Exploratory Data Analysis DoQA Cooking Test Output

```
pip install textatistic
```

```
Requirement already satisfied: textatistic in /usr/local/lib/python3.10/dist-packages (0.0.1)
Requirement already satisfied: pyhyphen>=2.0.5 in /usr/local/lib/python3.10/dist-packages (from textatistic) (4.
Requirement already satisfied: wheel>=0.36.0 in /usr/local/lib/python3.10/dist-packages (from pyhyphen>=2.0.5->t
Requirement already satisfied: setuptools>=52.0 in /usr/local/lib/python3.10/dist-packages (from pyhyphen>=2.0.5
Requirement already satisfied: appdirs>=1.4.0 in /usr/local/lib/python3.10/dist-packages (from pyhyphen>=2.0.5->
Requirement already satisfied: requests>=2.25 in /usr/local/lib/python3.10/dist-packages (from pyhyphen>=2.0.5->
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from request
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.25->pyh
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.2
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.2
```

```
import json
import pandas as pd
from textatistic import Textatistic
from pathlib import Path
import re


with open("/content/doqa-cooking-test-v2.1.json", 'r') as file:
    # Load the JSON data
    json_data = json.load(file)


# Access the 'data' key
data = json_data['data']


# Create lists to store the extracted data
titles = []
backgrounds = []
all_paragraphs = []

for entry in data:
   title = entry["title"]
   background = entry["background"]
   paragraphs = [c["context"] for c in entry["paragraphs"]]
   # Store each variable in separate lists
   titles.append(title)
   backgrounds.append(background)
   all_paragraphs.append(paragraphs)
#three separate lists: titles, backgrounds, and all_paragraphs
title_df = pd.DataFrame({'Title': titles})
background_df = pd.DataFrame({'Background': backgrounds})
paragraphs_df = pd.DataFrame({'Paragraphs': all_paragraphs})


title_df = title_df['Title'].tolist()
background_df = background_df['Background'].tolist()
paragraphs_df = paragraphs_df['Paragraphs'].tolist()
```

```
def clean_string(text): # Cleaning
    """re.sub(pattern, repl, string).
    Returns the string obtained by replacing the leftmost
    non-overlapping occurrences of pattern in string by the
    replacement thus removing any urls
    """
    return " ".join(re.sub("([^0-9A-Za-z \t])|(\w+:\/\/\S+)", "", str(text)).split())


TextOnlyTitle = [clean_string(Title) for Title in title_df]
TextOnlyBackground = [clean_string(Background) for Background in background_df]
TextOnlyParagraphs = [clean_string(Paragraphs) for Paragraphs in paragraphs_df]


TextOnlyTitle[:2]
```

```
['Canning chili with beans', 'Is rare duck breast safe']
```

```
TextOnlyBackground[:2]
```

```
['Were going to try canning some homemade chili with canned kidney beans Question Would it be best to add the beans right before canning so they dont get over cooked or do you think there would be little difference if they were cooked with the chili before canning Im trying to avoid mushy beans but would like to cook a big pot and can the left overs I usually add the beans in the last half hour of cooking anyway',
 'Other forms of poultry are regarded as undercooked at the slightest sign of pink juices and yet it is common practice to cook duck so that its medium rare Is the risk of food poisoning significantly lower with duck meat than other birdsOn the other hand if we could be sure that a piece of chicken could was salmonellafree would its texture and flavor be improved by not overcooking it I expect the sight of pink would put most people off but if one could overcome that could it actually taste better']
```

```
TextOnlyParagraphs[:2]
```

```
['If you are going to be canning this for long term storage you will need to be pressure canning it and as that will mean some pretty high heat youd be better off not cooking you beans too long before canning Id add them in a few minutes before the end of cooking just to get them warmed up in preparation for canning it as a half hour of cooking plus pressure canning would probably mean mush CANNOTANSWER',
 'Rare duck meat is safe to eat because it does NOT contain the same risk of Salmonella as does chicken meatPrimarily because ducks as mentioned above have not traditionally been raised in the same squalid conditions as factory raised chickens salmonella is a disease that is primarily transmitted through dirtdirty unclean conditions Now on the other hand as more and more ducks are being raised in industrial conditions they are also becoming more likely to contain strains of Salmonella CANNOTANSWER']
```

Title

```
a_list = TextOnlyTitle


new_list_a = [x for x in a_list if len(x) < 1000]


textfile_a = open("fileA_file.txt", "w")


for element in new_list_a:
    textfile_a.write(element + "\n")


textfile_a.close()


textA = Path('fileA_file.txt').read_text()


a_string = textA
```

```
Str_EndFixA = a_string.replace("\n", ".")
```

```
readability = Textatistic(Str_EndFixA)
```

```
%precision 3
```

```
'%.3f'
```

```
readability.dict()
```

```
{'char_count': 16234,
 'word_count': 3052,
 'sent_count': 400,
 'sybl_count': 4248,
 'notdalechall_count': 952,
 'polysyblword_count': 264,
 'flesch_score': 81.338,
 'fleschkincaid_score': 3.810,
 'gunningfog_score': 6.512,
 'smog_score': 7.770,
 'dalechall_score': 8.940}
```

For Background

```
b_list = TextOnlyBackground
```

```
new_list_b = [x for x in b_list if len(x) < 1000]
```

```
textfile_b = open("fileB_file.txt", "w")
```

```
for element in new_list_b:
    textfile_b.write(element + "\n")
```

```
textfile_b.close()
```

```
textB = Path('fileB_file.txt').read_text()
```

```
b_string = textB
```

```
Str_EndFixB = b_string.replace("\n", ".")
```

```
readability = Textatistic(Str_EndFixB)
```

```
%precision 3
```

```
'%.3f'
```

```
readability.dict()
```

```
{'char_count': 124810,
 'word_count': 28464,
 'sent_count': 370,
 'sybl_count': 36253,
 'notdalechall_count': 5487,
 'polysyblword_count': 1496,
 'flesch_score': 21.001,
```

```
 'fleschkincaid_score': 29.442,
 'gunningfog_score': 32.874,
 'smog_score': 14.616,
 'dalechall_score': 10.496}
```

For Paragraph

```
p_list = TextOnlyParagraphs
```

```
new_list_p = [x for x in p_list if len(x) < 1000]
```

```
textfile_p = open("fileP_file.txt", "w")
```

```
for element in new_list_p:
    textfile_p.write(element + "\n")
```

```
textfile_p.close()
```

```
textP = Path('fileP_file.txt').read_text()
```

```
p_string = textP
```

```
Str_EndFixP = p_string.replace("\n", ".")
```

```
readability = Textatistic(Str_EndFixP)
```

```
%precision 3
```

```
'%.3f'
```

```
readability.dict()
```

```
{'char_count': 160412,
 'word_count': 35565,
 'sent_count': 353,
 'sybl_count': 45467,
 'notdalechall_count': 7171,
 'polysyblword_count': 1958,
 'flesch_score': -3.581,
 'fleschkincaid_score': 38.788,
 'gunningfog_score': 42.502,
 'smog_score': 16.583,
 'dalechall_score': 11.817}
```