Code Snippet Edited

import requests

url = "<https://jsearch.p.rapidapi.com/search>"

querystring = {"query":"cybersecurity jobs on glassdoor","page":"1","num\_pages":"1","date\_posted":"all"}

headers = {

    "x-rapidapi-key": "5a87c59c2cmsh24ea8471a41758ep15cb7bjsn258ed147e434",

    "x-rapidapi-host": "jsearch.p.rapidapi.com"

}

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

print(response.json())

**INITIAL CODE:**

**Taken from Jsearch API to scrape for cybersecurity jobs on the website "glassdoor". VS Code was used to run all scripts.**

**Summary of the Code Flow:**

1. **Imports the requests library** to make HTTP requests.
2. **Defines the API endpoint** (url) you want to query.
3. **Sets up the query parameters** (querystring) that specify the search details (e.g., job type, page number).
4. **Adds the necessary headers** (headers), including your API key for authentication.
5. **Sends the HTTP GET request** to the API using requests.get() with the URL, headers, and parameters.
6. **Prints the JSON response** from the API in a readable format using .json().

Json/ CSV File Breakdown Edited

**Code ouput was running in terminal only**:

Your script is making a successful request to the RapidAPI endpoint to retrieve "cybersecurity jobs on Glassdoor." However, it prints the results directly to the terminal. If you want to see the output in the left panel (such as saving it into a file), here’s how you can modify the code to save the data into a file.

**Modify the Script to Save the JSON Response to a File**

You can save the response JSON into a file (e.g., cybersecurity\_jobs.json) in your project directory. Here's the updated version of your script:

import requests

import json

# Define the API endpoint and query parameters

url = "<https://jsearch.p.rapidapi.com/search>"

querystring = {"query":"cybersecurity jobs on glassdoor","page":"1","num\_pages":"1","date\_posted":"all"}

# Add your RapidAPI headers

headers = {

    "x-rapidapi-key": "5a87c59c2cmsh24ea8471a41758ep15cb7bjsn258ed147e434",

    "x-rapidapi-host": "jsearch.p.rapidapi.com"

}

# Make the request to the API

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

# Check if the request was successful (status code 200)

if response.status\_code == 200:

    # Parse the JSON response

    data = response.json()

    # Save the JSON response to a file

    with open("cybersecurity\_jobs.json", "w") as json\_file:

        json.dump(data, json\_file, indent=4)  # Pretty print with indentation for readability

    print("Data saved to 'cybersecurity\_jobs.json'")

else:

    print(f"Failed to retrieve data. Status code: {response.status\_code}")

**WHAT CHANGED:**

1. **Importing the json library**: To save the response in JSON format.
2. **json.dump()**: This function is used to write the JSON response into a file.
3. **File Creation**: The JSON response is saved to a file named cybersecurity\_jobs.json in your current directory.

**WHAT WILL HAPPEN:**

* When you run the script, it will fetch the job data from RapidAPI.
* Instead of just printing the data to the terminal, it will create a file cybersecurity\_jobs.json in your project directory. This file should appear in the left panel in VS Code (the file explorer).

CSV Breakdown/Execution Edited

**To also export the data to a CSV file, you can modify the script further by using Python's csv module to write the relevant job information into a CSV format.**

**Here's how to modify the script to export the data to a CSV file:**

**Updated Script for CSV Export**

import requests

import json

import csv

# Define the API endpoint and query parameters

url = "<https://jsearch.p.rapidapi.com/search>"

querystring = {"query":"cybersecurity jobs on glassdoor","page":"1","num\_pages":"1","date\_posted":"all"}

# Add your RapidAPI headers

headers = {

    "x-rapidapi-key": "5a87c59c2cmsh24ea8471a41758ep15cb7bjsn258ed147e434",

    "x-rapidapi-host": "jsearch.p.rapidapi.com"

}

# Make the request to the API

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

# Check if the request was successful (status code 200)

if response.status\_code == 200:

    # Parse the JSON response

    data = response.json()

    # Save the JSON response to a file

    with open("cybersecurity\_jobs.json", "w") as json\_file:

        json.dump(data, json\_file, indent=4)  # Pretty print with indentation for readability

    print("Data saved to 'cybersecurity\_jobs.json'")

    # Export relevant job data to CSV

    job\_listings = data.get("data", [])  # Assuming the job listings are stored in the "data" key

    if job\_listings:

        # Define the CSV file and headers

        with open("cybersecurity\_jobs.csv", "w", newline="", encoding="utf-8") as csv\_file:

            csv\_writer = csv.writer(csv\_file)

            # Write the CSV header

            csv\_writer.writerow(["Job Title", "Company", "Location", "Date Posted", "Job Description", "URL"])

            # Write the job data to the CSV file

            for job in job\_listings:

                job\_title = job.get("title", "N/A")

                company = job.get("company\_name", "N/A")

                location = job.get("location", "N/A")

                date\_posted = job.get("posted\_at", "N/A")

                description = job.get("description", "N/A")

                url = job.get("url", "N/A")

                # Write the row to CSV

                csv\_writer.writerow([job\_title, company, location, date\_posted, description, url])

        print("Data exported to 'cybersecurity\_jobs.csv'")

    else:

        print("No job listings found in the response.")

else:

    print(f"Failed to retrieve data. Status code: {response.status\_code}")

**Key Changes:**

* 1. **Importing the csv module**: This will allow us to write the job data to a CSV file.
  2. **Fetching the relevant data**: We're pulling specific fields like the job title, company name, location, date posted, description, and URL from the JSON response.
  3. **Writing to CSV**: We're opening a CSV file (cybersecurity\_jobs.csv) and writing the job data into it row by row, with headers.

**CSV Fields:**

* 1. **Job Title**: The title of the job.
  2. **Company**: The name of the company offering the job.
  3. **Location**: Where the job is located.
  4. **Date Posted**: When the job was posted.
  5. **Job Description**: A brief description of the job.
  6. **URL**: The link to the job listing.

**What Will Happen:**

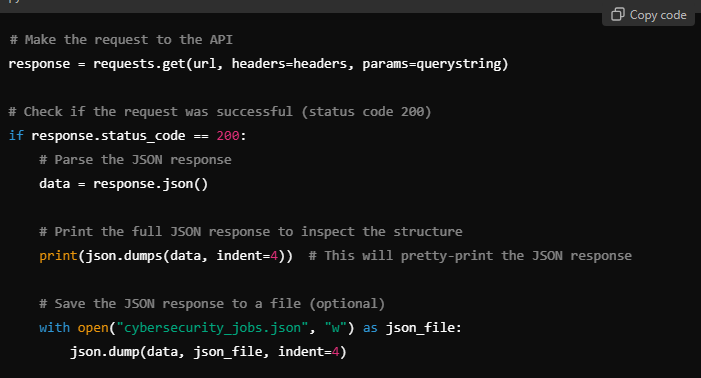
* 1. After the API request is completed, the script will now:
     1. Save the JSON response to cybersecurity\_jobs.json.
     2. Create and write the job listings to cybersecurity\_jobs.csv.

Both files should appear in your VS Code file explorer, and you’ll have the job details in a CSV file.

**\*\*\*At this point, the CSV file in VS CODE was returning missing values, Troubleshooting for that below:**

**Steps to Debug:**

**Print the JSON Response Structure**: Print the structure of the data variable to understand how the job listings are formatted. You can add a print(data) statement after the API call like this:



* 1. **Look for Relevant Fields**: Once you inspect the JSON response, locate the correct keys where job title, company, location, and other details are stored. They might have different names than we originally assumed (for example, job\_title instead of title).
  2. **Update the Key Names**: After identifying the correct field names in the JSON structure, update the job.get() statements in the CSV-writing part of the code.

**ASKED GPT TO GIVE ME THE FULL CODE TO INCLUDE THE EDITED SNIPPET:**

import requests

import json

import csv

# API request setup

url = "<https://jsearch.p.rapidapi.com/search>"

querystring = {"query": "cybersecurity jobs on glassdoor", "page": "1", "num\_pages": "1", "date\_posted": "all"}

headers = {

    "x-rapidapi-key": "5a87c59c2cmsh24ea8471a41758ep15cb7bjsn258ed147e434",

    "x-rapidapi-host": "jsearch.p.rapidapi.com"

}

# Make the request to the API

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

# Check if the request was successful (status code 200)

if response.status\_code == 200:

    # Parse the JSON response

    data = response.json()

    # Pretty print the JSON response to inspect the structure

    print(json.dumps(data, indent=4))  # Inspect the structure of the response

    # Save the JSON response to a file (optional, for later review)

    with open("cybersecurity\_jobs.json", "w") as json\_file:

        json.dump(data, json\_file, indent=4)

    # Open or create a CSV file to write job data

    with open("cybersecurity\_jobs.csv", mode="w", newline="", encoding="utf-8") as file:

        writer = csv.writer(file)

        # Write the headers

        writer.writerow(["Job Title", "Company", "Location", "Date Posted", "Job Description", "URL"])

        # Assuming the jobs data is in the 'data' key and 'jobs' key inside

        if 'data' in data:

            jobs = data['data']

            # Iterate over the job results

            for job in jobs:

                # Extract the relevant information from each job

                title = job.get('job\_title', 'N/A')  # Example key: Update based on actual JSON structure

                company = job.get('employer\_name', 'N/A')  # Example key

                location = job.get('location', 'N/A')  # Example key

                date\_posted = job.get('date\_posted', 'N/A')  # Example key

                job\_description = job.get('description', 'N/A')  # Example key

                job\_url = job.get('job\_apply\_link', 'N/A')  # Example key for the job URL

                # Write the row to the CSV

                writer.writerow([title, company, location, date\_posted, job\_description, job\_url])

        else:

            print("No job data found in the response.")

else:

    print(f"Failed to retrieve data: {response.status\_code}")

**Key Additions and Changes:**

* 1. **Pretty-print the JSON response**:
     1. **The print(json.dumps(data, indent=4)) line will output the entire JSON response with indentation, making it easier to inspect the structure of the data.**
  2. **CSV File Writing**:
     1. **The code writes the headers ["Job Title", "Company", "Location", "Date Posted", "Job Description", "URL"] into the CSV file.**
     2. **It then loops through the jobs list (assumed to be inside the data object), extracting relevant fields like job\_title, employer\_name, and location.**
  3. **Adjust Field Names**:
     1. **The keys used in job.get() are examples and may need to be updated based on the actual structure of the JSON response after you inspect it.**
  4. **Handle Missing Data**:
     1. **If any key does not exist for a job entry, the default value 'N/A' will be used.**

**STILL RETURNING MISSING FIELDS, PROMPTED GPT AGAIN FOR UPDATED CODE:**

It seems that some fields like "Location," "Date Posted," and "Job Description" are coming back as N/A in your CSV. This could mean the keys for these fields are either missing or named differently in the response.

Here's how you can modify the script to further inspect and potentially correct the missing fields by updating the key names if necessary.

**Updated Script to Handle Missing Fields:**

import requests

import json

import csv

# API request setup

url = "<https://jsearch.p.rapidapi.com/search>"

querystring = {"query": "cybersecurity jobs on glassdoor", "page": "1", "num\_pages": "1", "date\_posted": "all"}

headers = {

    "x-rapidapi-key": "5a87c59c2cmsh24ea8471a41758ep15cb7bjsn258ed147e434",

    "x-rapidapi-host": "jsearch.p.rapidapi.com"

}

# Make the request to the API

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

# Check if the request was successful (status code 200)

if response.status\_code == 200:

    # Parse the JSON response

    data = response.json()

    # Pretty print the JSON response to inspect the structure

    print(json.dumps(data, indent=4))  # Inspect the structure of the response

    # Save the JSON response to a file (optional, for later review)

    with open("cybersecurity\_jobs.json", "w") as json\_file:

        json.dump(data, json\_file, indent=4)

    # Open or create a CSV file to write job data

    with open("cybersecurity\_jobs.csv", mode="w", newline="", encoding="utf-8") as file:

        writer = csv.writer(file)

        # Write the headers

        writer.writerow(["Job Title", "Company", "Location", "Date Posted", "Job Description", "URL"])

        # Assuming the jobs data is in the 'data' key and 'jobs' key inside

        if 'data' in data:

            jobs = data['data']

            # Iterate over the job results

            for job in jobs:

                # Extract the relevant information from each job

                title = job.get('job\_title', 'N/A')  # Example key: Update based on actual JSON structure

                company = job.get('employer\_name', 'N/A')  # Example key

                location = job.get('location', 'N/A')  # Example key

                date\_posted = job.get('date\_posted', 'N/A')  # Example key

                job\_description = job.get('description', 'N/A')  # Example key

                job\_url = job.get('job\_apply\_link', 'N/A')  # Example key for the job URL

                # Write the row to the CSV

                writer.writerow([title, company, location, date\_posted, job\_description, job\_url])

        else:

            print("No job data found in the response.")

else:

    print(f"Failed to retrieve data: {response.status\_code}")

**Steps to Resolve the Missing Fields:**

* 1. **Inspect the Printed JSON**:
     1. **After running this script, inspect the pretty-printed JSON output in the terminal. Look for fields related to location, date\_posted, and description. You may find that these fields have different key names.**
  2. **Update Key Names**:
     1. **Once you see the correct keys for those fields in the JSON response, update the corresponding job.get() lines in the script.**

**STILL RETURNING MISSING FIELDS: FURTHER PROMPTING TO UPDATE CODE:**

It looks like the "Location," "Date Posted," and "Job Description" fields are still returning N/A. This is likely because the actual API response does not contain these fields under the expected key names, or they may not be available for every job listing.

To solve this, you'll need to verify the exact key names for these fields from the JSON response, adjust the script to match, and ensure that if the fields are missing, it defaults to N/A properly.

**Updated Plan:**

* 1. **Inspect the JSON Response**: After making the request, print out the entire response so we can see the structure.
  2. **Update Key Names**: Adjust the script based on what fields you find in the JSON response for location, date posted, and job description.

Here’s the updated script to help you inspect the response:

**Code to Inspect JSON Response and Adjust CSV Export**

import requests

import json

import csv

# API request setup

url = "<https://jsearch.p.rapidapi.com/search>"

querystring = {"query": "cybersecurity jobs on glassdoor", "page": "1", "num\_pages": "1", "date\_posted": "all"}

headers = {

    "x-rapidapi-key": "5a87c59c2cmsh24ea8471a41758ep15cb7bjsn258ed147e434",

    "x-rapidapi-host": "jsearch.p.rapidapi.com"

}

# Make the request to the API

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

# Check if the request was successful (status code 200)

if response.status\_code == 200:

    # Parse the JSON response

    data = response.json()

    # Pretty print the JSON response to inspect the structure

    print(json.dumps(data, indent=4))  # Inspect the structure of the response

    # Save the JSON response to a file (optional, for later review)

    with open("cybersecurity\_jobs.json", "w") as json\_file:

        json.dump(data, json\_file, indent=4)

    # Open or create a CSV file to write job data

    with open("cybersecurity\_jobs.csv", mode="w", newline="", encoding="utf-8") as file:

        writer = csv.writer(file)

        # Write the headers

        writer.writerow(["Job Title", "Company", "Location", "Date Posted", "Job Description", "URL"])

        # Assuming the jobs data is in the 'data' key

        if 'data' in data:

            jobs = data['data']

            # Iterate over the job results

            for job in jobs:

                # Extract the relevant information from each job

                title = job.get('job\_title', 'N/A')

                company = job.get('employer\_name', 'N/A')

                location = job.get('location', 'N/A')  # You need to inspect and adjust this key

                date\_posted = job.get('date\_posted', 'N/A')  # Inspect and adjust this key

                job\_description = job.get('description', 'N/A')  # Inspect and adjust this key

                job\_url = job.get('job\_apply\_link', 'N/A')

                # Write the row to the CSV

                writer.writerow([title, company, location, date\_posted, job\_description, job\_url])

        else:

            print("No job data found in the response.")

else:

    print(f"Failed to retrieve data: {response.status\_code}")

**Next Steps:**

* 1. **Run the script and inspect the print(json.dumps(data, indent=4)) output**.
  2. **Check the actual keys used for fields like location, date\_posted, and description.**
  3. **Replace job.get('location', 'N/A') with the correct key you find in the JSON.**

**Once you confirm the structure, the N/A values should be replaced with the correct data from the API response.**

**NOTE**: At this point, I encountered a few errors/mix-ups:

* 1. The CSV file ended up showing N/A in the location, date posted and job description fields in excel, even though the Json file extracted the information.

* 1. The headers in excel did not show up all the necessary criteria that was asked for in the assignment

* 1. Made a mistake editing the code, ended up getting confusing data outputs so had to modify it to re-run the desired information.

**TROUBLESHOOTING:** These are the changes I made to get the final code script:

* 1. ADJUST HEADERS: Added in the additional header fields in the "writer.writerow" section in the code so it shows exactly what I want in excel file (removed job description after inspecting the Json because it was outputting unfinished/unclear sentences, and missing information in other areas).

* 1. ITERATE OVER THE JOB RESULTS: Modified to include city, state, country for the location

* 1. EXTRACT: Skills, responsibilities, experience, qualifications, remote, on site or contactor( previous CSV did not include this)

 Production-Final Scraper

7:47pm 11/09/2024

**This currently fulfils the criteria for: job title, company, city, state, country, date posted, skills, experience and responsibilities.**

**\*\*\*Moved code from testbackup1.py to prod.py and this outputs to cybersecurity\_jobs\_101\_v3.csv**

**Next Steps:**

* **Separate experience and responsibilities into two separate columns (currently the 'experience' data is lumped in with the responsibilties section)**
* **Remove odd characters from the skills and qualifications columns**
* **Get the Remote, On Site, Contractor fields to show the accurate information (currently showing yes/no only even though the data states which posting is full time, remote etc)**

import requests

import json

import csv

from datetime import datetime

# Helper function to extract skills and qualifications

def extract\_skills(job\_data):

    potential\_keys = ['job\_required\_skills', 'skills', 'certifications', 'qualifications', 'job\_description']

    collected\_skills = []

    for key in potential\_keys:

        content = job\_data.get(key)

        if content:

            if isinstance(content, list):

                collected\_skills.extend(content)  # Extend list if content is a list

            elif 'qualification' in key.lower() or 'skill' in key.lower() or 'description' in key.lower():

                lines = content.split('.')

                for line in lines:

                    if "qualifications" in line.lower() or "skills" in line.lower():

                        collected\_skills.append(line.strip())

    return ', '.join(collected\_skills) if collected\_skills else 'N/A'

# API request setup

url = "<https://jsearch.p.rapidapi.com/search>"

querystring = {"query": "cybersecurity jobs on glassdoor", "page": "1", "num\_pages": "1", "date\_posted": "all"}

headers = {

    "x-rapidapi-key": "5a87c59c2cmsh24ea8471a41758ep15cb7bjsn258ed147e434",

    "x-rapidapi-host": "jsearch.p.rapidapi.com"

}

# Make the request to the API

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

# Check if the request was successful (status code 200)

if response.status\_code == 200:

    # Parse the JSON response

    data = response.json()

    # Open or create a CSV file to write job data

    with open("C:/Users/garne/Documents/cybersecurity\_jobs\_101\_v3.csv", mode="w", newline="", encoding="utf-8") as file:

        writer = csv.writer(file)

        # Write the headers

        writer.writerow(["Job Title", "Company", "City", "State", "Country", "Date Posted", "Skills", "Responsibilities", "Experience", "Qualifications", "On Site", "Remote", "Contractor", "URL"])

        # Assuming the jobs data is in the 'data' key

        if 'data' in data:

            jobs = data['data']

            # Iterate over the job results

            for job in jobs:

                # Extract the relevant information from each job

                title = job.get('job\_title', 'N/A')

                company = job.get('employer\_name', 'N/A')

                city = job.get('job\_city', 'N/A')

                state = job.get('job\_state', 'N/A')

                country = job.get('job\_country', 'N/A')

                timestamp = job.get('job\_posted\_at\_timestamp', None)

                date\_posted = datetime.utcfromtimestamp(timestamp).strftime('%Y-%m-%d') if timestamp else 'N/A'

                # Extract Skills with enhanced details

                skills = extract\_skills(job)

                # Responsibilities (inferred from job description)

                responsibilities = job.get('job\_description', 'N/A')

                # Experience (nested inside 'job\_required\_experience')

                experience = job.get('job\_required\_experience', {}).get('experience', 'N/A')

                # Qualifications (extracted and processed separately)

                qualifications = extract\_skills(job)  # Reusing the function to get detailed qualifications too

                # On Site / Remote / Contractor status

                on\_site = 'Yes' if job.get('job\_is\_on\_site', False) else 'No'

                remote = 'Yes' if job.get('job\_is\_remote', False) else 'No'

                contractor = 'Yes' if job.get('job\_is\_contract', False) else 'No'

                # Extract the job application link

                job\_url = job.get('job\_apply\_link', 'N/A')

                # Write the row to the CSV

                writer.writerow([title, company, city, state, country, date\_posted, skills, responsibilities, experience, qualifications, on\_site, remote, contractor, job\_url])

        else:

            print("No job data found in the response.")

else:

    print(f"Failed to retrieve data: {response.status\_code}")