Reddit Scraper

This notebook uses the **official Reddit API with OAuth (PRAW)** to download **posts** and **comments** from a subreddit and writes them to **Data/raw/** without metrics or preprocessing.

Libraries

In [1]:
```
# Run this cell to install the following dependencies:
# ```bash
# pip install praw pandas tqdm python-dotenv
# ```
```

In [2]:
```
from dataclasses import dataclass, asdict
from typing import List, Optional, Tuple
from pathlib import Path
import os, json, re, time
from datetime import datetime, timezone
import pandas as pd
from tqdm import tqdm
from dotenv import load_dotenv
import praw
```

Parameters and Credentials

In [3]:
```
# Edit the .env file with your Reddit API credentials
load_dotenv()
# Reddit API credentials
# Before running this cell, define the following variables in a .env file:
# - `REDDIT_CLIENT_ID`
# - `REDDIT_CLIENT_SECRET`
# - `REDDIT_USER_AGENT`
# Or define them directly here:
# REDDIT_CLIENT_ID     = "your_client_id"
# REDDIT_CLIENT_SECRET = "your_client_secret"
# REDDIT_USER_AGENT    = "your_user_agent"
REDDIT_CLIENT_ID     = os.getenv("REDDIT_CLIENT_ID")
REDDIT_CLIENT_SECRET = os.getenv("REDDIT_CLIENT_SECRET")
REDDIT_USER_AGENT    = os.getenv("REDDIT_USER_AGENT")

# Parameters
SUBREDDIT                   = "USCIS"    # <-- target subreddit
SORT                        = "top"      # "hot" | "new" | "top" | "rising"
TIME_FILTER                 = "year"     # if SORT="top", use: "day"|"week"|
MIN_POSTS                   = 20         # minimum number of posts to fetch
MAX_POSTS                   = 200        # maximum number of posts to fetch
MAX_COMMENTS_PER_POST       = 80         # limit on comments per post
DOWNLOAD_IMAGES             = False      # download images locally if URL pc
```

```
REQUEST_SLEEP_S             = 0.7        # respect Reddit API rate limits
DATA_DIR                    = Path("data") # base data directory
RAW_DIR                     = DATA_DIR / "raw" # directory for unprocessed d

# Create directories if they don't exist
RAW_DIR.mkdir(parents=True, exist_ok=True)

# Check credentials
if not REDDIT_CLIENT_ID or not REDDIT_CLIENT_SECRET:
    raise RuntimeError("Missing credentials: define REDDIT_CLIENT_ID and RED
```

Utilities

In [4]:

```
def make_reddit() -> "praw.Reddit":
    """Creates an authenticated PRAW Reddit API client instance."""
    return praw.Reddit(
        client_id=REDDIT_CLIENT_ID,
        client_secret=REDDIT_CLIENT_SECRET,
        user_agent=REDDIT_USER_AGENT,
    )


def is_image_url(u: str) -> bool:
    """Checks if a URL points to an image (jpg, jpeg, png, gif)."""
    return bool(re.search(r"\.(jpg|jpeg|png|gif)(?:\?.*)?$", (u or ""), flag


def to_iso(ts_utc: Optional[float]) -> Optional[str]:
    """Converts a UTC timestamp to ISO 8601 format."""
    try:
        return datetime.fromtimestamp(float(ts_utc), tz=timezone.utc).isofor
    except Exception:
        return None


def rsleep():
    """Sleeps for REQUEST_SLEEP_S seconds if greater than 0."""
    if REQUEST_SLEEP_S > 0:
        time.sleep(REQUEST_SLEEP_S)
```

Raw Output Schema for Posts and Comments

In [5]:

```
@dataclass
class PostRow:
    post_id: str
    title: Optional[str]
    author: Optional[str]
    score: Optional[int]
    num_comments: Optional[int]
    created: Optional[str]
    permalink: str
    is_self: bool
    image_urls: List[str]
```

```
    selftext: Optional[str]
    subreddit: Optional[str]


@dataclass
class CommentRow:
    post_id: str
    comment_id: str
    author: Optional[str]
    created: Optional[str]
    score: Optional[int]
    body: str
```

Data Extraction

In [6]:

```
# Create Reddit client
reddit = make_reddit()
# Select subreddit
sub = reddit.subreddit(SUBREDDIT)

# Get post listing based on criteria
if SORT == "hot":
    listing = sub.hot(limit=MAX_POSTS)
elif SORT == "new":
    listing = sub.new(limit=MAX_POSTS)
elif SORT == "rising":
    listing = sub.rising(limit=MAX_POSTS)
elif SORT == "top":
    listing = sub.top(time_filter=TIME_FILTER, limit=MAX_POSTS)
else:
    listing = sub.hot(limit=MAX_POSTS)

# Initialize result lists
posts_rows: List[PostRow] = []
comments_rows: List[CommentRow] = []

# Iterate posts
for p in tqdm(listing, total=MAX_POSTS, desc=f"Posts r/{SUBREDDIT}"):
    rsleep()
    img_urls = [p.url] if is_image_url(getattr(p, "url", "")) else []

    pr = PostRow(
        post_id=p.id,
        title=getattr(p, "title", None),
        author=f"u/{p.author}" if getattr(p, "author", None) else None,
        score=int(getattr(p, "score", 0)) if getattr(p, "score", None) is nc
        num_comments=int(getattr(p, "num_comments", 0)) if getattr(p, "num_c
        created=to_iso(getattr(p, "created_utc", None)),
        permalink=f"https://www.reddit.com{getattr(p, 'permalink', '')}",
        is_self=bool(getattr(p, "is_self", False)),
        image_urls=img_urls,
        selftext=(getattr(p, "selftext", None) or None),
        subreddit=str(getattr(p, "subreddit", SUBREDDIT)) if getattr(p, "sub
    )
```

```
    # Comments
    try:
        p.comments.replace_more(limit=0)
        com_list = p.comments.list()[:MAX_COMMENTS_PER_POST]
    except Exception as e:
        print(f"[WARN] Comments {e} in {pr.permalink}")
        com_list = []

    for c in com_list:
        text = getattr(c, "body", "") or ""
        comments_rows.append(CommentRow(
            post_id=pr.post_id,
            comment_id=getattr(c, "id", ""),
            author=f"u/{c.author}" if getattr(c, "author", None) else None,
            created=to_iso(getattr(c, "created_utc", None)),
            score=int(getattr(c, "score", 0)) if getattr(c, "score", None) i
            body=text,
        ))

    posts_rows.append(pr)

print(f"Posts: {len(posts_rows)} | Comments: {len(comments_rows)}")
```

```
Posts r/USCIS: 100%|██████████| 200/200 [05:55<00:00,  1.78s/it]
Posts: 200 | Comments: 15131
```

Save Data (CSV + JSONL)

In [7]:

```
posts_df = pd.DataFrame([asdict(p) for p in posts_rows])
comments_df = pd.DataFrame([asdict(c) for c in comments_rows])

posts_csv = RAW_DIR / "posts.csv"
comments_csv = RAW_DIR / "comments.csv"
posts_jsonl = RAW_DIR / "posts.jsonl"
comments_jsonl = RAW_DIR / "comments.jsonl"

posts_df.to_csv(posts_csv, index=False, encoding="utf-8-sig")
comments_df.to_csv(comments_csv, index=False, encoding="utf-8-sig")

with open(posts_jsonl, "w", encoding="utf-8") as f:
    for _, row in posts_df.iterrows():
        f.write(json.dumps(row.to_dict(), ensure_ascii=False) + "\n")

with open(comments_jsonl, "w", encoding="utf-8") as f:
    for _, row in comments_df.iterrows():
        f.write(json.dumps(row.to_dict(), ensure_ascii=False) + "\n")

print("Saved to:", RAW_DIR)
```

```
Saved to: data\raw
```

Quick View

In [8]:
```
print("Posts:")
display(posts_df.head())
print("Comments:")
display(comments_df.head())
```

Posts:

	post_id	title	author	score	num_comments	create
0	1lcftjz	Got my mom her green card by enlisting in the ...	u/SuperiorT	4159	537	2025-0(16T00:52:02+00:0
1	1grmeq4	Today I became a US citizen	u/adepojus	3873	261	2024-1(15T02:45:34+00:0
2	1gkfbph	Today I became a US citizen	u/Asteroids19_9	3688	142	2024-1(05T19:40:39+00:0
3	1glflxy	So, what now? An immigration attorney perspect...	u/Honest-Grape-9352	2898	714	2024-1(07T02:01:16+00:0
4	1ltlanr	Became a Citizen after 26 years!!	u/Ajax4557	2653	165	2025-0(07T04:44:32+00:0

Comments:

	post_id	comment_id	author	created	score	
0	1lcftjz	my05pdb	u/DrummerHistorical493	2025-06-16T01:03:23+00:00	202	What a
1	1lcftjz	my05mw5	u/Thedippyhoe	2025-06-16T01:02:57+00:00	212	Congra Big h mo
2	1lcftjz	my0c8ez	u/WonderfulVariation93	2025-06-16T01:44:40+00:00	190	You fro Mothe
3	1lcftjz	my08ip3	u/GeekNoy	2025-06-16T01:21:16+00:00	50	Cong mor aw
4	1lcftjz	my067yj	u/Greedy_Disaster_3130	2025-06-16T01:06:41+00:00	128	Th benefi ser