Portfolio Assignment 3 - Scraping

In [ ]:
```python
# Install packages
import praw
import pandas as pd
from praw.models import MoreComments
import json
import os
import datetime
```

API

In [ ]:
```python
# The weird letters are the cresidentials from my API

reddit = praw.Reddit(client_id ='hQafbgnPnl6Jl_wabPofuA',
                     client_secret ='OZ0EsYse7J-JiYE7kPk3mddQgMZ2nA',
                     user_agent ='WebScraper/Own-Biscotti6249')

# to verify whether the instance is authorized instance or not
print(reddit.read_only)
```

Scraping

In [ ]:
```python
# convert data
def date_to_timestamp(date_str):
    return int(datetime.datetime.strptime(date_str, "%Y-%m-%d").timestamp())

# gets posts within a subreddit and date range
def fetch_submission_ids(subreddit_name, start_timestamp, end_timestamp):
    submission_ids = []
    subreddit = reddit.subreddit(subreddit_name)
    print(f"Fetching posts from r/{subreddit_name} between {start_timestamp} and {end

    for submission in subreddit.new(limit=None):  # posts in chronological order
        # dont get posts out of the desired time range
        if submission.created_utc < start_timestamp:
            break
        if start_timestamp <= submission.created_utc <= end_timestamp:
            print(f"Found post: {submission.title} (ID: {submission.id})")
            submission_ids.append(submission.id)

    if not submission_ids:
        print("No posts found in the specified date range.")
    return submission_ids

# fetch comments and their replies
def fetch_comments_and_replies(comment, submission_id, submission_title, start_timest
    if start_timestamp <= comment.created_utc <= end_timestamp:
        comment_body = comment.body if comment.body != '[deleted]' else None
        comment_author = comment.author.name if comment.author else None

        comment_data = {
            "submission_id": submission_id,
            "submission_title": submission_title,
            "comment_id": comment.id,
            "comment": comment_body,
```

```
            "author": comment_author,
            "created": datetime.datetime.fromtimestamp(comment.created_utc).strftime(
            "upvotes": comment.score,
            "depth": depth,
            "replies": []
        }

        if comment.replies:
            for reply in comment.replies:
                if isinstance(reply, praw.models.Comment) and start_timestamp <= repl
                    comment_data['replies'].append(fetch_comments_and_replies(reply,

        return comment_data
    return None

# Fcomments from specific submissions
def fetch_comments_from_submissions(submission_ids, start_timestamp, end_timestamp):
    all_comments = []

    for submission_id in submission_ids:
        try:
            submission = reddit.submission(id=submission_id)
            print(f"Fetching comments from: {submission.title} (ID: {submission.id})"

            submission.comments.replace_more(limit=None)
            for comment in submission.comments.list():
                comment_data = fetch_comments_and_replies(comment, submission.id, sub
                if comment_data:
                    all_comments.append(comment_data)

        except Exception as e:
            print(f"An error occurred while fetching comments from {submission_id}: {

    return all_comments

# save to JSON
def save_comments_to_json(comments, json_file_name):
    folder_path = 'reddit_data_new'
    if not os.path.exists(folder_path):
        os.makedirs(folder_path)

    json_file_path = os.path.join(folder_path, json_file_name)

    with open(json_file_path, 'w', encoding='utf-8') as json_file:
        json.dump(comments, json_file, ensure_ascii=False, indent=4)

    print(f"Comments saved to {json_file_path}")

# putting it all together
def main(subreddit_name, start_date, end_date, json_file_name):
    start_timestamp = date_to_timestamp(start_date)
    end_timestamp = date_to_timestamp(end_date)

    # posts from the subreddit
    submission_ids = fetch_submission_ids(subreddit_name, start_timestamp, end_timest

    # comments from the sposts
    comments = fetch_comments_from_submissions(submission_ids, start_timestamp, end_t

    # Save
    save_comments_to_json(comments, json_file_name)
```

In [ ]:
```python
# Run the script for democrats
if __name__ == "__main__":
    subreddit_name = "democrats"
    start_date = "2024-11-06"
    end_date = "2024-11-18"
    json_file_name = "democrat_comments.json"

    main(subreddit_name, start_date, end_date, json_file_name)
```

In [ ]:
```python
# Run the script for republicans
if __name__ == "__main__":
    subreddit_name = "republican"
    start_date = "2024-11-05"  #
    end_date = "2024-11-19"
    json_file_name = "republican_comments.json"

    main(subreddit_name, start_date, end_date, json_file_name)
```