axis=1)` were used to get the information in a series datatype.



	location	year	lowest_mortality
0	Austria	deaths_2021_all_ages	3885.0
1	Belgium	deaths_2021_all_ages	2423.0
2	Bulgaria	deaths_2021_all_ages	4932.0
3	Canada	deaths_2021_all_ages	0.0
4	Chile	deaths_2010_all_ages	0.0
5	Croatia	deaths_2021_all_ages	0.0

Figure1

Suggestion

Data analysis along with visualisations is suggested.