Data Extraction

```
In [9]: import pandas as pd
        import bs4
        import requests
```

```
In [52]: df=pd.read_csv("Input.xlsx.csv")          # To read input file
         df['URL_ID'] = df['URL_ID'].astype(str).replace('\.0', '', regex=True) #For accurate file name
         print("Dataframe Created")
```

```
Dataframe Created
```

```
In [55]: #This Block of Code Takes Time To Complete
         import time

         start = time.time()

         print("Extraction Started")
         for index, row in df.iterrows():
             result=requests.get(row["URL"])
             soup=bs4.BeautifulSoup(result.text,'lxml')
             try:                                                   #Using Try try block for error handeling
                 title=soup.select("h1")[0].text
                 filecontent=soup.select(".td-post-content")[0].text
             except:
                 filecontent="Ooops... Error 404"
             with open(f".//output_files//{row['URL_ID']}.txt", "w", encoding="utf-8") as f:
                 f.write(title)
                 f.write(filecontent)

         end = time.time()



         print("Extraction Complete")
         print("Time taken for extraction ",end-start)
```

```
Extraction Started
Extraction Complete
Time taken for extraction  253.02700448036194
```

For Data Analysis

```
In [56]: import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk import punkt
from textblob import TextBlob
from nltk.corpus import cmudict
# Download if not already downloaded
# nltk.download('vader_lexicon')
# nltk.download('punkt')
# nltk.download('stopwords')
# nltk.download('corpus')


#Output Dataframe
output_df = pd.DataFrame(
    columns=['URL_ID','URL','POSITIVE SCORE', 'NEGATIVE SCORE', 'POLARITY SCORE', 'SUBJECTIVITY SCORE',
             'AVG SENTENCE LENGTH', 'PERCENTAGE OF COMPLEX WORDS', 'FOG INDEX', 'AVG NUMBER OF WORDS PER SENTENCE',
             'COMPLEX WORD COUNT', 'WORD COUNT', 'SYLLABLE PER WORD', 'PERSONAL PRONOUNS', 'AVG WORD LENGTH',])

def count_syllables(word, dic):
    return len(dic.inserted(word).split('-'))


#Iterating input.csv file's df for reference
for index, row in df.iterrows():
    output_df.loc[index,"URL_ID"]=row["URL_ID"]
    output_df.loc[index,"URL"]=row["URL"]
    with open(f".//output_files//{row['URL_ID']}.txt", "r", encoding="utf-8") as f:
        data=f.read()

        sia = SentimentIntensityAnalyzer()
        # Get sentiment scores
        sentiment_scores = sia.polarity_scores(data)
        # Positive score
        output_df.loc[index,"POSITIVE SCORE"] = sentiment_scores['pos']
        output_df.loc[index,'NEGATIVE SCORE'] = sentiment_scores['neg']



        blob = TextBlob(data)
```

```python
        output_df.loc[index,'POLARITY SCORE'] = blob.sentiment.polarity
        output_df.loc[index,'SUBJECTIVITY SCORE'] = blob.sentiment.subjectivity


        sentences = nltk.sent_tokenize(data)
        words = nltk.word_tokenize(data)




        # Count total number of words
        total_words = sum(len(nltk.word_tokenize(sentence)) for sentence in sentences)
        # Count total number of sentences
        total_sentences = len(sentences)
        # Calculate average sentence length
        avg_sent_length=total_words / total_sentences

        output_df.loc[index,'AVG SENTENCE LENGTH'] = avg_sent_length

        output_df.loc[index,'AVG NUMBER OF WORDS PER SENTENCE'] = avg_sent_length

        # Initialize pyphen dictionary
        dic = pyphen.Pyphen(lang='en')

        # Define a threshold for complex words (e.g., words with more than three syllables)
        complex_word_threshold = 3

        # Process text in smaller chunks
        chunk_size = 100  # Adjust as needed
        complex_word_count = sum(1 for word in words if count_syllables(word, dic) > complex_word_threshold)
        total_word_count = len(words)

        for i in range(0, len(words), chunk_size):
            chunk = words[i:i+chunk_size]
            for word in chunk:
                total_word_count += 1
                if count_syllables(word, dic) > complex_word_threshold:
                    complex_word_count += 1

        # Calculate the percentage of complex words
        percent_of_complex_words=(complex_word_count / total_word_count) * 100

        output_df.loc[index,'PERCENTAGE OF COMPLEX WORDS'] = percent_of_complex_words
        output_df.loc[index,'COMPLEX WORD COUNT'] = complex_word_count
        output_df.loc[index,'WORD COUNT'] = total_word_count

        FOG_Index = 0.4 * (avg_sent_length + percent_of_complex_words)

        output_df.loc[index,'FOG INDEX']=FOG_Index



        total_syllables = sum(count_syllables(word, dic) for word in words)
        average_syllables_per_word = total_syllables / total_words
        output_df.loc[index,'SYLLABLE PER WORD']=average_syllables_per_word


        personal_pronouns = ["I", "you", "he", "she", "it", "we", "they", "me", "him", "her", "us", "them"]

        # Count the occurrences of personal pronouns
        personal_pronoun_count = sum(1 for word in words if word.lower() in personal_pronouns)
        output_df.loc[index,'PERSONAL PRONOUNS']=personal_pronoun_count



        total_characters = sum(len(word) for word in words)
        average_word_length = total_characters / total_words
        output_df.loc[index,'AVG WORD LENGTH']=average_word_length
print("Analysis Complete")
```

```
Analysis Complete
```

In [57]:

```
output_df
```

Out[57]:

	URL_ID	URL	POSITIVE SCORE	NEGATIVE SCORE	POLARITY SCORE	SUBJECTIVITY SCORE	AVG SENTENCE LENGTH	PERCENTAGE OF COMPLEX WORDS	FOG INDEX	AVG NUMBER OF WORDS PER SENTENCE
0	123	https://insights.blackcoffer.com/rise-of-telem...	0.137	0.02	0.136034	0.43728	23.25	3.494624	10.697849	23.25
1	321	https://insights.blackcoffer.com/rise-of-e-hea...	0.171	0.01	0.111801	0.615704	27.04	6.213018	13.301207	27.04
2	2345	https://insights.blackcoffer.com/rise-of-e-hea...	0.131	0.047	0.086835	0.459666	17.768116	3.344209	8.44493	17.768116
3	4321	https://insights.blackcoffer.com/rise-of-telem...	0.205	0.055	0.139706	0.385624	23.066667	3.323699	10.556146	23.066667
4	432	https://insights.blackcoffer.com/rise-of-telem...	0.205	0.055	0.139706	0.385624	23.066667	3.323699	10.556146	23.066667
...	...	...	...	...	...	...	...	...	...	...
109	50921	https://insights.blackcoffer.com/coronavirus-i...	0.032	0.049	0.077797	0.432445	25.733333	4.145078	11.951364	25.733333
110	51382.8	https://insights.blackcoffer.com/coronavirus-i...	0.067	0.106	0.013231	0.401288	37.84	2.114165	15.981666	37.84
111	51844.6	https://insights.blackcoffer.com/what-are-the-...	0.119	0.023	0.132965	0.455201	27.873239	3.183426	12.422666	27.873239
112	52306.4	https://insights.blackcoffer.com/marketing-dri...	0.089	0.06	0.073452	0.434461	26.915254	3.02267	11.97517	26.915254
113	52768.2	https://insights.blackcoffer.com/continued-dem...	0.147	0.083	0.0561	0.44689	27.585366	5.835544	13.368364	27.585366

114 rows × 15 columns