Quality Solution Delivered

Project Guide

Enhancing Business Intelligence with LinkedIn Data Using Apollo.io API

Technical Project Lead:
Krishna Kumar
Cloud Engineer Cum BA at EncapTechno

Resources Allocated:
Sahib Khan
Developer at Encaptechno

Project Duration:
2 days

Date of Completion:
01/04/2024

Organisation:
Encaptechno India Pvt Ltd.

Industry/Field:
IT & Consulting

Target Audience:
Sales and Prospect Mining

Quality Solution Delivered

Project Overview:

The project aims to develop a script using Python to fetch and enrich business-related data from LinkedIn profiles using the Apollo.io API. By leveraging this script, businesses can streamline their lead generation, sales, and marketing efforts by obtaining comprehensive information about potential prospects and organisations.

Project Steps:

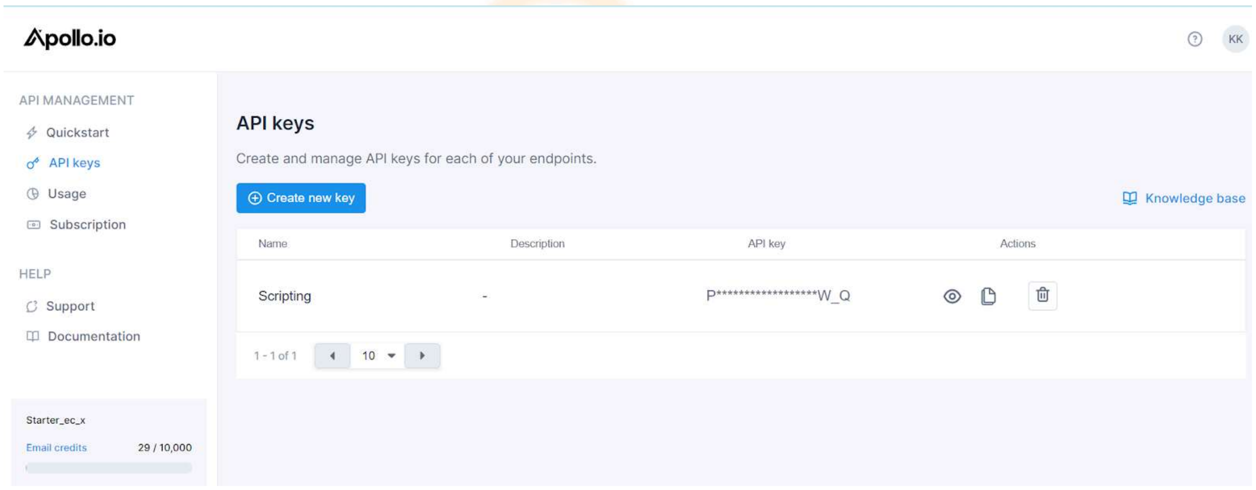
Step 1: Generating Apollo.io API Key

- The initial step involves logging into the Apollo.io platform to generate an API key. This key will be used for authentication purposes when making requests to the Apollo.io API.

Steps to Generate API keys from Apollo.io Console.

1. Go to Settings in Apollo.io Console.
2. Click on Integrations.
3. Click on Connect API.
4. Click on API Keys and create API Key.

Take reference from Figure no 1.1.



API keys

Create and manage API keys for each of your endpoints.

Name	Description	API key	Actions
Scripting	-	P******************W_Q	

Figure 1.1

#NOTE# Your Account should be Admin to get API Key.

Step 2: Utilising Apollo.io API for Data Enrichment

- The script utilises the Apollo.io API for people enrichment by sending a POST request to the API endpoint (`https://api.apollo.io/v1/people/match`).
- Various parameters such as API key, LinkedIn URL, first name, last name, organisation name, email, hashed email, domain, and reveal personal emails are included in the request payload.

- The script is designed to retrieve information associated with the provided LinkedIn URL, such as contact details, professional background, and organisational affiliations.

For More information please visit this URL by Apollo:
https://apolloio.github.io/apollo-api-docs/?python#bulk-organization-enrichment

Here is the relevant raw code that we took as a reference from the previous link.

```
import requests

url = "https://api.apollo.io/v1/people/match"

data = {
    "api_key": "YOUR API KEY HERE",
    "id": "583f2f7ed9ced98ab5bfXXXX",
    "first_name": "Tim",
    "last_name": "Zheng",
    "organization_name": "Apollo",
    "email": "name@domain.io",
    "hashed_email": "8d935115b9ff4489f2d1f9249503cadf",
    "domain": "apollo.io",
    "reveal_personal_emails": true,
    "linkedin_url": "http://www.linkedin.com/in/tim-zheng-677ba010"
}
headers = {
    'Cache-Control': 'no-cache',
    'Content-Type': 'application/json'
}

response = requests.request("POST", url, headers=headers, json=data)

print(response.text)
```

Step 3: Setting up Python Environment and Necessary Modules

- The Python environment is set up, and essential modules including `requests`, `pandas`, `json`, and `csv` are imported to facilitate data manipulation and interaction with the API.
- LinkedIn URLs of the individuals or organisations from which data needs to be fetched are listed for batch processing.

Run these commands in the Terminal of your IDE to provide Environment.

1. pip install json
2. Pip install pandas
3. pip install requests
4. pip install csv

Step 4: Iterating Over LinkedIn URLs and Fetching Data

- The script iterates over the list of LinkedIn URLs, sending POST requests to the Apollo.io API for each URL.
- For each successful request (status code 200), the JSON response containing enriched data is extracted and appended to the results list.
- In case of any errors during the request, appropriate error messages are printed for debugging purposes.

// Below is the full fledged code is Provided.

Step 5: Converting Results to DataFrame and Exporting to CSV

- Once data retrieval is complete, the results are converted into a pandas DataFrame for easy manipulation and analysis.
- The DataFrame is then exported to a CSV file, including the index, for further processing or integration with other systems.

// Below is the full fledged code is Provided.

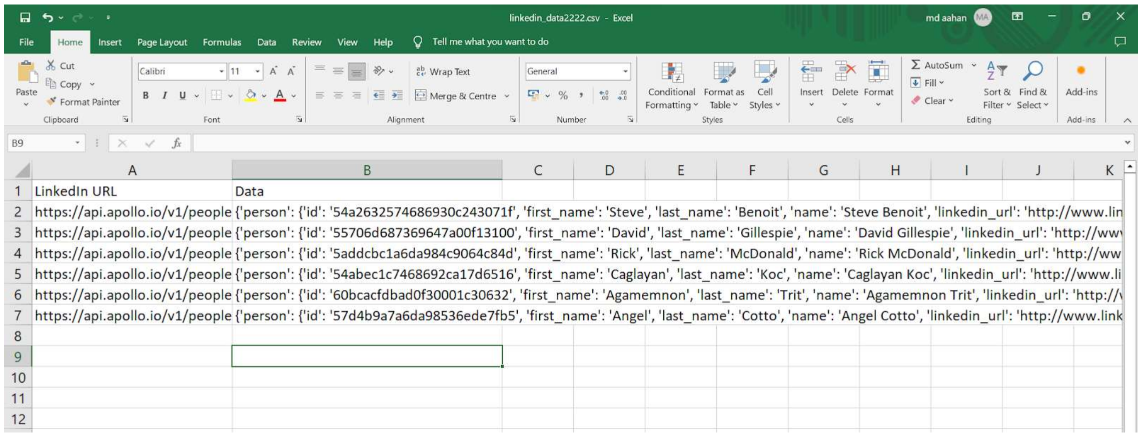
Run this Code in your IDE:

```
import requests
import pandas as pd
import json
import csv

# List of LinkedIn URLs
linke din_urls = [
    "https://www.linkedin.com/in/krishnakumar2409//",
    "https://www.linkedin.com/in/krishnakumar2409/",
    "https://www.linkedin.com/in/krishnakumar2409/",
  ] #These are your Prospects (Contact info).

#List to store results
results = []

# API endpoint
url = "https://api.apollo.io/v1/people/match"

# Headers
headers = {
    'Cache-Control': 'no-cache',
    'Content-Type': 'application/json'
}

# Iterate over LinkedIn URLs
for urls in linkedin_urls:
    # Data for the request
    data = {
        "api_key": "API KEY HERE",
        "linkedin_url": urls
    }

    # Making the HTTP POST request
    response = requests.post(url, headers=headers, json=data)

    # Checking if the request was successful
    if response.status_code == 200:
        # Extracting JSON data from the response
        json_data = response.json()
        results.append(json_data)
    else:
        # Print error message
        print(f"Error fetching data for URL: {url}")
```

```
# Converting the list of results to a DataFrame
df = pd.DataFrame(results)

# Writing DataFrame to a CSV file with index included
csv_file = 'Records_1111.csv'
df.to_csv(csv_file, index=True)
print(f"Data successfully saved to {csv_file}")
```

Result:

In this diagram, you can observe that the output has been compiled and generated as a CSV file. You now have the flexibility to manually adjust the data to conform to your specific formatting requirements.



	A	B
1	LinkedIn URL	Data
2	https://api.apollo.io/v1/people	{'person': {'id': '54a2632574686930c243071f', 'first_name': 'Steve', 'last_name': 'Benoit', 'name': 'Steve Benoit', 'linkedin_url': 'http://www.lin
3	https://api.apollo.io/v1/people	{'person': {'id': '55706d687369647a00f13100', 'first_name': 'David', 'last_name': 'Gillespie', 'name': 'David Gillespie', 'linkedin_url': 'http://ww
4	https://api.apollo.io/v1/people	{'person': {'id': '5addcbc1a6da984c9064c84d', 'first_name': 'Rick', 'last_name': 'McDonald', 'name': 'Rick McDonald', 'linkedin_url': 'http://ww
5	https://api.apollo.io/v1/people	{'person': {'id': '54abec1c7468692ca17d6516', 'first_name': 'Caglayan', 'last_name': 'Koc', 'name': 'Caglayan Koc', 'linkedin_url': 'http://www.li
6	https://api.apollo.io/v1/people	{'person': {'id': '60bcacfdbad0f30001c30632', 'first_name': 'Agamemnon', 'last_name': 'Trit', 'name': 'Agamemnon Trit', 'linkedin_url': 'http://
7	https://api.apollo.io/v1/people	{'person': {'id': '57d4b9a7a6da98536ede7fb5', 'first_name': 'Angel', 'last_name': 'Cotto', 'name': 'Angel Cotto', 'linkedin_url': 'http://www.link
8		
9		
10		
11		
12		

Conclusion:

By implementing this script, businesses can efficiently gather and enrich valuable business intelligence data from LinkedIn profiles using the Apollo.io API. This data can be instrumental in enhancing lead generation, sales prospecting, market segmentation, and overall business strategy. Moreover, the automation provided by the script saves time and effort, enabling businesses to focus on core activities while leveraging the power of data-driven insights.

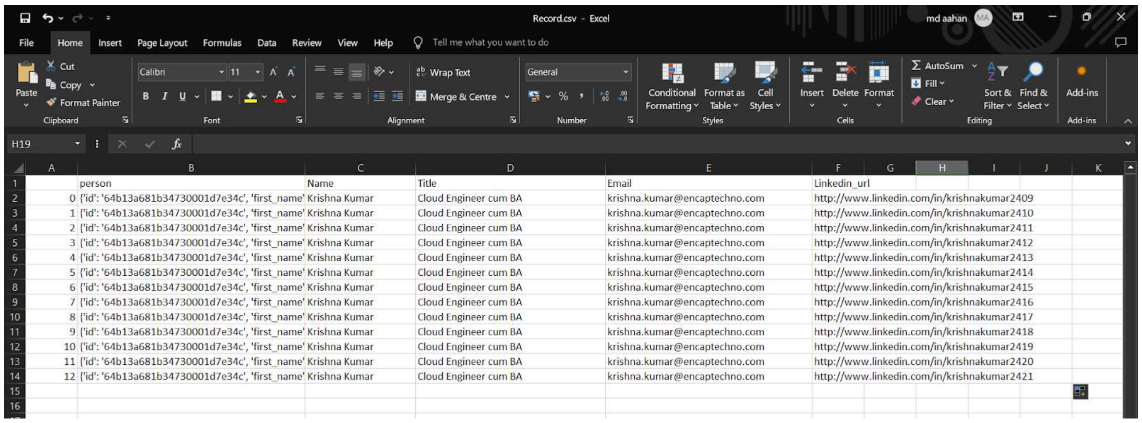
encoap
Quality Solution Delivered

CR:Formatize the CSV file within the Script.

Objective: This line of code is updated to Structurise the fields as per the need in CSV.

```
import requests
import pandas as pd
import time

# List of LinkedIn URLs
linkedin_urls = [
"http://www.linkedin.com/in/krishnakumar2409"

]

# List to store results
results = []

# API endpoint
api_endpoint = "https://api.apollo.io/v1/people/match"

# Headers
headers = {
    'Cache-Control': 'no-cache',
    'Content-Type': 'application/json'
}

# Chunk size to control the buffer
chunk_size = 1 # Adjust as needed

# Delay between chunks (in seconds)
delay_between_chunks = 1

# Iterate over LinkedIn URLs in chunks
for i in range(0, len(linkedin_urls), chunk_size):
    chunk_urls = linkedin_urls[i:i+chunk_size]

    # Iterate over chunked LinkedIn URLs
    for url in chunk_urls:
        try:
            # Data for the request
            data = {
                "api_key": "Your API KEY HERE",
                "linkedin_url": url
            }

            # Making the HTTP POST request
            response = requests.post(api_endpoint, headers=headers, json=data)

            # Checking if the request was successful
            if response.status_code == 200:
                # Extracting JSON data from the response
                json_data = response.json()

                results.append(json_data)
            else:
                # Print error message
                print(f"Error fetching data for URL: {url}. Status Code:
{response.status_code}")
        except Exception as e:
            print(f"An error occurred while fetching data for URL: {url}. Error: {str(e)}")

    # Introduce a delay between chunks to avoid overwhelming the server
    time.sleep(delay_between_chunks)

# Converting the list of results to a DataFrame
df = pd.DataFrame(results)

# Extracting emails, LinkedIn URLs, names, and titles from the results
emails = [item['person']['email'] if 'email' in item['person'] else 'N/A' for item in
results]
linkedin_urls = [item['person']['linkedin_url'] if 'linkedin_url' in item['person'] else
'N/A' for item in results]
names = [item['person']['name'] if 'name' in item['person'] else 'N/A' for item in
results]
titles = [item['person']['title'] if 'title' in item['person'] else 'N/A' for item in results]

# Adding emails, LinkedIn URLs, names, and titles to the DataFrame
df['Name'] = names
df['Title'] = titles
df['Email'] = emails
df['Linkedin_url'] = linkedin_urls

# Writing DataFrame to a CSV file with index included
csv_file = str(input("Enter your file name "))
df.to_csv(csv_file, index=True)
print(f"Data successfully saved to {csv_file}")
```

Output:-



	person	Name	Title	Email	Linkedin_url
0	{'id': '64b13a681b34730001d7e34c', 'first_name'	Krishna Kumar	Cloud Engineer cum BA	krishna.kumar@encaptechno.com	http://www.linkedin.com/in/krishnakumar2409
1	{'id': '64b13a681b34730001d7e34c', 'first_name'	Krishna Kumar	Cloud Engineer cum BA	krishna.kumar@encaptechno.com	http://www.linkedin.com/in/krishnakumar2410
2	{'id': '64b13a681b34730001d7e34c', 'first_name'	Krishna Kumar	Cloud Engineer cum BA	krishna.kumar@encaptechno.com	http://www.linkedin.com/in/krishnakumar2411
3	{'id': '64b13a681b34730001d7e34c', 'first_name'	Krishna Kumar	Cloud Engineer cum BA	krishna.kumar@encaptechno.com	http://www.linkedin.com/in/krishnakumar2412
4	{'id': '64b13a681b34730001d7e34c', 'first_name'	Krishna Kumar	Cloud Engineer cum BA	krishna.kumar@encaptechno.com	http://www.linkedin.com/in/krishnakumar2413
5	{'id': '64b13a681b34730001d7e34c', 'first_name'	Krishna Kumar	Cloud Engineer cum BA	krishna.kumar@encaptechno.com	http://www.linkedin.com/in/krishnakumar2414
6	{'id': '64b13a681b34730001d7e34c', 'first_name'	Krishna Kumar	Cloud Engineer cum BA	krishna.kumar@encaptechno.com	http://www.linkedin.com/in/krishnakumar2415
7	{'id': '64b13a681b34730001d7e34c', 'first_name'	Krishna Kumar	Cloud Engineer cum BA	krishna.kumar@encaptechno.com	http://www.linkedin.com/in/krishnakumar2416
8	{'id': '64b13a681b34730001d7e34c', 'first_name'	Krishna Kumar	Cloud Engineer cum BA	krishna.kumar@encaptechno.com	http://www.linkedin.com/in/krishnakumar2417
9	{'id': '64b13a681b34730001d7e34c', 'first_name'	Krishna Kumar	Cloud Engineer cum BA	krishna.kumar@encaptechno.com	http://www.linkedin.com/in/krishnakumar2418
10	{'id': '64b13a681b34730001d7e34c', 'first_name'	Krishna Kumar	Cloud Engineer cum BA	krishna.kumar@encaptechno.com	http://www.linkedin.com/in/krishnakumar2419
11	{'id': '64b13a681b34730001d7e34c', 'first_name'	Krishna Kumar	Cloud Engineer cum BA	krishna.kumar@encaptechno.com	http://www.linkedin.com/in/krishnakumar2420
12	{'id': '64b13a681b34730001d7e34c', 'first_name'	Krishna Kumar	Cloud Engineer cum BA	krishna.kumar@encaptechno.com	http://www.linkedin.com/in/krishnakumar2421