```
pip install requests beautifulsoup4 pandas openpyxl
```

```
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (2.32.3)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (4.12.3)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.1.4)
Requirement already satisfied: openpyxl in /usr/local/lib/python3.10/dist-packages (3.1.5)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests) (2024.7.4)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4) (2.6)
Requirement already satisfied: numpy<2,>=1.22.4 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.1)
Requirement already satisfied: et-xmlfile in /usr/local/lib/python3.10/dist-packages (from openpyxl) (1.1.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
```

```
import requests
from bs4 import BeautifulSoup
import pandas as pd

# Define the list of companies with their websites
companies = [
    {"id": 5875, "name": "Solarkal", "website": "https://www.solarkal.com/"},
    {"id": 11917, "name": "H2Scan", "website": "https://h2scan.com/"},
    {"id": 34005, "name": "Eo Charging", "website": "https://www.eocharging.com/"},
    {"id": 65212, "name": "Prewave", "website": "https://www.prewave.com/"},
    {"id": 18533, "name": "Viriciti", "website": "https://www.chargepoint.com/"},
    {"id": 2805, "name": "EasyMile", "website": "https://www.easymile.com/"},
    {"id": 101741, "name": "Everstream", "website": "https://www.everstream.ai/"},
    {"id": 110133, "name": "Altus Power", "website": "https://www.altuspower.com/"},
    {"id": 12605, "name": "Charm Industrial", "website": "https://www.charmindustrial.com/"},
    {"id": 105894, "name": "Isotropic Systems", "website": "https://www.all.space/"},
    {"id": 400, "name": "Caban Systems", "website": "https://www.cabanenergy.com/"},
    {"id": 34204, "name": "BioBTX", "website": "https://biobtx.com/"},
    {"id": 6134, "name": "Hydrogenious LOHC", "website": "https://hydrogenious.net/"},
    {"id": 12008, "name": "Iogen", "website": "https://www.iogen.com/"},
    {"id": 6997, "name": "Infinited Fiber Company", "website": "https://www.infinitedfiber.com/"}
]

# Function to scrape data for a company
def scrape_company_data(company):
    try:
        response = requests.get(company['website'])
        soup = BeautifulSoup(response.text, 'html.parser')

        # Extract description
        description = soup.find('meta', attrs={'name': 'description'})['content'] if soup.find('meta', attrs={'name': 'description'}) else "N

        # Placeholder data for HQ and Offices, Clients, and News (to be implemented as per website structure)
        hq_offices = "HQ and Offices details to be implemented"
        clients = "Clients details to be implemented"
        news = "News details to be implemented"

        return {
            "Company ID": company['id'],
            "Company Name": company['name'],
            "Website": company['website'],
            "Description": description,
            "HQ and Offices": hq_offices,
            "Clients": clients,
            "News": news
        }
    except Exception as e:
        print(f"Error scraping {company['name']}: {e}")
        return None

# Scrape data for all companies
data = []
for company in companies:
    company_data = scrape_company_data(company)
    if company_data:
        data.append(company_data)

# Create a DataFrame
```

```
df = pd.DataFrame(data)

# Save DataFrame to Excel mimicking SQL table structure
df.to_excel('company_data.xlsx', index=False)

print("Data scraping complete. The results have been saved to 'company_data.xlsx'.")
```

```
Data scraping complete. The results have been saved to 'company_data.xlsx'.
```

```
import pandas as pd

# Read the Excel file
df = pd.read_excel('company_data.xlsx')

# Display the contents
print(df)
```

```
8        12605         Charm Industrial  https://www.charmindustrial.com/
9       105894        Isotropic Systems            https://www.all.space/
10         400            Caban Systems      https://www.cabanenergy.com/
11       34204                   BioBTX               https://biobtx.com/
12        6134        Hydrogenious LOHC         https://hydrogenious.net/
13       12008                   Iogen            https://www.iogen.com/
14        6997  Infinited Fiber Company   https://www.infinitedfiber.com/

                                    Description  \
0   SolarKal is the leading commercial solar advis...
1   H2scan’s proven sensing technology, based on R...
2   Our commercial EV charging infrastructure solu...
3   Supplier monitoring for purchasing, supply cha...
4                           No description available
5   Driverless vehicle solutions and full-service ...
6                           No description available
7   Take control of your sustainability and decarb...
8   Charm Industrial provides high-quality carbon ...
9   ALL.SPACE is revolutionising communications wi...
10  Reimagining how we power the planet. Energy st...
11  We at BioBTX developed a technology to produce...
12  We store and transport hydrogen in a liquid or...
13  Iogen carbon-negative fuel production technolo...
14  Let's make textile circularity an everyday rea...

                           HQ and Offices  \
0   HQ and Offices details to be implemented
1   HQ and Offices details to be implemented
2   HQ and Offices details to be implemented
3   HQ and Offices details to be implemented
4   HQ and Offices details to be implemented
5   HQ and Offices details to be implemented
6   HQ and Offices details to be implemented
7   HQ and Offices details to be implemented
8   HQ and Offices details to be implemented
9   HQ and Offices details to be implemented
10  HQ and Offices details to be implemented
11  HQ and Offices details to be implemented
12  HQ and Offices details to be implemented
13  HQ and Offices details to be implemented
14  HQ and Offices details to be implemented

                           Clients                          News
0   Clients details to be implemented  News details to be implemented
1   Clients details to be implemented  News details to be implemented
2   Clients details to be implemented  News details to be implemented
3   Clients details to be implemented  News details to be implemented
4   Clients details to be implemented  News details to be implemented
5   Clients details to be implemented  News details to be implemented
6   Clients details to be implemented  News details to be implemented
7   Clients details to be implemented  News details to be implemented
8   Clients details to be implemented  News details to be implemented
9   Clients details to be implemented  News details to be implemented
10  Clients details to be implemented  News details to be implemented
11  Clients details to be implemented  News details to be implemented
12  Clients details to be implemented  News details to be implemented
13  Clients details to be implemented  News details to be implemented
14  Clients details to be implemented  News details to be implemented
```

```
from google.colab import files

# Download the file
files.download('company_data.xlsx')
```

Adding

1) Description Extraction

2)HQ and Offices Extraction

3) News Extraction

4) Clients Extraction

```
import requests
from bs4 import BeautifulSoup
import pandas as pd

# Define the list of companies with their websites
companies = [
    {"id": 5875, "name": "Solarkal", "website": "https://www.solarkal.com/"},
    {"id": 11917, "name": "H2Scan", "website": "https://h2scan.com/"},
    {"id": 34005, "name": "Eo Charging", "website": "https://www.eocharging.com/"},
    {"id": 65212, "name": "Prewave", "website": "https://www.prewave.com/"},
    {"id": 18533, "name": "Viriciti", "website": "https://www.chargepoint.com/"},
    {"id": 2805, "name": "EasyMile", "website": "https://www.easymile.com/"},
    {"id": 101741, "name": "Everstream", "website": "https://www.everstream.ai/"},
    {"id": 110133, "name": "Altus Power", "website": "https://www.altuspower.com/"},
    {"id": 12605, "name": "Charm Industrial", "website": "https://www.charmindustrial.com/"},
    {"id": 105894, "name": "Isotropic Systems", "website": "https://www.all.space/"},
    {"id": 400, "name": "Caban Systems", "website": "https://www.cabanenergy.com/"},
    {"id": 34204, "name": "BioBTX", "website": "https://biobtx.com/"},
    {"id": 6134, "name": "Hydrogenious LOHC", "website": "https://hydrogenious.net/"},
    {"id": 12008, "name": "Iogen", "website": "https://www.iogen.com/"},
    {"id": 6997, "name": "Infinited Fiber Company", "website": "https://www.infinitedfiber.com/"}
]

# Function to extract company description
def get_description(soup):
    description = soup.find('meta', attrs={'name': 'description'})
    if description:
        return description.get('content')
    fallback_description = soup.find('section', {'class': 'description'}) or soup.find('div', {'class': 'about-us'})
    return fallback_description.get_text(strip=True) if fallback_description else "No description available"

# Function to extract HQ and office locations
def get_hq_offices(soup):
    offices_section = soup.find('div', {'class': 'office-locations'}) or \
                      soup.find('section', {'id': 'contact'}) or \
                      soup.find(string=lambda t: "office" in t.lower())
    if offices_section:
        if isinstance(offices_section, str):
            return offices_section.strip()
        offices = [office.get_text(strip=True) for office in offices_section.find_all('li')]
        return ', '.join(offices) if offices else offices_section.get_text(strip=True)
    return "No offices found"

# Function to extract clients
def get_clients(soup):
    clients_section = soup.find('div', {'class': 'client-logos'}) or \
                      soup.find('section', {'id': 'clients'}) or \
                      soup.find_all('img', alt=True)
    if clients_section:
        if isinstance(clients_section, list):
            clients = [client.get('alt') for client in clients_section if client.get('alt')]
        else:
            clients = [client.get('alt') for client in clients_section.find_all('img')]
        return ', '.join(clients) if clients else "No clients found"
    return "No clients found"

# Function to extract latest news
def get_news(soup):
```

```
    def get_news(soup):
        news_list = []
        news_section = soup.find('section', {'id': 'news'}) or \
                       soup.find('div', {'class': 'latest-news'}) or \
                       soup.find_all('article')
        if news_section:
            articles = news_section.find_all('article') if hasattr(news_section, 'find_all') else news_section
            for article in articles:
                title = article.find('h2').get_text(strip=True) if article.find('h2') else "No title"
                date = article.find('time').get_text(strip=True) if article.find('time') else "No date"
                url = article.find('a').get('href') if article.find('a') else "No URL"
                summary = article.find('p').get_text(strip=True) if article.find('p') else "No summary"
                news_list.append(f"Title: {title}, Date: {date}, URL: {url}, Summary: {summary}")
        return ' | '.join(news_list) if news_list else "No news found"

    # Function to scrape data for a company
    def scrape_company_data(company):
        try:
            response = requests.get(company['website'])
            soup = BeautifulSoup(response.text, 'html.parser')

            # Extract data
            description = get_description(soup)
            hq_offices = get_hq_offices(soup)
            clients = get_clients(soup)
            news = get_news(soup)

            return {
                "Company ID": company['id'],
                "Company Name": company['name'],
                "Website": company['website'],
                "Description": description,
                "HQ and Offices": hq_offices,
                "Clients": clients,
                "News": news
            }
        except Exception as e:
            print(f"Error scraping {company['name']}: {e}")
            return None

    # Scrape data for all companies
    data = []
    for company in companies:
        company_data = scrape_company_data(company)
        if company_data:
            data.append(company_data)

    # Create a DataFrame
    df2 = pd.DataFrame(data)

    # Save DataFrame to Excel mimicking SQL table structure
    df2.to_excel('company_data_enhanced.xlsx', index=False)

    print("Enhanced data scraping complete. The results have been saved to 'company_data_enhanced.xlsx'.")
```

```
Enhanced data scraping complete. The results have been saved to 'company_data_enhanced.xlsx'.
```

```
    # Read the Excel file
    df2 = pd.read_excel('company_data_enhanced.xlsx')

    # Display the contents
    print(df2)
```

```
5                        No offices found
6                        No offices found
7                        No offices found
8                        No offices found
9                        No offices found
10                       No offices found
11                       No offices found
12                       No offices found
13                       No offices found
14                       No offices found

                                          Clients  \
0                                  No clients found
1   abb, alabamapower, blueorigin, boeing, bp, con...
2   EO Genius, London Bus charging, EO Cloud Lapto...
3   ClickCease, Mercedes-benz-logo, Audi_logo.svg,...
4   Home, Home, Careers-nav-thumb, Laptop displayi...
5   EZTow autonomous tow tractor, Autonomous Termi...
6                                  No clients found
7   Close Icon, Birds eye view of commercial rooft...
8   Charm Industrial Homepage, Frontier, Stripe, A...
9                                  No clients found
10  Caban's hardware, Enduro and Monaco, and softw...
11                                 No clients found
12  Hydrogenious LOHC tanker truck, Hydrogenious L...
13  Iogen Icons Clean Fuel, Iogen Icons Net Zero, ...
14  Infinited Fiber, Infinited Fiber, Infinited Fiber

                                              News
0                                  No news found
1   Title: No title, Date: No date, URL: https://h...
2                                  No news found
3                                  No news found
4   Title: Connect with drivers when they want to ...
5   Title: There Is Money To Be Made In Driverless...
6                                  No news found
7                                  No news found
8   Title: No title, Date: No date, URL: No URL, S...
9                                  No news found
10                                 No news found
11                                 No news found
12  Title: No title, Date: No date, URL: /what/#st...
13                                 No news found
14  Title: Introducing Infinna™– the circular fibe...
```

```
from google.colab import files

# Download the file
files.download('company_data_enhanced.xlsx')
```

GenAi

```
pip install openai
```

```
Collecting openai
  Downloading openai-1.42.0-py3-none-any.whl.metadata (22 kB)
Requirement already satisfied: anyio<5,>=3.5.0 in /usr/local/lib/python3.10/dist-packages (from openai) (3.7.1)
Requirement already satisfied: distro<2,>=1.7.0 in /usr/lib/python3/dist-packages (from openai) (1.7.0)
Collecting httpx<1,>=0.23.0 (from openai)
  Downloading httpx-0.27.0-py3-none-any.whl.metadata (7.2 kB)
Collecting jiter<1,>=0.4.0 (from openai)
  Downloading jiter-0.5.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (3.6 kB)
Requirement already satisfied: pydantic<3,>=1.9.0 in /usr/local/lib/python3.10/dist-packages (from openai) (2.8.2)
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-packages (from openai) (1.3.1)
Requirement already satisfied: tqdm>4 in /usr/local/lib/python3.10/dist-packages (from openai) (4.66.5)
Requirement already satisfied: typing-extensions<5,>=4.11 in /usr/local/lib/python3.10/dist-packages (from openai) (4.12.2)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist-packages (from anyio<5,>=3.5.0->openai) (3.7)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio<5,>=3.5.0->openai) (1.2.2)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from httpx<1,>=0.23.0->openai) (2024.7.4)
Collecting httpcore==1.* (from httpx<1,>=0.23.0->openai)
  Downloading httpcore-1.0.5-py3-none-any.whl.metadata (20 kB)
Collecting h11<0.15,>=0.13 (from httpcore==1.*->httpx<1,>=0.23.0->openai)
  Downloading h11-0.14.0-py3-none-any.whl.metadata (8.2 kB)
Requirement already satisfied: annotated-types>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1.9.0->openai) (0.7.
Requirement already satisfied: pydantic-core==2.20.1 in /usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1.9.0->openai) (2.20.
Downloading openai-1.42.0-py3-none-any.whl (362 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 362.9/362.9 kB 3.4 MB/s eta 0:00:00
Downloading httpx-0.27.0-py3-none-any.whl (75 kB)
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 75.6/75.6 kB 5.4 MB/s eta 0:00:00
Downloading httpcore-1.0.5-py3-none-any.whl (77 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 77.9/77.9 kB 5.2 MB/s eta 0:00:00
Downloading jiter-0.5.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (318 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 318.9/318.9 kB 14.3 MB/s eta 0:00:00
Downloading h11-0.14.0-py3-none-any.whl (58 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.3/58.3 kB 3.7 MB/s eta 0:00:00
Installing collected packages: jiter, h11, httpcore, httpx, openai
Successfully installed h11-0.14.0 httpcore-1.0.5 httpx-0.27.0 jiter-0.5.0 openai-1.42.0
```

```
pip install openai==0.28
```

```
Collecting openai==0.28
  Downloading openai-0.28.0-py3-none-any.whl.metadata (13 kB)
Requirement already satisfied: requests>=2.20 in /usr/local/lib/python3.10/dist-packages (from openai==0.28) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from openai==0.28) (4.66.5)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from openai==0.28) (3.10.5)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai==0.28) (
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai==0.28) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai==0.28) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai==0.28) (2024.7
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai==0.28) (2.4.0)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai==0.28) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai==0.28) (24.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai==0.28) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai==0.28) (6.0.5)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai==0.28) (1.9.4)
Requirement already satisfied: async-timeout<5.0,>=4.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai==0.28) (4.0.3)
Downloading openai-0.28.0-py3-none-any.whl (76 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 76.5/76.5 kB 1.1 MB/s eta 0:00:00
Installing collected packages: openai
  Attempting uninstall: openai
    Found existing installation: openai 1.42.0
    Uninstalling openai-1.42.0:
      Successfully uninstalled openai-1.42.0
Successfully installed openai-0.28.0
WARNING: The following packages were previously imported in this runtime:
  [openai]
You must restart the runtime in order to use newly installed versions.
```

RESTART SESSION

```
import openai
import requests
from bs4 import BeautifulSoup
import pandas as pd

# Initialize OpenAI API (Replace 'your-api-key' with your actual OpenAI API key)
openai.api_key = 'sk-proj-MZbq2mYuJxUWZNE5vJcqZOzhBjiHx7jbyHX12pKyENjmqCX8Jgfj8sLoefT3BlbkFJTIKD4zjsVd5uKNDzspir86HnActmMEf4AqKNH4HNIGKpmO2TnI


# Define the list of companies with their websites
companies = [
    {"id": 5875, "name": "Solarkal", "website": "https://www.solarkal.com/"},
    {"id": 11917, "name": "H2Scan", "website": "https://h2scan.com/"},
    {"id": 34005, "name": "Eo Charging", "website": "https://www.eocharging.com/"},
    {"id": 65212, "name": "Prewave", "website": "https://www.prewave.com/"},
    {"id": 18533, "name": "Viriciti", "website": "https://www.chargepoint.com/"},
    {"id": 2805, "name": "EasyMile", "website": "https://www.easymile.com/"},
    {"id": 101741, "name": "Everstream", "website": "https://www.everstream.ai/"},
    {"id": 110133, "name": "Altus Power", "website": "https://www.altuspower.com/"},
    {"id": 12605, "name": "Charm Industrial", "website": "https://www.charmindustrial.com/"},
    {"id": 105894, "name": "Isotropic Systems", "website": "https://www.all.space/"},
    {"id": 400, "name": "Caban Systems", "website": "https://www.cabanenergy.com/"},
    {"id": 34204, "name": "BioBTX", "website": "https://biobtx.com/"},
    {"id": 6134, "name": "Hydrogenious LOHC", "website": "https://hydrogenious.net/"},
    {"id": 12008, "name": "Iogen", "website": "https://www.iogen.com/"},
    {"id": 6997, "name": "Infinited Fiber Company", "website": "https://www.infinitedfiber.com/"}
]

# Function to call GPT-4 to assist in parsing and summarizing content
def call_gpt4(prompt):

  response = openai.Completion.create(
    engine="gpt-3.5-turbo",
    prompt=prompt,
    max_tokens=150
```

```
        max_tokens=150,
        n=1,
        stop=None,
        temperature=0.7
    )


      return response.choices[0].text.strip()

    # Function to scrape and use GPT-4 for processing
    def scrape_company_data(company):
        try:
            response = requests.get(company['website'])
            soup = BeautifulSoup(response.text, 'html.parser')

            # Generate prompts for GPT-4
            description_prompt = f"Extract and summarize the company description from this HTML:\n\n{soup.prettify()}"
            hq_offices_prompt = f"Extract the headquarters and office locations from this HTML:\n\n{soup.prettify()}"
            clients_prompt = f"List the clients of the company from this HTML:\n\n{soup.prettify()}"
            news_prompt = f"Summarize the latest news articles from this HTML:\n\n{soup.prettify()}"

            # Call GPT-4
            description = call_gpt4(description_prompt)
            hq_offices = call_gpt4(hq_offices_prompt)
            clients = call_gpt4(clients_prompt)
            news = call_gpt4(news_prompt)

            return {
                "Company ID": company['id'],
                "Company Name": company['name'],
                "Website": company['website'],
                "Description": description,
                "HQ and Offices": hq_offices,
                "Clients": clients,
                "News": news
            }
        except Exception as e:
            print(f"Error scraping {company['name']}: {e}")
            return None

    # Scrape data for all companies
    data = []
    for company in companies:
        company_data = scrape_company_data(company)
        if company_data:
            data.append(company_data)

    # Create a DataFrame
    df = pd.DataFrame(data)

    # Save DataFrame to Excel mimicking SQL table structure
    df.to_excel('company_data_with_gpt4.xlsx', index=False)

    print("Data scraping with GPT-4 complete. The results have been saved to 'company_data_with_gpt4.xlsx'.")
```