In [4]:
```
# Arabic News Scraper for Al Jazeera Arabic using XML Sitemaps
import os, re, html, json, time
from datetime import datetime, timedelta, timezone
from typing import List, Dict, Optional, Tuple
import pandas as pd
import requests
from bs4 import BeautifulSoup
from fpdf import FPDF

# ---------------------------- SETTINGS ----------------------------
DAYS_BACK = 1000
TARGET_MAX = 20000
REQUEST_TIMEOUT = 20
PAUSE_BETWEEN_REQUESTS = 0.15
MAX_CHILD_SITEMAPS = 600

SITEMAP_CANDIDATES = [
    "https://www.aljazeera.net/sitemap.xml",
]

HEADERS = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)"
}

BUCKET_RULES = {
    "الشرق الأوسط": [r"/middleeast", r"الشرق الأوسط"],
    "فلسطين": [r"/palestine", r"فلسطين"],
    "سوريا": [r"/syria", r"سوريا"],
    "الاقتصاد": [r"/economy", r"الاقتصاد"],
    "الصحة": [r"/health", r"الصحة"],
    "الرياضة": [r"/sports", r"رياضة"],
    "التكنولوجيا": [r"/tech", r"تكنولوجيا"],
    "الرأي": [r"/opinion", r"رأي"],
    "فيديو": [r"/video", r"فيديو"],
    "أخبار": [r"/news", r"أخبار"],
}

# ---------------------------- UTILS ----------------------------
session = requests.Session()

def http_get(url, *, stream=False):
    for attempt in range(3):
        try:
            r = session.get(url, headers=HEADERS, timeout=REQUEST_TIMEOUT, str
            if r.status_code == 200:
                return r
            if r.status_code in (429, 500, 502, 503, 504):
                time.sleep(1 + attempt)
        except requests.RequestException:
            time.sleep(1 + attempt)
    return None

def parse_xml(xml_bytes):
    return BeautifulSoup(xml_bytes, "xml")

def iso_to_dt(s: str) -> Optional[datetime]:
    if not s:
```

```
        return None
    s = s.replace("Z", "+00:00")
    try:
        return datetime.fromisoformat(s)
    except:
        try:
            return datetime.strptime(s[:10], "%Y-%m-%d")
        except:
            return None

def collect_from_index(soup, start_dt, end_dt):
    items = []
    for sm in soup.find_all("sitemap"):
        loc = sm.loc.text.strip() if sm.loc else None
        lastmod = iso_to_dt(sm.lastmod.text.strip()) if sm.lastmod else None
        if not loc or "aljazeera.net" not in loc:
            continue
        if lastmod and lastmod.replace(tzinfo=None) < start_dt - timedelta(day
            continue
        items.append((loc, lastmod))
    items.sort(key=lambda x: x[1] or datetime.min, reverse=True)
    return items

def collect_sitemap_urls(sitemap_url, start_dt, end_dt):
    r = http_get(sitemap_url)
    if r is None:
        return []
    soup = parse_xml(r.content)
    out = []

    sitemap_nodes = soup.find_all("sitemap")
    if sitemap_nodes:
        children = collect_from_index(soup, start_dt, end_dt)
        for i, (child_url, _) in enumerate(children[:MAX_CHILD_SITEMAPS]):
            r2 = http_get(child_url)
            if r2 is None:
                continue
            soup2 = parse_xml(r2.content)
            for u in soup2.find_all("url"):
                loc = u.loc.text.strip() if u.loc else None
                lastmod = u.lastmod.text.strip() if u.lastmod else ""
                if loc and "aljazeera.net" in loc:
                    out.append({"loc": loc, "lastmod": lastmod})
        return out

    for u in soup.find_all("url"):
        loc = u.loc.text.strip() if u.loc else None
        lastmod = u.lastmod.text.strip() if u.lastmod else ""
        if loc and "aljazeera.net" in loc:
            out.append({"loc": loc, "lastmod": lastmod})
    return out

def within_date(lastmod_str, start_dt, end_dt):
    if not lastmod_str:
        return True
    dt = iso_to_dt(lastmod_str)
    if not dt:
```

```
        return True
    return start_dt <= dt.replace(tzinfo=None) <= end_dt

def clean_text(txt: Optional[str]) -> str:
    if not txt:
        return ""
    t = BeautifulSoup(txt, "lxml").get_text(" ", strip=True)
    return html.unescape(t)

def extract_article(url: str) -> Dict[str, str]:
    r = http_get(url)
    if r is None:
        return {}
    soup = BeautifulSoup(r.content, "lxml")

    title = soup.find("h1")
    title = title.get_text(strip=True) if title else soup.title.string if soup

    summary = ""
    first_p = soup.find("p")
    if first_p:
        summary = first_p.get_text(strip=True)

    date_meta = soup.find("meta", attrs={"property": "article:published_time"}
    date = date_meta["content"] if date_meta and date_meta.get("content") else

    return {
        "title": clean_text(title),
        "summary": clean_text(summary),
        "date": date
    }

def guess_bucket(url: str, extra_text: str = "") -> str:
    hay = f"{url} || {extra_text}".lower()
    for bucket, patterns in BUCKET_RULES.items():
        for pat in patterns:
            if re.search(pat, hay, flags=re.I):
                return bucket
    return "أخرى"

def save_pdf(category: str, df: pd.DataFrame, out_path: str, font_path="arial.
    pdf = FPDF(orientation="P", unit="mm", format="A4")
    pdf.add_page()
    pdf.add_font("Arabic", "", font_path, uni=True)
    pdf.set_font("Arabic", "", 14)
    pdf.cell(0, 10, f"{category} - الجزيرة", ln=True, align="R")
    pdf.ln(4)
    for _, row in df.iterrows():
        pdf.set_font("Arabic", "", 12)
        pdf.multi_cell(0, 8, f"📌 العنوان: {row['title']}", align="R")
        pdf.set_font("Arabic", "", 10)
        if row['date']:
            pdf.multi_cell(0, 8, f"🕒 التاريخ: {row['date']}", align="R")
        if row['summary']:
            pdf.multi_cell(0, 8, f"📄 الملخص: {row['summary']}", align="R")
        if row['url']:
            pdf.multi_cell(0, 8, f"🔗 الرابط: {row['url']}", align="R")
```

```
        pdf.ln(3)
    pdf.output(out_path)

# ---------------------------- MAIN ----------------------------
start_dt = (datetime.now(timezone.utc) - timedelta(days=DAYS_BACK)).replace(tz
end_dt = datetime.now(timezone.utc).replace(tzinfo=None)
print(f"🔎 Scraping from {start_dt.date()} to {end_dt.date()}")

# Discover sitemap
all_entries = []
for sm in SITEMAP_CANDIDATES:
    entries = collect_sitemap_urls(sm, start_dt, end_dt)
    all_entries.extend(entries)

print(f"📦 Found {len(all_entries)} raw URLs")

# Filter by date & remove duplicates
seen = set()
filtered = []
for e in all_entries:
    loc = e["loc"]
    if loc in seen:
        continue
    if within_date(e.get("lastmod", ""), start_dt, end_dt):
        filtered.append(loc)
        seen.add(loc)

if len(filtered) > TARGET_MAX:
    filtered = filtered[:TARGET_MAX]
print(f"✅ Retained {len(filtered)} articles")

# Scrape articles
rows = []
for i, url in enumerate(filtered, 1):
    meta = extract_article(url)
    bucket = guess_bucket(url, meta.get("title", ""))
    rows.append({
        "bucket": bucket,
        "title": meta.get("title", ""),
        "url": url,
        "date": meta.get("date", ""),
        "summary": meta.get("summary", ""),
    })
    if i % 20 == 0:
        print(f"… fetched {i}/{len(filtered)}")

df = pd.DataFrame(rows)
df.to_excel("arabic_news.xlsx", index=False)
df.to_csv("arabic_news.csv", index=False, encoding="utf-8-sig")
save_pdf("جميع الأخبار", df, "arabic_news.pdf")

print("✅ Exported: arabic_news.xlsx / arabic_news.csv / arabic_news.pdf")
```

```
… fetched 19660/20000
… fetched 19680/20000
… fetched 19700/20000
… fetched 19720/20000
… fetched 19740/20000
… fetched 19760/20000
… fetched 19780/20000
… fetched 19800/20000
… fetched 19820/20000
… fetched 19840/20000
… fetched 19860/20000
… fetched 19880/20000
… fetched 19900/20000
… fetched 19920/20000
… fetched 19940/20000
… fetched 19960/20000
… fetched 19980/20000
… fetched 20000/20000
```

In [ ]: