CITS1401 Project 1 marking guide, Semester 1, 2023

There are 22 marks for functionality which are divided into 4 outputs test cases. Each test case checks each output with two different files. Marks for each test case are mentioned in the test cases mentioned below.

Mention in comments why student lost the marks. In addition to any other comments. You may add a note in comments which includes something like x/22 (test) + x/5 (style) + z/3 (efficiency). **Please note that** you can't leave the comment section empty in Moodle. You can add their awarded marks for each criteria (x/22 (test) + x/5 (style) + z/3 (efficiency)) in the comment section.

If output is not returned in proper format as required and simply printed, then deduct 4 marks extra as mentioned in the end and grade it accordingly. This is considered as marker's intervention to fix student's code. You will get a good idea from test cases output that what to expect from student's code.

Following is the table that shows the marking criteria for each calculation they complete in Python:

Maximum awarded marks	**Criteria**
6	Correlation calculation.
4	Average and Standard deviation calculation.
4	Density calculation.
4	Maximum and minimum calculation.
4	Error cases
8	Style and efficiency
Total: 30	

Minimum marks in each cannot be lower than zero.

Output test cases:

1. Correlation (6 Marks):

   If the outputs of both test files match then assign 6 marks, else
   If the output of only one test file fully matches then assign 3 marks.

2. Standard deviation and Average (4 Marks):

   If the outputs of both test files match then assign 4 marks, else
   If the output of only one test file fully matches then assign 2 marks.

3. Maximum and Minimum (4 Marks):

   If the outputs of both test files match then assign 4 marks, else
   If the output of only one test file fully matches then assign 2 marks.

4. Density (4 Marks):

   If the outputs of both test files match then assign 4 marks, else
   If the output of only one test file fully matches then assign 2 marks.

5. Error State (4 Marks):

For successfully passing each error state assign 1 mark. We are checking following error state:

i) Zero division: Calculations should not get divide by zero error in any case.
ii) Single region input: What will happen if there is a single row of data of specific region only.
iii) Random rows for the input region.
iv) All data zero for the input region.

Note: Deduct 1 mark per calculation if it is failed in rounding output.

<u>Style and efficiency: (be lenient but mention in comments to improve for next time)</u>

- Style (5/8) which involves intuitive variable and function names, consistent indentation, comments, etc.
    - Default is 5.
    - Deduct 2 marks if person's name or student id is not on the file to identify author of the code.
    - Deduct 2 marks if scant comments are provided.
    - Deduct 2 marks if appropriate naming convention is not used.
    - Deduct 2 marks if two or less functions are created.
    - Minimum can be zero.

- Efficiency (3/8):
    - Default is 3 marks.
    - Deduct 1 mark if readline() function is used in a loop or file is opened multiple times.
    - Deduct 2 marks if the code includes repeated blocks instead of loops or code has more loops than necessary.
    - Deduct 2 mark if the code is taking long time to compile the big testing file.

For any intervention to fix the code, deduct 4 marks extra for each intervention. However, it should be quick and simple. Normally missing a colon or wrong indentation or missing a character of some instruction is considered as a simple debug. Re-writing or adding or removing the code is not part of the grading and code should be graded accordingly as mentioned above.

If student has imported any module then comment it out and run the code, grade it accordingly and assign 0/3 in efficiency.

Sample Outcomes are provided below which can used with provided data files to test your code:

```
>>> sMaxMin_result,sstdAverage_result,sdensity_result,scorr_result = main('countries.csv','Africa')
```

```
>>> sMaxMin_result,sstdAverage_result,sdensity_result,scorr_result
```

```
(['Western Sahara', 'Seychelles'], [291620.6667, 239936.4497], [['Mayotte', 727.5067], ['Sao Tome & Principe', 228.2906], ['Cabo Verde', 137.962], ['Saint Helena', 15.5821], ['Seychelles', 11.0008], ['Western Sahara', 2.2456]], 0.6243)
```

```
>>>sMaxMin_result1,sstdAverage_result1,sdensity_result1,scorr_result
1=main('World_countries.csv','Asia')

>>>sMaxMin_result1,sstdAverage_result1,sdensity_result1,scorr_result
1

(['China', 'Brunei'], [91001074.0196, 274892875.7162], [['Macao',
21644.5], ['Singapore', 8357.6314], ['Hong Kong', 7139.9819],
['Bahrain', 2238.9145], ['Maldives', 1801.8133], ['Bangladesh',
1265.1869], ['State of Palestine', 847.411], ['Taiwan', 672.6003],
['Lebanon', 667.1989], ['South Korea', 527.298], ['India', 464.1494],
['Israel', 399.9785], ['Philippines', 367.5121], ['Japan', 346.9338],
['Sri Lanka', 341.4647], ['Vietnam', 313.9245], ['Pakistan',
286.5457], ['Qatar', 248.1527], ['Kuwait', 239.6504], ['North Korea',
214.092], ['Nepal', 203.2564], ['China', 153.3118], ['Indonesia',
150.9871], ['Thailand', 136.6243], ['Cyprus', 130.6666],
['Azerbaijan', 122.6642], ['United Arab Emirates', 118.3062],
['Jordan', 114.926], ['Turkey', 109.5839], ['Armenia', 104.083],
['Malaysia', 98.5116], ['Syria', 95.3039], ['Cambodia', 94.7143],
['Iraq', 92.6103], ['Timor-Leste', 88.6648], ['Myanmar', 83.2858],
['Brunei', 83.0131], ['Uzbekistan', 78.677], ['Tajikistan', 68.1455],
['Afghanistan', 59.6274], ['Georgia', 57.4063], ['Yemen', 56.4918],
['Iran', 51.5753], ['Kyrgyzstan', 34.0156], ['Laos', 31.5232],
['Bhutan', 20.2431], ['Oman', 16.4996], ['Saudi Arabia', 16.1948],
['Turkmenistan', 12.8343], ['Kazakhstan', 6.9551], ['Mongolia',
2.1102]], 0.8192)
```

#Testing for random rows

```
>>>MaxMin_result3,stdAverage_result3,density_result3,corr_result3=ma
in('World_countries_1.csv','Asia')

>>> MaxMin_result3,stdAverage_result3,density_result3,corr_result3

(['China', 'Brunei'], [91001074.0196, 274892875.7162], [['Macao',
21644.5], ['Singapore', 8357.6314], ['Hong Kong', 7139.9819],
['Bahrain', 2238.9145], ['Maldives', 1801.8133], ['Bangladesh',
1265.1869], ['State of Palestine', 847.411], ['Taiwan', 672.6003],
['Lebanon', 667.1989], ['South Korea', 527.298], ['India', 464.1494],
['Israel', 399.9785], ['Philippines', 367.5121], ['Japan', 346.9338],
['Sri Lanka', 341.4647], ['Vietnam', 313.9245], ['Pakistan',
286.5457], ['Qatar', 248.1527], ['Kuwait', 239.6504], ['North Korea',
214.092], ['Nepal', 203.2564], ['China', 153.3118], ['Indonesia',
150.9871], ['Thailand', 136.6243], ['Cyprus', 130.6666],
['Azerbaijan', 122.6642], ['United Arab Emirates', 118.3062],
['Jordan', 114.926], ['Turkey', 109.5839], ['Armenia', 104.083],
['Malaysia', 98.5116], ['Syria', 95.3039], ['Cambodia', 94.7143],
['Iraq', 92.6103], ['Timor-Leste', 88.6648], ['Myanmar', 83.2858],
['Brunei', 83.0131], ['Uzbekistan', 78.677], ['Tajikistan', 68.1455],
['Afghanistan', 59.6274], ['Georgia', 57.4063], ['Yemen', 56.4918],
['Iran', 51.5753], ['Kyrgyzstan', 34.0156], ['Laos', 31.5232],
['Bhutan', 20.2431], ['Oman', 16.4996], ['Saudi Arabia', 16.1948],
```

['Turkmenistan', 12.8343], ['Kazakhstan', 6.9551], ['Mongolia', 2.1102]], 0.8192)

#Testing for error state where all data is zero")

```
>>>
MaxMin_result1,stdAverage_result1,density_result1,corr_result1=main(
'World_countries.csv','Testing_zero')

>>> MaxMin_result1,stdAverage_result1,density_result1,corr_result1

(0, [0.0, 0.0], 0, 0)
```

#Testing for error state where all data is dividing by zero and single region

```
>>>MaxMin_result2,stdAverage_result2,density_result2,corr_result2=ma
in('World_countries.csv','Testing_single')

>>> MaxMin_result2,stdAverage_result2,density_result2,corr_result2

(0, None, [], None)
```