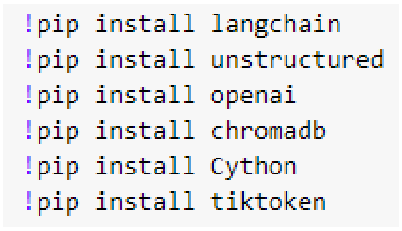
RAG-BASED CHAT APPLICATION

STEP1:

Installing the Packages



```
!pip install langchain
!pip install unstructured
!pip install openai
!pip install chromadb
!pip install Cython
!pip install tiktoken
```

Langchain:

LangChain is an open source framework for developing applications powered by language models and that allows AI developers to combine Large Language Models (LLMs) like GPT-4 with external data. In the Langchain package by uploading the data we ask the questions, but without uploading the data we cannot answer any questions.

Unstructured:

The Unstructured package refers to a collection of code files or modules, It is used for the data where the lines are not fixed width, or they are just HTML, image or pdf files. Such data is known as unstructured data.

Openai:

OpenAI is a research organization that aims to create artificial intelligence in a safe and beneficial way.It used to create a model,They offer an API that allows developers to access their cutting-edge models and use them in their own applications.

Cython:

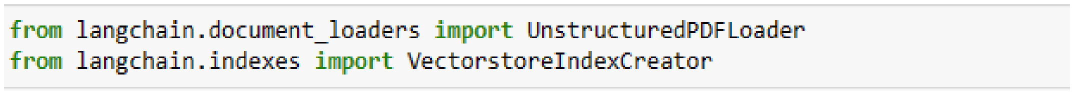
Cython is a programming language that allows you to write C-like code with Python-like syntax. It is designed to bridge the gap between Python's ease of use and C/C++'s performance.

tiktoken:

tiktoken is a fast BPE (Byte pair encoding) tokenizer for use with OpenAI's models.Splitting text strings into tokens is useful because GPT models see text in the form of tokens.

STEP2:

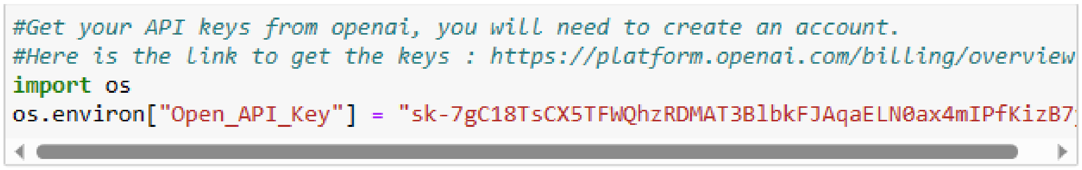
Loading the required packages



```
from langchain.document_loaders import UnstructuredPDFLoader
from langchain.indexes import VectorstoreIndexCreator
```

STEP3:

Creating our API keys from openai

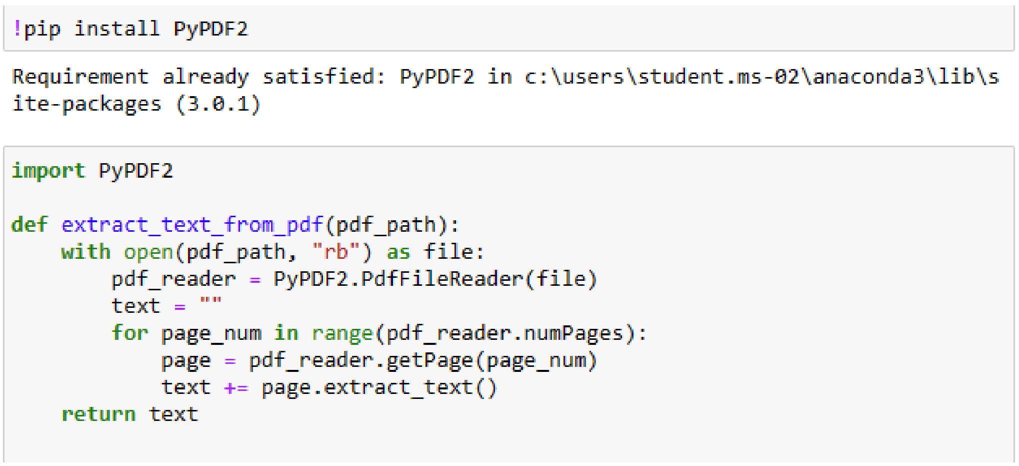


```
#Get your API keys from openai, you will need to create an account.
#Here is the link to get the keys : https://platform.openai.com/billing/overview
import os
os.environ["Open_API_Key"] = "sk-7gC18TsCX5TFWQhzRDMAT3BlbkFJAqaELN0ax4mIPfKizB7
```

STEP4:

To extract the text from multiple PDF files and save the text content into separate text files, you can use the PyPDF2 .

These library is used to read the PDF files and the built-in Python file I/O functions to write the text content into individual text files.



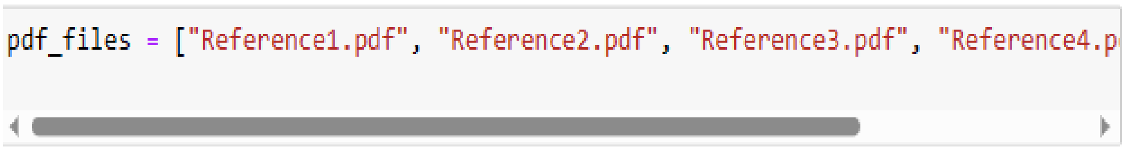
```
!pip install PyPDF2
```

```
Requirement already satisfied: PyPDF2 in c:\users\student.ms-02\anaconda3\lib\s
ite-packages (3.0.1)
```

```python
import PyPDF2

def extract_text_from_pdf(pdf_path):
    with open(pdf_path, "rb") as file:
        pdf_reader = PyPDF2.PdfFileReader(file)
        text = ""
        for page_num in range(pdf_reader.numPages):
            page = pdf_reader.getPage(page_num)
            text += page.extract_text()
    return text
```

STEP5:

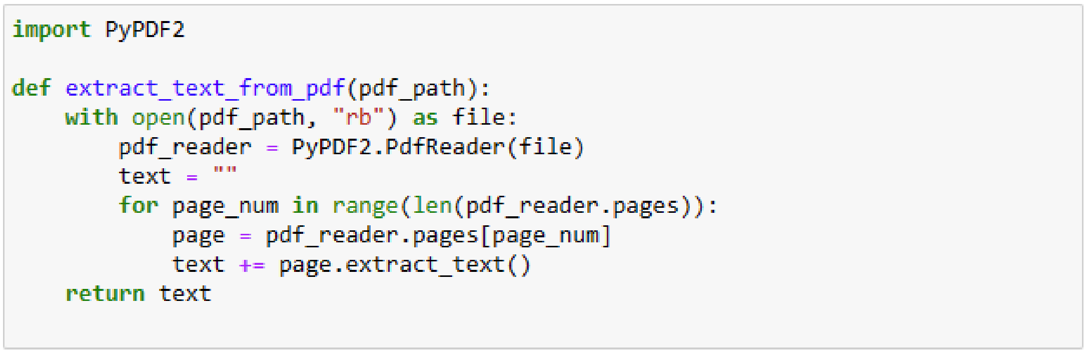
Defining a list of PDF files to extract the text.



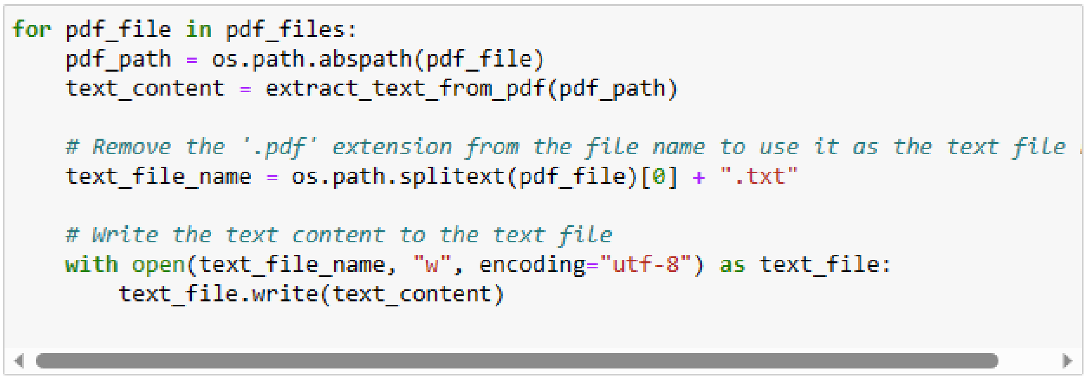
```python
pdf_files = ["Reference1.pdf", "Reference2.pdf", "Reference3.pdf", "Reference4.p
```

STEP6:

Using PdfReader to extract text from pdf. The pdfreader is a Pythonic API to PDF documents which follows PDF-1.7 specification. It allows to parse documents, extract texts, images, fonts, CMaps, and other data; access different objects within PDF documents.



```python
import PyPDF2

def extract_text_from_pdf(pdf_path):
    with open(pdf_path, "rb") as file:
        pdf_reader = PyPDF2.PdfReader(file)
        text = ""
        for page_num in range(len(pdf_reader.pages)):
            page = pdf_reader.pages[page_num]
            text += page.extract_text()
    return text
```



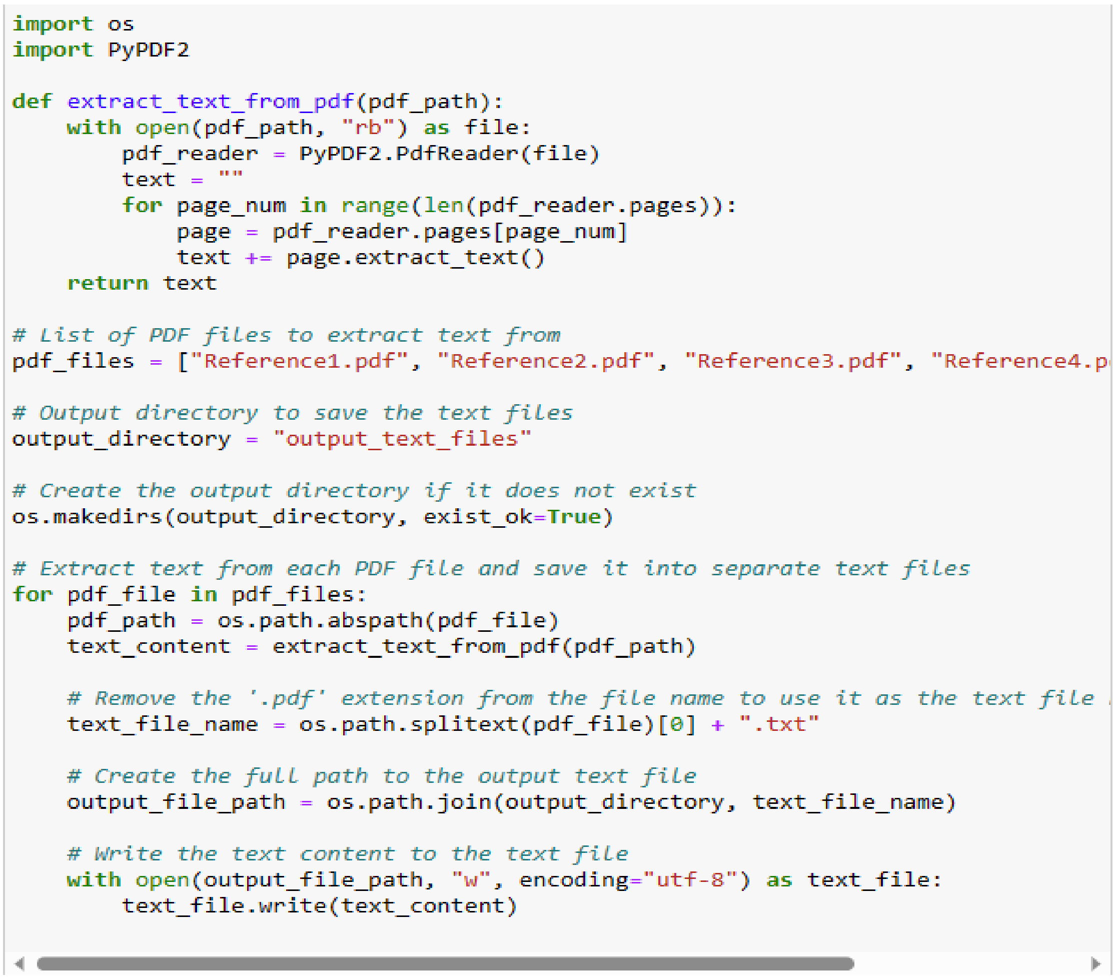
```python
for pdf_file in pdf_files:
    pdf_path = os.path.abspath(pdf_file)
    text_content = extract_text_from_pdf(pdf_path)

    # Remove the '.pdf' extension from the file name to use it as the text file
    text_file_name = os.path.splitext(pdf_file)[0] + ".txt"

    # Write the text content to the text file
    with open(text_file_name, "w", encoding="utf-8") as text_file:
        text_file.write(text_content)
```

STEP7:

After extracting the text content from the PDF files, we have to save the text into separate text files.

This code specifies the list of PDF files (pdf_files) from which we want to extract the text and also defines the output directory (output_directory) where we want to save the extracted text files. If the output directory doesn't exist, the code will create it.

Then, for each PDF file in the list, we extract the text using the extract_text_from_pdf function and save it to a separate text file with the same name as the corresponding PDF file, but with the .txt extension.

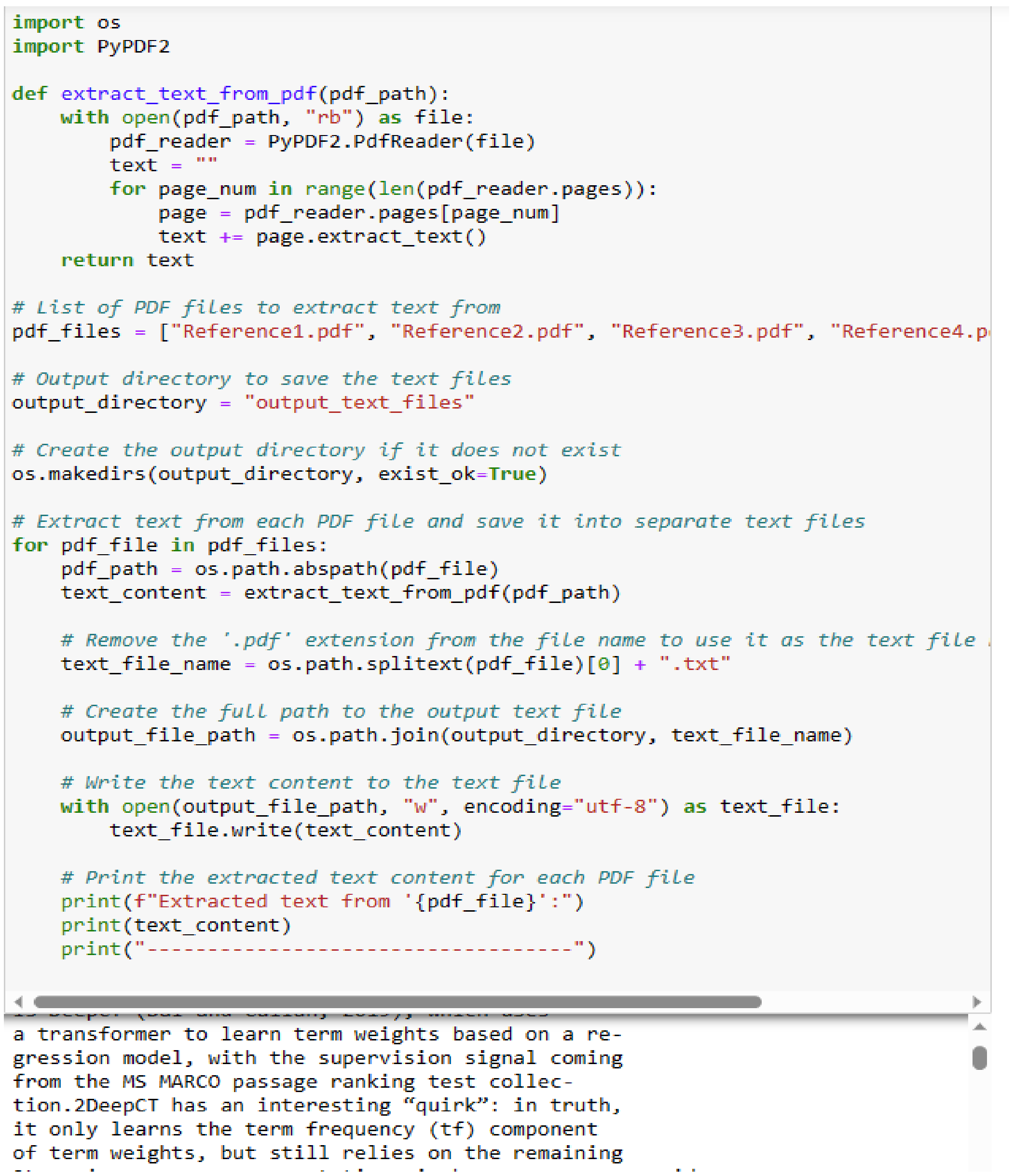


```
import os
import PyPDF2

def extract_text_from_pdf(pdf_path):
    with open(pdf_path, "rb") as file:
        pdf_reader = PyPDF2.PdfReader(file)
        text = ""
        for page_num in range(len(pdf_reader.pages)):
            page = pdf_reader.pages[page_num]
            text += page.extract_text()
    return text

# List of PDF files to extract text from
pdf_files = ["Reference1.pdf", "Reference2.pdf", "Reference3.pdf", "Reference4.p

# Output directory to save the text files
output_directory = "output_text_files"

# Create the output directory if it does not exist
os.makedirs(output_directory, exist_ok=True)

# Extract text from each PDF file and save it into separate text files
for pdf_file in pdf_files:
    pdf_path = os.path.abspath(pdf_file)
    text_content = extract_text_from_pdf(pdf_path)

    # Remove the '.pdf' extension from the file name to use it as the text file
    text_file_name = os.path.splitext(pdf_file)[0] + ".txt"

    # Create the full path to the output text file
    output_file_path = os.path.join(output_directory, text_file_name)

    # Write the text content to the text file
    with open(output_file_path, "w", encoding="utf-8") as text_file:
        text_file.write(text_content)
```

After running this code, you will have individual text files in the output_text_files directory, each containing the extracted text from the corresponding PDF file.

STEP8:

To see the extracted text content in output,we have to modify the code to print the content after extraction.

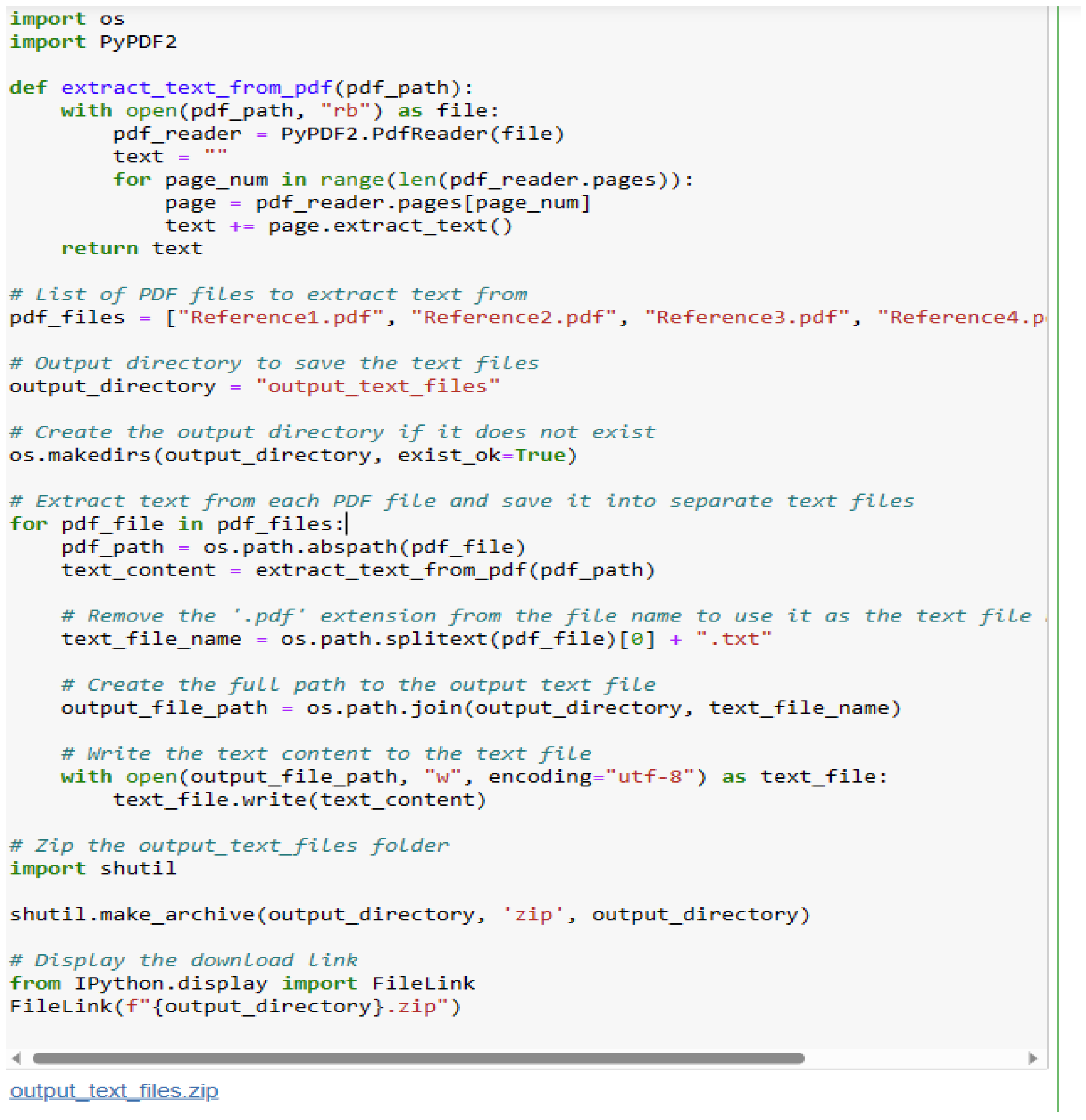


```python
import os
import PyPDF2

def extract_text_from_pdf(pdf_path):
    with open(pdf_path, "rb") as file:
        pdf_reader = PyPDF2.PdfReader(file)
        text = ""
        for page_num in range(len(pdf_reader.pages)):
            page = pdf_reader.pages[page_num]
            text += page.extract_text()
    return text

# List of PDF files to extract text from
pdf_files = ["Reference1.pdf", "Reference2.pdf", "Reference3.pdf", "Reference4.p

# Output directory to save the text files
output_directory = "output_text_files"

# Create the output directory if it does not exist
os.makedirs(output_directory, exist_ok=True)

# Extract text from each PDF file and save it into separate text files
for pdf_file in pdf_files:
    pdf_path = os.path.abspath(pdf_file)
    text_content = extract_text_from_pdf(pdf_path)

    # Remove the '.pdf' extension from the file name to use it as the text file
    text_file_name = os.path.splitext(pdf_file)[0] + ".txt"

    # Create the full path to the output text file
    output_file_path = os.path.join(output_directory, text_file_name)

    # Write the text content to the text file
    with open(output_file_path, "w", encoding="utf-8") as text_file:
        text_file.write(text_content)

    # Print the extracted text content for each PDF file
    print(f"Extracted text from '{pdf_file}':")
    print(text_content)
    print("-----------------------------------")
```

```
a transformer to learn term weights based on a re-
gression model, with the supervision signal coming
from the MS MARCO passage ranking test collec-
tion.2DeepCT has an interesting “quirk”: in truth,
it only learns the term frequency (tf) component
of term weights, but still relies on the remaining
```

STEP9:

To save the extracted text content into a separate text file and create a folder in Jupyter Notebook to store these text files.



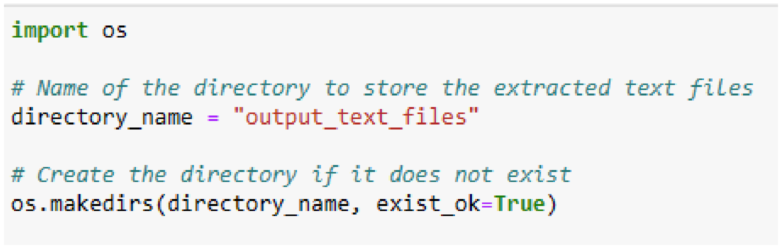
```
import os
import PyPDF2

def extract_text_from_pdf(pdf_path):
    with open(pdf_path, "rb") as file:
        pdf_reader = PyPDF2.PdfReader(file)
        text = ""
        for page_num in range(len(pdf_reader.pages)):
            page = pdf_reader.pages[page_num]
            text += page.extract_text()
    return text

# List of PDF files to extract text from
pdf_files = ["Reference1.pdf", "Reference2.pdf", "Reference3.pdf", "Reference4.p

# Output directory to save the text files
output_directory = "output_text_files"

# Create the output directory if it does not exist
os.makedirs(output_directory, exist_ok=True)

# Extract text from each PDF file and save it into separate text files
for pdf_file in pdf_files:
    pdf_path = os.path.abspath(pdf_file)
    text_content = extract_text_from_pdf(pdf_path)

    # Remove the '.pdf' extension from the file name to use it as the text file
    text_file_name = os.path.splitext(pdf_file)[0] + ".txt"

    # Create the full path to the output text file
    output_file_path = os.path.join(output_directory, text_file_name)

    # Write the text content to the text file
    with open(output_file_path, "w", encoding="utf-8") as text_file:
        text_file.write(text_content)

# Zip the output_text_files folder
import shutil

shutil.make_archive(output_directory, 'zip', output_directory)

# Display the download link
from IPython.display import FileLink
FileLink(f"{output_directory}.zip")
```

output_text_files.zip

The above code provides a zip file named "output_text_files.zip" containing individual text files for each PDF file. These text files contain the extracted text content from the corresponding PDF files.

STEP10:

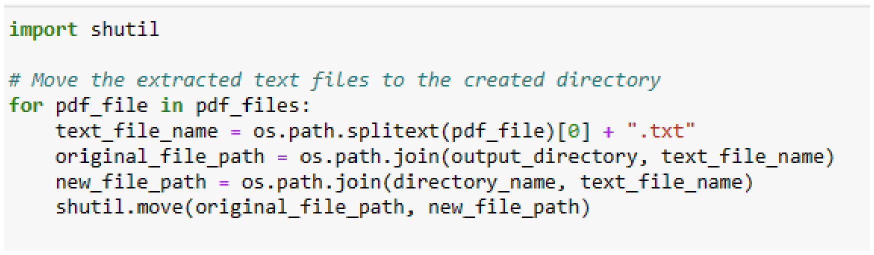
After extracting the text from the PDF files and saving them into individual text files, you can create a directory path to organize the extracted files, and then proceed to ask questions related to the content.



```python
import os

# Name of the directory to store the extracted text files
directory_name = "output_text_files"

# Create the directory if it does not exist
os.makedirs(directory_name, exist_ok=True)
```

STEP11:

Moving the extracted text files into the created directory



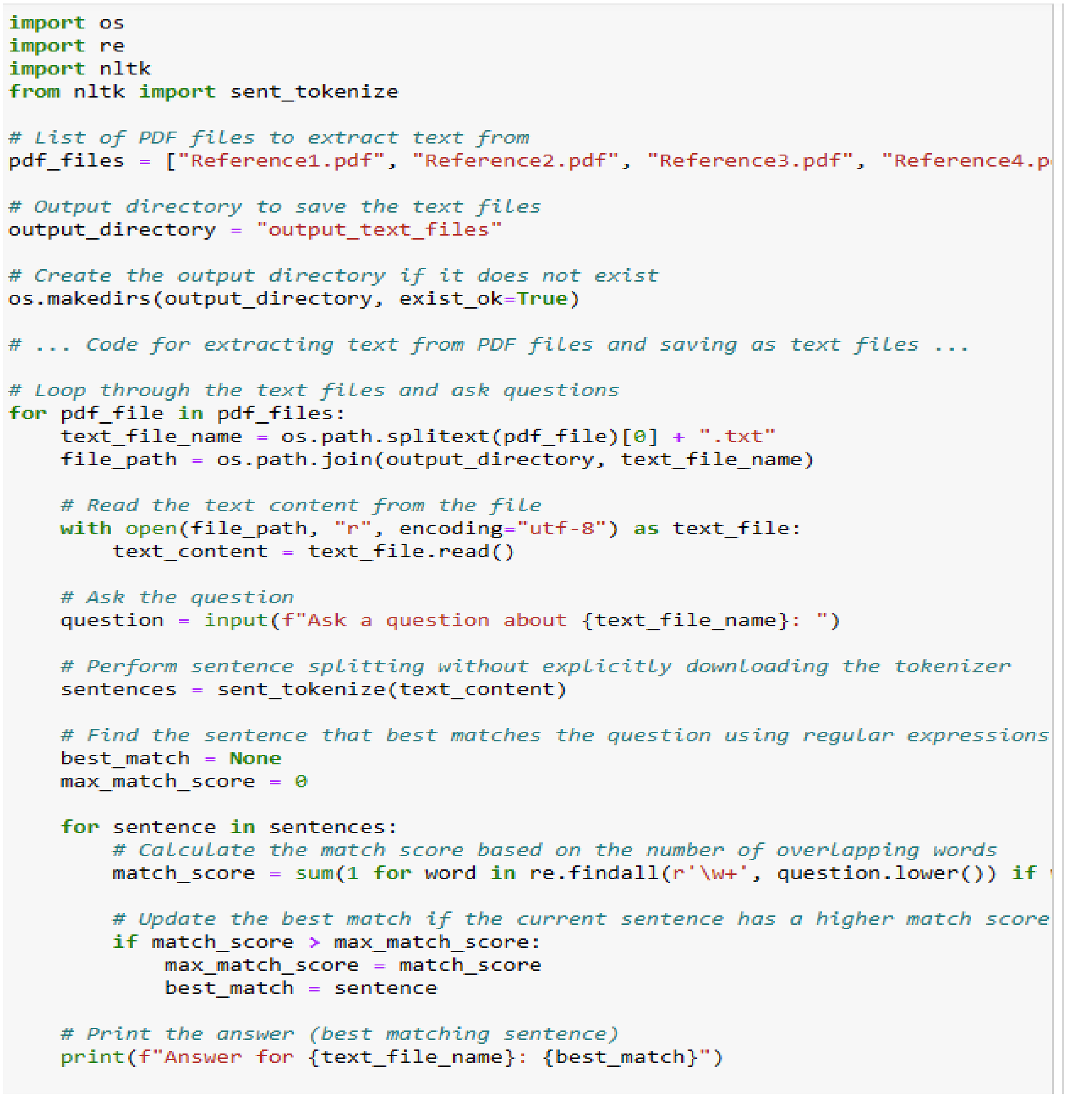
```python
import shutil

# Move the extracted text files to the created directory
for pdf_file in pdf_files:
    text_file_name = os.path.splitext(pdf_file)[0] + ".txt"
    original_file_path = os.path.join(output_directory, text_file_name)
    new_file_path = os.path.join(directory_name, text_file_name)
    shutil.move(original_file_path, new_file_path)
```

STEP12:

Now we extracted text files that were organized in the extracted_text_files directory. we can ask questions related to the content of these text files.

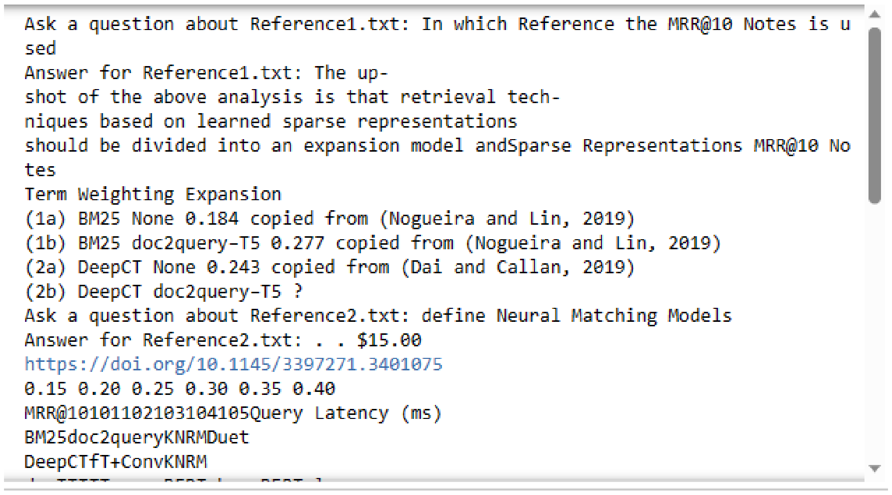
To ask questions, we can use any natural language processing library or chatbot framework.

By using the sent_tokenize function without downloading the Punkt tokenizer, the code should not encounter the "No space left on device" error.



```python
import os
import re
import nltk
from nltk import sent_tokenize

# List of PDF files to extract text from
pdf_files = ["Reference1.pdf", "Reference2.pdf", "Reference3.pdf", "Reference4.p

# Output directory to save the text files
output_directory = "output_text_files"

# Create the output directory if it does not exist
os.makedirs(output_directory, exist_ok=True)

# ... Code for extracting text from PDF files and saving as text files ...

# Loop through the text files and ask questions
for pdf_file in pdf_files:
    text_file_name = os.path.splitext(pdf_file)[0] + ".txt"
    file_path = os.path.join(output_directory, text_file_name)

    # Read the text content from the file
    with open(file_path, "r", encoding="utf-8") as text_file:
        text_content = text_file.read()

    # Ask the question
    question = input(f"Ask a question about {text_file_name}: ")

    # Perform sentence splitting without explicitly downloading the tokenizer
    sentences = sent_tokenize(text_content)

    # Find the sentence that best matches the question using regular expressions
    best_match = None
    max_match_score = 0

    for sentence in sentences:
        # Calculate the match score based on the number of overlapping words
        match_score = sum(1 for word in re.findall(r'\w+', question.lower()) if 

        # Update the best match if the current sentence has a higher match score
        if match_score > max_match_score:
            max_match_score = match_score
            best_match = sentence

    # Print the answer (best matching sentence)
    print(f"Answer for {text_file_name}: {best_match}")
```

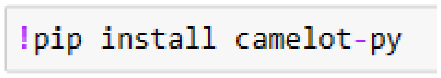
OUTPUT:



```
Ask a question about Reference1.txt: In which Reference the MRR@10 Notes is u
sed
Answer for Reference1.txt: The up-
shot of the above analysis is that retrieval tech-
niques based on learned sparse representations
should be divided into an expansion model andSparse Representations MRR@10 No
tes
Term Weighting Expansion
(1a) BM25 None 0.184 copied from (Nogueira and Lin, 2019)
(1b) BM25 doc2query-T5 0.277 copied from (Nogueira and Lin, 2019)
(2a) DeepCT None 0.243 copied from (Dai and Callan, 2019)
(2b) DeepCT doc2query-T5 ?
Ask a question about Reference2.txt: define Neural Matching Models
Answer for Reference2.txt: . . $15.00
https://doi.org/10.1145/3397271.3401075
0.15 0.20 0.25 0.30 0.35 0.40
MRR@10101102103104105Query Latency (ms)
BM25doc2queryKNRMDuet
DeepCTfT+ConvKNRM
```

STEP13:

To extract tables from the PDF files, we can use the camelot library in Python. This library provides functionality for extracting tables from PDF documents.

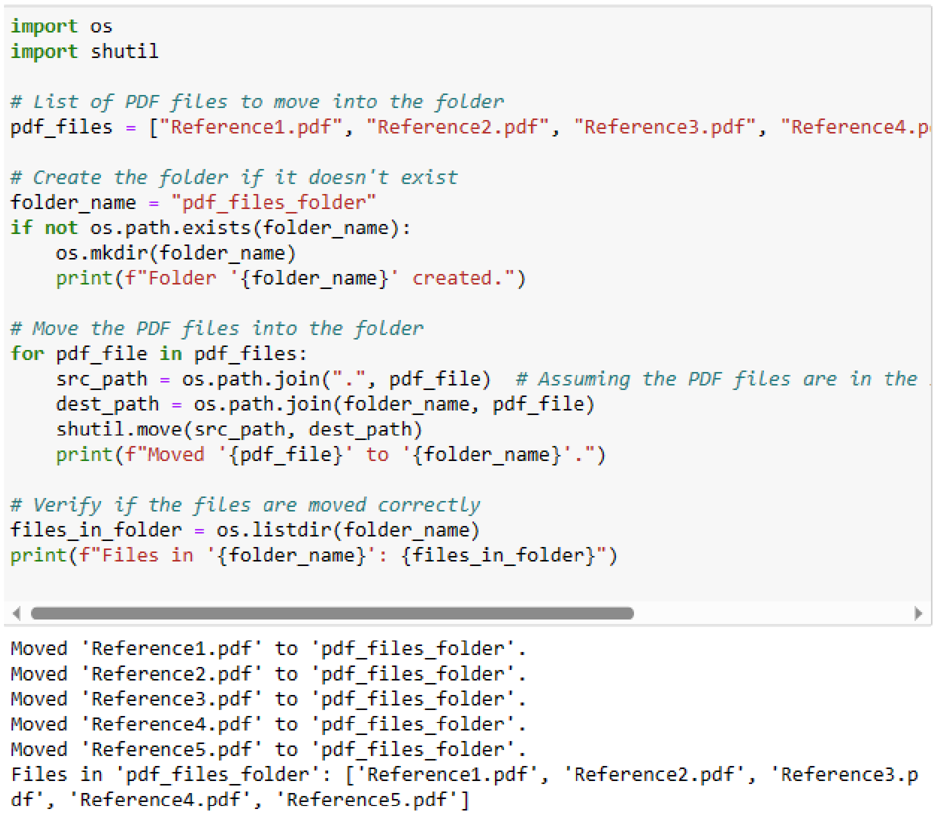


```
!pip install camelot-py
```

STEP14:

Created the "pdf_files_folder" directory by Moving the PDF files into the "pdf_files_folder" directory.

To open the "pdf_files_folder" ,we can use the os module to call the appropriate system command. This method is for opening a folder can vary depending on your operating system.

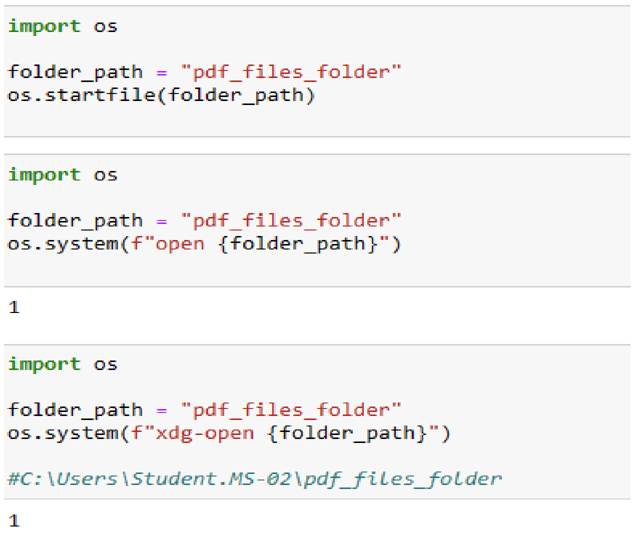


```
import os
import shutil

# List of PDF files to move into the folder
pdf_files = ["Reference1.pdf", "Reference2.pdf", "Reference3.pdf", "Reference4.p

# Create the folder if it doesn't exist
folder_name = "pdf_files_folder"
if not os.path.exists(folder_name):
    os.mkdir(folder_name)
    print(f"Folder '{folder_name}' created.")

# Move the PDF files into the folder
for pdf_file in pdf_files:
    src_path = os.path.join(".", pdf_file)  # Assuming the PDF files are in the
    dest_path = os.path.join(folder_name, pdf_file)
    shutil.move(src_path, dest_path)
    print(f"Moved '{pdf_file}' to '{folder_name}'.")

# Verify if the files are moved correctly
files_in_folder = os.listdir(folder_name)
print(f"Files in '{folder_name}': {files_in_folder}")
```

```
Moved 'Reference1.pdf' to 'pdf_files_folder'.
Moved 'Reference2.pdf' to 'pdf_files_folder'.
Moved 'Reference3.pdf' to 'pdf_files_folder'.
Moved 'Reference4.pdf' to 'pdf_files_folder'.
Moved 'Reference5.pdf' to 'pdf_files_folder'.
Files in 'pdf_files_folder': ['Reference1.pdf', 'Reference2.pdf', 'Reference3.p
df', 'Reference4.pdf', 'Reference5.pdf']
```

STEP15:

By Choosing the appropriate code snippet based on the operating system and executing it to open the "pdf_files_folder". the folder will open by using the default file explorer of your system.



```
import os

folder_path = "pdf_files_folder"
os.startfile(folder_path)
```

```
import os

folder_path = "pdf_files_folder"
os.system(f"open {folder_path}")
```

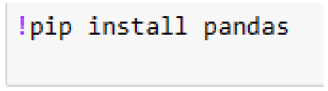
```
1
```

```
import os

folder_path = "pdf_files_folder"
os.system(f"xdg-open {folder_path}")

#C:\Users\Student.MS-02\pdf_files_folder
```

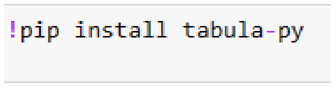
```
1
```

STEP16:

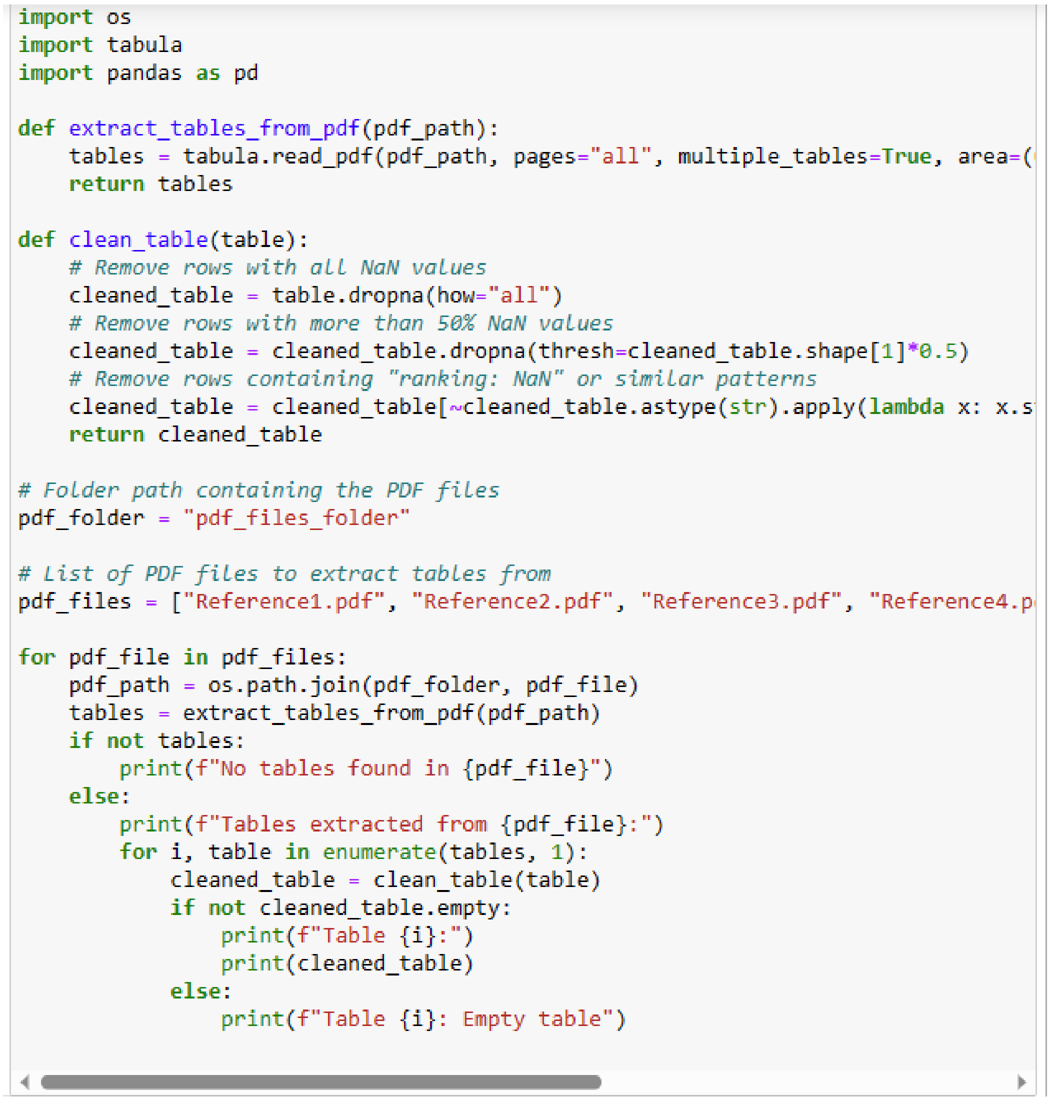
Extracted tables from PDF files and displaying them. To view the extracted tables from the PDF files, we need to run the modified code in a Python environment that has the required libraries we want to install (Pandas and Tabula).



```
!pip install pandas
```



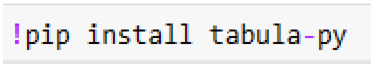
```
!pip install tabula-py
```



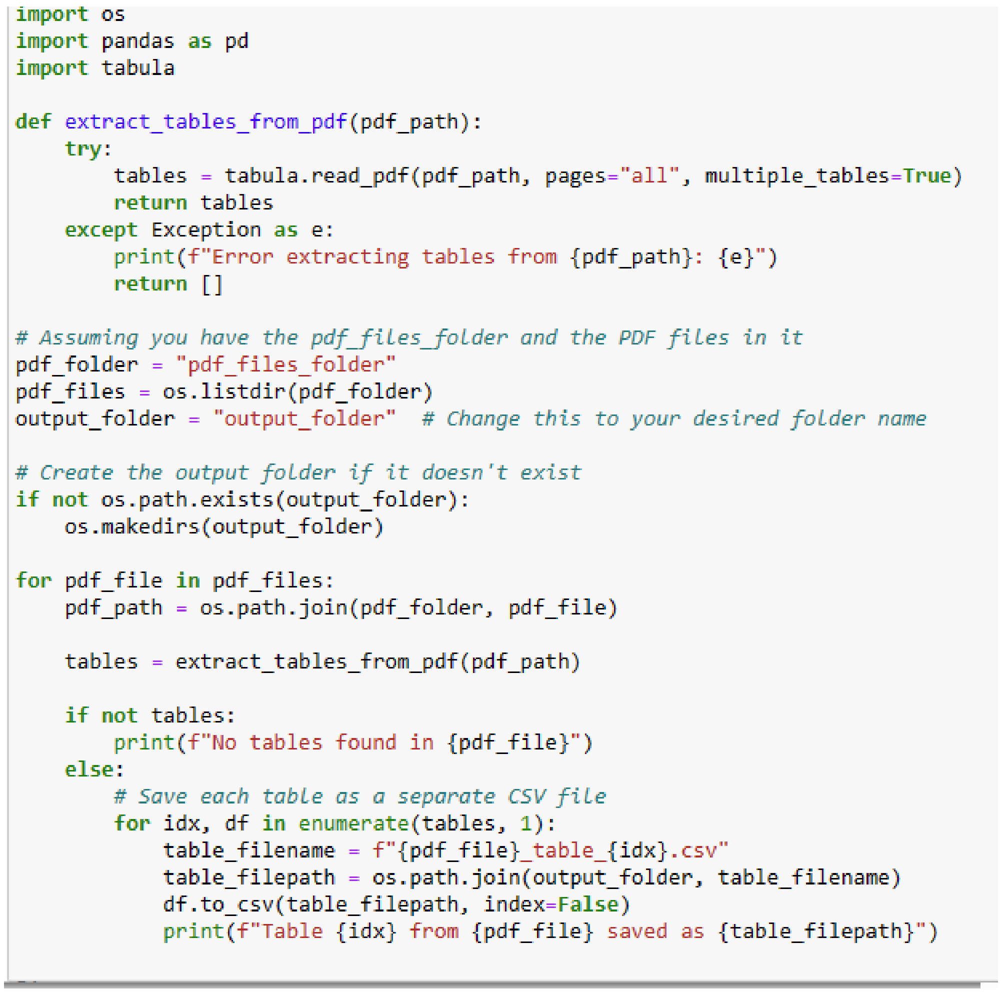
```python
import os
import tabula
import pandas as pd

def extract_tables_from_pdf(pdf_path):
    tables = tabula.read_pdf(pdf_path, pages="all", multiple_tables=True, area=(
    return tables

def clean_table(table):
    # Remove rows with all NaN values
    cleaned_table = table.dropna(how="all")
    # Remove rows with more than 50% NaN values
    cleaned_table = cleaned_table.dropna(thresh=cleaned_table.shape[1]*0.5)
    # Remove rows containing "ranking: NaN" or similar patterns
    cleaned_table = cleaned_table[~cleaned_table.astype(str).apply(lambda x: x.s
    return cleaned_table

# Folder path containing the PDF files
pdf_folder = "pdf_files_folder"

# List of PDF files to extract tables from
pdf_files = ["Reference1.pdf", "Reference2.pdf", "Reference3.pdf", "Reference4.p

for pdf_file in pdf_files:
    pdf_path = os.path.join(pdf_folder, pdf_file)
    tables = extract_tables_from_pdf(pdf_path)
    if not tables:
        print(f"No tables found in {pdf_file}")
    else:
        print(f"Tables extracted from {pdf_file}:")
        for i, table in enumerate(tables, 1):
            cleaned_table = clean_table(table)
            if not cleaned_table.empty:
                print(f"Table {i}:")
                print(cleaned_table)
            else:
                print(f"Table {i}: Empty table")
```

STEP17:

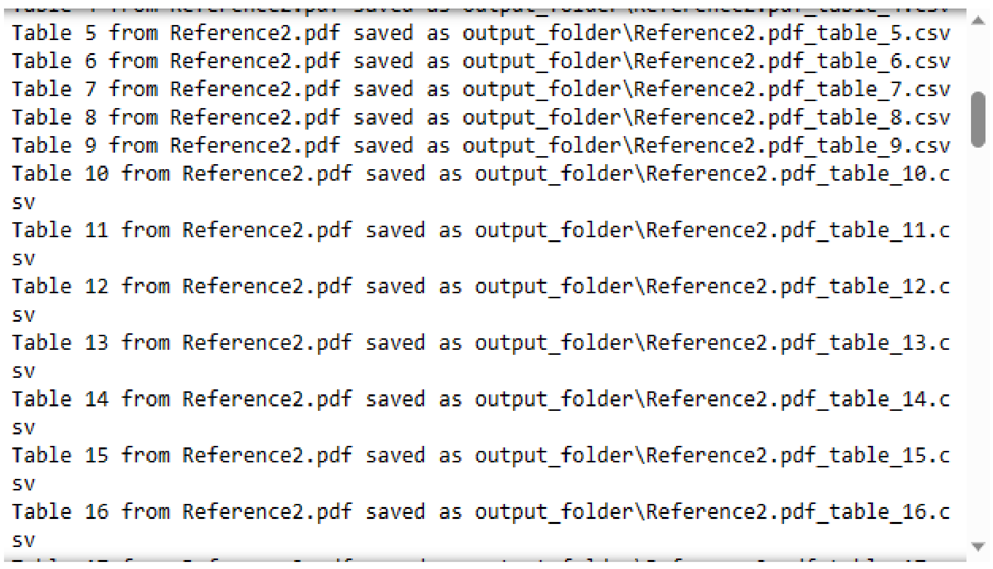
To save the extracted tables as separate CSV files, you can modify the extract_tables_from_pdf function to save each table as a separate CSV file.



```
!pip install tabula-py
```

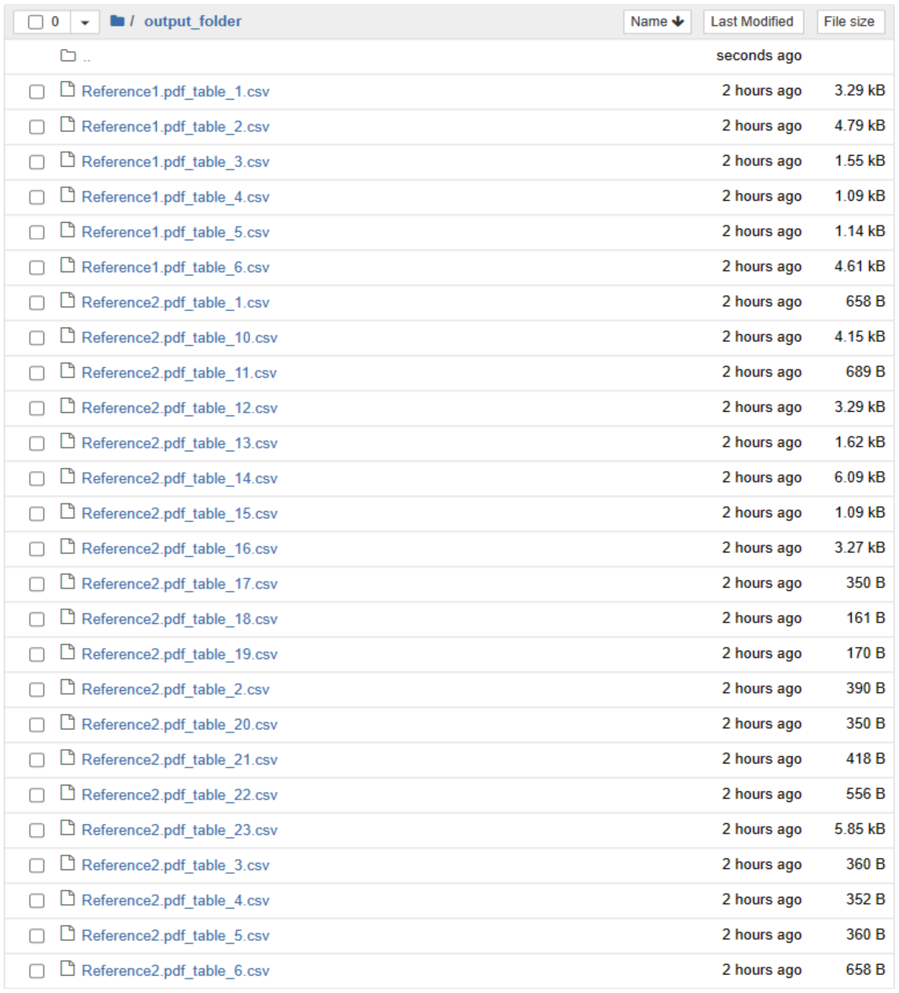


```python
import os
import pandas as pd
import tabula

def extract_tables_from_pdf(pdf_path):
    try:
        tables = tabula.read_pdf(pdf_path, pages="all", multiple_tables=True)
        return tables
    except Exception as e:
        print(f"Error extracting tables from {pdf_path}: {e}")
        return []

# Assuming you have the pdf_files_folder and the PDF files in it
pdf_folder = "pdf_files_folder"
pdf_files = os.listdir(pdf_folder)
output_folder = "output_folder"  # Change this to your desired folder name

# Create the output folder if it doesn't exist
if not os.path.exists(output_folder):
    os.makedirs(output_folder)

for pdf_file in pdf_files:
    pdf_path = os.path.join(pdf_folder, pdf_file)

    tables = extract_tables_from_pdf(pdf_path)

    if not tables:
        print(f"No tables found in {pdf_file}")
    else:
        # Save each table as a separate CSV file
        for idx, df in enumerate(tables, 1):
            table_filename = f"{pdf_file}_table_{idx}.csv"
            table_filepath = os.path.join(output_folder, table_filename)
            df.to_csv(table_filepath, index=False)
            print(f"Table {idx} from {pdf_file} saved as {table_filepath}")
```



```
Table 5 from Reference2.pdf saved as output_folder\Reference2.pdf_table_5.csv
Table 6 from Reference2.pdf saved as output_folder\Reference2.pdf_table_6.csv
Table 7 from Reference2.pdf saved as output_folder\Reference2.pdf_table_7.csv
Table 8 from Reference2.pdf saved as output_folder\Reference2.pdf_table_8.csv
Table 9 from Reference2.pdf saved as output_folder\Reference2.pdf_table_9.csv
Table 10 from Reference2.pdf saved as output_folder\Reference2.pdf_table_10.c
sv
Table 11 from Reference2.pdf saved as output_folder\Reference2.pdf_table_11.c
sv
Table 12 from Reference2.pdf saved as output_folder\Reference2.pdf_table_12.c
sv
Table 13 from Reference2.pdf saved as output_folder\Reference2.pdf_table_13.c
sv
Table 14 from Reference2.pdf saved as output_folder\Reference2.pdf_table_14.c
sv
Table 15 from Reference2.pdf saved as output_folder\Reference2.pdf_table_15.c
sv
Table 16 from Reference2.pdf saved as output_folder\Reference2.pdf_table_16.c
sv
```

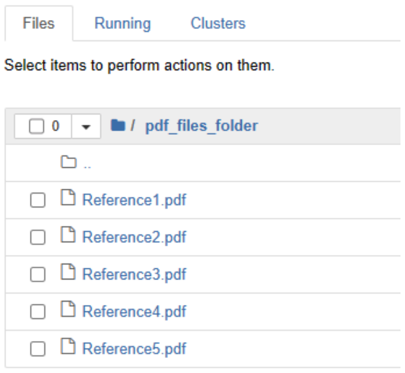
Output Folder:



0 / output_folder   Name   Last Modified   File size

Name	Last Modified	File size
..	seconds ago	
Reference1.pdf_table_1.csv	2 hours ago	3.29 kB
Reference1.pdf_table_2.csv	2 hours ago	4.79 kB
Reference1.pdf_table_3.csv	2 hours ago	1.55 kB
Reference1.pdf_table_4.csv	2 hours ago	1.09 kB
Reference1.pdf_table_5.csv	2 hours ago	1.14 kB
Reference1.pdf_table_6.csv	2 hours ago	4.61 kB
Reference2.pdf_table_1.csv	2 hours ago	658 B
Reference2.pdf_table_10.csv	2 hours ago	4.15 kB
Reference2.pdf_table_11.csv	2 hours ago	689 B
Reference2.pdf_table_12.csv	2 hours ago	3.29 kB
Reference2.pdf_table_13.csv	2 hours ago	1.62 kB
Reference2.pdf_table_14.csv	2 hours ago	6.09 kB
Reference2.pdf_table_15.csv	2 hours ago	1.09 kB
Reference2.pdf_table_16.csv	2 hours ago	3.27 kB
Reference2.pdf_table_17.csv	2 hours ago	350 B
Reference2.pdf_table_18.csv	2 hours ago	161 B
Reference2.pdf_table_19.csv	2 hours ago	170 B
Reference2.pdf_table_2.csv	2 hours ago	390 B
Reference2.pdf_table_20.csv	2 hours ago	350 B
Reference2.pdf_table_21.csv	2 hours ago	418 B
Reference2.pdf_table_22.csv	2 hours ago	556 B
Reference2.pdf_table_23.csv	2 hours ago	5.85 kB
Reference2.pdf_table_3.csv	2 hours ago	360 B
Reference2.pdf_table_4.csv	2 hours ago	352 B
Reference2.pdf_table_5.csv	2 hours ago	360 B
Reference2.pdf_table_6.csv	2 hours ago	658 B

PDF file Folder:



Files   Running   Clusters

Select items to perform actions on them.

0 / pdf_files_folder

- ..
- Reference1.pdf
- Reference2.pdf
- Reference3.pdf
- Reference4.pdf
- Reference5.pdf

20I125_Kaaviyasree.SK