Covid Data Information Retrieval and Analysis

*Here text data is comprised of scholarly articles about COVID-19, SARS-COV-2, and related coronaviruses. The dataset is known as CORD-19, prepared by the White House and a coalition of leading research groups. This freely available dataset is provided to the global research community to apply recent advances in natural language processing and other AI techniques to generate new insights in support of the ongoing fight against this infectious disease. There is a growing urgency for these approaches because of the rapid acceleration in new coronavirus literature, making it difficult for the medical research community to keep up.*

***1.We use:***

*Python 3.6, spacy, Tensorflow, NLTK, Dask*

# ***2. Dataset Description***

*The dataset used here is from* [*Kaggle*](https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge) *competition. The dataset contains doc\_id, source, title, abstract, and text body. doc\_id is document id, the source is the place from where the paper is extracted like biorxiv, abstract is a summary of the paper, and text body is body or content of the research paper.*

# ***3.Tasks***

*Our task is to retrieve information based on the following questions.*

## ***Task 1 (COVID-19 Open Research Dataset Challenge (CORD-19))***

* ***What is known about transmission, incubation, and environmental stability?***
* ***What do we know about natural history, transmission, and diagnostics for the virus? What have we learned about infection prevention and control?***

*Specifically, we want to know what the literature reports about:*

* *Range of incubation periods for the disease in humans (and how this varies across age and health status) and how long individuals are contagious, even after recovery.*
* *Prevalence of asymptomatic shedding and transmission (e.g., particularly children).*
* *Seasonality of transmission.*
* *Physical science of the coronavirus (e.g., charge distribution, adhesion to hydrophilic/phobic surfaces, environmental survival to inform decontamination efforts for affected areas and provide information about viral shedding).*
* *Persistence and stability on a multitude of substrates and sources (e.g., nasal discharge, sputum, urine, faecal matter, blood).*
* *Persistence of virus on surfaces of different materials (e,g., copper, stainless steel, plastic).*
* *Natural history of the virus and shedding of it from an infected person*
* *Implementation of diagnostics and products to improve clinical processes*
* *Disease models, including animal models for infection, disease and transmission*
* *Tools and studies to monitor phenotypic change and potential adaptation of the virus*
* *Immune response and immunity*
* *Effectiveness of movement control strategies to prevent secondary transmission in health care and community settings*
* *Effectiveness of personal protective equipment (PPE) and its usefulness to reduce risk of transmission in health care and community settings*
* *Role of the environment in transmission*

# ***4The Process***

## ***4.1 Import required libraries***

*Pandas, OS and json are required to get preprocess the dataset and extract the required features.*

## ***4.2 Create a dataset from the json files***

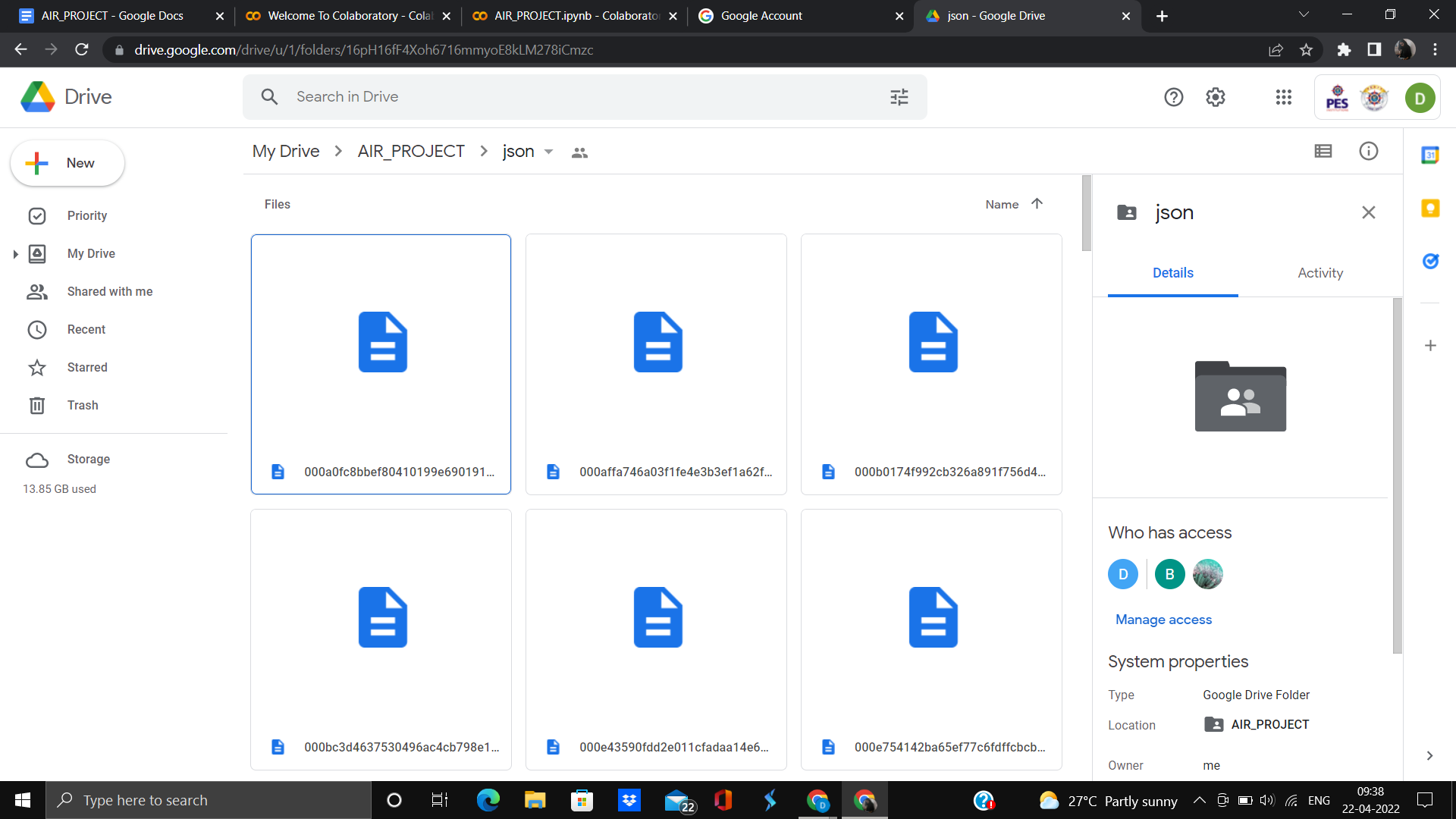
## ***4.2.1Read the dataset uploaded on google drive***

*Mount google drive to colab and authorize the request to access google colab.Load the dataset from there.*

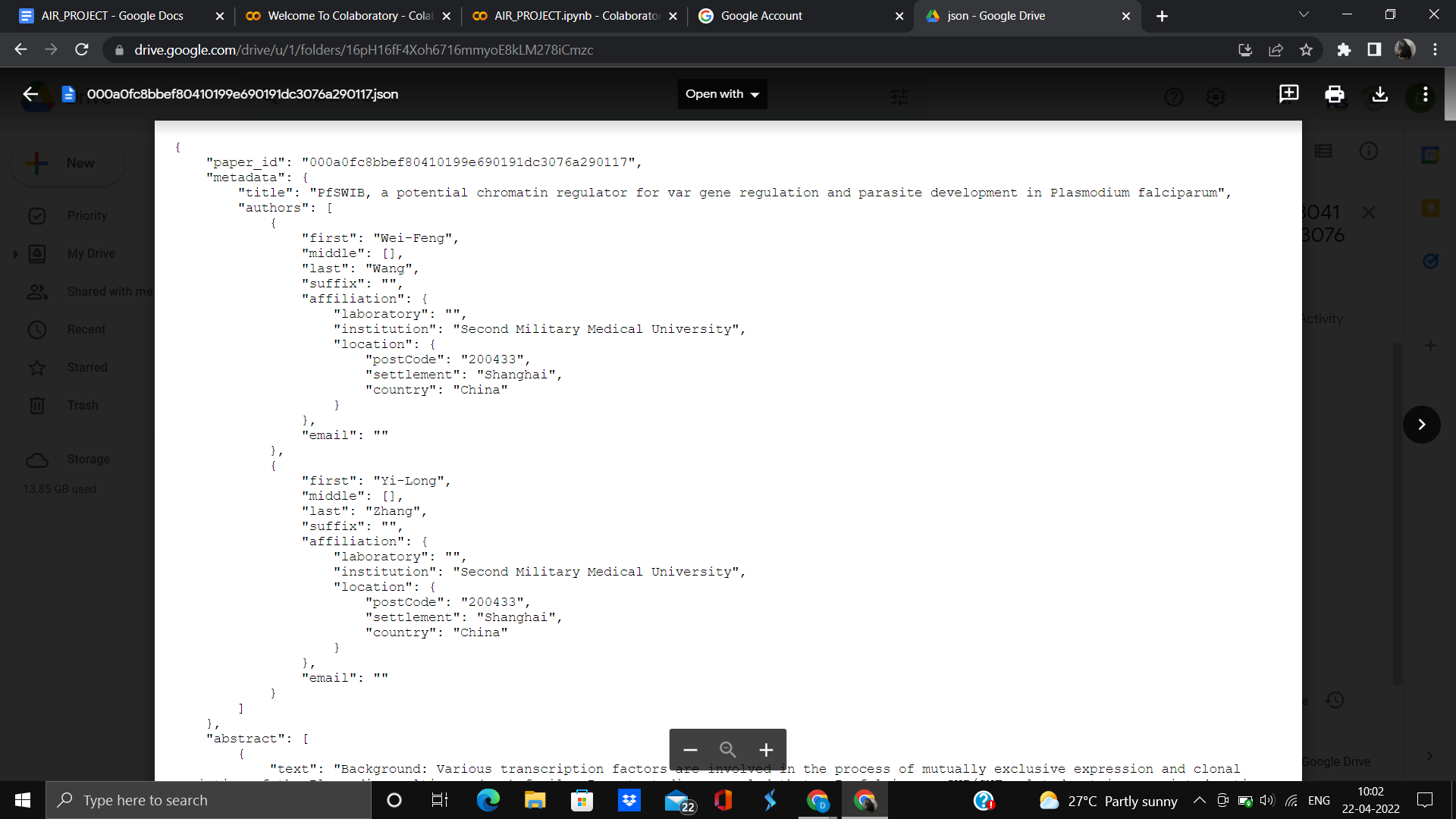
## ***4.2.2 Write a function to preprocess the json files and extract the text***

*The json file has a key ‘body\_text’. Apply pd.Series to get all the sub-keys under the current field. Then extract the value of the key ‘text’ and merge the values to get the text of the json file.*

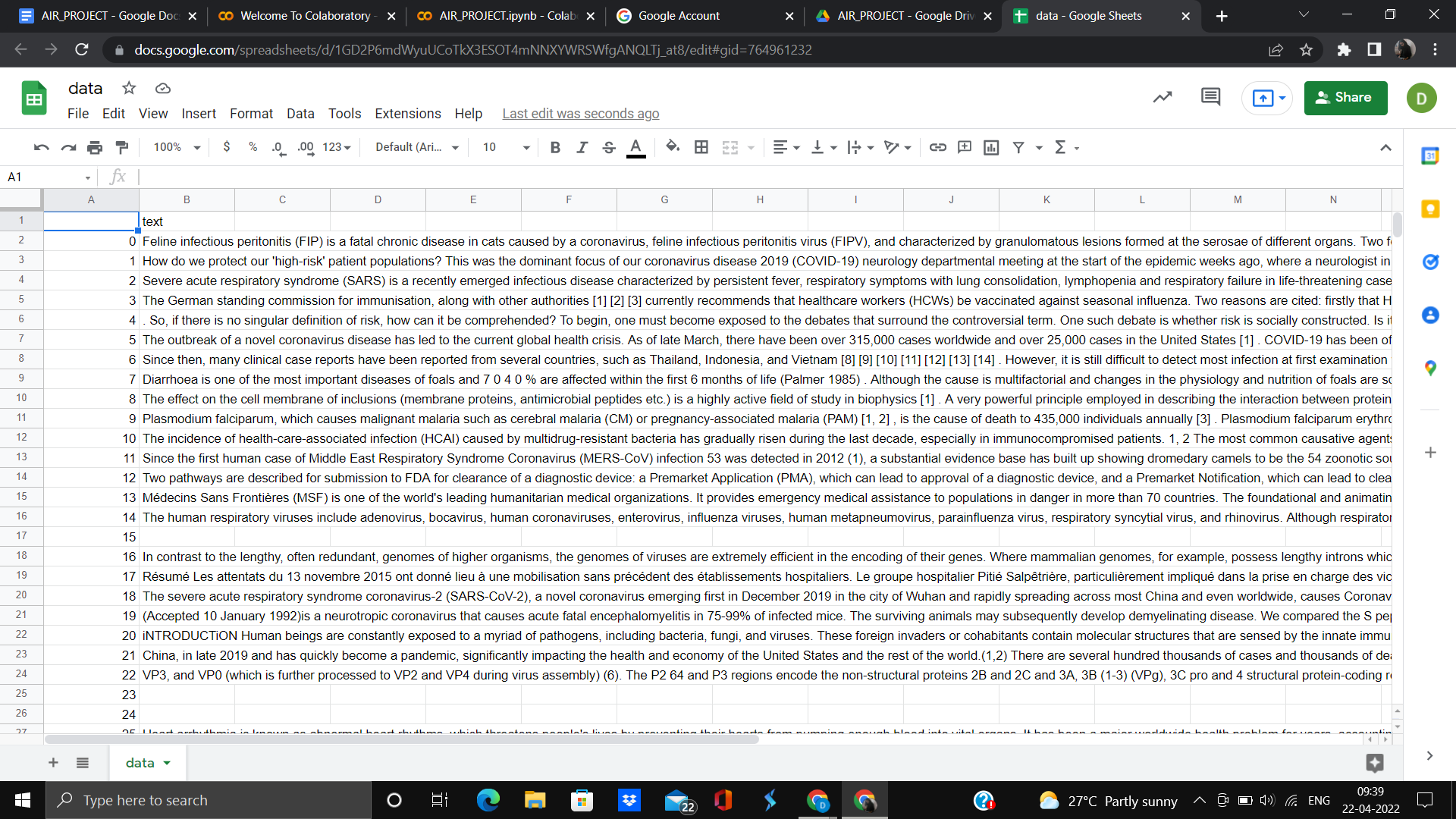
## ***4.2.3 Traverse through all the json files in the folder and apply the function to get the dataset. Export the data frame to a csv file.***



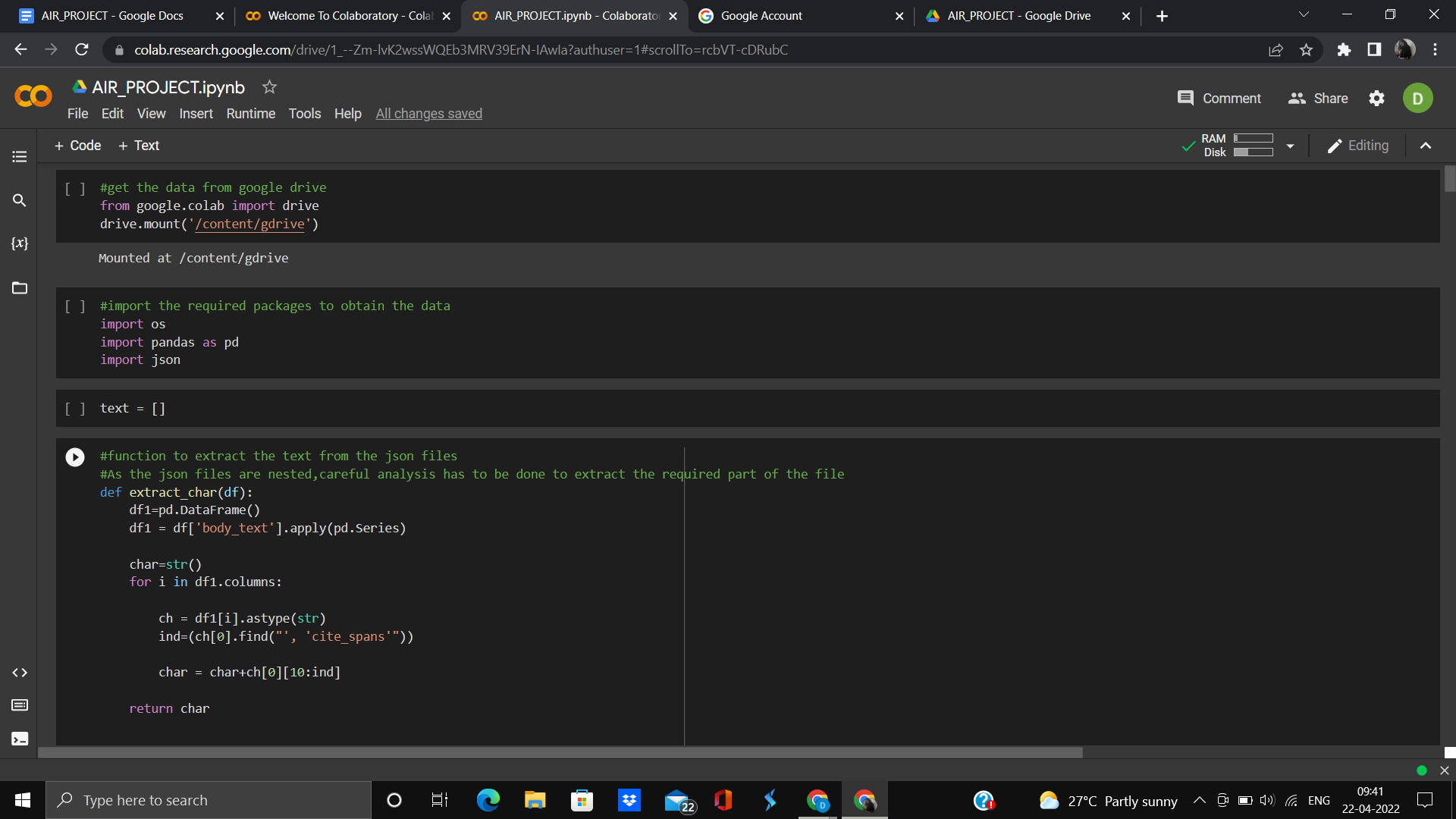
The directory of the json files

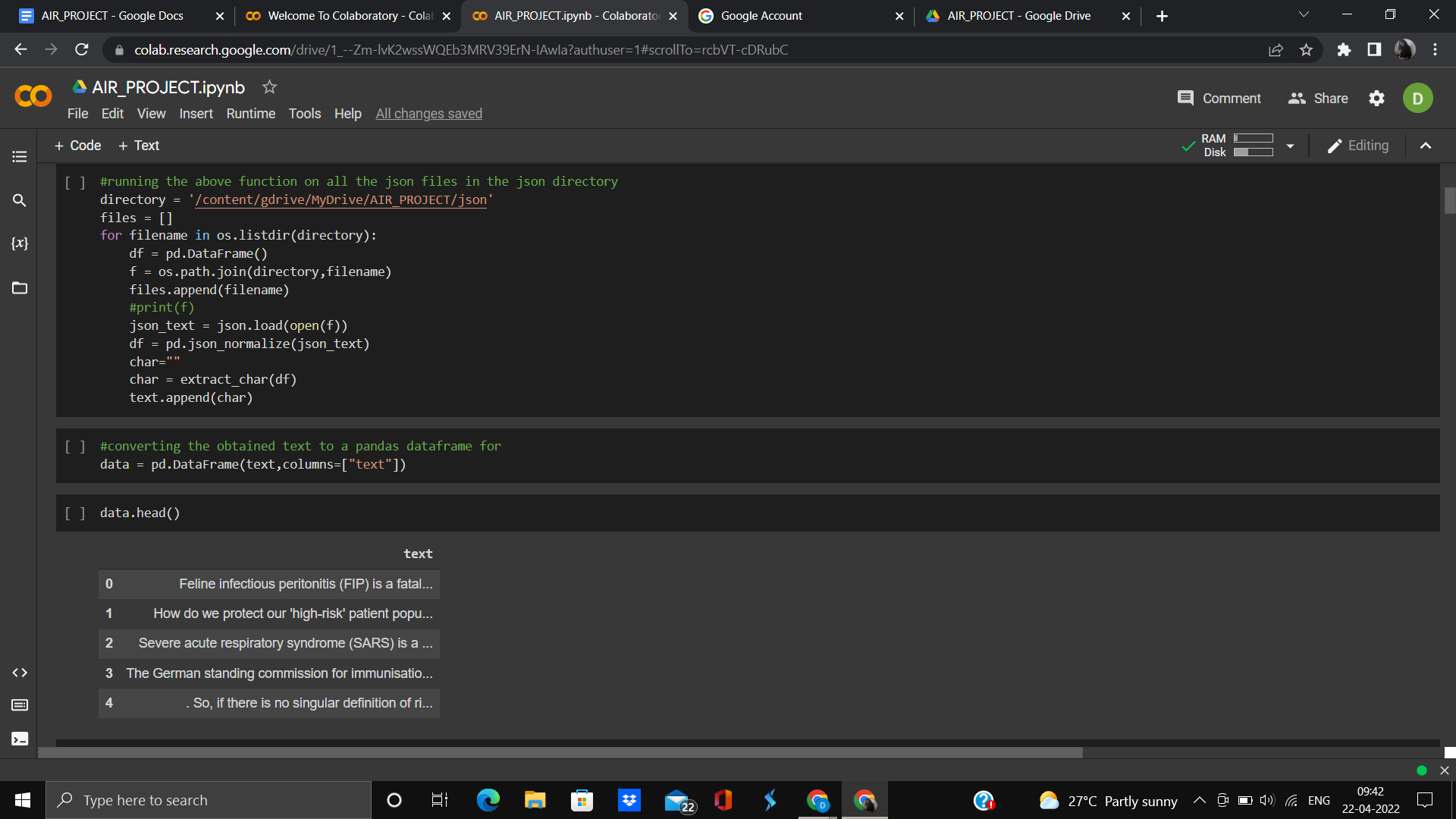


The json file



The csv file





Preprocessing the json files and storing the text in a dataframe

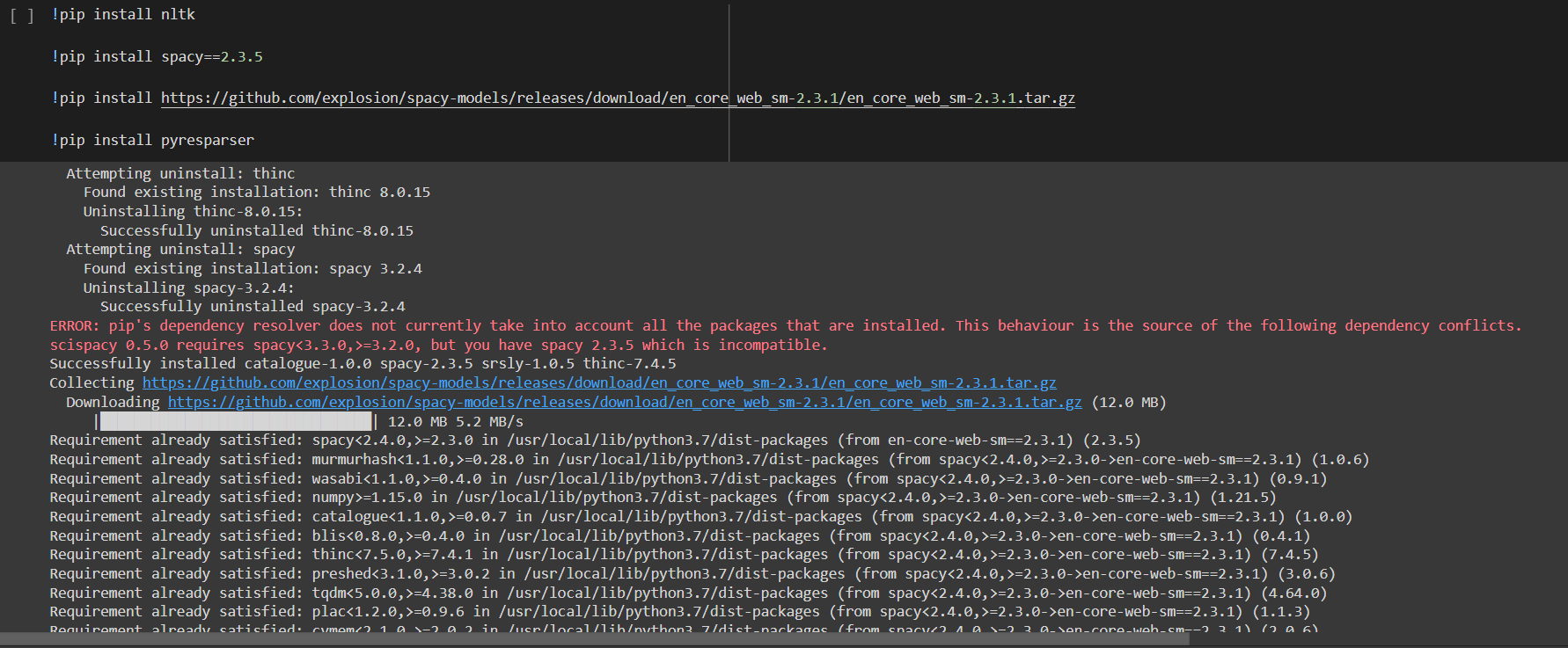
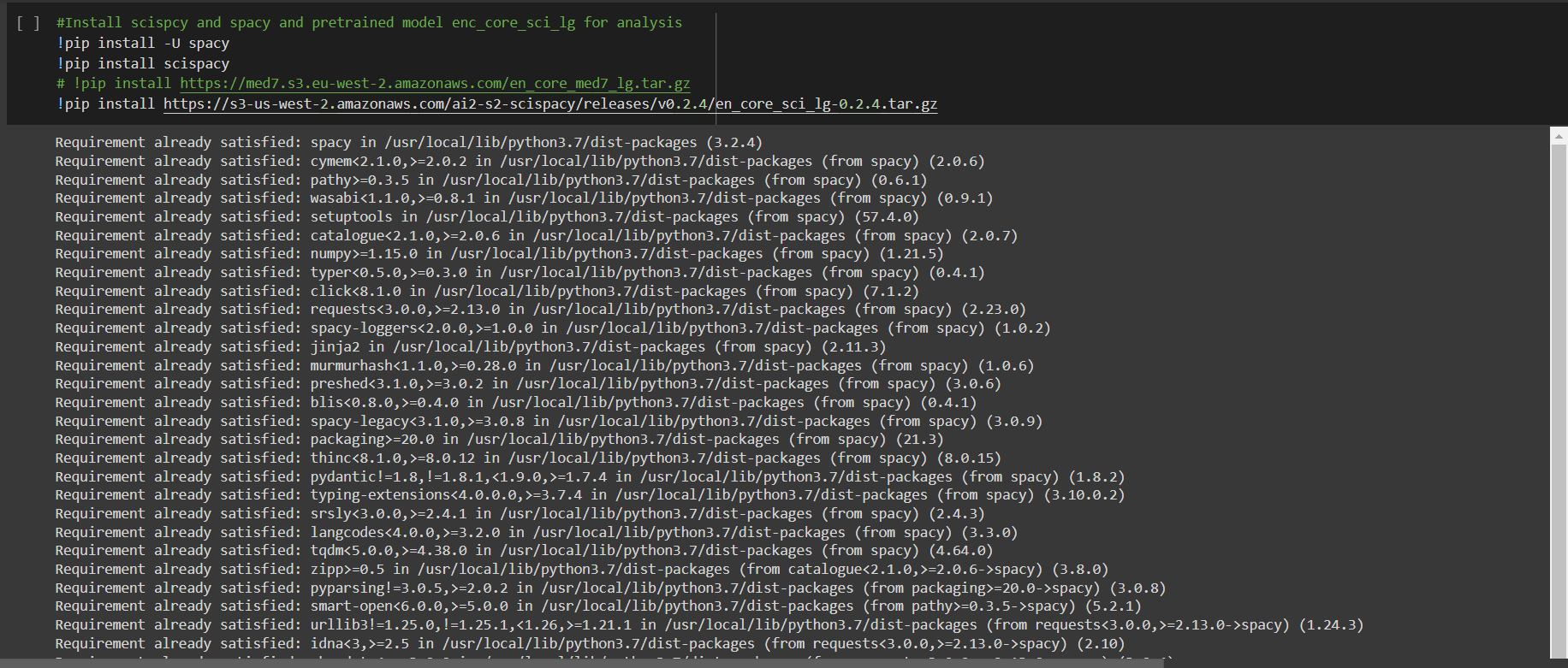
## ***4.3 EDA***

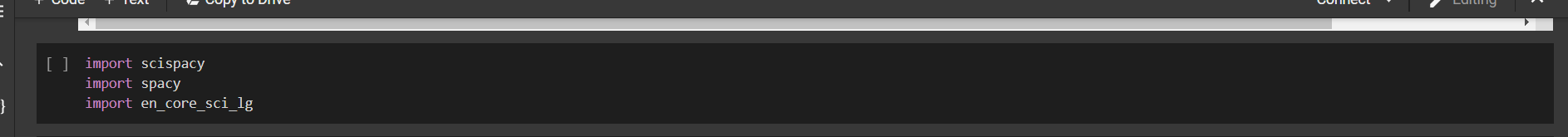
*Exploratory Data Analysis (EDA) is the process by which the data analyst becomes acquainted with their data in order to drive intuition. This process typically makes use of descriptive statistics and visualizations. Visually representing the content of a text document is one of the most important tasks in the field of text mining as a Data Scientist or NLP specialist. However, there are some gaps between visualizing unstructured (text) data and structured data.*

*It is a good practice to understand the data first and try to gather as many insights from it. EDA is all about making sense of data in hand, before getting them dirty with it.*

*Let’s install some required packages and import them. we will be using the scispacy for analysis.* ***scispaCy*** *is a Python package containing spaCy models for processing biomedical, scientific or clinical text.*

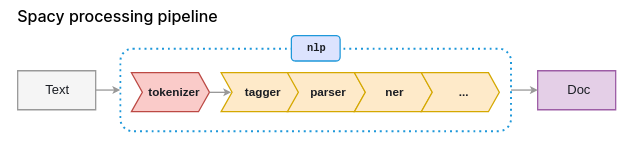
*import them as usual*

**

**

## ***Spacy Language Processing Pipeline***

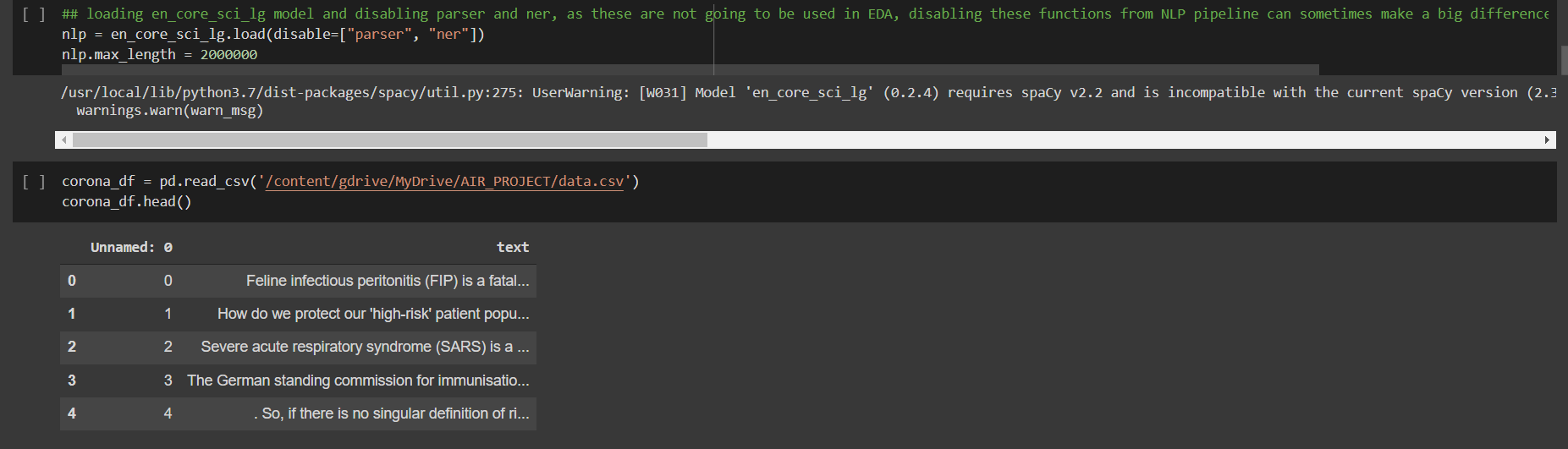
*When you call nlp on a text, spaCy first tokenizes the text to produce a Doc object. The Doc is then processed in several different steps – this is also referred to as the* ***processing pipeline****. The pipeline used by the* [*default models*](https://spacy.io/models) *consists of a tagger, a parser and an entity recognizer. Each pipeline component returns the processed Doc, which is then passed on to the next component.*

**

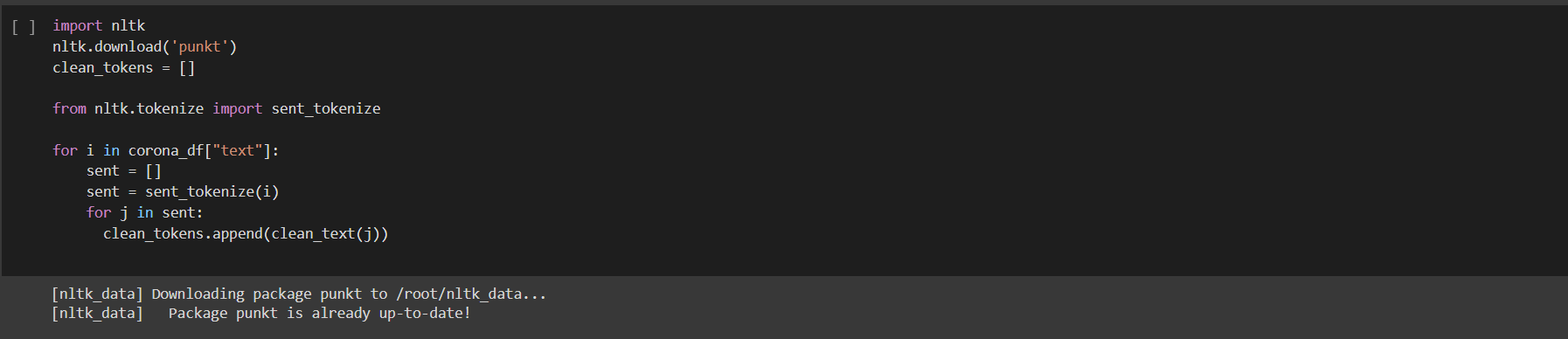
*Source:* [*https://spacy.io/usage/processing-pipelines/*](https://spacy.io/usage/processing-pipelines/)

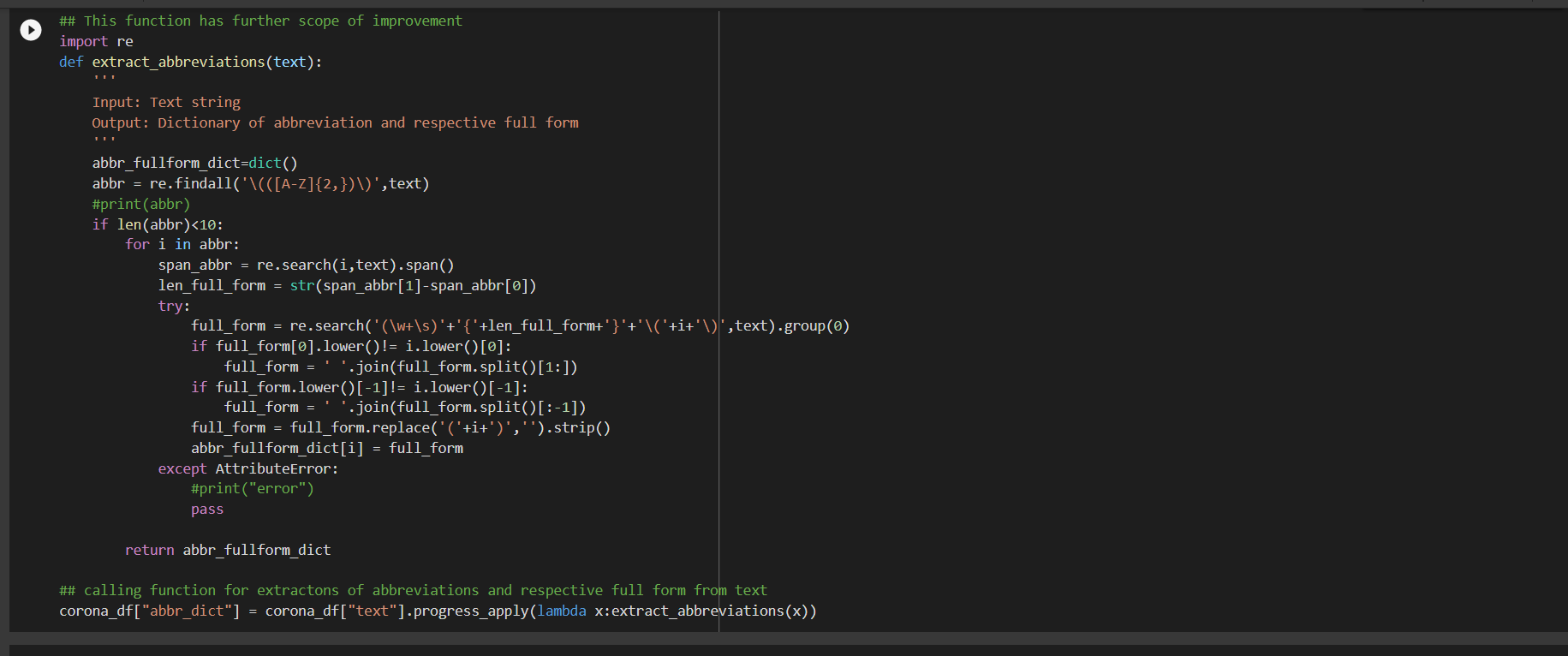
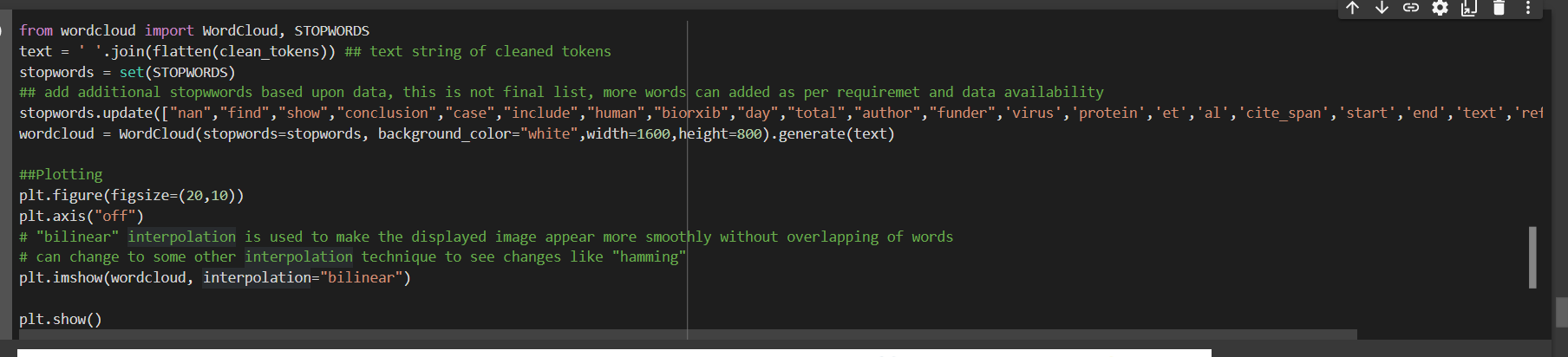
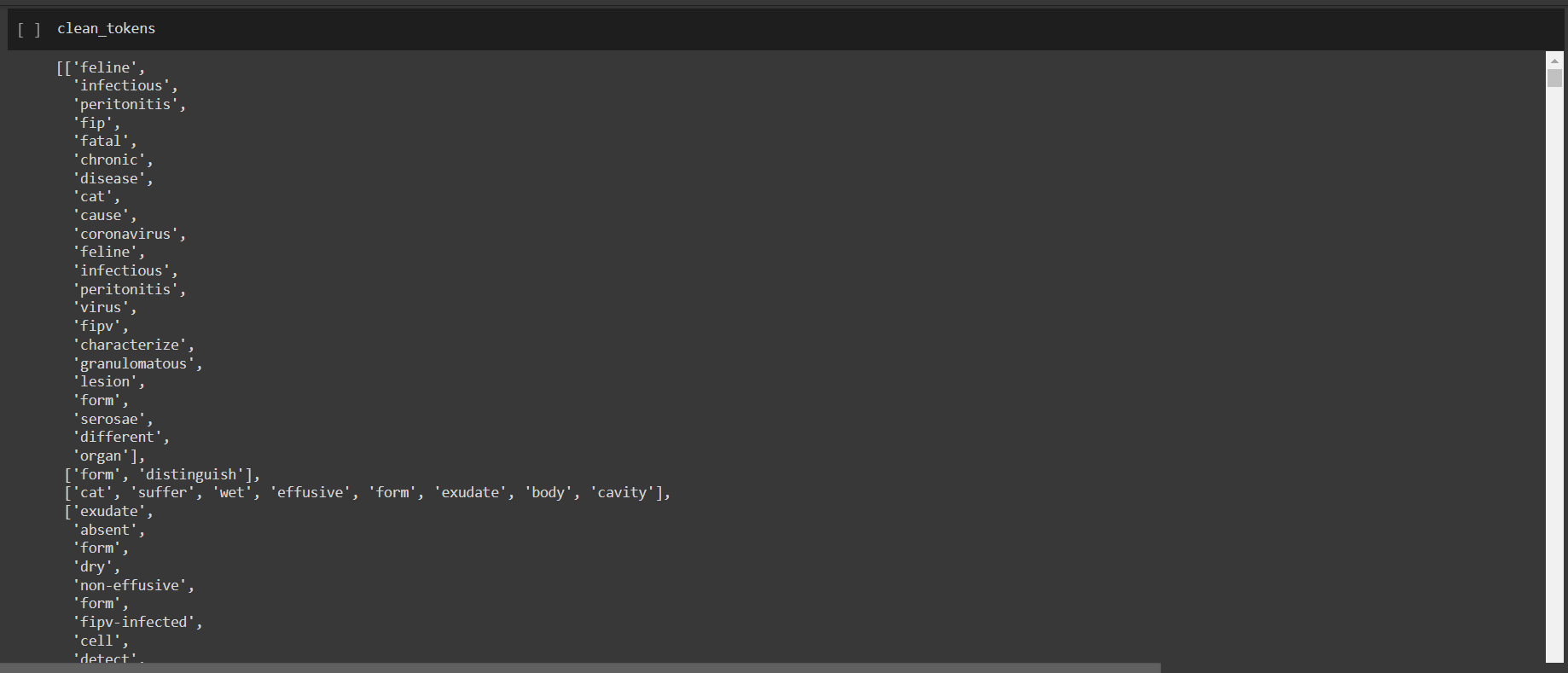
* ***tokenizer****: Segment text into tokens.*
* ***tagger****: Assign part-of-speech tags.*
* ***parser****: Assign dependency labels.*
* ***ner****: Detect and label named entities.*

*The processing pipeline always* ***depends on the statistical model*** *and its capabilities. For example, a pipeline can only include an entity recognizer component if the model includes data to make predictions of entity labels.*

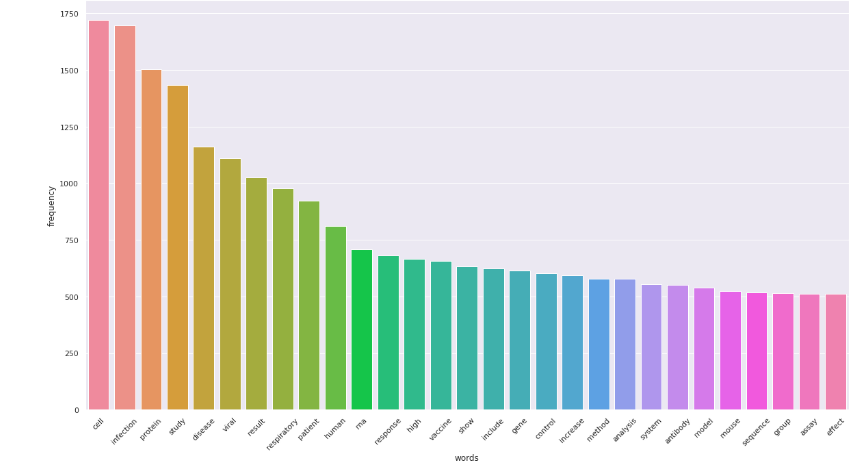
***NOTE****: you can create own pipeline components for further processing*

*now, create a function for cleaning data, this will remove all the stop words, punctuations, extra spaces, URLs, email, and currency. sometimes verbs, adverbs, pronouns are not required and can be removed. this function will return the base form of word which is known as lemmatization.*

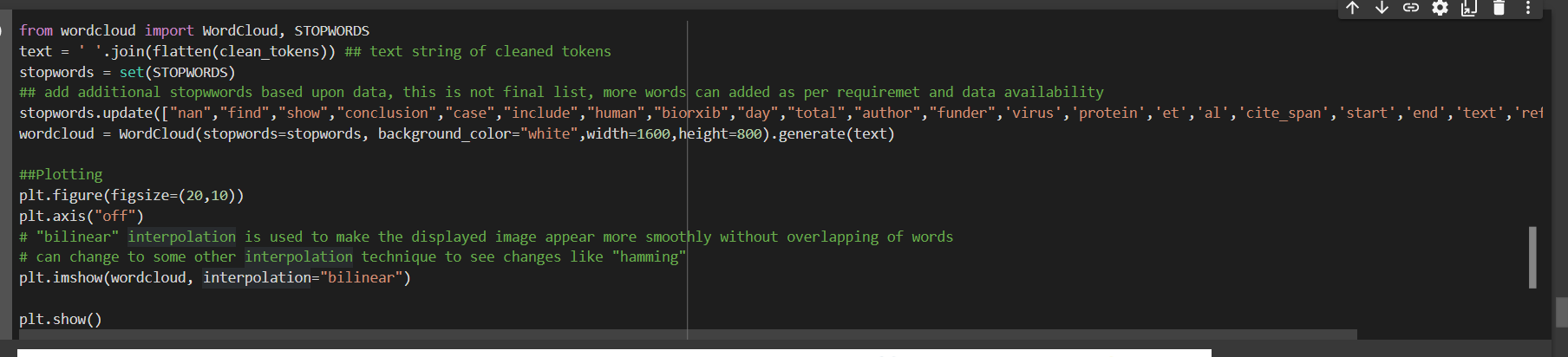
**

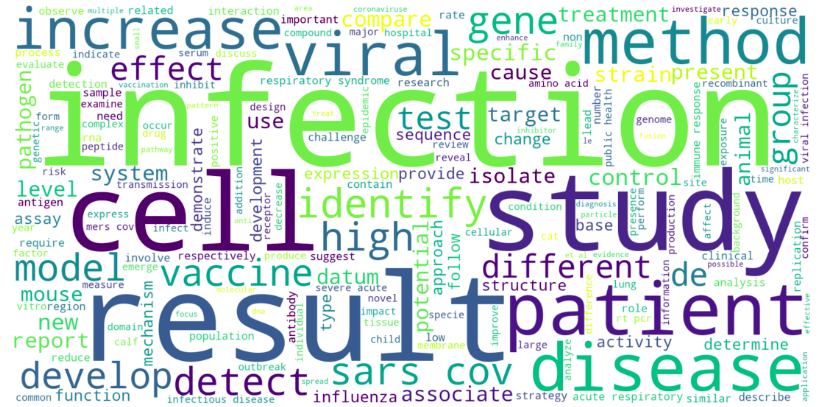
**

***4.3.1 Frequency graph***

********

## ***4.3.2 Wordcloud***

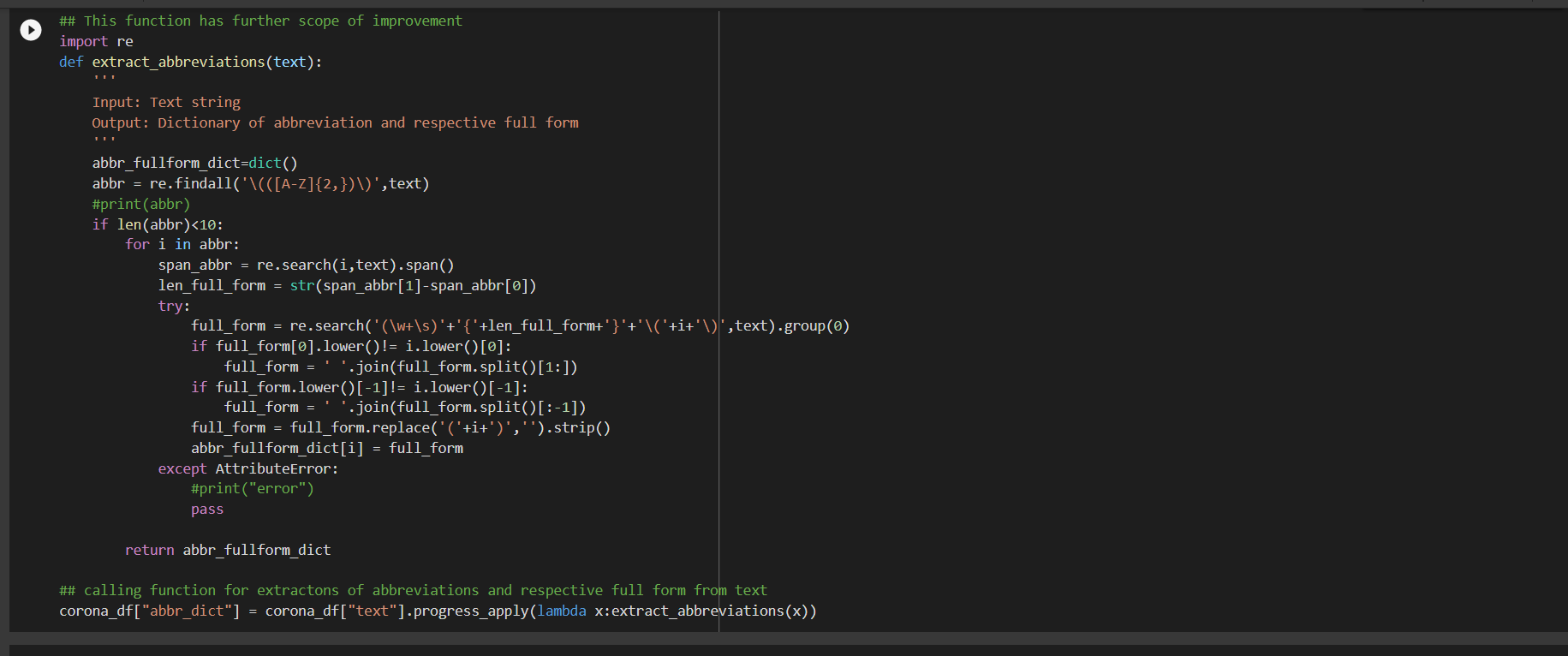


******

# ***4.4 Preprocessing***

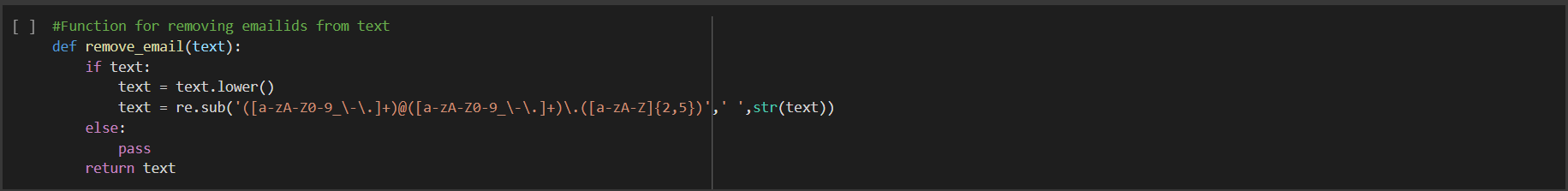
## ***4.4.1 Extract all abbreviations and full form from text***

*Text contains lots of abbreviations and full form, this code is used to extract all abbreviations and respective full forms within the text and put them in a dictionary where keys are abbreviations and values are full forms. Then, we need to create a function to replace all abbreviations with their respective full forms.*

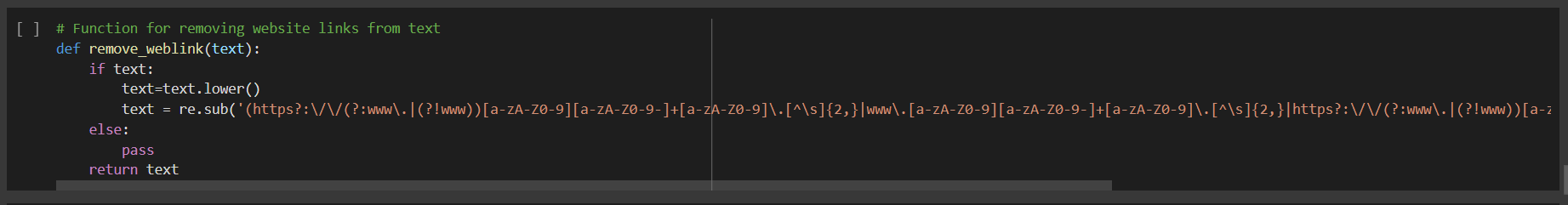
**

*similarily, we will create functions for removing email ids, weblink, paper references, extra spaces, unwanted characters, etc.*

## ***4.4.2 Remove Email Id***

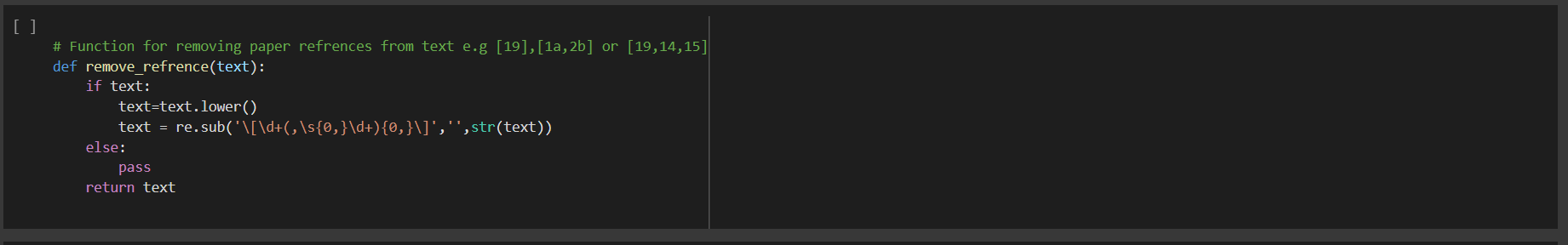


## ***4.4.3 Remove web-link***

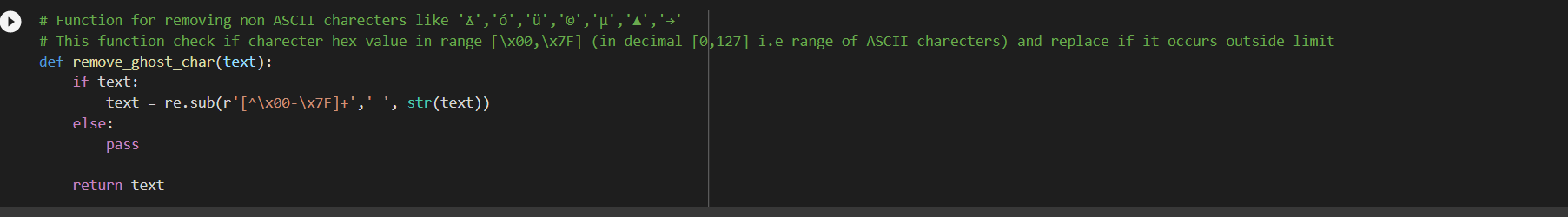


## 

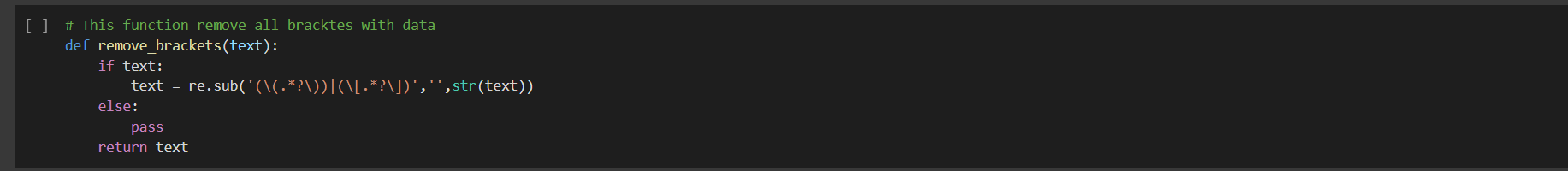
## ***4.4.4 Remove References***



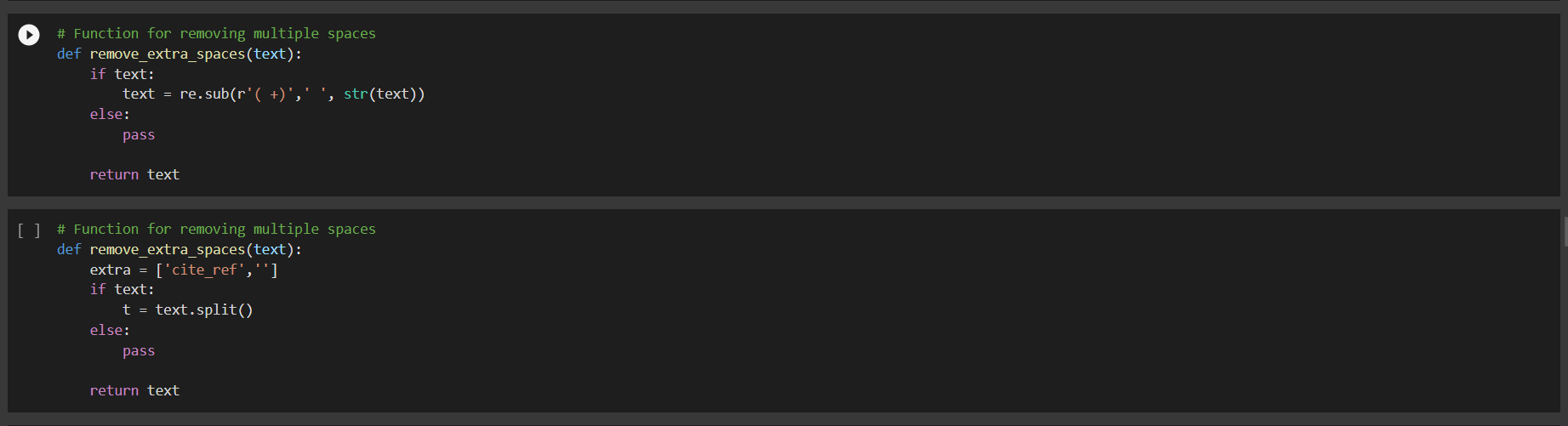
## ***4.4.5 Remove Ghost characters***



## ***4.4.6 Remove Brackets***



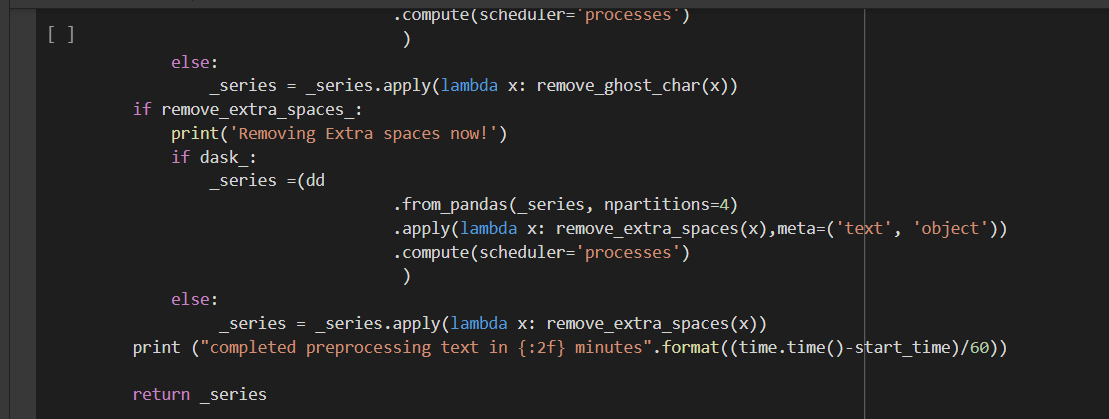
## ***4.4.7 Remove Extra spaces***



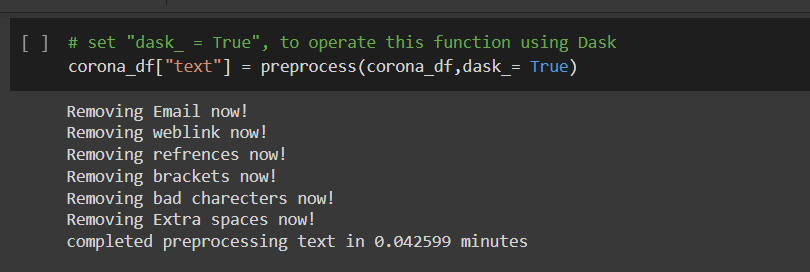
## ***4.4.8 Implementation of Dask for fast processing and Better utilization of CPU***

*here, we’re going to use* [*Dask*](https://dask.org/) *for processing, Dask is a flexible library for parallel computing in python, provides multi-core execution on larger-than-memory datasets.*

“Dask provides advanced parallelism for analytics, enabling performance at scale for the tools you love” — [*https://dask.pydata.org/en/latest/*](https://dask.pydata.org/en/latest/)

**

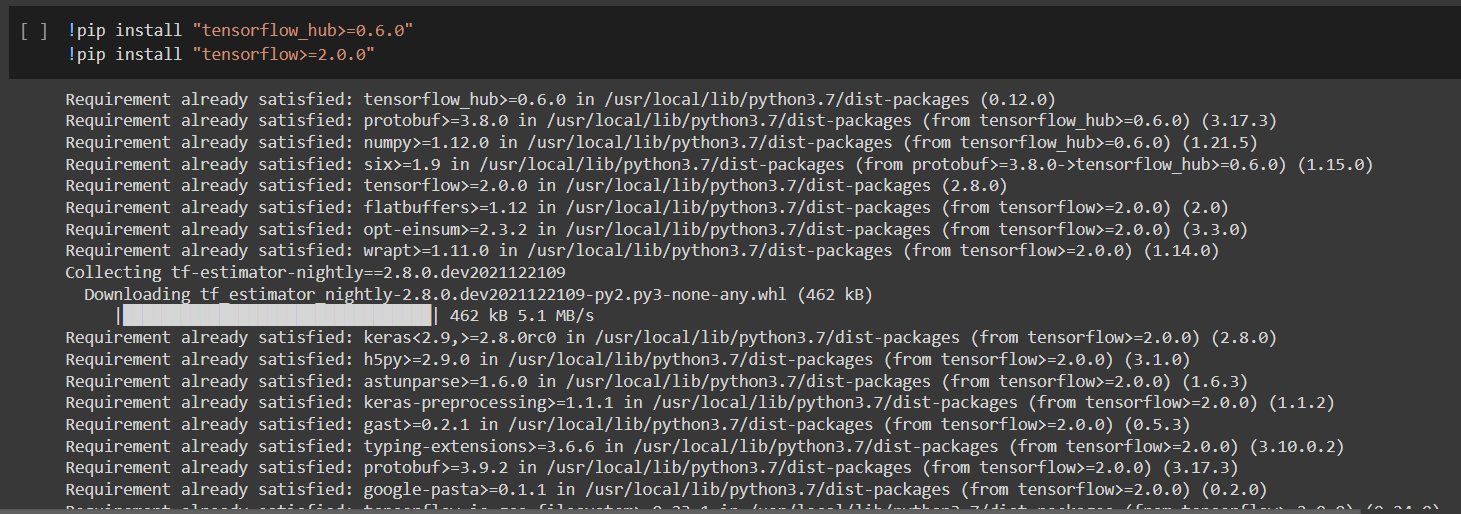
*now, call above function to preprocess text*

**

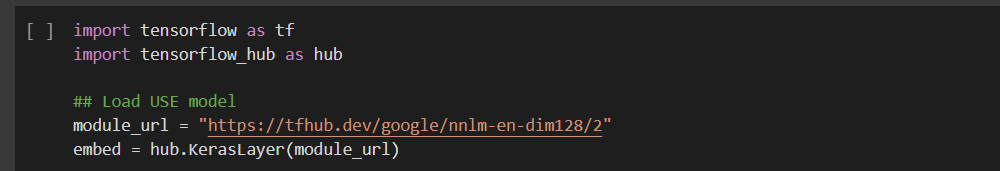
# ***4.5 Word embeddings using USE***

*The model is trained and optimized for greater-than-word length text, such as sentences, phrases or short paragraphs. It is trained on a variety of data sources and a variety of tasks with the aim of dynamically accommodating a wide variety of natural language understanding tasks. The input is variable length English text and the output is a* ***128-dimensional*** *vector. The* [*universal-sentence-encoder*](https://ai.googleblog.com/2019/07/multilingual-universal-sentence-encoder.html) *model has trained with a* ***deep averaging network*** *(DAN) encoder.*

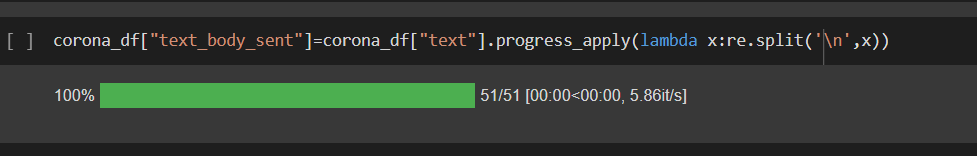
*let's install required packages for USE*

**

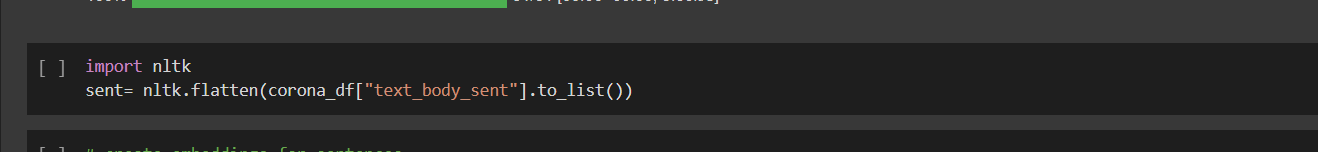
*load model from its URL*

**

*now, split text body of research papers by new line into list separate paragraphs, so that we can create embeddings for each paragraph. for example, if a text has 100 paragraphs, then we will get a list of these 100 elements.*

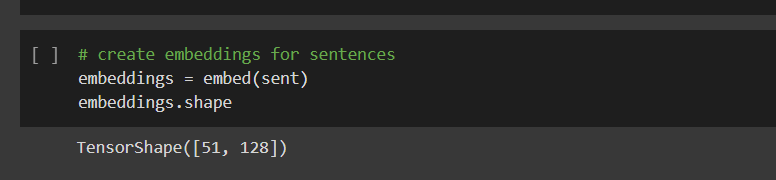
**

*Convert the different list of paragraphs for each row of text into a single list*

**

# ***4.5.1 Create embeddings(Deep-Learning)***

*now create embeddings*

**

*this will return a tensor object of shape nx128, where n is the number of paragraphs or sentences, as number of sentences are 1052935, therefore the shape of tensor will be 1052935x128*

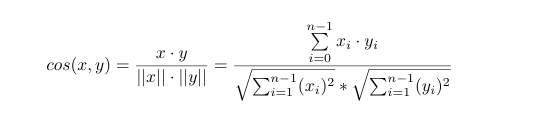
*TensorShape([1052935, 128])*

# ***4.6 cosine similarity***

[***Cosine similarity***](https://www.sciencedirect.com/topics/computer-science/cosine-similarity) *is a measure of similarity that can be used to compare documents or, say, give a ranking of documents with respect to a given vector of query words.*

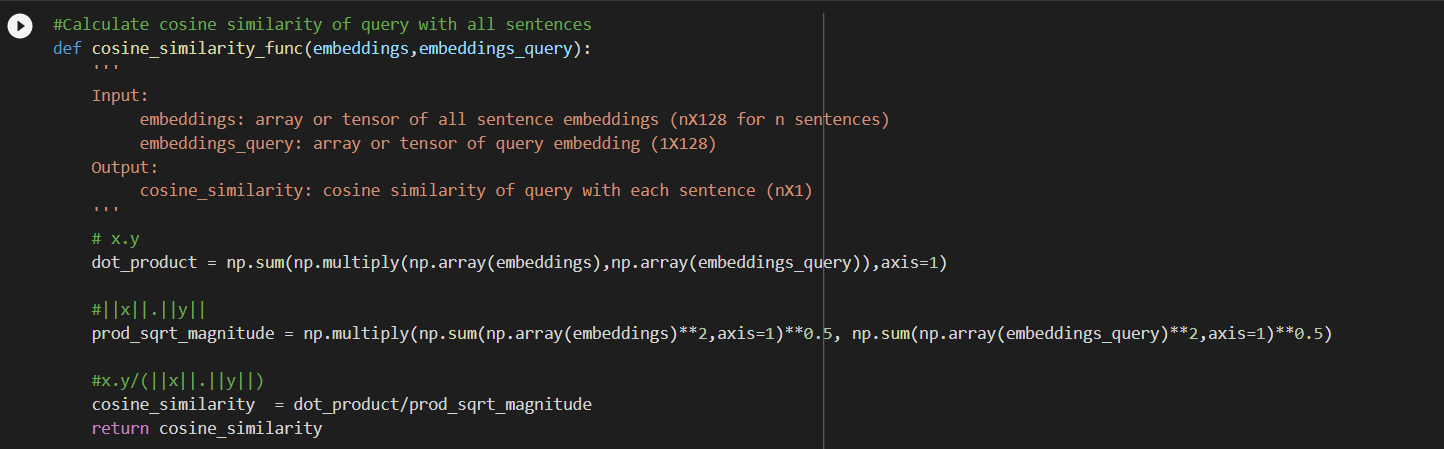
*Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. In this context, the two vectors I am talking about are arrays containing the embeddings of two documents.*

*Let* ***x*** *and* ***y*** *be two vectors for comparison. Using the cosine measure as a similarity function, we have*

**

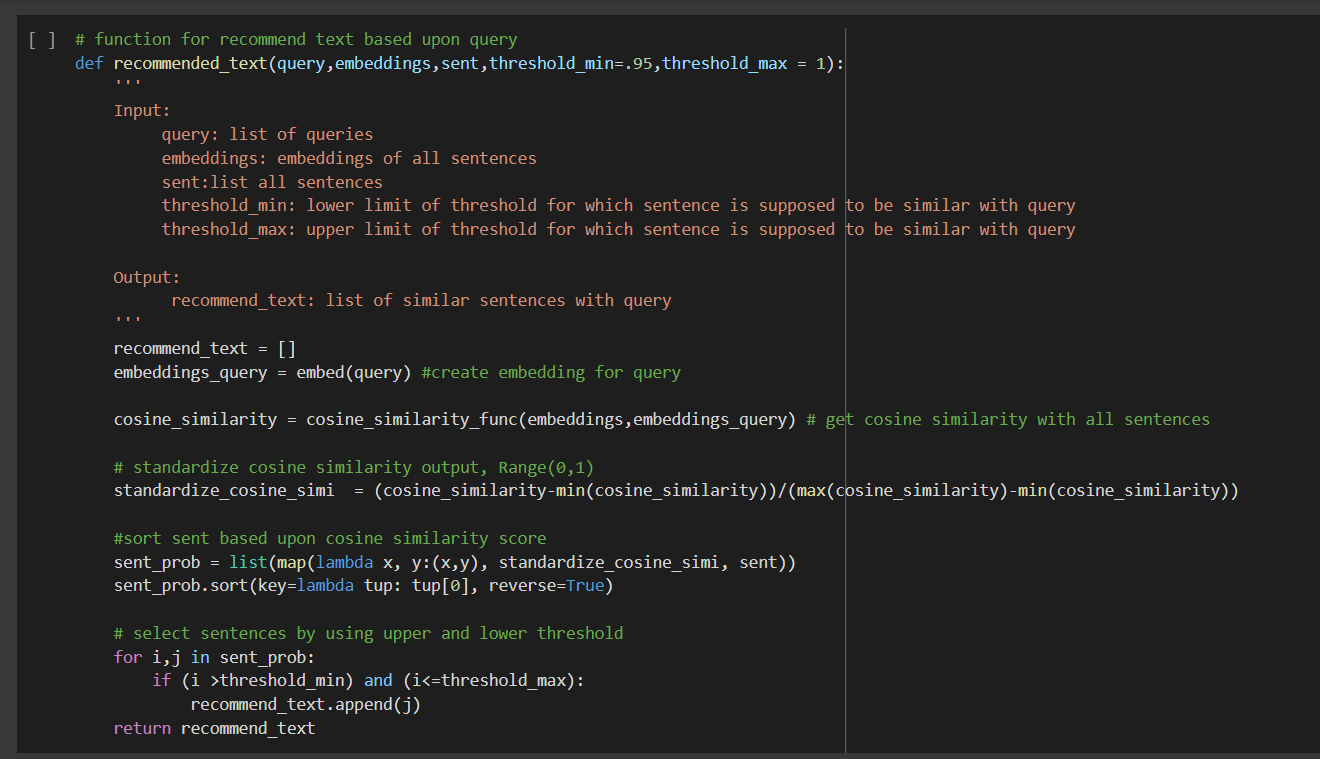
*where ||****x****|| is the* [*Euclidean norm*](https://www.sciencedirect.com/topics/computer-science/euclidean-norm) *of vector x=(x1,x2,…,xn), defined as x1^2+x2^2+⋯+xn^2. Conceptually, it is the length of the vector. Similarly, ||****y****|| is the Euclidean norm of vector* ***y****. The measure computes the cosine of the angle between vectors* ***x*** *and* ***y****. A cosine value of 0 means that the two vectors are at 90 degrees to each other (orthogonal) and have no match. The closer the cosine value to 1, the smaller the angle, and the greater the match between vectors.*

*The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance because of the size, they could still have a smaller angle between them. Smaller the angle, higher the similarity.*

**

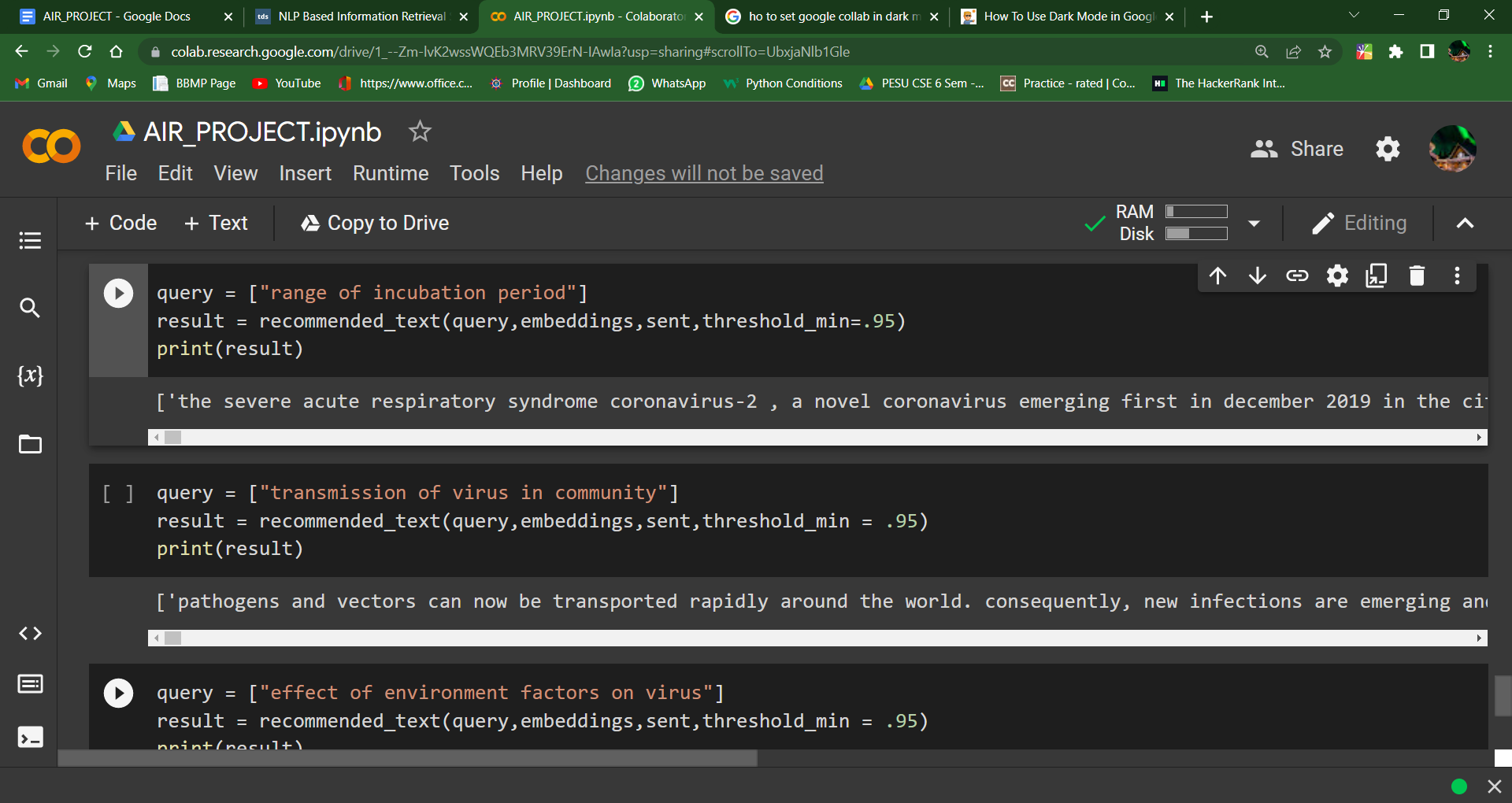
# ***4.7 Recommend text***

*This function extracts information from corpus, based upon query. it calculates cosine similarity of query with all paragraphs or sentences, standardize it to fix its range b/w 0 to 1, and return sentences based upon threshold.*

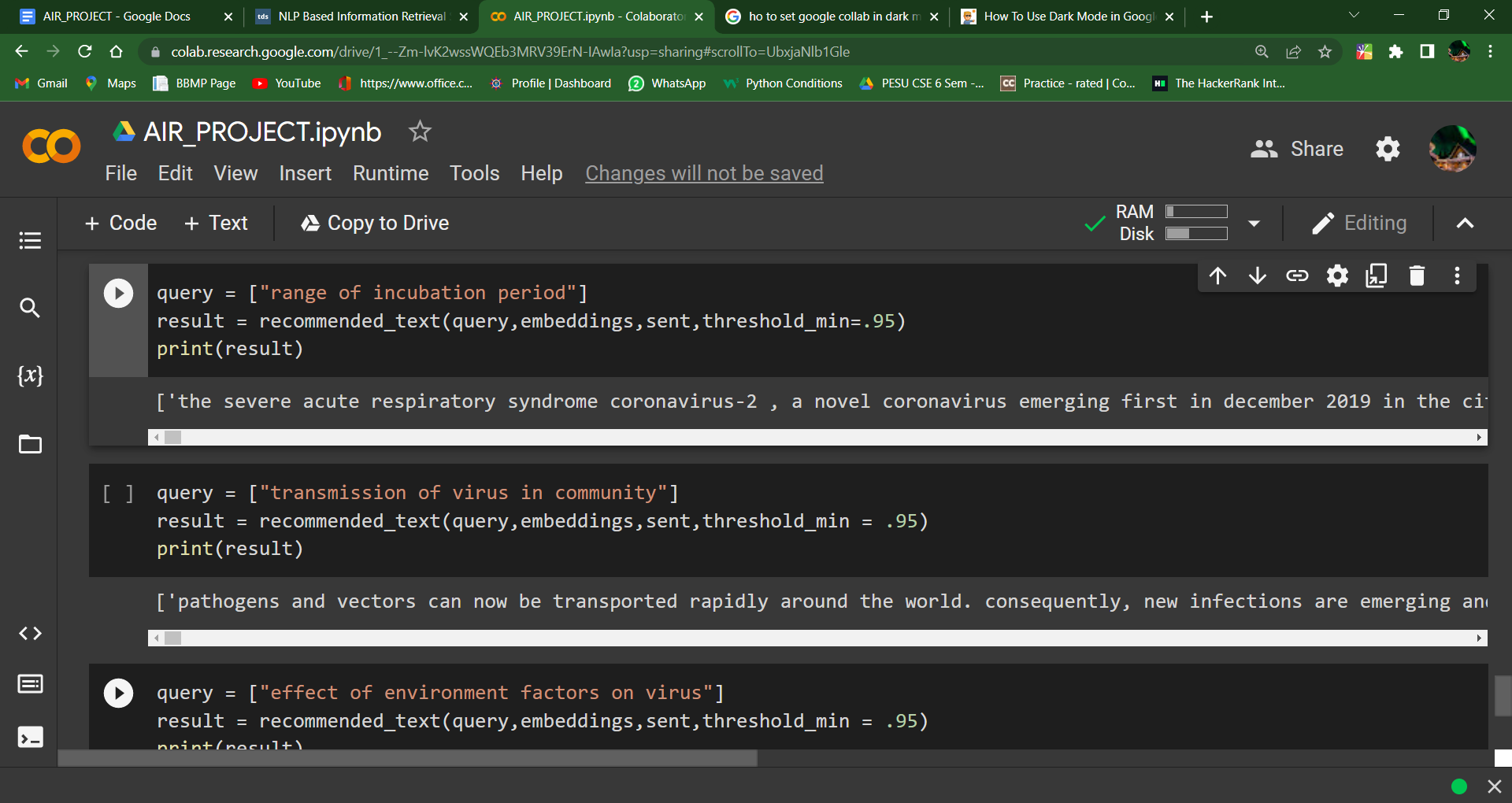
**

# ***4.8 Task Results***

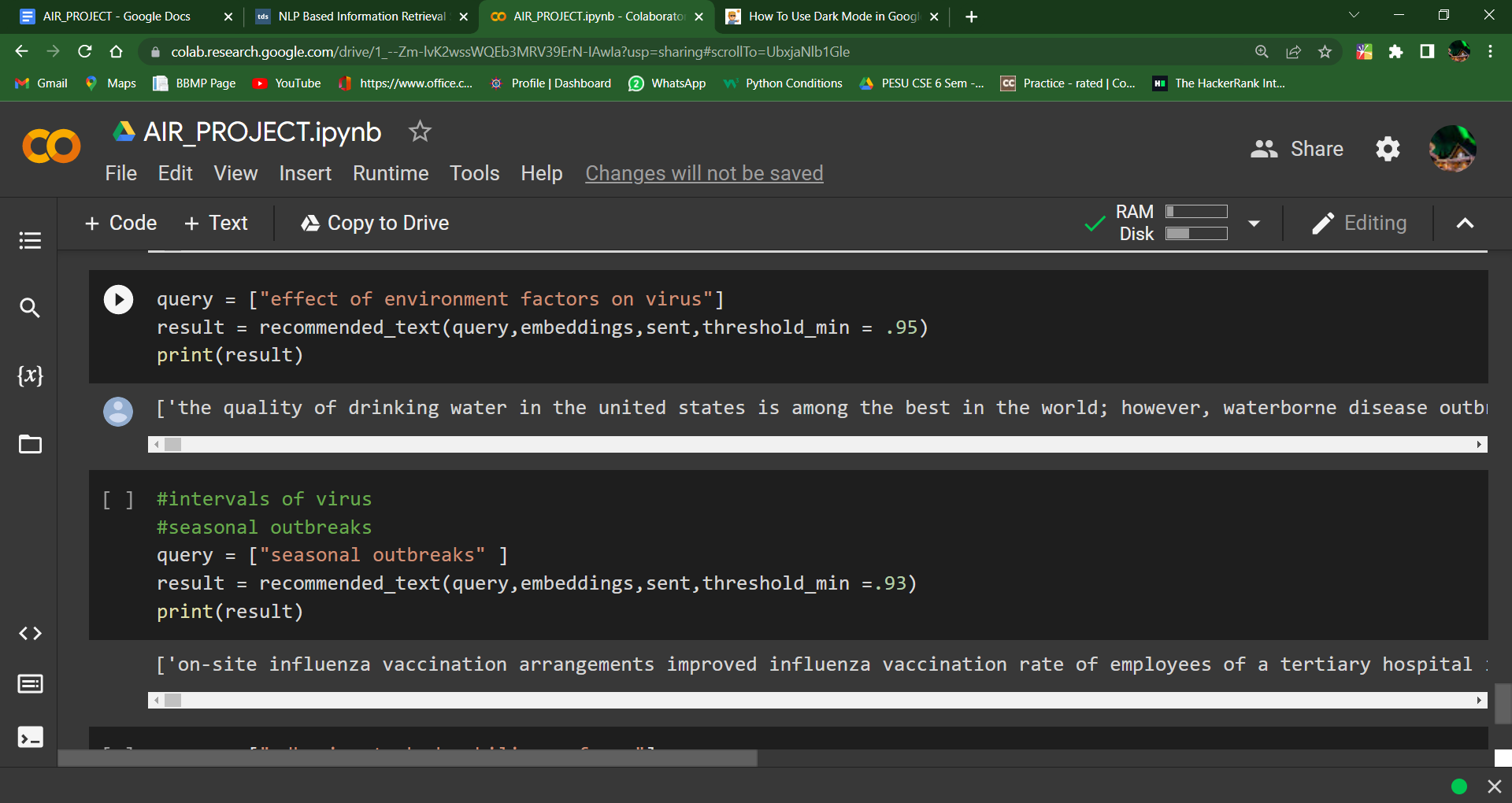
***4.8.1 Incubation period***



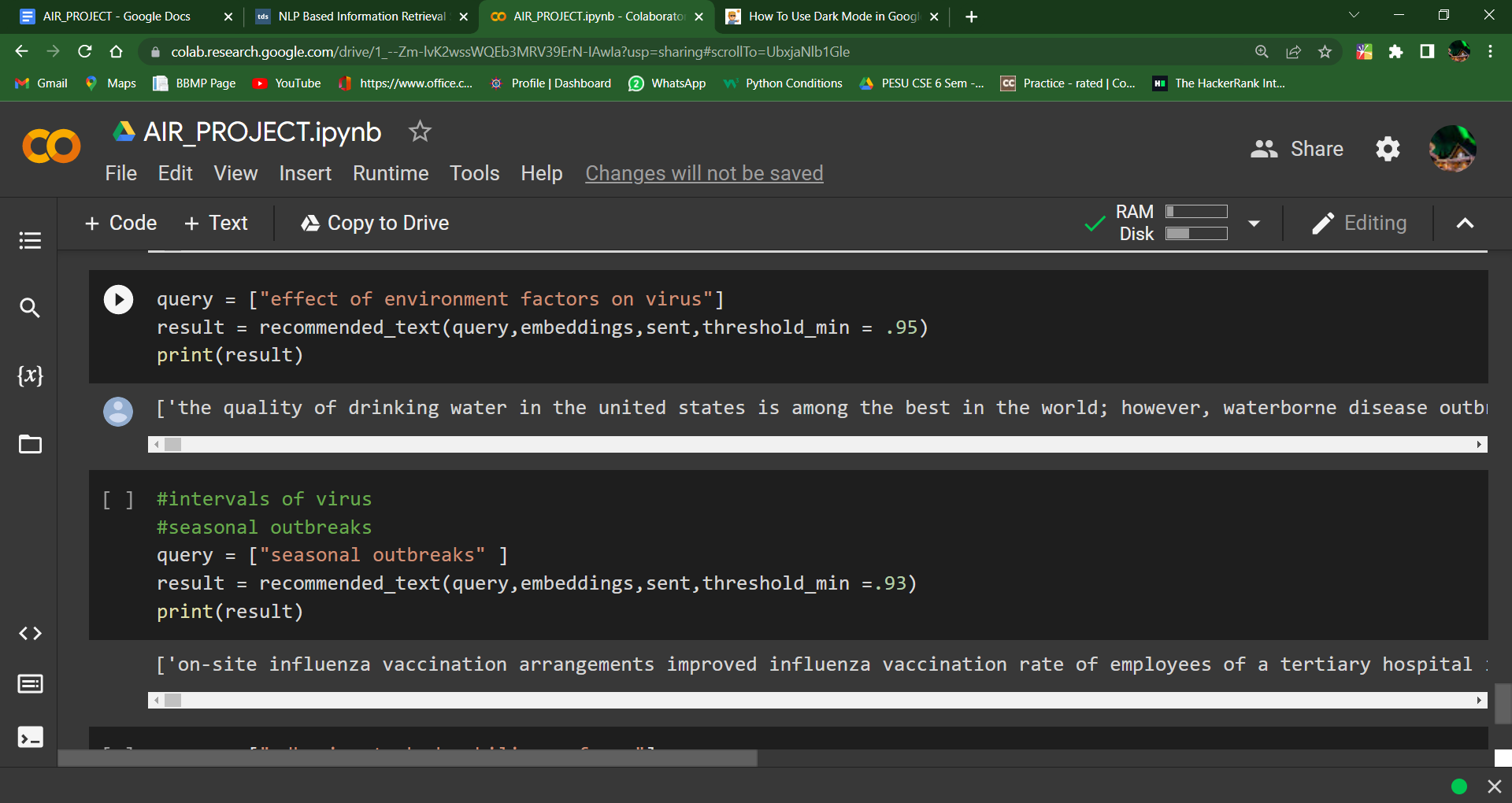
## ***4.8.2 Transmission***



***4.8.3 Role of environment in Transmission***

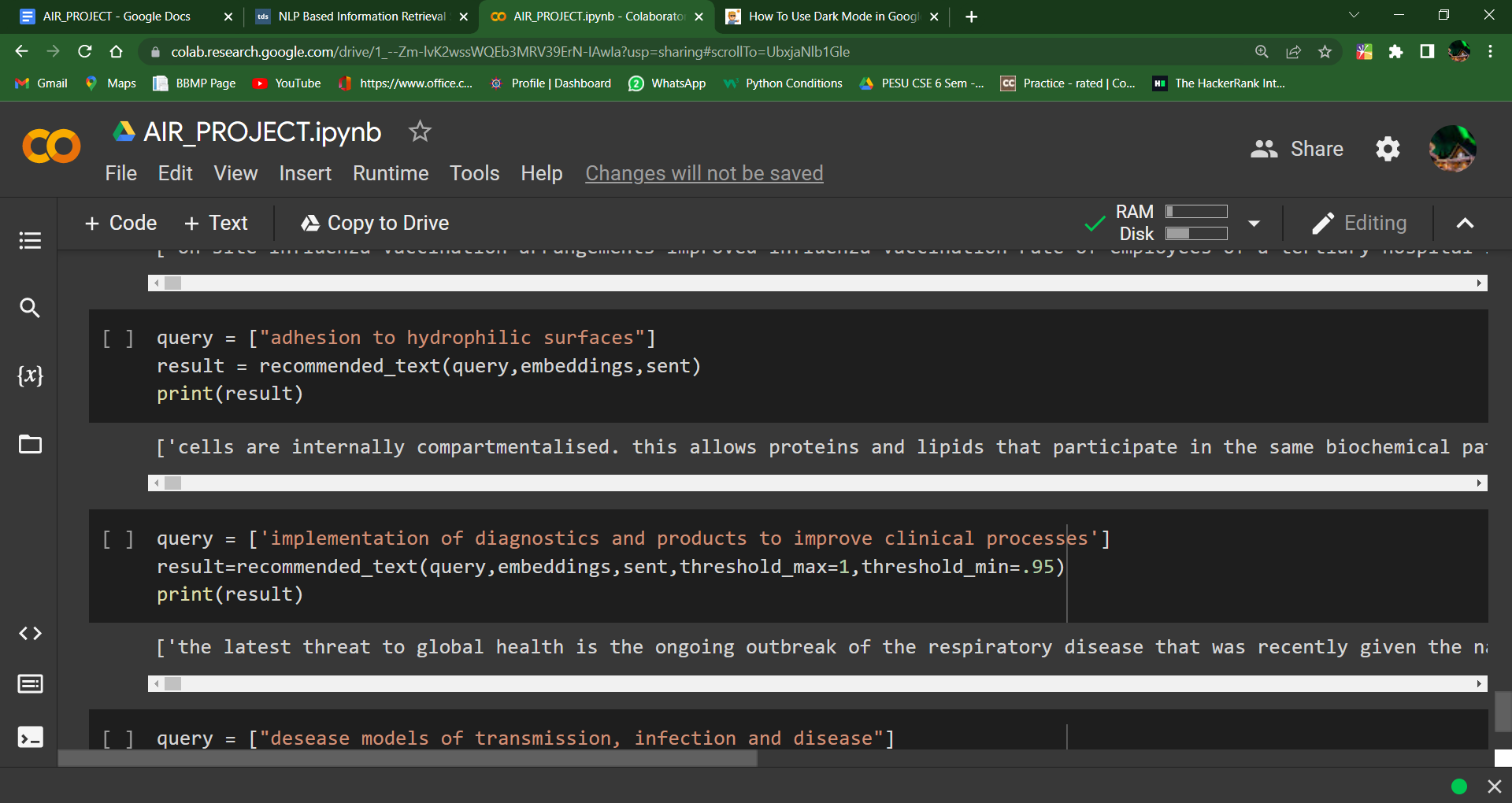


## ***4.8.4 Seasonality of transmission***

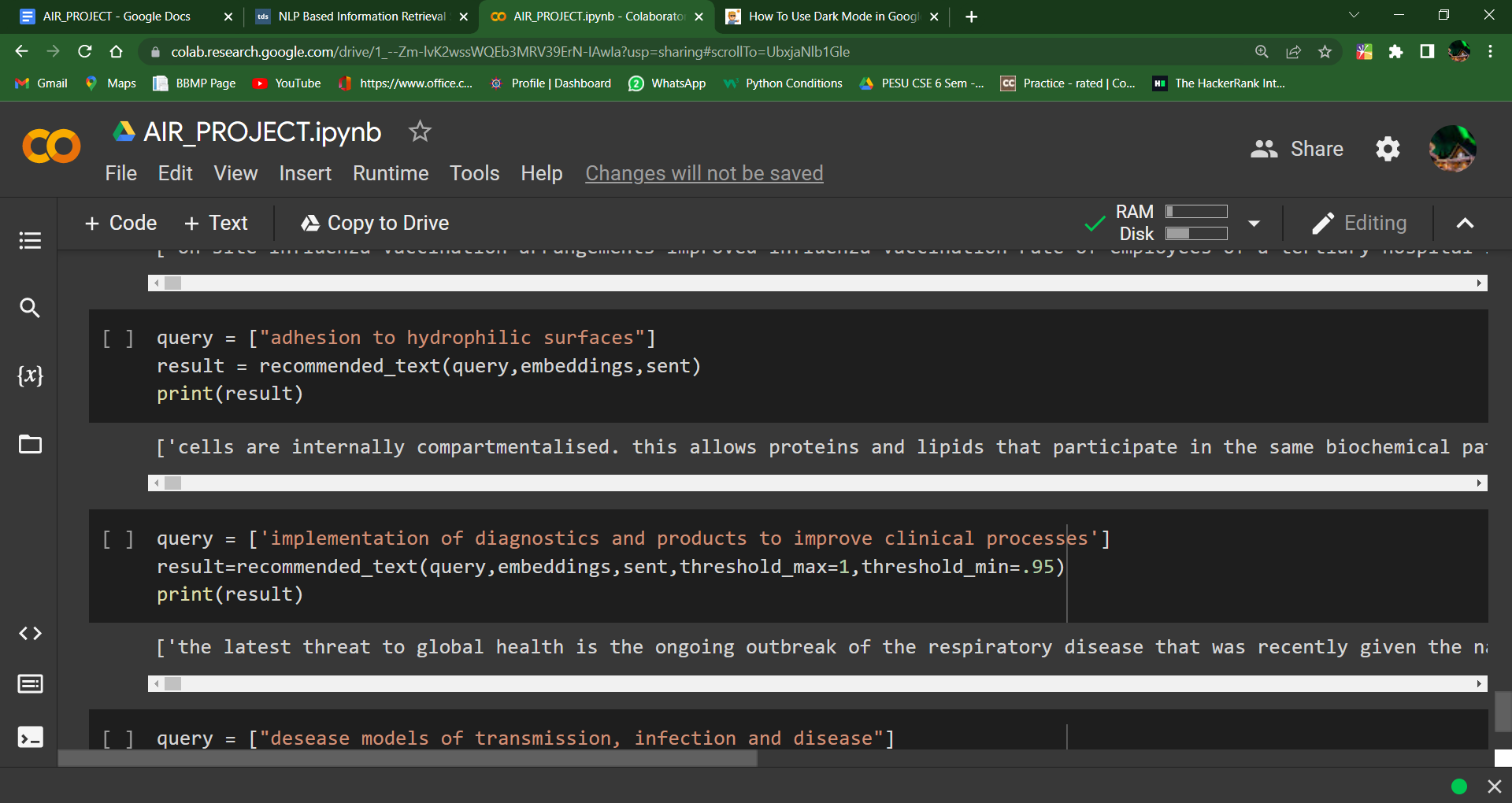


## ***4.8.5 The physical science of the coronavirus***

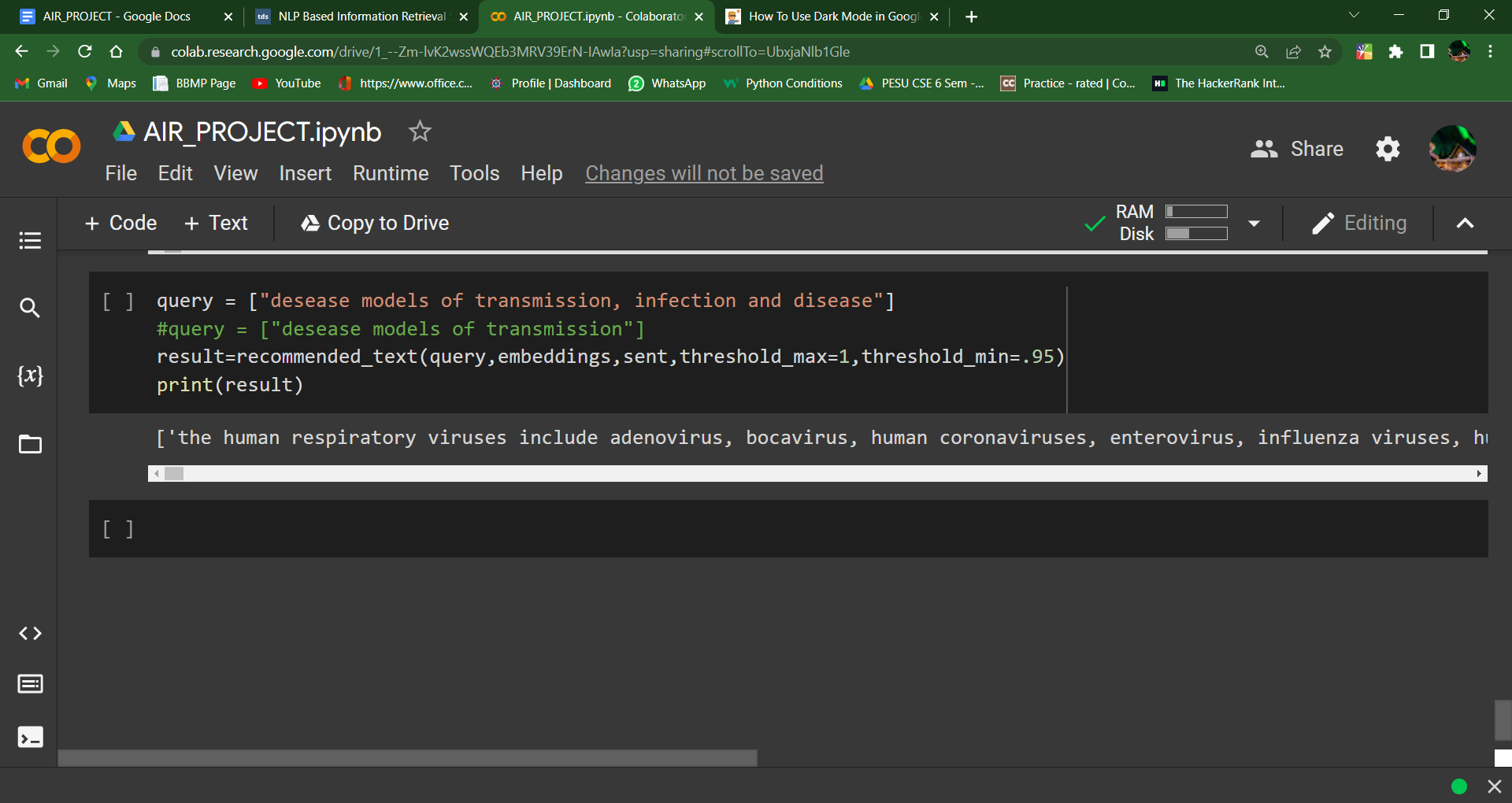
*(e.g., charge distribution, adhesion to hydrophilic/phobic surfaces, environmental survival to inform decontamination efforts for affected areas and provide information about viral shedding).*



## ***4.8.6 Implementation of diagnostics and products to improve clinical processes***



## ***4.8.7 Disease models, including animal models for infection, disease and transmission***

**