group_CVE.py

```
# group_CVE.py (CHANGES MADE)
from collections import Counter, defaultdict
from datetime import datetime

def group_by_severity(cves):
    #Count up all CVEs by severity (CRITICAL, HIGH, MEDIUM, LOW).
    #Return a dict ready for dashboard charts.
    counts = Counter()
    for cve in cves:
        sev = cve.get("Severity", "UNKNOWN").upper()
        counts[sev] += 1
    #Only pick canonical severity keys; drop weird stuff from the API
    return {k: counts.get(k, 0) for k in ['CRITICAL', 'HIGH', 'MEDIUM',
'LOW']}

def group_by_cwe(cves):
    #Counts the number of CVEs per CWE ID.
    #Lets us show "top weaknesses" or run radar/bar charts.
    counts = defaultdict(int)
    for cve in cves:
        cwe = cve.get("CWE")
        if cwe:
            counts[cwe] += 1
    return dict(counts)

def timeline_group(cves):
    #Sorts CVEs by published month (YYYY-MM), counts them.
    #Used for timeline graphs. Handles weird/missing dates gracefully!
    timeline = defaultdict(int)
    for cve in cves:
        date = cve.get("Published", "")
        if date:
            try:
                month = date[:7]  # "YYYY-MM"
                timeline[month] += 1
            except:
                continue
    sorted_months = sorted(timeline.items())
    labels = [item[0] for item in sorted_months]
    values = [item[1] for item in sorted_months]
    return {"labels": labels, "values": values}
```

```
def top_n_cwes(cwe_counts, n=7):
    #Find the top N CWEs (by count). Used for leaderboards/bar charts.
    return sorted(cwe_counts, key=cwe_counts.get, reverse=True)[:n]

def yearly_trends(cves):
    #Count how many CVEs were published in each of the last 5 years.
    #Used for yearly trend graphs.
    current_year = datetime.now().year
    year_counts = {year: 0 for year in range(current_year - 4, current_year +
1)}

    for cve in cves:
        published = cve.get('Published', '')
        if published:
            try:
                year = int(published[:4])
                if year in year_counts:
                    year_counts[year] += 1
            except ValueError:
                continue

    return year_counts
```