Web Scraping and SEO Analysis Tool Documentation

- Snekha.S (Advisory Intern - CEDA)

1. Introduction

This document provides an overview of a **Python-based web scraping tool** that extracts text data from a specified website, performs **SEO analysis**, and stores the results in both a **CSV** file and a **SQLite** database.

2. Objective

The primary goal of this project is to:
- **Scrape** data from a target website.
- Perform **SEO analysis** by counting word occurrences.
- Remove **stopwords** to improve accuracy.
- Save the output to both CSV and SQLite database for future reference.

3. Requirements

To run the project, you need the following libraries installed:
- `**requests**` – To send HTTP requests and fetch website data.
- `**beautifulsoup4**` – To parse HTML content.
- `**pandas**` – To manage and manipulate structured data.
- `**sqlalchemy**` – To handle database interactions.

You can install the required libraries using:
```
**pip install requests beautifulsoup4 pandas sqlalchemy**
```

4. Code Overview

4.1 Target Website

The target website is defined as a constant URL:
```python
URL='https://www.coursera.org/in/articles/what-is-python-used-for-a-beginners-guide-to-using-python'
```

4.2 Scraping Function

The `**scrape_website(url)**` function sends a GET request to the target URL and parses the HTML content using BeautifulSoup:
```
response = requests.get(url)
response.raise_for_status()
```

```
soup = BeautifulSoup(response.text, 'html.parser')
text = soup.get_text()
```

4.3 SEO Analysis

The `**analyze_seo(text)**` function splits the extracted text into individual words and counts the occurrences of each word, excluding common stopwords

4.4 Saving Data

The **`save_data`** function creates a DataFrame from the word count dictionary and saves it to a CSV and a DB.

4.5 Error Handling

If the HTTP request fails, the exception is caught, and an error message is printed:

```
except requests.exceptions.RequestException as e:
    print(f"Error fetching data: {e}")
```

4.6 Execution

The `**main()**` function orchestrates the execution by calling the scrape, analysis, and save functions:

```
if __name__ == "__main__":
    main()
```

5. Output

Upon successful execution, the following files are generated:

- **seo_analysis.csv** – Contains the list of words and their occurrence count.
- **seo_analysis.db** – SQLite database with a table `seo_data` storing the same data.

6. Testing

1. Run the script using:

```
python assignment.py
```

2. Verify that the CSV file and database are created correctly.
3. Check for any missing or incorrect data.