```
[48] from google.colab import drive
     drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[49] #Loading Libraries
     import nltk
     nltk.download('punkt')
     nltk.download('stopwords')
     from nltk.corpus import stopwords
     from nltk.tokenize import word_tokenize, sent_tokenize
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
[68] #Loading and reading file
     filename="/content/drive/MyDrive/COVID_19_dataset/COVID_19_dataset/documents/008.txt"

     f = open((filename), "r")
     text=f.read() #append each line in the file to a list
     f.close()
```

```
[69] #Pre-processing
     sent_tokens = nltk.sent_tokenize(text)
     word_tokens = nltk.word_tokenize(text)
     word_tokens_lower=[word.lower() for word in word_tokens]
     stopWords = list(set(stopwords.words("english")))
     word_tokens_refined=[x for x in word_tokens_lower if x not in stopWords]
     print(len(word_tokens_refined))
```

```
310
```

```
[70] #Crete Frequency Distribution of words in the document
     freqTable = dict()
     for word in word_tokens_refined:
         if word in freqTable:
             freqTable[word] += 1
         else:
             freqTable[word] = 1
     print(len(freqTable))
```

```
200
```

```
[71] #Compute score of each sentence
     sentenceValue = dict()
     for sentence in sent_tokens:
         sentenceValue[sentence]=0
         for word, freq in freqTable.items():
             if word in sentence.lower():
                 sentenceValue[sentence] += freq
     print(sentenceValue.values())
```

```
dict_values([51, 83, 44, 39, 29, 67, 67, 45, 45, 48, 66, 72, 36, 31, 40, 58, 82, 66, 53, 64, 51, 28])
```

```
[72] #Compute average sentence score in the document
     sumValues = 0
     for sentence in sentenceValue:
         sumValues += sentenceValue[sentence]
     average = int(sumValues / len(sentenceValue))
     print(average)
     # Storing sentences into our summary.
     summary = ''
     for sentence in sent_tokens:
         if (sentence in sentenceValue) and (sentenceValue[sentence] > (1.3*average)):
             summary += " " + sentence
     print(summary)
```

```
52
 Belgium could soon be overwhelmed by new coronavirus infections, the health minister has warned, amid soaring case numbers across the country. From Monday, u
```

```
[73] # Import packages
     import matplotlib.pyplot as plt
     %matplotlib inline
```

```
[74] # Define a function to plot word cloud
     def plot_cloud(wordcloud):
         # Set figure size
```

```
        plt.figure(figsize=(40, 30))
        # Display image
        plt.imshow(wordcloud)
        # No axis details
        plt.axis("off");
```

```
[75] # Import package
     from wordcloud import WordCloud, STOPWORDS
     # Generate word cloud
     wordcloud = WordCloud(width = 3000, height = 2000, random_state=1, background_color='salmon', colormap='Pastel1', collocations=False, stopwords = STOPWORDS).
     # Plot
     plot_cloud(wordcloud)
```