Exploratory Data Analysis DoQA Travel 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-travel-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]
```

```
['Japanese train etiquette Is it acceptable to take a baby in the Shinkansen Green Car',
 'UK online visa application travelling with someone who doesnt have visa yet']
```

```
TextOnlyBackground[:2]
```

```
['The green carriage is of course first class This would be our first preference with our 5 month old baby in
lap because of the presumably larger toilets for changing and whatnot However the baby being a Western white
baby will occasionally babble screech or cry uncontrollably for a little whileSpeaking in terms of Japanese
culture should we avoid Green class tickets if we are taking aboard a potentially disruptive baby',
 'I am filling in the UK online visa application on visa4ukgovuk They ask if I am travelling with anyone My
plans have been made in conjunction with someone else we arrive at the same time and leave at the same time and
some of our tickets have both our names on them although our movements will differ some of the time and we are
doing separate interviews I would like to give them every detail possible to as not to be accused of
withholding information However there is this question about does this person have a valid visa for the UK The
answer is not yet he will not have a visa until his own application process is complete just like me However
the options are yes and no So the correct answer is currently no Will this cause my own application hassles']
```

```
TextOnlyParagraphs[:2]
```

```
['Yes it is acceptable However it is etiquette that if the baby does start to make noise that you take it to
the deck area beyond the doors where the bathrooms telephone vending machines etc are This is the same protocol
if you have to make or receive a telephone call want to have a loud conversation with your seatmate or do
anything that might disturb other passengers I should note that this has nothing to do with the green car the
main cars nominally operate under these protocols its just that the other passengers and conductors are less
likely to enforce them CANNOTANSWER',
 'In general they are trying to find out if for example a child is travelling with their parent or if a person
is travelling with their spouse Also they are trying to establish the persons premise for visiting the UK and
sometimes if a person is travelling with a group the info can be useful Finally the info is helpful in
understanding the applicants connection to the UK if there is one and if their visit is contingent on another
person getting a visa Based upon what you wrote you are travelling with a friendcompanion You would answer the
question honestly and check the NO box if the person does not have a valid UK visa Optionally you can then go
to Part 9 of the form and enter a brief explanation of how and when your friendcompanion will apply for their
visa In fact you can simply reword your text above ie your question and copy it to Part 9 It is OK to be
explicit about your relationship with your friendcompanion because it helps establish the premise of both your
visits 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': 20485,
 'word_count': 3993,
 'sent_count': 396,
 'sybl_count': 5697,
 'notdalechall_count': 1416,
 'polysyblword_count': 394,
 'flesch_score': 75.898,
 'fleschkincaid_score': 5.178,
 'gunningfog_score': 7.980,
 'smog_score': 8.827,
 'dalechall_score': 9.736}
```

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': 129145,
 'word_count': 29217,
 'sent_count': 335,
 'sybl_count': 38435,
 'notdalechall_count': 6871,
 'polysyblword_count': 1900,
 'flesch_score': 7.020,
 'fleschkincaid_score': 33.947,
 'gunningfog_score': 37.487,
 'smog_score': 16.734,
 'dalechall_score': 11.676}
```

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': 161799,
 'word_count': 35911,
 'sent_count': 337,
 'sybl_count': 46946,
 'notdalechall_count': 7683,
 'polysyblword_count': 2358,
 'flesch_score': -11.921,
 'fleschkincaid_score': 41.395,
 'gunningfog_score': 45.251,
 'smog_score': 18.240,
 'dalechall_score': 12.300}
```