1

import requests

from bs4 import BeautifulSoup

def search\_amazon(product):

base\_url = 'https://www.amazon.in/s'

params = {'k': product}

headers = {

'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3'}

response = requests.get(base\_url, params=params, headers=headers)

if response.status\_code == 200:

soup = BeautifulSoup(response.content, 'html.parser')

products = soup.find\_all('div', {'class': 'sg-col-inner'})

for product in products:

product\_name = product.find('span', {'class': 'a-size-medium'}).text.strip() if product.find('span', {'class': 'a-size-medium'}) else "-"

product\_price = product.find('span', {'class': 'a-price-whole'}).text.strip() if product.find('span', {'class': 'a-price-whole'}) else "-"

product\_rating = product.find('span', {'class': 'a-icon-alt'}).text.strip() if product.find('span', {'class': 'a-icon-alt'}) else "-"

product\_reviews = product.find('span', {'class': 'a-size-base'}).text.strip() if product.find('span', {'class': 'a-size-base'}) else "-"

print("Product:", product\_name)

print("Price:", product\_price)

print("Rating:", product\_rating)

print("Reviews:", product\_reviews)

print("\n")

else:

print("Failed to retrieve search results.")

if \_\_name\_\_ == "\_\_main\_\_":

user\_input = input("Enter the product you want to search on Amazon.in: ")

search\_amazon(user\_input)

2.

import requests

from bs4 import BeautifulSoup

import pandas as pd

def scrape\_product\_details(product\_name):

base\_url = 'https://www.amazon.in/s'

headers = {

'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3'}

all\_products = []

for page\_num in range(1, 4):

params = {'k': product\_name, 'page': page\_num}

response = requests.get(base\_url, params=params, headers=headers)

if response.status\_code == 200:

soup = BeautifulSoup(response.content, 'html.parser')

products = soup.find\_all('div', {'class': 'sg-col-inner'})

for product in products:

details = {}

details['Brand Name'] = product.find('span', {'class': 'a-size-base-plus a-color-base a-text-normal'}).text.strip() if product.find('span', {'class': 'a-size-base-plus a-color-base a-text-normal'}) else "-"

details['Name of the Product'] = product.find('span', {'class': 'a-size-base-plus a-color-base a-text-normal'}).text.strip() if product.find('span', {'class': 'a-size-base-plus a-color-base a-text-normal'}) else "-"

details['Price'] = product.find('span', {'class': 'a-price-whole'}).text.strip() if product.find('span', {'class': 'a-price-whole'}) else "-"

details['Return/Exchange'] = product.find('span', {'class': 'a-text-bold'}).text.strip() if product.find('span', {'class': 'a-text-bold'}) else "-"

details['Expected Delivery'] = product.find('span', {'class': 'a-text-bold'}).text.strip() if product.find('span', {'class': 'a-text-bold'}) else "-"

details['Availability'] = product.find('span', {'class': 'a-size-base'}).text.strip() if product.find('span', {'class': 'a-size-base'}) else "-"

details['Product URL'] = 'https://www.amazon.in' + product.find('a', {'class': 'a-link-normal a-text-normal'})['href'] if product.find('a', {'class': 'a-link-normal a-text-normal'}) else "-"

all\_products.append(details)

else:

print(f"Failed to retrieve search results for page {page\_num}")

return all\_products

def save\_to\_csv(product\_name, products):

df = pd.DataFrame(products)

df.to\_csv(f'{product\_name}\_products.csv', index=False)

print("Data saved to CSV successfully.")

if \_\_name\_\_ == "\_\_main\_\_":

user\_input = input("Enter the product you want to search on Amazon.in: ")

products\_data = scrape\_product\_details(user\_input)

save\_to\_csv(user\_input, products\_data)

3

from selenium import webdriver

from selenium.webdriver.common.keys import Keys

import time

import os

import requests

from bs4 import BeautifulSoup

def scrape\_images(keyword, num\_images):

driver = webdriver.Chrome(executable\_path="chromedriver.exe") # Provide path to your chromedriver

driver.get("https://images.google.com/")

search\_bar = driver.find\_element\_by\_name("q")

search\_bar.clear()

search\_bar.send\_keys(keyword)

search\_bar.send\_keys(Keys.RETURN)

last\_height = driver.execute\_script("return document.body.scrollHeight")

while True:

driver.execute\_script("window.scrollTo(0, document.body.scrollHeight);")

time.sleep(2)

new\_height = driver.execute\_script("return document.body.scrollHeight")

if new\_height == last\_height:

break

last\_height = new\_height

image\_urls = set()

soup = BeautifulSoup(driver.page\_source, "html.parser")

for img in soup.find\_all("img"):

if img.has\_attr("src"):

image\_urls.add(img["src"])

image\_urls = [url for url in image\_urls if not url.startswith("data:image")]

os.makedirs(keyword, exist\_ok=True)

for i, url in enumerate(image\_urls[:num\_images]):

try:

response = requests.get(url)

with open(os.path.join(keyword, f"{keyword}\_{i+1}.jpg"), "wb") as f:

f.write(response.content)

print(f"Downloaded image {i+1} for '{keyword}'")

except Exception as e:

print(f"Failed to download image {i+1} for '{keyword}': {e}")

driver.quit()

if \_\_name\_\_ == "\_\_main\_\_":

keywords = ['fruits', 'cars', 'Machine Learning', 'Guitar', 'Cakes']

num\_images\_per\_keyword = 10

for keyword in keywords:

print(f"Scraping images for '{keyword}'...")

scrape\_images(keyword, num\_images\_per\_keyword)

print()

print("Scraping complete.")

4

import requests

from bs4 import BeautifulSoup

import pandas as pd

def scrape\_flipkart\_smartphones(keyword):

base\_url = f"https://www.flipkart.com/search?q={keyword}&otracker=search&otracker1=search&marketplace=FLIPKART&as-show=on&as=off"

headers = {

'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3'}

response = requests.get(base\_url, headers=headers)

if response.status\_code == 200:

soup = BeautifulSoup(response.content, 'html.parser')

products = soup.find\_all('div', {'class': '\_1AtVbE'})

results = []

for product in products:

details = {}

details['Brand Name'] = product.find('div', {'class': '\_4rR01T'}).text.strip() if product.find('div', {'class': '\_4rR01T'}) else "-"

details['Smartphone Name'] = product.find('a', {'class': 'IRpwTa'}).text.strip() if product.find('a', {'class': 'IRpwTa'}) else "-"

details['Colour'] = product.find('a', {'class': '\_1WPlpC'}).text.strip() if product.find('a', {'class': '\_1WPlpC'}) else "-"

details['RAM'] = [spec.text.strip() for spec in product.find\_all('li', {'class': 'rgWa7D'}) if 'RAM' in spec.text.strip()][0] if any('RAM' in spec.text.strip() for spec in product.find\_all('li', {'class': 'rgWa7D'})) else "-"

details['Storage(ROM)'] = [spec.text.strip() for spec in product.find\_all('li', {'class': 'rgWa7D'}) if 'ROM' in spec.text.strip()][0] if any('ROM' in spec.text.strip() for spec in product.find\_all('li', {'class': 'rgWa7D'})) else "-"

details['Primary Camera'] = [spec.text.strip() for spec in product.find\_all('li', {'class': 'rgWa7D'}) if 'MP' in spec.text.strip() and 'Camera' in spec.text.strip()][0] if any('MP' in spec.text.strip() and 'Camera' in spec.text.strip() for spec in product.find\_all('li', {'class': 'rgWa7D'})) else "-"

details['Secondary Camera'] = [spec.text.strip() for spec in product.find\_all('li', {'class': 'rgWa7D'}) if 'MP' in spec.text.strip() and 'Front' in spec.text.strip()][0] if any('MP' in spec.text.strip() and 'Front' in spec.text.strip() for spec in product.find\_all('li', {'class': 'rgWa7D'})) else "-"

details['Display Size'] = [spec.text.strip() for spec in product.find\_all('li', {'class': 'rgWa7D'}) if 'inch' in spec.text.strip()][0] if any('inch' in spec.text.strip() for spec in product.find\_all('li', {'class': 'rgWa7D'})) else "-"

details['Battery Capacity'] = [spec.text.strip() for spec in product.find\_all('li', {'class': 'rgWa7D'}) if 'mAh' in spec.text.strip()][0] if any('mAh' in spec.text.strip() for spec in product.find\_all('li', {'class': 'rgWa7D'})) else "-"

details['Price'] = product.find('div', {'class': '\_30jeq3'}).text.strip() if product.find('div', {'class': '\_30jeq3'}) else "-"

details['Product URL'] = 'https://www.flipkart.com' + product.find('a', {'class': 'IRpwTa'})['href'] if product.find('a', {'class': 'IRpwTa'}) else "-"

results.append(details)

return results

else:

print("Failed to retrieve search results.")

return []

def save\_to\_csv(keyword, results):

df = pd.DataFrame(results)

df.to\_csv(f'{keyword}\_smartphones.csv', index=False)

print("Data saved to CSV successfully.")

if \_\_name\_\_ == "\_\_main\_\_":

keyword = input("Enter the smartphone you want to search on Flipkart: ")

results = scrape\_flipkart\_smartphones(keyword)

if results:

save\_to\_csv(keyword, results)

5

from selenium import webdriver

from selenium.webdriver.common.keys import Keys

import time

def scrape\_coordinates(city\_name):

driver = webdriver.Chrome(executable\_path="chromedriver.exe")

driver.maximize\_window()

driver.get("https://www.google.com/maps")

search\_bar = driver.find\_element\_by\_css\_selector("input[aria-label='Search Google Maps']")

search\_bar.clear()

search\_bar.send\_keys(city\_name)

search\_bar.send\_keys(Keys.RETURN)

time.sleep(5)

current\_url = driver.current\_url

driver.quit()

if "/@" in current\_url:

coordinates\_index = current\_url.index("/@") + 2

coordinates\_string = current\_url[coordinates\_index:].split(",")[0]

latitude = coordinates\_string.split(",")[0]

longitude = coordinates\_string.split(",")[1]

return latitude, longitude

else:

print("Failed to scrape coordinates. Please try again.")

return None, None

if \_\_name\_\_ == "\_\_main\_\_":

city = input("Enter the name of the city to search on Google Maps: ")

latitude, longitude = scrape\_coordinates(city)

if latitude and longitude:

print(f"Coordinates for {city}: Latitude {latitude}, Longitude {longitude}")

6

import requests

from bs4 import BeautifulSoup

import pandas as pd

def scrape\_gaming\_laptops():

url = "https://www.digit.in/top-products/best-gaming-laptops-40.html"

headers = {

"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3"

}

response = requests.get(url, headers=headers)

if response.status\_code == 200:

soup = BeautifulSoup(response.content, "html.parser")

laptops = soup.find\_all("div", class\_="TopNumbeHeading sticky-footer")

laptop\_details = []

for laptop in laptops:

details = {}

details["Name"] = laptop.find("div", class\_="heading-wraper").text.strip()

details["Price"] = laptop.find("td", class\_="smprice").text.strip()

details["Specifications"] = laptop.find("div", class\_="Section-center").text.strip()

laptop\_details.append(details)

return laptop\_details

else:

print("Failed to retrieve data from the website.")

return []

def save\_to\_csv(data):

df = pd.DataFrame(data)

df.to\_csv("gaming\_laptops.csv", index=False)

print("Data saved to CSV successfully.")

if \_\_name\_\_ == "\_\_main\_\_":

gaming\_laptops\_data = scrape\_gaming\_laptops()

if gaming\_laptops\_data:

save\_to\_csv(gaming\_laptops\_data)

7

import requests

from bs4 import BeautifulSoup

import pandas as pd

def scrape\_forbes\_billionaires():

url = "https://www.forbes.com/billionaires/"

headers = {

"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3"

}

response = requests.get(url, headers=headers)

if response.status\_code == 200:

soup = BeautifulSoup(response.content, "html.parser")

billionaires = soup.find\_all("div", class\_="personName")

billionaires\_data = []

for billionaire in billionaires:

details = {}

details["Rank"] = billionaire.find\_previous("div", class\_="rank").text.strip()

details["Name"] = billionaire.text.strip()

details["Net Worth"] = billionaire.find\_next("div", class\_="netWorth").text.strip()

details["Age"] = billionaire.find\_next("div", class\_="age").text.strip()

details["Citizenship"] = billionaire.find\_next("div", class\_="countryOfCitizenship").text.strip()

details["Source"] = billionaire.find\_next("div", class\_="source-column").text.strip()

details["Industry"] = billionaire.find\_next("div", class\_="category").text.strip()

billionaires\_data.append(details)

return billionaires\_data

else:

print("Failed to retrieve data from the website.")

return []

def save\_to\_csv(data):

df = pd.DataFrame(data)

df.to\_csv("forbes\_billionaires.csv", index=False)

print("Data saved to CSV successfully.")

if \_\_name\_\_ == "\_\_main\_\_":

billionaires\_data = scrape\_forbes\_billionaires()

if billionaires\_data:

save\_to\_csv(billionaires\_data)

8

from googleapiclient.discovery import build

import datetime

API\_KEY = "YOUR\_API\_KEY"

def extract\_comments(video\_id, max\_results=500):

youtube = build("youtube", "v3", developerKey=API\_KEY)

comments = []

next\_page\_token = None

total\_results = 0

while total\_results < max\_results:

request = youtube.commentThreads().list(

part="snippet",

videoId=video\_id,

maxResults=min(100, max\_results - total\_results),

pageToken=next\_page\_token if next\_page\_token else ""

)

response = request.execute()

for item in response["items"]:

comment = item["snippet"]["topLevelComment"]["snippet"]["textDisplay"]

comment\_time = item["snippet"]["topLevelComment"]["snippet"]["publishedAt"]

comment\_time = datetime.datetime.strptime(comment\_time, "%Y-%m-%dT%H:%M:%SZ").strftime("%Y-%m-%d %H:%M:%S")

comments.append({"Comment": comment, "Time": comment\_time})

total\_results += 1

next\_page\_token = response.get("nextPageToken")

if not next\_page\_token:

break

return comments

if \_\_name\_\_ == "\_\_main\_\_":

video\_id = input("Enter the YouTube video ID: ")

comments = extract\_comments(video\_id)

print(f"Total comments extracted: {len(comments)}")

for idx, comment in enumerate(comments, start=1):

print(f"\nComment {idx}:")

print(f"Text: {comment['Comment']}")

print(f"Time: {comment['Time']}")

9

import requests

from bs4 import BeautifulSoup

import pandas as pd

def scrape\_hostels\_in\_london():

url = "https://www.hostelworld.com/findabed.php/ChosenCity.London/ChosenCountry.England"

headers = {

"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3"

}

response = requests.get(url, headers=headers)

if response.status\_code == 200:

soup = BeautifulSoup(response.content, "html.parser")

hostels = soup.find\_all("div", class\_="fabresult")

hostel\_data = []

for hostel in hostels:

details = {}

details["Hostel Name"] = hostel.find("h2", class\_="title-5").text.strip()

details["Distance from City Centre"] = hostel.find("span", class\_="description").text.strip()

details["Ratings"] = hostel.find("div", class\_="score orange big").text.strip()

details["Total Reviews"] = hostel.find("div", class\_="reviews").text.strip().split()[0]

details["Overall Reviews"] = hostel.find("div", class\_="keyword").text.strip()

details["Privates from Price"] = hostel.find("span", class\_="price title-5").text.strip().split()[0]

details["Dorms from Price"] = hostel.find("span", class\_="price").text.strip().split()[0]

details["Facilities"] = ", ".join([item.text.strip() for item in hostel.find\_all("div", class\_="facilities")])

details["Property Description"] = hostel.find("div", class\_="rating-factors prop-card-tablet rating-factors small").text.strip()

hostel\_data.append(details)

return hostel\_data

else:

print("Failed to retrieve data from the website.")

return []

def save\_to\_csv(data):

df = pd.DataFrame(data)

df.to\_csv("hostels\_in\_london.csv", index=False)

print("Data saved to CSV successfully.")

if \_\_name\_\_ == "\_\_main\_\_":

hostels\_data = scrape\_hostels\_in\_london()

if hostels\_data:

save\_to\_csv(hostels\_data)