In [ ]:

```python
from selenium.common.exceptions import NoSuchElementException, WebDriverException
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

import time
import pandas as pd

# Function to click the "Show More" button to expand job details
def click_show_more(driver):
    try:
        show_more_less = WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.CSS_SELECTOR, ".css
    except:
        show_more_less = None

    if show_more_less:
        # Scroll the element into view
        driver.execute_script("arguments[0].scrollIntoView();", show_more_less)
        driver.execute_script("arguments[0].click();", show_more_less)
        return True
    else:
        return False

# Function to click the "Next" button for pagination
def click_next_button(driver):
    try:
        time.sleep(2)
        next_button = WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.XPATH, '//button[@data
        driver.execute_script("arguments[0].scrollIntoView();", next_button)
        driver.execute_script("arguments[0].click();", next_button)
        time.sleep(3)
    except NoSuchElementException:
        print("Reached the last page or unable to find the 'Next' button")
        return False
    return True

# Function to extract job details from a listing
def extract_job_details(listing, driver):
    job_details = {}

    # Extract job title
    try:
         job_title = listing.find_element(By.CLASS_NAME, "job-title").text
    except NoSuchElementException:
         job_title = "N/A"

    # Extract location
    try:
          location = listing.find_element(By.CLASS_NAME, "location").text
    except NoSuchElementException:
          location = "N/A"

    # Extract salary estimate
    try:
         salary_estimate_element = listing.find_element(By.CSS_SELECTOR, 'div.salary-estimate[data-test="detail
         salary_estimate = salary_estimate_element.text
    except NoSuchElementException:
         salary_estimate = "N/A"

    # Extract employer name
    try:
         employer_name_element = driver.find_element(By.XPATH, './/div[@data-test="employerName"]')
         employer_name = employer_name_element.text.strip().split('\n')[0]
    except NoSuchElementException:
        employer_name = "N/A"

    # Extract job description
    job_description = ""
    try:
         job_description_element = driver.find_element(By.CLASS_NAME, "jobDescriptionContent")
         elements = job_description_element.find_elements(By.XPATH, ".//*")
         job_description = "\n".join([element.text for element in elements if element.text])
         job_description = job_description.replace('\n', ' ')  # Remove newlines within job description
    except NoSuchElementException:
         job_description = "N/A"

    # Extract rating
    try:
         rating = driver.find_element(By.CSS_SELECTOR, '[data-test="detailRating"]').text
    except NoSuchElementException:
         rating = "N/A"

    # Create the job details dictionary
    job_details["Location"] = location
    job_details["Job Title"] = job_title
    job_details["Salary Estimate"] = salary_estimate
    job_details["Employer Name"] =  employer_name
    job_details["Job Description"] = job_description.replace('\n', ' ')
    job_details["Rating"] = rating

    # Extract company info
    try:
        company_container = driver.find_element(By.ID, "CompanyContainer")

        # Extract company size
        try:
            size_element = company_container.find_element(By.XPATH, './/span[text()="Size"]/following-sibling::
            size = size_element.text
        except NoSuchElementException:
            size = "N/A"

        # Extract founded year
        try:
            founded_element = company_container.find_element(By.XPATH, './/span[text()="Founded"]/following-sib
            founded = founded_element.text
        except NoSuchElementException:
            founded = "N/A"

        # Extract company type
        try:
            company_type_element = company_container.find_element(By.XPATH, './/span[text()="Type"]/following-s
            company_type = company_type_element.text
        except NoSuchElementException:
            company_type = "N/A"

        # Extract industry
        try:
            industry_element = company_container.find_element(By.XPATH, './/span[text()="Industry"]/following-s
            industry = industry_element.text
        except NoSuchElementException:
            industry = "N/A"

        # Extract sector
        try:
            sector_element = company_container.find_element(By.XPATH, './/span[text()="Sector"]/following-sibli
            sector = sector_element.text
        except NoSuchElementException:
            sector = "N/A"

        # Extract revenue
        try:
            revenue_element = company_container.find_element(By.XPATH, './/span[text()="Revenue"]/following-sib
            revenue = revenue_element.text
        except NoSuchElementException:
            revenue = "N/A"

        # Add company info to job details dictionary
        job_details["Company Size"] = size
        job_details["Founded"] = founded
        job_details["Type"] = company_type
        job_details["Industry"] = industry
        job_details["Sector"] = sector
        job_details["Revenue"] = revenue

    except NoSuchElementException:
        print("Company info not found.")

    return job_details


# Function to initiate the WebDriver with retries
def initiate_webdriver():
    max_attempts = 5  # Maximum number of attempts to establish WebDriver

    for attempt in range(1, max_attempts + 1):
        try:
            options = webdriver.EdgeOptions()
            driver = webdriver.Edge(options=options)
            driver.set_window_size(1120, 1000)
            return driver
        except WebDriverException:
            print(f"Attempt {attempt}/{max_attempts}: WebDriver initiation failed. Retrying...")
            time.sleep(5)  # Wait before retrying

    print(f"Failed to initiate WebDriver after {max_attempts} attempts.")
    return None

# Main function to initiate scraping
def main(job_title):
    print("="*50)
    print("Fetching for: ", job_title)

    driver = initiate_webdriver()
    if driver is None:
        print("Exiting due to WebDriver initiation failure.")
        return []

    # Construct the URL for the Glassdoor job search
    url = f"https://www.glassdoor.com/Job/jobs.htm?suggestCount=0&suggestChosen=false&clickSource=searchBtn&typ
    driver.get(url)

    # Extract total number of pages for pagination
    pagination_footer = driver.find_element(By.CLASS_NAME, "paginationFooter").text
    page_numbers = pagination_footer.split()[-1]

    jobs = []

    for _ in range(int(2)):
        job_listings = driver.find_elements(By.CLASS_NAME, "react-job-listing")

        # Iterate over job listings
        for listing in job_listings:
            driver.execute_script("arguments[0].click();", listing)
            time.sleep(3)
            click_show_more(driver)
            time.sleep(2)
            job_details = extract_job_details(listing, driver)
            if job_details not in jobs:
                print(job_details)
                print("*"*50)
                jobs.append(job_details)

        if not click_next_button(driver):
            break

    driver.quit()  # Close the WebDriver

    return jobs

## Main function call
# job_titles = ['data_scientist', 'data analyst', 'machine learning', 'data engineer', 'business intelligence a

job_titles = ['data_scientist', 'data analyst']

jobs = []

# Scraping job listings and creating DataFrames
for title in job_titles:
    job_listings = main(title)
    job_df = pd.DataFrame(job_listings)

    # Check for duplicate job listings before appending
    for _, row in job_df.iterrows():
        if row.to_dict() not in jobs:
            jobs.append(row.to_dict())

# Create a DataFrame from the combined job listings
jobs_df = pd.DataFrame(jobs)

# Save the combined DataFrame to a CSV file
jobs_df.to_csv('combined_data_jobs.csv', index=False, encoding='utf-8-sig')

print("Jobs data saved to 'combined_data_jobs.csv'")
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js