To scrape the data

import requests

import pandas as pd

# Constants

GITHUB\_API\_URL = "https://api.github.com"

TOKEN = "ghp\_hEZv4ZaAvrETPhEJnxiKLOdMj1hlvM2HqgB2"  # Replace with your GitHub token

# Set headers for authentication

headers = {

    "Authorization": f"token {TOKEN}",

    "Accept": "application/vnd.github.v3+json"

}

# Function to search for users in Dublin

def search\_users\_in\_dublin():

    users = []

    page = 1

    while True:

        response = requests.get(f"{GITHUB\_API\_URL}/search/users?q=location:Dublin+followers:>50&page={page}", headers=headers)

        data = response.json()

        if response.status\_code != 200:

            print(f"Error: {data.get('message', 'Unknown error')}")

            break

        if 'items' not in data or len(data['items']) == 0:

            break

        users.extend(data['items'])

        page += 1

    return users

# Function to fetch detailed user info

def fetch\_user\_details(username):

    response = requests.get(f"{GITHUB\_API\_URL}/users/{username}", headers=headers)

    return response.json()

# Function to fetch repositories for a user

def fetch\_user\_repositories(username):

    repos = []

    page = 1

    while True:

        response = requests.get(f"{GITHUB\_API\_URL}/users/{username}/repos?per\_page=100&page={page}", headers=headers)

        data = response.json()

        if response.status\_code != 200:

            print(f"Error fetching repos for {username}: {data.get('message', 'Unknown error')}")

            break

        if len(data) == 0:

            break

        repos.extend(data)

        page += 1

    return repos

# Main script

def main():

    users = search\_users\_in\_dublin()

    user\_data = []

    repo\_data = []

    for user in users:

        user\_info = fetch\_user\_details(user["login"])

        user\_data.append({

            "login": user\_info.get("login"),

            "name": user\_info.get("name", ""),

            "company": (user\_info.get("company") or "").strip("@").upper(),

            "location": user\_info.get("location", ""),

            "email": user\_info.get("email", ""),

            "hireable": user\_info.get("hireable", False),

            "bio": user\_info.get("bio", ""),

            "public\_repos": user\_info.get("public\_repos", 0),

            "followers": user\_info.get("followers", 0),

            "following": user\_info.get("following", 0),

            "created\_at": user\_info.get("created\_at", "")

        })

        # Fetch and store user repositories

        repos = fetch\_user\_repositories(user\_info["login"])

        for repo in repos:

            repo\_data.append({

              "login": user\_info.get("login"),

              "full\_name": repo.get("full\_name", ""),

              "created\_at": repo.get("created\_at", ""),

              "stargazers\_count": repo.get("stargazers\_count", 0),

              "watchers\_count": repo.get("watchers\_count", 0),

              "language": repo.get("language", ""),

              "has\_projects": repo.get("has\_projects", False),

              "has\_wiki": repo.get("has\_wiki", False),

              "license\_name": repo.get("license").get("name", "") if repo.get("license") else ""

          })

    # Create DataFrames and save to CSV

    users\_df = pd.DataFrame(user\_data)

    users\_df.to\_csv('users.csv', index=False)

    repos\_df = pd.DataFrame(repo\_data)

    repos\_df.to\_csv('repositories.csv', index=False)

    print("Data scraped and saved to users.csv and repositories.csv.")

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

    main()

qUESTION 1

1. Who are the top 5 users in Dublin with the highest number of followers? List their login in order, comma-separated.

orta,jeromeetienne,jonataslaw,steventroughtonsmith,axic

//reading from csv directly

import pandas as pd

# Load the data

data = pd.read\_csv('users.csv')

# Filter for users in Dublin

dublin\_users = data[data['location'].str.contains('Dublin', na=False)]

# Sort users by number of followers in descending order and select top 5

top\_users = dublin\_users.nlargest(5, 'followers')

# Get the logins of the top users

top\_user\_logins = top\_users['login'].tolist()

# Print the result as a comma-separated string

print(", ".join(top\_user\_logins))

//READING FROM GITHUB

import pandas as pd

# URL to the raw CSV file in the GitHub repository

url = 'https://raw.githubusercontent.com/manjuiitm/Dublin1/main/users.csv'

# Read the CSV file directly from the URL

data = pd.read\_csv(url)

# Filter for users in Dublin

dublin\_users = data[data['location'].str.contains('Dublin', na=False)]

# Sort users by number of followers in descending order and select top 5

top\_users = dublin\_users.nlargest(5, 'followers')

# Get the logins of the top users

top\_user\_logins = top\_users['login'].tolist()

# Print the result as a comma-separated string

print(", ".join(top\_user\_logins))

2. Who are the 5 earliest registered GitHub users in Dublin? List their login in ascending order of created\_at, comma-separated.

paulca,adrian,GavinJoyce,amir,ciaranlee

//FROM GITHUB DIRECTLY

import pandas as pd

# URL to the raw CSV file in the GitHub repository

url = 'https://raw.githubusercontent.com/manjuiitm/Dublin1/main/users.csv'

# Read the CSV file directly from the URL

data = pd.read\_csv(url)

# Filter for users in Dublin

dublin\_users = data[data['location'].str.contains('Dublin', na=False)]

# Convert the created\_at column to datetime format

dublin\_users['created\_at'] = pd.to\_datetime(dublin\_users['created\_at'])

# Sort users by created\_at in ascending order and select the top 5

earliest\_users = dublin\_users.nsmallest(5, 'created\_at')

# Get the logins of the earliest registered users

earliest\_user\_logins = earliest\_users['login'].tolist()

# Print the result as a comma-separated string

print(", ".join(earliest\_user\_logins))

//from excel file

import pandas as pd

# Load the data from the CSV file

file\_path = 'users.csv' # Ensure the file is in the same directory or provide the full path

data = pd.read\_csv(file\_path)

# Filter for users in Dublin

dublin\_users = data[data['location'].str.contains('Dublin', na=False)]

# Convert the created\_at column to datetime format

dublin\_users['created\_at'] = pd.to\_datetime(dublin\_users['created\_at'])

# Sort users by created\_at in ascending order and select the top 5

earliest\_users = dublin\_users.nsmallest(5, 'created\_at')

# Get the logins of the earliest registered users

earliest\_user\_logins = earliest\_users['login'].tolist()

# Print the result as a comma-separated string

print(", ".join(earliest\_user\_logins))

3. What are the 3 most popular license among these users? Ignore missing licenses. List the license\_name in order, comma-separated.

MIT License, Apache License 2.0, Other

import pandas as pd

# Load the data from the CSV file

file\_path = 'repositories.csv'  # Ensure the file is in the same directory or provide the full path

data = pd.read\_csv(file\_path)

# Print the columns of the DataFrame

print("Columns in the DataFrame:", data.columns)

# After identifying the correct column name, proceed with counting licenses

# Replace 'license\_name' with the actual column name if it's different

license\_counts = data['license\_name'].value\_counts(dropna=True)

# Get the top 3 most common licenses

top\_licenses = license\_counts.nlargest(3)

# Prepare the output as a comma-separated string

top\_license\_names = top\_licenses.index.tolist()

# Print the result

print(", ".join(top\_license\_names))

## 4. Which company do the majority of these developers work at?

MICROSOFT

import pandas as pd

# Load the data from the CSV file

file\_path = 'users.csv' # Ensure the file is in the same directory or provide the full path

data = pd.read\_csv(file\_path)

# Count the occurrences of each company, ignoring missing values

company\_counts = data['company'].value\_counts(dropna=True)

# Identify the company with the maximum count

most\_common\_company = company\_counts.idxmax()

most\_common\_count = company\_counts.max()

# Print the result

print(f"The majority of developers work at: {most\_common\_company} (Count: {most\_common\_count})")

## 5. Which programming language is most popular among these users?

jAVASCRIPT

import pandas as pd

# Load the data from the CSV file

file\_path = 'repositories.csv' # Ensure the file is in the same directory or provide the full path

data = pd.read\_csv(file\_path)

# Print the columns of the DataFrame

print("Columns in the DataFrame:", data.columns)

# After identifying the correct column name, proceed with counting languages

# Replace 'programming\_language' with the actual column name if it's different

language\_counts = data['language'].value\_counts(dropna=True) # Update this line

# Identify the most popular programming language

most\_popular\_language = language\_counts.idxmax()

most\_popular\_count = language\_counts.max()

# Print the result

print(f"The most popular programming language is: {most\_popular\_language} (Count: {most\_popular\_count})")

## 7. Which language has the highest average number of stars per repository?

## import pandas as pd

## # Load the data from the CSV file

## file\_path = 'repositories.csv'

## data = pd.read\_csv(file\_path)

## # Print the columns of the DataFrame

## print("Columns in the DataFrame:", data.columns)

## # Replace 'stargazers\_count' and 'language' with your actual column names

## data['stars'] = pd.to\_numeric(data['stargazers\_count'], errors='coerce') # Correct column for stars

## data = data.dropna(subset=['language', 'stars']) # Correct column for language

## # Group by language and calculate the average number of stars

## average\_stars = data.groupby('language')['stars'].mean().sort\_values(ascending=False)

## # Identify the language with the highest average number of stars

## if not average\_stars.empty:

## highest\_average\_language = average\_stars.idxmax()

## highest\_average\_stars = average\_stars.max()

## # Print the result

## print(f"The language with the highest average number of stars per repository is: {highest\_average\_language} (Average Stars: {highest\_average\_stars})")

## else:

print("No data available to calculate averages.")

## 8. Let's define leader\_strength as followers / (1 + following). Who are the top 5 in terms of leader\_strength? List their login in order, comma-separated.

flaviohenriquealmeida,zalando,AnikSarker,wix,CardinalHealth

import pandas as pd

# Load the data from the CSV file

file\_path = 'users.csv' # Ensure this path is correct

data = pd.read\_csv(file\_path)

# Print the columns of the DataFrame (for troubleshooting)

print("Columns in the DataFrame:", data.columns)

# Ensure that 'followers' and 'following' are numeric

data['followers'] = pd.to\_numeric(data['followers'], errors='coerce')

data['following'] = pd.to\_numeric(data['following'], errors='coerce')

# Calculate leader\_strength

data['leader\_strength'] = data['followers'] / (1 + data['following'])

# Drop rows with NaN in leader\_strength

data = data.dropna(subset=['leader\_strength'])

# Sort by leader\_strength in descending order and get the top 5

top\_users = data.nlargest(5, 'leader\_strength')

# Get the login names of the top users

top\_user\_logins = top\_users['login'].tolist()

# Print the result

print("Top 5 users in terms of leader\_strength:", ', '.join(top\_user\_logins))

## 9. What is the correlation between the number of followers and the number of public repositories among users in Dublin?

0.555

import pandas as pd

# Load the data from the CSV file

file\_path = 'users.csv' # Ensure this path is correct

data = pd.read\_csv(file\_path)

# Print the columns of the DataFrame (for troubleshooting)

print("Columns in the DataFrame:", data.columns)

# Filter for users located in Dublin

dublin\_users = data[data['location'].str.contains('Dublin', na=False)]

# Ensure followers and public\_repos are numeric

dublin\_users['followers'] = pd.to\_numeric(dublin\_users['followers'], errors='coerce')

dublin\_users['public\_repos'] = pd.to\_numeric(dublin\_users['public\_repos'], errors='coerce')

# Drop rows with NaN in followers or public\_repos

dublin\_users = dublin\_users.dropna(subset=['followers', 'public\_repos'])

# Calculate the correlation

correlation = dublin\_users['followers'].corr(dublin\_users['public\_repos'])

# Print the result

print(f"The correlation between the number of followers and the number of public repositories among users in Dublin is: {correlation:.2f}")

## 13. Some developers write long bios. Does that help them get more followers? What's the impact of the length of their bio (in Unicode words, split by whitespace) with followers? (Ignore people without bios)

## import pandas as pd

## import numpy as np

## import statsmodels.api as sm

## # Read the CSV file

## df = pd.read\_csv('users.csv')

## # Clean the data

## df = df[df['bio'].notna()] # Keep only users with bios

## df['bio\_word\_count'] = df['bio'].apply(lambda x: len(x.split())) # Count words in bio

## # Prepare the data for regression

## X = df[['bio\_word\_count']] # Independent variable

## y = df['followers'] # Dependent variable

## # Add a constant to the independent variable

## X = sm.add\_constant(X)

## # Perform regression

## model = sm.OLS(y, X).fit()

## # Get the slope for bio word count

## slope\_bio = model.params['bio\_word\_count']

## # Print the slope rounded to three decimal places

print(f"Slope of followers on bio word count: {slope\_bio:.3f}")

7.232

16. Let's assume that the last word in a user's name is their surname (ignore missing names, trim and split by whitespace.) What's the most common surname? (If there's a tie, list them all, comma-separated, alphabetically)

Most common surname(s)

import pandas as pd

from collections import Counter

# Read the CSV file

df = pd.read\_csv('users.csv')

# Function to extract surname

def extract\_surname(name):

if pd.isna(name) or not isinstance(name, str):

return None

# Split the name by whitespace and get the last word

return name.strip().split()[-1]

# Extract surnames from the 'name' column

df['surname'] = df['name'].apply(extract\_surname)

# Filter out None values (missing names)

surnames = df['surname'].dropna()

# Count occurrences of each surname

surname\_counts = Counter(surnames)

# Find the most common surname(s)

max\_count = max(surname\_counts.values())

most\_common\_surnames = [surname for surname, count in surname\_counts.items() if count == max\_count]

# Sort surnames alphabetically

most\_common\_surnames.sort()

# Output the result

result = ', '.join(most\_common\_surnames)

print(f"Most common surname(s): {result}")