Data Engineering Project

Data Pipeline

Introduction:

The purpose of this project is to extract data from multiple sources, transform it, and then load it into a data storage solution.
The targeted subject is "A.I News".
Extraction from .CSV files, .PDF files, and web scraping will be implemented.

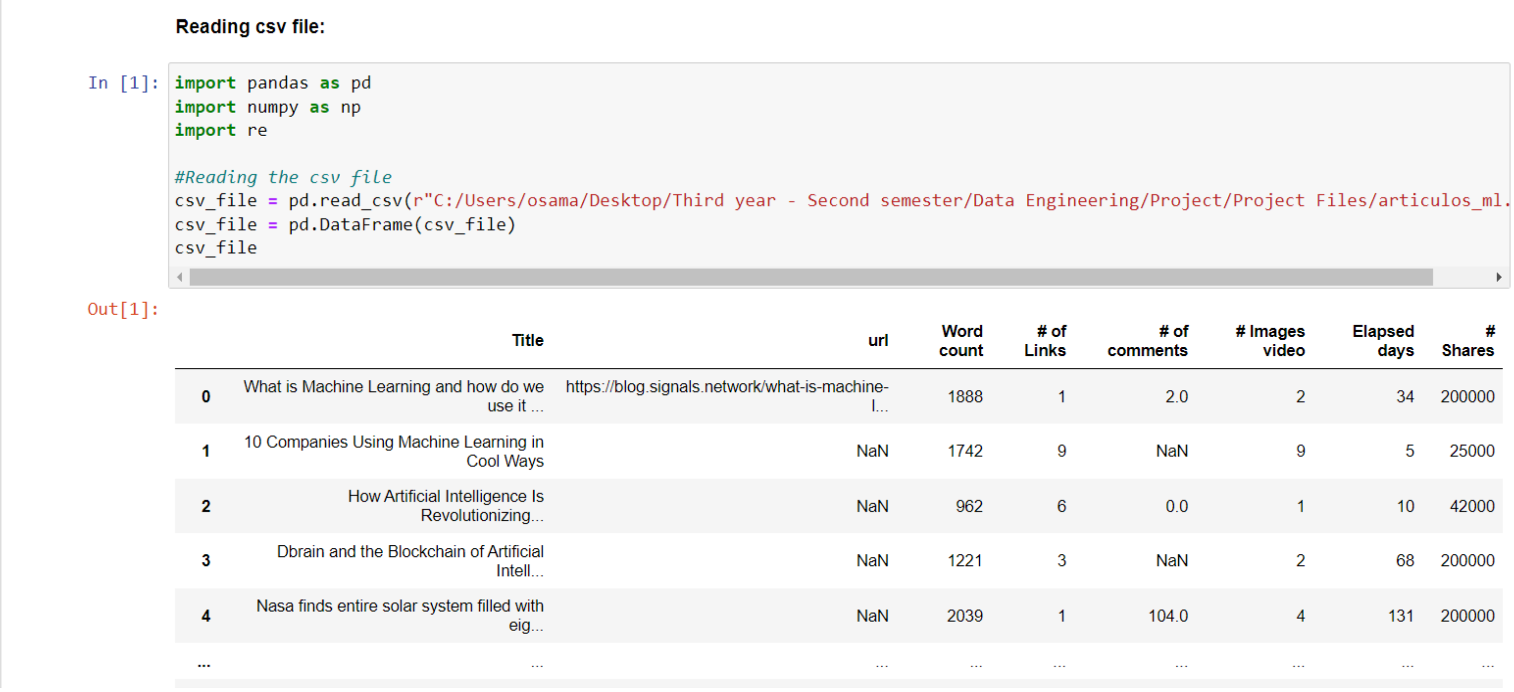
1. Data Ingestion:

- **CSV Extraction:**
  File link: https://github.com/jbagnato/machine-learning/blob/master/articulos_ml.csv

The csv file used contained relevant and irrelevant data to A.I/A.I News, we're going to extract the relevant data.

Figure 1: Reading the csv file:

Figure 1, Reading the csv file:



Reading csv file:

```
In [1]: import pandas as pd
        import numpy as np
        import re

        #Reading the csv file
        csv_file = pd.read_csv(r"C:/Users/osama/Desktop/Third year - Second semester/Data Engineering/Project/Project Files/articulos_ml.
        csv_file = pd.DataFrame(csv_file)
        csv_file
```

Out[1]:

	Title	url	Word count	# of Links	# of comments	# Images video	Elapsed days	# Shares
0	What is Machine Learning and how do we use it ...	https://blog.signals.network/what-is-machine-l...	1888	1	2.0	2	34	200000
1	10 Companies Using Machine Learning in Cool Ways	NaN	1742	9	NaN	9	5	25000
2	How Artificial Intelligence Is Revolutionizing...	NaN	962	6	0.0	1	10	42000
3	Dbrain and the Blockchain of Artificial Intell...	NaN	1221	3	NaN	2	68	200000
4	Nasa finds entire solar system filled with eig...	NaN	2039	1	104.0	4	131	200000
...	...	...	...	...	...	...	...	...

Data Engineering Project

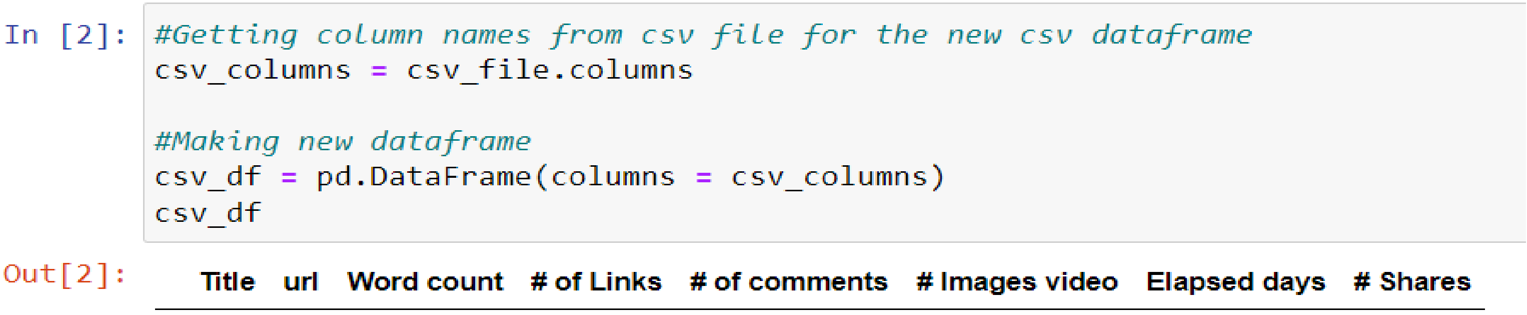
After reading the csv file we need to clean and transform the data, refer to the figures below.

Figure 2: Creating the data frame using the columns from the csv file, which we will use to save the transformed data.

Figure 3: Cleaning and transforming the csv file data, the data contained missing values and inconsistencies.

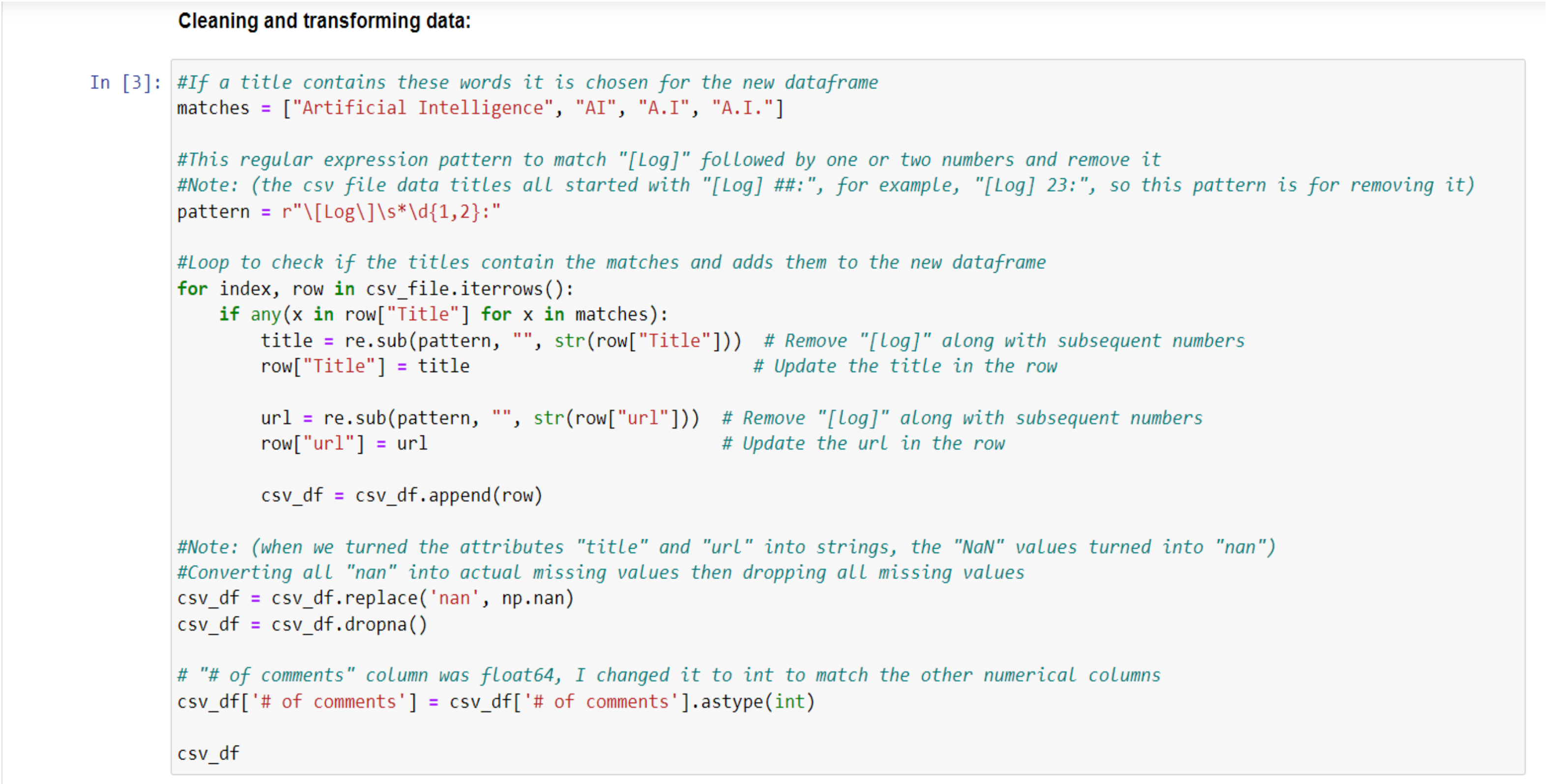
Figure 4: Displays the data after transformation.

Figure 2, Data frame creation.



```
In [2]: #Getting column names from csv file for the new csv dataframe
        csv_columns = csv_file.columns

        #Making new dataframe
        csv_df = pd.DataFrame(columns = csv_columns)
        csv_df

Out[2]:
          Title  url  Word count  # of Links  # of comments  # Images video  Elapsed days  # Shares
```

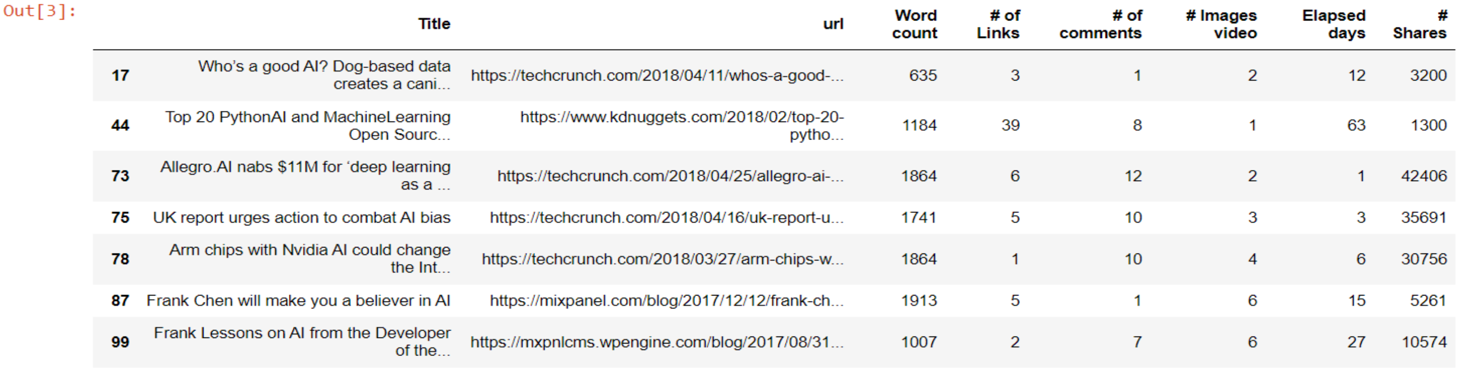
Figure 3, Cleaning and transforming.



Cleaning and transforming data:

```
In [3]: #If a title contains these words it is chosen for the new dataframe
        matches = ["Artificial Intelligence", "AI", "A.I", "A.I."]

        #This regular expression pattern to match "[Log]" followed by one or two numbers and remove it
        #Note: (the csv file data titles all started with "[Log] ##:", for example, "[Log] 23:", so this pattern is for removing it)
        pattern = r"\[Log\]\s*\d{1,2}:"

        #Loop to check if the titles contain the matches and adds them to the new dataframe
        for index, row in csv_file.iterrows():
            if any(x in row["Title"] for x in matches):
                title = re.sub(pattern, "", str(row["Title"]))  # Remove "[log]" along with subsequent numbers
                row["Title"] = title                            # Update the title in the row

                url = re.sub(pattern, "", str(row["url"]))  # Remove "[log]" along with subsequent numbers
                row["url"] = url                             # Update the url in the row

                csv_df = csv_df.append(row)

        #Note: (when we turned the attributes "title" and "url" into strings, the "NaN" values turned into "nan")
        #Converting all "nan" into actual missing values then dropping all missing values
        csv_df = csv_df.replace('nan', np.nan)
        csv_df = csv_df.dropna()

        # "# of comments" column was float64, I changed it to int to match the other numerical columns
        csv_df['# of comments'] = csv_df['# of comments'].astype(int)

        csv_df
```

Figure 4, Data after cleaning and transformation.



```
Out[3]:
```

	Title	url	Word count	# of Links	# of comments	# Images video	Elapsed days	# Shares
17	Who's a good AI? Dog-based data creates a cani...	https://techcrunch.com/2018/04/11/whos-a-good-...	635	3	1	2	12	3200
44	Top 20 PythonAI and MachineLearning Open Sourc...	https://www.kdnuggets.com/2018/02/top-20-pytho...	1184	39	8	1	63	1300
73	Allegro.AI nabs $11M for 'deep learning as a ...	https://techcrunch.com/2018/04/25/allegro-ai-...	1864	6	12	2	1	42406
75	UK report urges action to combat AI bias	https://techcrunch.com/2018/04/16/uk-report-u...	1741	5	10	3	3	35691
78	Arm chips with Nvidia AI could change the Int...	https://techcrunch.com/2018/03/27/arm-chips-w...	1864	1	10	4	6	30756
87	Frank Chen will make you a believer in AI	https://mixpanel.com/blog/2017/12/12/frank-ch...	1913	5	1	6	15	5261
99	Frank Lessons on AI from the Developer of the...	https://mxpnlcms.wpengine.com/blog/2017/08/31...	1007	2	7	6	27	10574

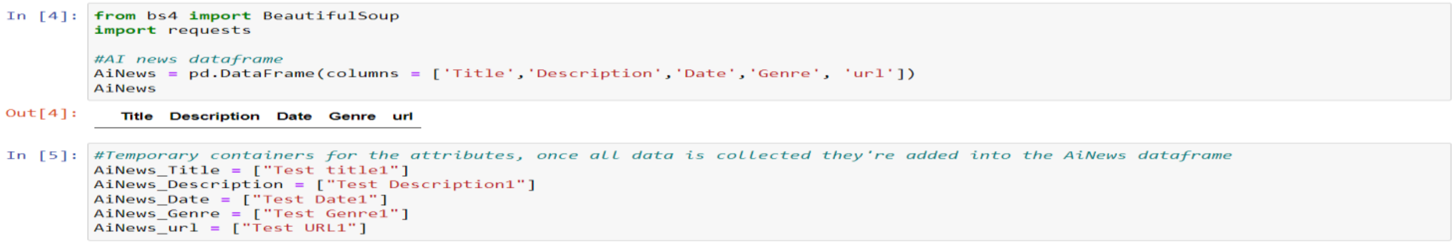
- ## Web Scraping Extraction:

Website: https://www.artificialintelligence-news.com/

Used BeautifulSoup library to scrap from the website which contains A.I News articles, the data scraped from each article is: Title, Description, Date, Genre, URL.

Figure 5: Creating the data frame which will store the finalized data, and the temporary containers will be used to store the scraped data directly.
Figure 6: details of the functions used for web scraping.
Figure 7: The AiNews_create() function is used once when initializing web scraping.
Figure 8: The AiNews_add() function will be used later in a thread to continually add new data to the data frame.
Figure 9: The AiNews_create() function is called to initialize web scraping, then the AiNews_add() function repeats every 2 hours to keep the data up to date.

Figure 5, data frame and containers creation.

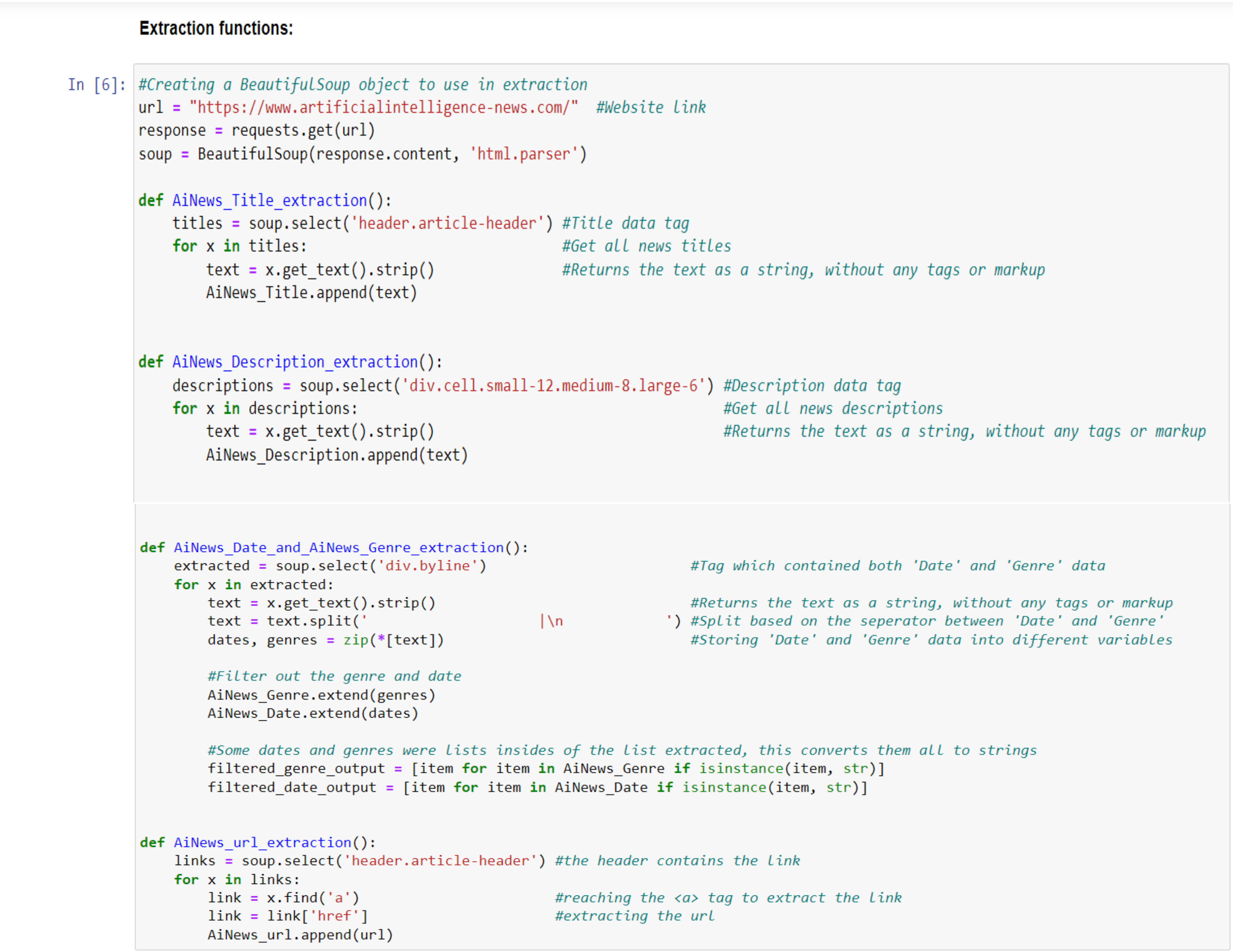


```
In [4]: from bs4 import BeautifulSoup
        import requests

        #AI news dataframe
        AiNews = pd.DataFrame(columns = ['Title','Description','Date','Genre', 'url'])
        AiNews
```

```
Out[4]:   Title  Description  Date  Genre  url
```

```
In [5]: #Temporary containers for the attributes, once all data is collected they're added into the AiNews dataframe
        AiNews_Title = ["Test title1"]
        AiNews_Description = ["Test Description1"]
        AiNews_Date = ["Test Date1"]
        AiNews_Genre = ["Test Genre1"]
        AiNews_url = ["Test URL1"]
```

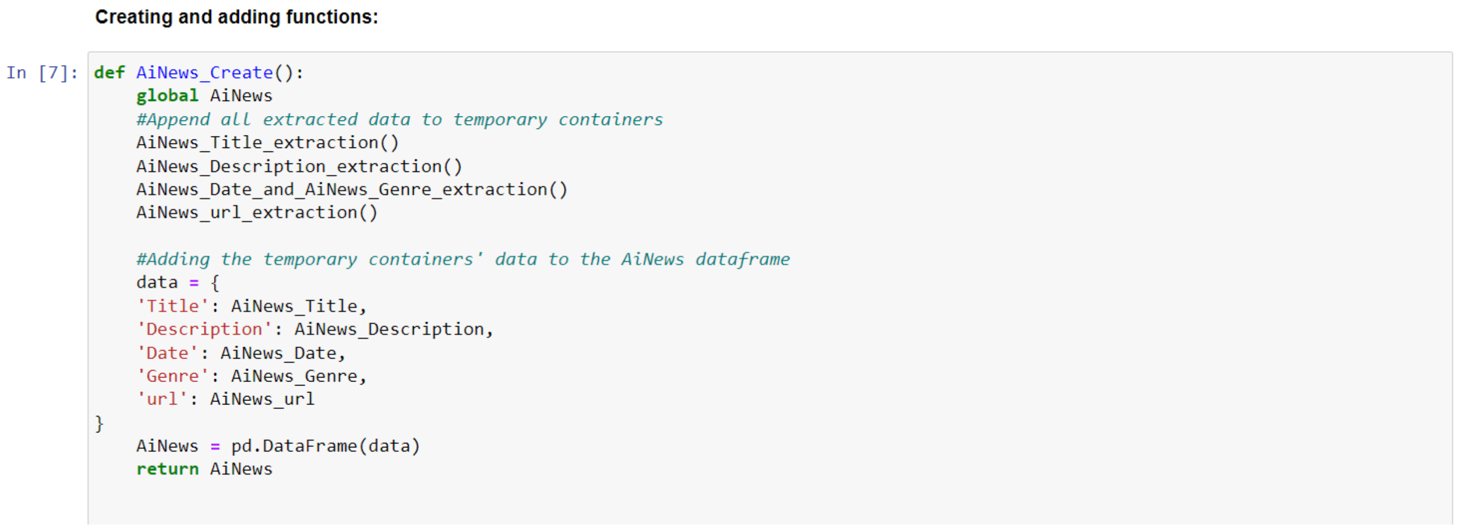
Figure 6, functions used in extracting the data.



Extraction functions:

```
In [6]: #Creating a BeautifulSoup object to use in extraction
        url = "https://www.artificialintelligence-news.com/"  #Website link
        response = requests.get(url)
        soup = BeautifulSoup(response.content, 'html.parser')

        def AiNews_Title_extraction():
            titles = soup.select('header.article-header') #Title data tag
            for x in titles:                              #Get all news titles
                text = x.get_text().strip()               #Returns the text as a string, without any tags or markup
                AiNews_Title.append(text)


        def AiNews_Description_extraction():
            descriptions = soup.select('div.cell.small-12.medium-8.large-6') #Description data tag
            for x in descriptions:                                           #Get all news descriptions
                text = x.get_text().strip()                                  #Returns the text as a string, without any tags or markup
                AiNews_Description.append(text)


        def AiNews_Date_and_AiNews_Genre_extraction():
            extracted = soup.select('div.byline')                       #Tag which contained both 'Date' and 'Genre' data
            for x in extracted:
                text = x.get_text().strip()                             #Returns the text as a string, without any tags or markup
                text = text.split('                  |\n             ') #Split based on the seperator between 'Date' and 'Genre'
                dates, genres = zip(*[text])                            #Storing 'Date' and 'Genre' data into different variables

                #Filter out the genre and date
                AiNews_Genre.extend(genres)
                AiNews_Date.extend(dates)

                #Some dates and genres were lists insides of the list extracted, this converts them all to strings
                filtered_genre_output = [item for item in AiNews_Genre if isinstance(item, str)]
                filtered_date_output = [item for item in AiNews_Date if isinstance(item, str)]


        def AiNews_url_extraction():
            links = soup.select('header.article-header') #the header contains the link
            for x in links:
                link = x.find('a')                       #reaching the <a> tag to extract the link
                link = link['href']                      #extracting the url
                AiNews_url.append(url)
```

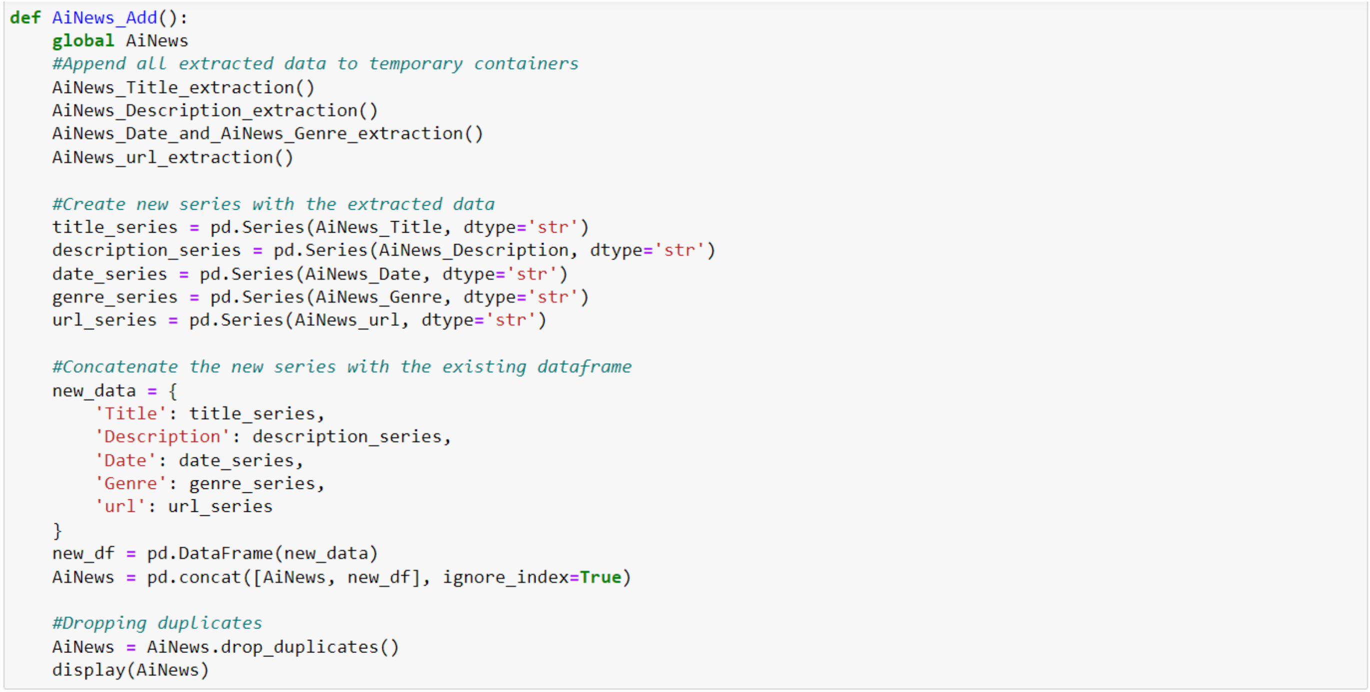
Figure 7, The AiNews Data frame creation function.



Creating and adding functions:

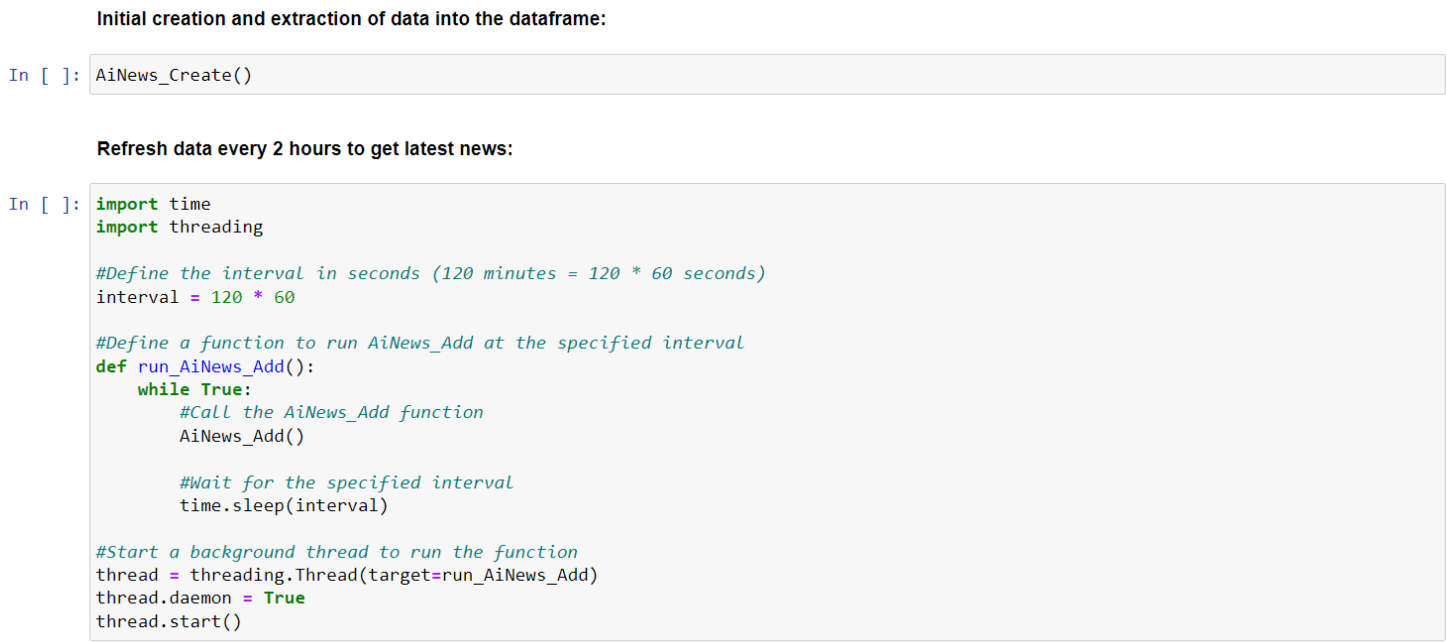
```
In [7]: def AiNews_Create():
            global AiNews
            #Append all extracted data to temporary containers
            AiNews_Title_extraction()
            AiNews_Description_extraction()
            AiNews_Date_and_AiNews_Genre_extraction()
            AiNews_url_extraction()

            #Adding the temporary containers' data to the AiNews dataframe
            data = {
            'Title': AiNews_Title,
            'Description': AiNews_Description,
            'Date': AiNews_Date,
            'Genre': AiNews_Genre,
            'url': AiNews_url
        }
            AiNews = pd.DataFrame(data)
            return AiNews
```

Figure 8, The AiNews adding function.



```
def AiNews_Add():
    global AiNews
    #Append all extracted data to temporary containers
    AiNews_Title_extraction()
    AiNews_Description_extraction()
    AiNews_Date_and_AiNews_Genre_extraction()
    AiNews_url_extraction()

    #Create new series with the extracted data
    title_series = pd.Series(AiNews_Title, dtype='str')
    description_series = pd.Series(AiNews_Description, dtype='str')
    date_series = pd.Series(AiNews_Date, dtype='str')
    genre_series = pd.Series(AiNews_Genre, dtype='str')
    url_series = pd.Series(AiNews_url, dtype='str')

    #Concatenate the new series with the existing dataframe
    new_data = {
        'Title': title_series,
        'Description': description_series,
        'Date': date_series,
        'Genre': genre_series,
        'url': url_series
    }
    new_df = pd.DataFrame(new_data)
    AiNews = pd.concat([AiNews, new_df], ignore_index=True)

    #Dropping duplicates
    AiNews = AiNews.drop_duplicates()
    display(AiNews)
```

Figure 9, web scrape initialization and addition every 2 hours.



Initial creation and extraction of data into the dataframe:

```
In [ ]: AiNews_Create()
```

Refresh data every 2 hours to get latest news:

```
In [ ]: import time
        import threading

        #Define the interval in seconds (120 minutes = 120 * 60 seconds)
        interval = 120 * 60

        #Define a function to run AiNews_Add at the specified interval
        def run_AiNews_Add():
            while True:
                #Call the AiNews_Add function
                AiNews_Add()

                #Wait for the specified interval
                time.sleep(interval)

        #Start a background thread to run the function
        thread = threading.Thread(target=run_AiNews_Add)
        thread.daemon = True
        thread.start()
```

- ## PDF Extraction:

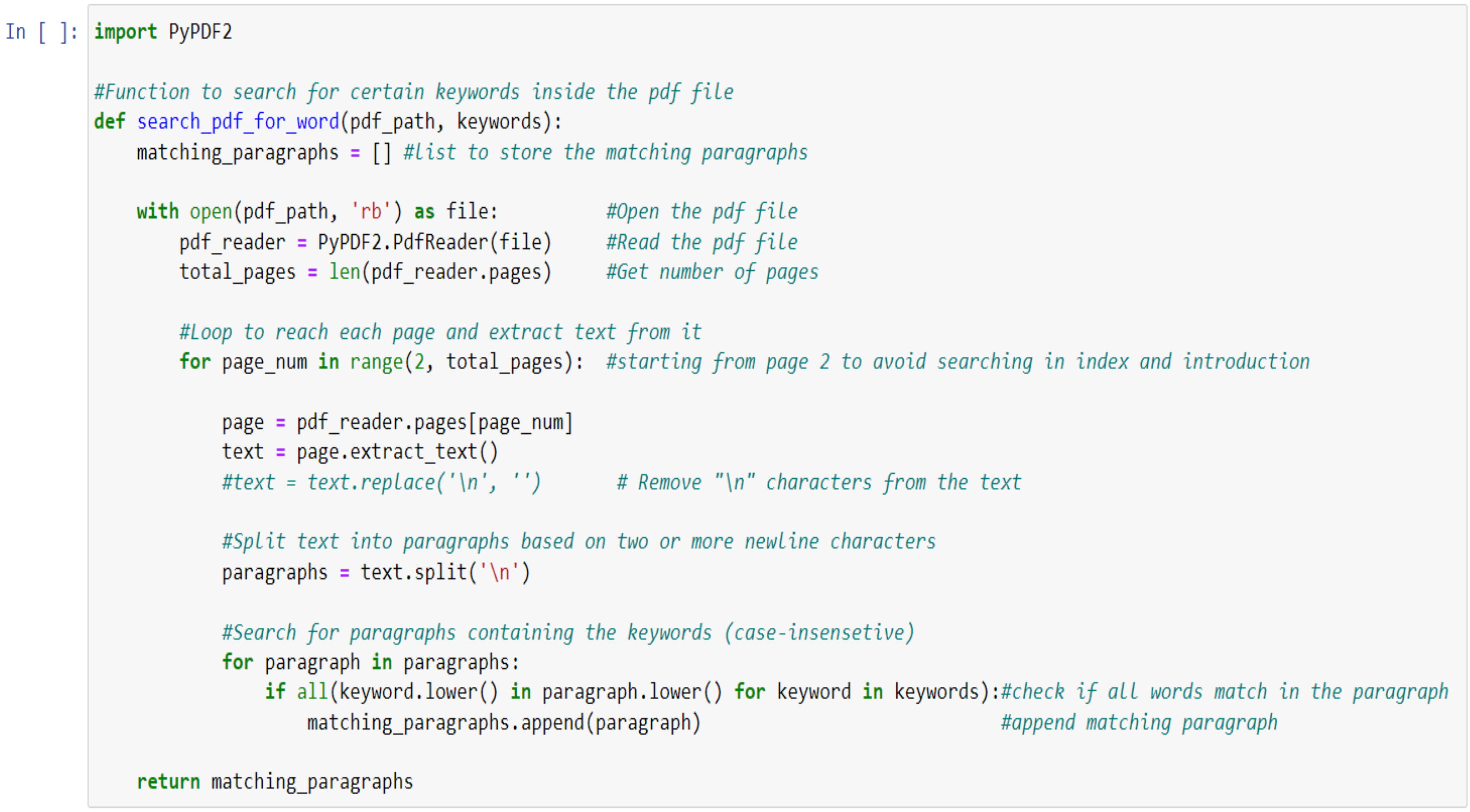
File Link:
https://www.tandfonline.com/doi/pdf/10.1080/21670811.2022.2063150

The pdf file has information about A.I in the news, we're going to extract the paragraphs which contain the words ["AI", "News"] and store them in a data frame.

Figure 10: Used PyPDF2 to read a pdf file and then extract the relevant paragraphs to our subject (A.I news).

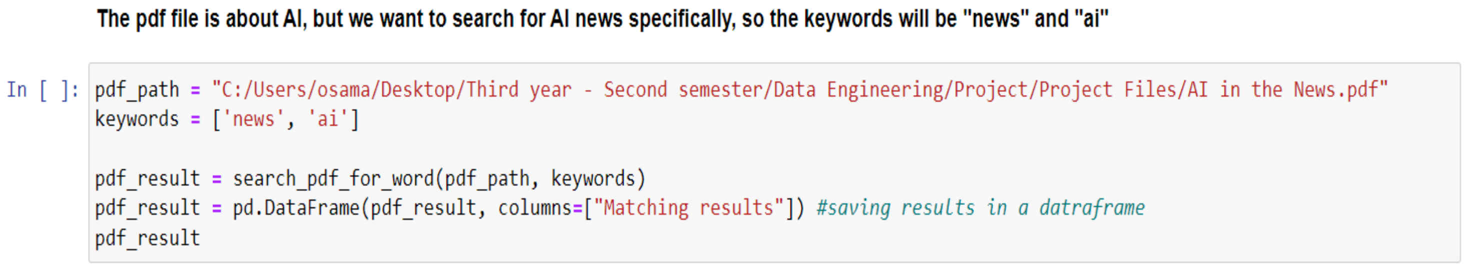
Figure 11: Called the search_pdf_for_word() function with the desired pdf path and keyword, which we then stored the result of in a data frame.

Figure 10, Function to search for matching words in paragraphs.



```
In [ ]: import PyPDF2

        #Function to search for certain keywords inside the pdf file
        def search_pdf_for_word(pdf_path, keywords):
            matching_paragraphs = [] #list to store the matching paragraphs

            with open(pdf_path, 'rb') as file:          #Open the pdf file
                pdf_reader = PyPDF2.PdfReader(file)     #Read the pdf file
                total_pages = len(pdf_reader.pages)     #Get number of pages

                #Loop to reach each page and extract text from it
                for page_num in range(2, total_pages):  #starting from page 2 to avoid searching in index and introduction

                    page = pdf_reader.pages[page_num]
                    text = page.extract_text()
                    #text = text.replace('\n', '')       # Remove "\n" characters from the text

                    #Split text into paragraphs based on two or more newline characters
                    paragraphs = text.split('\n')

                    #Search for paragraphs containing the keywords (case-insensetive)
                    for paragraph in paragraphs:
                        if all(keyword.lower() in paragraph.lower() for keyword in keywords):#check if all words match in the paragraph
                            matching_paragraphs.append(paragraph)                            #append matching paragraph

            return matching_paragraphs
```

Figure 11, Calling the function using the pdf file path and the keywords we're searching for



The pdf file is about AI, but we want to search for AI news specifically, so the keywords will be "news" and "ai"

```
In [ ]: pdf_path = "C:/Users/osama/Desktop/Third year - Second semester/Data Engineering/Project/Project Files/AI in the News.pdf"
        keywords = ['news', 'ai']

        pdf_result = search_pdf_for_word(pdf_path, keywords)
        pdf_result = pd.DataFrame(pdf_result, columns=["Matching results"]) #saving results in a datraframe
        pdf_result
```

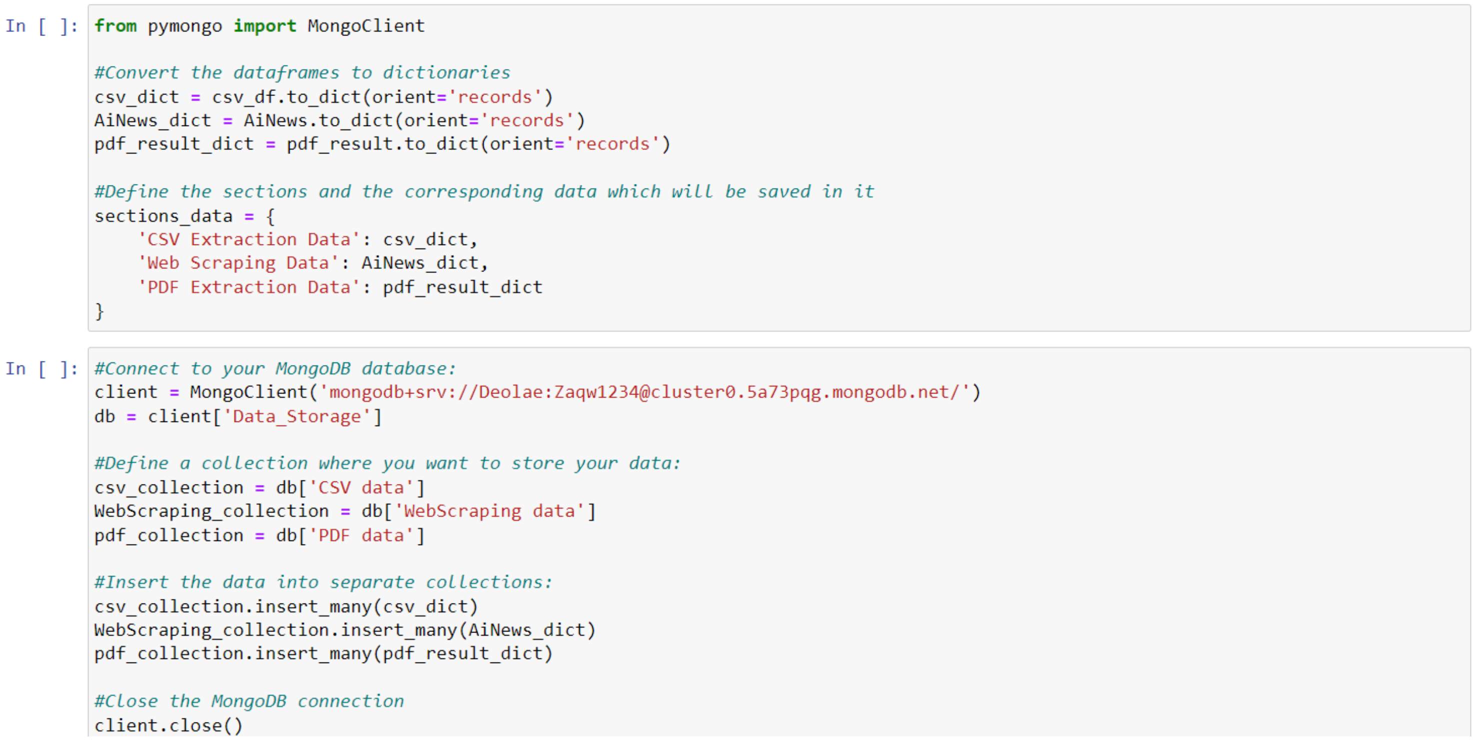
2. Data Storage:

• Storing data in MongoDB:

Used NoSQL MongoDB to store the data in separate collection depending on where they came from (CSV, web scraping, PDF).

Figure 12: Convert all data frames to dictionaries to be saved in JSON format inside MongoDB, and then connect to the MongoDB and store the data in the appropriate collection.

Figure 12, Building model for data storing to MongoDB.



```
In [ ]: from pymongo import MongoClient

        #Convert the dataframes to dictionaries
        csv_dict = csv_df.to_dict(orient='records')
        AiNews_dict = AiNews.to_dict(orient='records')
        pdf_result_dict = pdf_result.to_dict(orient='records')

        #Define the sections and the corresponding data which will be saved in it
        sections_data = {
            'CSV Extraction Data': csv_dict,
            'Web Scraping Data': AiNews_dict,
            'PDF Extraction Data': pdf_result_dict
        }

In [ ]: #Connect to your MongoDB database:
        client = MongoClient('mongodb+srv://Deolae:Zaqw1234@cluster0.5a73pqg.mongodb.net/')
        db = client['Data_Storage']

        #Define a collection where you want to store your data:
        csv_collection = db['CSV data']
        WebScraping_collection = db['WebScraping data']
        pdf_collection = db['PDF data']

        #Insert the data into separate collections:
        csv_collection.insert_many(csv_dict)
        WebScraping_collection.insert_many(AiNews_dict)
        pdf_collection.insert_many(pdf_result_dict)

        #Close the MongoDB connection
        client.close()
```

Data Engineering Project

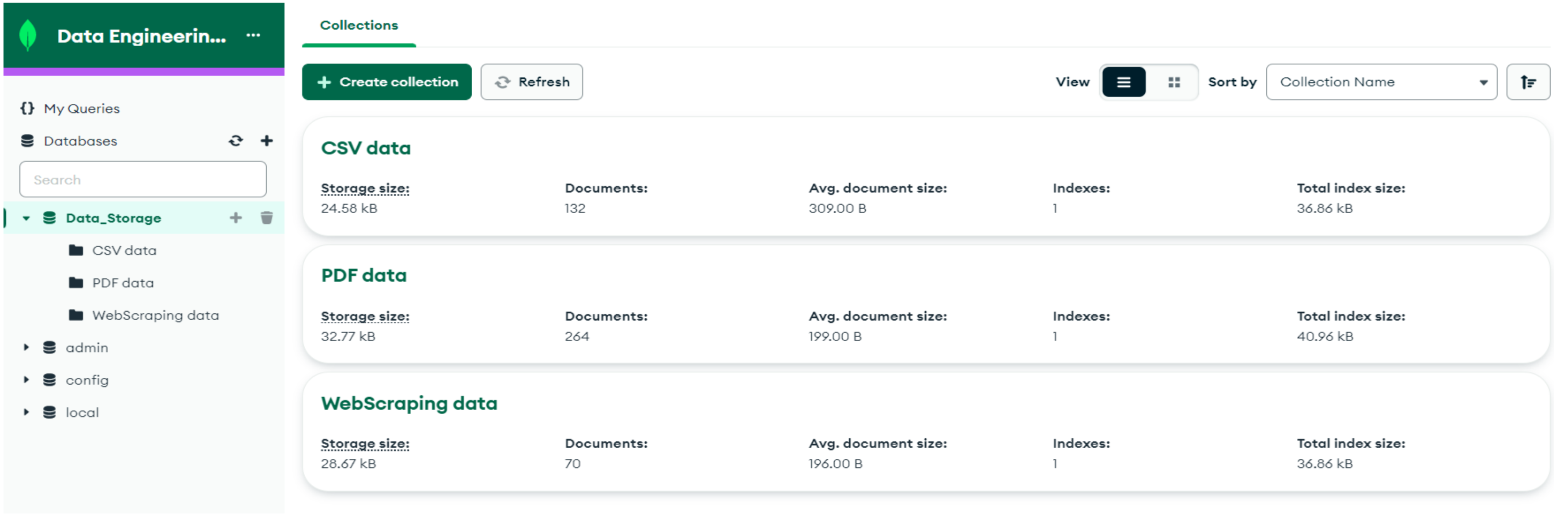
- **MongoDB:**

The database and collections are automatically made by the python code when you visit the MongoDB:

Figure 13: Overview of the database and collections.

Figure 14: Example of data inside the collection.

Figure 13, Overview.



CSV data

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
24.58 kB	132	309.00 B	1	36.86 kB

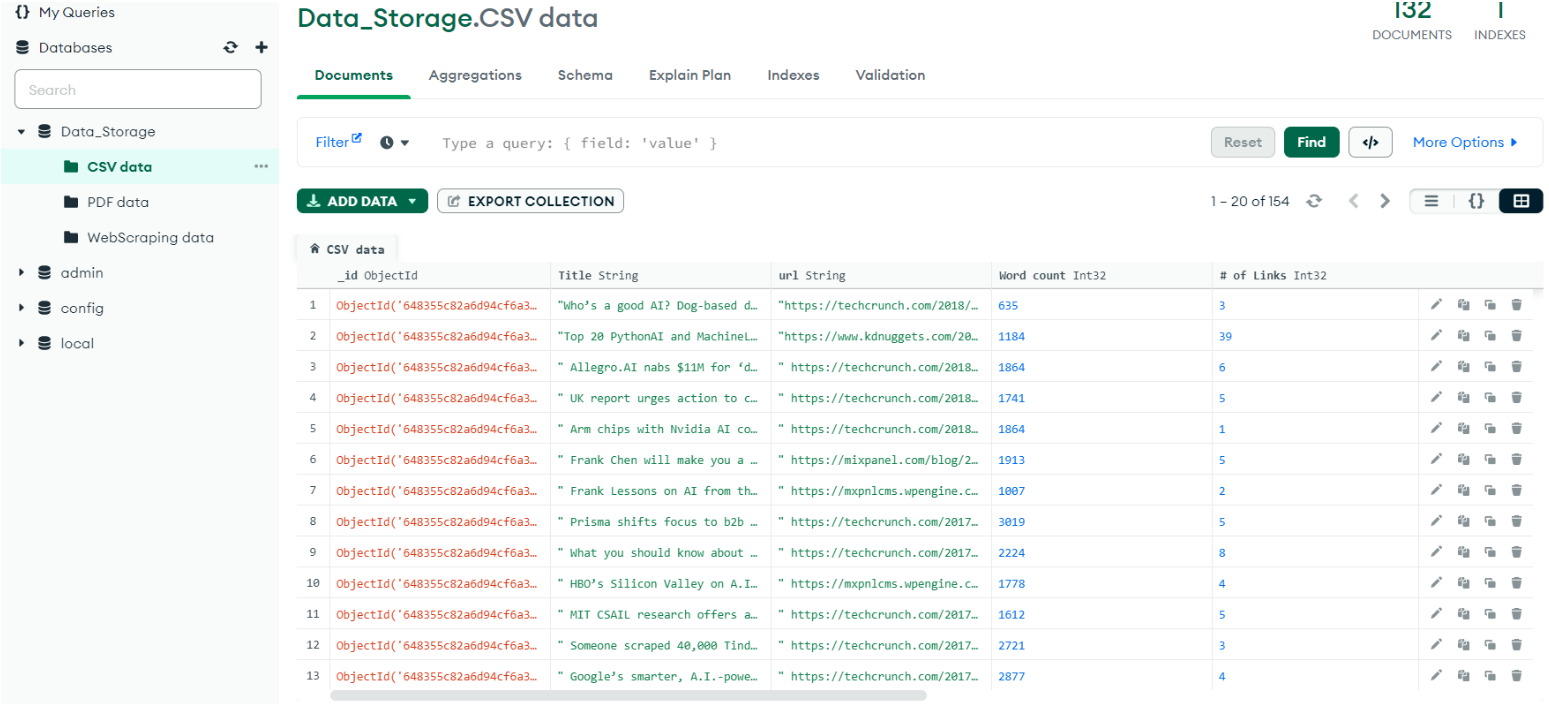
PDF data

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
32.77 kB	264	199.00 B	1	40.96 kB

WebScraping data

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
28.67 kB	70	196.00 B	1	36.86 kB

Figure 14, CSV Data extracted.



Data_Storage.CSV data

132 DOCUMENTS

1 INDEXES

Type a query: { field: 'value' }

CSV data

	_id ObjectId	Title String	url String	Word count Int32	# of Links Int32
1	ObjectId('648355c82a6d94cf6a3…	"Who's a good AI? Dog-based d…	"https://techcrunch.com/2018/…	635	3
2	ObjectId('648355c82a6d94cf6a3…	"Top 20 PythonAI and MachineL…	"https://www.kdnuggets.com/20…	1184	39
3	ObjectId('648355c82a6d94cf6a3…	" Allegro.AI nabs $11M for 'd…	" https://techcrunch.com/2018…	1864	6
4	ObjectId('648355c82a6d94cf6a3…	" UK report urges action to c…	" https://techcrunch.com/2018…	1741	5
5	ObjectId('648355c82a6d94cf6a3…	" Arm chips with Nvidia AI co…	" https://techcrunch.com/2018…	1864	1
6	ObjectId('648355c82a6d94cf6a3…	" Frank Chen will make you a …	" https://mixpanel.com/blog/2…	1913	5
7	ObjectId('648355c82a6d94cf6a3…	" Frank Lessons on AI from th…	" https://mxpnlcms.wpengine.c…	1007	2
8	ObjectId('648355c82a6d94cf6a3…	" Prisma shifts focus to b2b …	" https://techcrunch.com/2017…	3019	5
9	ObjectId('648355c82a6d94cf6a3…	" What you should know about …	" https://techcrunch.com/2017…	2224	8
10	ObjectId('648355c82a6d94cf6a3…	" HBO's Silicon Valley on A.I…	" https://mxpnlcms.wpengine.c…	1778	4
11	ObjectId('648355c82a6d94cf6a3…	" MIT CSAIL research offers a…	" https://techcrunch.com/2017…	1612	5
12	ObjectId('648355c82a6d94cf6a3…	" Someone scraped 40,000 Tind…	" https://techcrunch.com/2017…	2721	3
13	ObjectId('648355c82a6d94cf6a3…	" Google's smarter, A.I.-powe…	" https://techcrunch.com/2017…	2877	4