FINAL_LayoffsFYI_scraper

March 16, 2025

[154]:
```
%%capture
%pip install pandas
%pip install matplotlib
%pip install selenium
%pip install bs4
```

[119]:
```
import requests
import pandas as pd
import matplotlib.pyplot as plt
import time
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.action_chains import ActionChains

import re
import time
```

[129]:
```
driver = webdriver.Chrome()
driver.get('https://airtable.com/app1PaujS9zxVGUZ4/shrqYt5kSqMzHV9R5/
 ↪tbl8c8kanuNB6bPYr?backgroundColor=green&viewControls=on',)
time.sleep(1)
```

[130]:
```
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

try:
    close_button = WebDriverWait(driver, 10).until(
        EC.element_to_be_clickable((By.XPATH, "//
 ↪*[@id='onetrust-close-btn-container']/button"))
    )

    # Click the close button
    close_button.click()
    print("Popup closed successfully!")
```

```
except Exception as e:
    print("No popup found or an error occurred:", str(e))
```

```
Popup closed successfully!
```

[131]:
```
scrollbar = "//*[@id= 'view']/div/div[1]/div[1]/div[3]"
element = driver.find_element(By.XPATH, scrollbar)
```

[132]:
```
leftPane = []
rightPane = []

for _ in range(1240):
    start_time = time.time()  # Start time of the loop

    # Scroll bar
    ActionChains(driver).click_and_hold(element).move_by_offset(0, 1).perform()
    ActionChains(driver).reset_actions()

    html = driver.page_source
    parsed_html = BeautifulSoup(html, "html.parser")

    # Append data
    left_data = parsed_html.find_all('div', class_='dataRow leftPane␣
 ↪rowExpansionEnabled rowSelectionEnabled')
    right_data = parsed_html.find_all('div', class_='dataRow rightPane␣
 ↪rowExpansionEnabled rowSelectionEnabled')
    leftPane.append(left_data)
    rightPane.append(right_data)

    end_time = time.time()
    elapsed_time = end_time - start_time

    #Rate limiting activity
    if elapsed_time < 1/5:
        time.sleep(1/5 - elapsed_time)
```

[140]:
```
matches = {}

# HTML Pattern
pattern = r'<div class="line-height-4 overflow-hidden truncate">(.*?)<\/div>'

for element in leftPane:
    for sub_element in element:
        # Convert the sub-element to a string and search for the pattern
        match = re.search(pattern, str(sub_element))
```

```
        if match:
            key = str(sub_element)[70:100]
            value = str(match.group(1))

            matches[key] = value
```

[141]:
```
data_dict = {}

# Define patterns
url_pattern = r'<span class="url">(.*?)</span>'
common_pattern = r'<div class="flex-auto truncate-pre" title="(.*?)">'
pattern = r'<div class="flex-auto truncate line-height-4 right-align␣
 ↪tabular-nums" style="padding:6px">(.*?)</div>'
date_pattern = r'<div class="truncate css-10jy3hn">(.*?)</div></div></div>'

for i, right_pane_row in enumerate(rightPane):
    for j, right_pane_item in enumerate(right_pane_row):
        right_pane_str = str(right_pane_item)
        key = matches.get(right_pane_str[71:101])

        if key:
            html = right_pane_str
            company_name = matches.get(str(leftPane[i][j])[70:100])

            data_dict[key] = {
                'company_name': company_name,
                'percentage': '',
                'location': '',
                'url': '',
                'industry': '',
                '#layoff': '',
                'stage': '',
                'date': ''
            }

            # Get URL and common info
            url_match = re.search(url_pattern, html)
            if url_match:
                data_dict[key]['url'] = url_match.group(1)

            common_match = re.findall(common_pattern, html)
            if common_match:
                data_dict[key]['location'] = common_match[0]
                if len(common_match) == 4:
                    data_dict[key]['country'] = common_match[3]
                    data_dict[key]['stage'] = common_match[2]
                    data_dict[key]['industry'] = common_match[1]
```

```
                elif len(common_match) == 3:
                    data_dict[key]['country'] = common_match[2]
                else:
                    data_dict[key]['country'] = common_match[4]
                    data_dict[key]['industry'] = common_match[2]

            match = re.findall(pattern, right_pane_str)
            if match:
                if len(match) == 3:
                    data_dict[key]['#layoff'] = match[0]
                    data_dict[key]['percentage'] = match[1]
                    data_dict[key]['raised'] = match[2]
                elif len(match) == 1:
                    data_dict[key]['raised'] = match[0]
                elif len(match) == 2:
                    if '%' in match[0]:
                        data_dict[key]['percentage'] = match[0]
                        data_dict[key]['raised'] = match[1]
                    elif '%' in match[1]:
                        data_dict[key]['#layoff'] = match[0]
                        data_dict[key]['percentage'] = match[1]
                    else:
                        data_dict[key]['#layoff'] = match[0]
                        data_dict[key]['raised'] = match[1]

            date_match = re.search(date_pattern, html)
            if date_match:
                data_dict[key]['date'] = date_match.group(1)
```

[153]:
```
import pandas as pd

df = pd.DataFrame.from_dict(data_dict, orient='index')
df.reset_index(drop=True, inplace=True)

new_column_names = {
    'company_name': 'Company',
    'percentage': 'Percentage',
    'location': 'Location HQ',
    'url': 'URL',
    'industry': 'Industry',
    '#layoff': '# Laid Off',
    'stage': 'Stage',
    'date': 'Date',
    'country': 'Country',
    'raised': '$ Raised (mm)'
}
```

```
df.rename(columns=new_column_names, inplace=True)

if 'Date' in df.columns:
    df['Date'] = pd.to_datetime(df['Date'], errors='coerce')

    # Sort the dataframe by 'Date' in descending order
    df.sort_values(by='Date', ascending=False, inplace=True)

    print("DataFrame sorted by Date in descending order.")
else:
    print("The 'Date' column is not found in the DataFrame.")

df.to_csv("sorted_layoffs_data.csv", index=False)
print("Sorted data saved as 'sorted_layoffs_data.csv'")
```

```
DataFrame sorted by Date in descending order.
Sorted data saved as 'sorted_layoffs_data.csv'
```

[ ]: