```
pip install requests transformers torch tqdm newspaper3k
```

```
  Downloading requests_file-2.1.0-py2.py3-none-any.whl (4.2 kB)
Building wheels for collected packages: tinysegmenter, feedfinder2, jieba3k, sgmllib3k
  Building wheel for tinysegmenter (setup.py) ... done
  Created wheel for tinysegmenter: filename=tinysegmenter-0.3-py3-none-any.whl size=13540 sha256=3a606e01028a505ecab6baab2
  Stored in directory: /root/.cache/pip/wheels/fc/ab/f8/cce3a9ae6d828bd346be695f7ff54612cd22b7cbd7208d68f3
  Building wheel for feedfinder2 (setup.py) ... done
  Created wheel for feedfinder2: filename=feedfinder2-0.0.4-py3-none-any.whl size=3341 sha256=556057a7b58676e55d9edfdfbd4a
  Stored in directory: /root/.cache/pip/wheels/80/d5/72/9cd9eccc819636436c6a6e59c22a0fb1ec167beef141f56491
  Building wheel for jieba3k (setup.py) ... done
  Created wheel for jieba3k: filename=jieba3k-0.35.1-py3-none-any.whl size=7398380 sha256=7c5c2ae31a52359e993cdca2c8717abb
  Stored in directory: /root/.cache/pip/wheels/3a/a1/46/8e68055c1713f9c4598774c15ad0541f26d5425ee7423b6493
  Building wheel for sgmllib3k (setup.py) ... done
  Created wheel for sgmllib3k: filename=sgmllib3k-1.0.0-py3-none-any.whl size=6046 sha256=5b3fe8aa6c204baa9334fd337a3f9865
  Stored in directory: /root/.cache/pip/wheels/3b/25/2a/105d6a15df6914f4d15047691c6c28f9052cc1173e40285d03
Successfully built tinysegmenter feedfinder2 jieba3k sgmllib3k
Installing collected packages: tinysegmenter, sgmllib3k, jieba3k, nvidia-nvjitlink-cu12, nvidia-curand-cu12, nvidia-cufft-
  Attempting uninstall: nvidia-nvjitlink-cu12
    Found existing installation: nvidia-nvjitlink-cu12 12.5.82
    Uninstalling nvidia-nvjitlink-cu12-12.5.82:
      Successfully uninstalled nvidia-nvjitlink-cu12-12.5.82
  Attempting uninstall: nvidia-curand-cu12
    Found existing installation: nvidia-curand-cu12 10.3.6.82
    Uninstalling nvidia-curand-cu12-10.3.6.82:
      Successfully uninstalled nvidia-curand-cu12-10.3.6.82
  Attempting uninstall: nvidia-cufft-cu12
    Found existing installation: nvidia-cufft-cu12 11.2.3.61
    Uninstalling nvidia-cufft-cu12-11.2.3.61:
      Successfully uninstalled nvidia-cufft-cu12-11.2.3.61
  Attempting uninstall: nvidia-cuda-runtime-cu12
    Found existing installation: nvidia-cuda-runtime-cu12 12.5.82
    Uninstalling nvidia-cuda-runtime-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-runtime-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-nvrtc-cu12
    Found existing installation: nvidia-cuda-nvrtc-cu12 12.5.82
    Uninstalling nvidia-cuda-nvrtc-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-nvrtc-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-cupti-cu12
    Found existing installation: nvidia-cuda-cupti-cu12 12.5.82
    Uninstalling nvidia-cuda-cupti-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-cupti-cu12-12.5.82
  Attempting uninstall: nvidia-cublas-cu12
    Found existing installation: nvidia-cublas-cu12 12.5.3.2
    Uninstalling nvidia-cublas-cu12-12.5.3.2:
      Successfully uninstalled nvidia-cublas-cu12-12.5.3.2
  Attempting uninstall: nvidia-cusparse-cu12
    Found existing installation: nvidia-cusparse-cu12 12.5.1.3
    Uninstalling nvidia-cusparse-cu12-12.5.1.3:
      Successfully uninstalled nvidia-cusparse-cu12-12.5.1.3
  Attempting uninstall: nvidia-cudnn-cu12
    Found existing installation: nvidia-cudnn-cu12 9.3.0.75
    Uninstalling nvidia-cudnn-cu12-9.3.0.75:
      Successfully uninstalled nvidia-cudnn-cu12-9.3.0.75
  Attempting uninstall: nvidia-cusolver-cu12
    Found existing installation: nvidia-cusolver-cu12 11.6.3.83
    Uninstalling nvidia-cusolver-cu12-11.6.3.83:
      Successfully uninstalled nvidia-cusolver-cu12-11.6.3.83
Successfully installed cssselect-1.3.0 feedfinder2-0.0.4 feedparser-6.0.11 jieba3k-0.35.1 newspaper3k-0.2.8 nvidia-cublas-
```

```
pip install lxml[html_clean]
```

```
Requirement already satisfied: lxml[html_clean] in /usr/local/lib/python3.11/dist-packages (5.4.0)
Collecting lxml_html_clean (from lxml[html_clean])
  Downloading lxml_html_clean-0.4.2-py3-none-any.whl.metadata (2.4 kB)
Downloading lxml_html_clean-0.4.2-py3-none-any.whl (14 kB)
Installing collected packages: lxml_html_clean
Successfully installed
```



```
pip install newspaper3k
```

```
Requirement already satisfied: newspaper3k in /usr/local/lib/python3.11/dist-packages (0.2.8)
Requirement already satisfied: beautifulsoup4>=4.4.1 in /usr/local/lib/python3.11/dist-packages (from newspaper3k) (4.13.4)
Requirement already satisfied: Pillow>=3.3.0 in /usr/local/lib/python3.11/dist-packages (from newspaper3k) (11.2.1)
Requirement already satisfied: PyYAML>=3.11 in /usr/local/lib/python3.11/dist-packages (from newspaper3k) (6.0.2)
Requirement already satisfied: cssselect>=0.9.2 in /usr/local/lib/python3.11/dist-packages (from newspaper3k) (1.3.0)
Requirement already satisfied: lxml>=3.6.0 in /usr/local/lib/python3.11/dist-packages (from newspaper3k) (5.4.0)
Requirement already satisfied: nltk>=3.2.1 in /usr/local/lib/python3.11/dist-packages (from newspaper3k) (3.9.1)
Requirement already satisfied: requests>=2.10.0 in /usr/local/lib/python3.11/dist-packages (from newspaper3k) (2.32.3)
Requirement already satisfied: feedparser>=5.2.1 in /usr/local/lib/python3.11/dist-packages (from newspaper3k) (6.0.11)
Requirement already satisfied: tldextract>=2.0.1 in /usr/local/lib/python3.11/dist-packages (from newspaper3k) (5.3.0)
Requirement already satisfied: feedfinder2>=0.0.4 in /usr/local/lib/python3.11/dist-packages (from newspaper3k) (0.0.4)
Requirement already satisfied: jieba3k>=0.35.1 in /usr/local/lib/python3.11/dist-packages (from newspaper3k) (0.35.1)
Requirement already satisfied: python-dateutil>=2.5.3 in /usr/local/lib/python3.11/dist-packages (from newspaper3k) (2.9.0.p
Requirement already satisfied: tinysegmenter==0.3 in /usr/local/lib/python3.11/dist-packages (from newspaper3k) (0.3)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.11/dist-packages (from beautifulsoup4>=4.4.1->newspap
Requirement already satisfied: typing-extensions>=4.0.0 in /usr/local/lib/python3.11/dist-packages (from beautifulsoup4>=4.4
Requirement already satisfied: six in /usr/local/lib/python3.11/dist-packages (from feedfinder2>=0.0.4->newspaper3k) (1.17.0
Requirement already satisfied: sgmllib3k in /usr/local/lib/python3.11/dist-packages (from feedparser>=5.2.1->newspaper3k) (1
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packages (from nltk>=3.2.1->newspaper3k) (8.2.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (from nltk>=3.2.1->newspaper3k) (1.5.1)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.11/dist-packages (from nltk>=3.2.1->newspaper3k) (2
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from nltk>=3.2.1->newspaper3k) (4.67.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests>=2.10.0->n
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests>=2.10.0->newspaper3k)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests>=2.10.0->newspap
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests>=2.10.0->newspap
Requirement already satisfied: requests-file>=1.4 in /usr/local/lib/python3.11/dist-packages (from tldextract>=2.0.1->newspa
Requirement already satisfied: filelock>=3.0.8 in /usr/local/lib/python3.11/dist-packages (from tldextract>=2.0.1->newspaper
```

First Attempts

```python
# summarize_gnews.py

import requests
from datetime import datetime, timedelta
from transformers import pipeline
from newspaper import Article
import json
from tqdm import tqdm

GNEWS_API_KEY = "API KEY"
CATEGORIES = ["world", "business", "technology"]
MAX_RESULTS = 30  # total per day across all categories

summarizer = pipeline("summarization", model="t5-small", tokenizer="t5-small", framework="pt")

def fetch_articles(category):
    date_yesterday = (datetime.utcnow() - timedelta(days=1)).strftime("%Y-%m-%d")
    url = f"https://gnews.io/api/v4/search?q={category}&lang=en&from={date_yesterday}&max=10&token={GNEWS_API_KEY}"
    res = requests.get(url)
    return res.json().get("articles", [])

def extract_full_text(article_url):
    try:
        a = Article(article_url)
        a.download()
        a.parse()
        return a.text
    except:
        return None

def summarize_text(text):
    if len(text.split()) < 40:
        return text.strip()  # too short to summarize
    trimmed = " ".join(text.split()[:300])  # reduce token load
    summary = summarizer("summarize: " + trimmed, max_length=50, min_length=20, do_sample=False)
    return summary[0]["summary_text"].strip()

def main():
    summarized_articles = []
```

```
    for category in CATEGORIES:
        articles = fetch_articles(category)
        for article in tqdm(articles, desc=f"Processing {category}"):
            full_text = extract_full_text(article["url"]) or article.get("description", "")
            if not full_text:
                continue
            summary = summarize_text(full_text)
            summarized_articles.append({
                "title": article["title"],
                "summary": summary,
                "url": article["url"],
                "category": category,
                "published": article.get("publishedAt", "")
            })
            if len(summarized_articles) >= MAX_RESULTS:
                break

    with open("daily_summary.json", "w", encoding="utf-8") as f:
        json.dump(summarized_articles, f, indent=2)

    print(f"✅ Saved {len(summarized_articles)} summaries to daily_summary.json")

if __name__ == "__main__":
    main()
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
config.json: 100%                                                  1.21k/1.21k [00:00<00:00, 27.9kB/s]
model.safetensors: 100%                                              242M/242M [00:07<00:00, 34.1MB/s]
generation_config.json: 100%                                         147/147 [00:00<00:00, 2.78kB/s]
tokenizer_config.json: 100%                                        2.32k/2.32k [00:00<00:00, 167kB/s]
spiece.model: 100%                                                 792k/792k [00:00<00:00, 1.92MB/s]
tokenizer.json: 100%                                               1.39M/1.39M [00:00<00:00, 6.64MB/s]
Device set to use cpu
Processing world:   0%|          | 0/10 [00:00<?, ?it/s]Both `max_new_tokens` (=256) and `max_length`(=50) seem to have been
Processing world:  10%|▉         | 1/10 [00:08<01:19,  8.88s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem to h
Processing world:  20%|█▊        | 2/10 [00:15<00:59,  7.45s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem to h
Processing world:  30%|██▋       | 3/10 [00:20<00:44,  6.42s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem to h
Processing world:  40%|███▌      | 4/10 [00:26<00:38,  6.41s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem to h
Processing world:  50%|████▌     | 5/10 [00:31<00:27,  5.58s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem to h
Processing world:  60%|█████▍    | 6/10 [00:34<00:19,  4.88s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem to h
Processing world:  70%|██████▎   | 7/10 [00:39<00:15,  5.02s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem to h
Processing world:  80%|███████▏  | 8/10 [00:43<00:09,  4.68s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem to h
Processing world:  90%|████████▏ | 9/10 [00:47<00:04,  4.38s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem to h
Processing world: 100%|█████████| 10/10 [00:51<00:00,  5.12s/it]
Processing business:   0%|          | 0/10 [00:00<?, ?it/s]Both `max_new_tokens` (=256) and `max_length`(=50) seem to have b
Processing business:  10%|▉         | 1/10 [00:04<00:44,  4.91s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem t
Processing business:  20%|█▊        | 2/10 [00:09<00:39,  4.97s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem t
Processing business:  30%|██▋       | 3/10 [00:19<00:48,  6.95s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem t
Processing business:  40%|███▌      | 4/10 [00:27<00:43,  7.30s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem t
Processing business:  50%|████▌     | 5/10 [00:32<00:33,  6.64s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem t
Processing business:  60%|█████▍    | 6/10 [00:37<00:24,  6.15s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem t
Processing business:  70%|██████▎   | 7/10 [00:46<00:20,  6.93s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem t
Processing business:  80%|███████▏  | 8/10 [00:53<00:14,  7.14s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem t
Processing business:  90%|████████▏ | 9/10 [01:00<00:06,  6.98s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem t
Processing business: 100%|█████████| 10/10 [01:10<00:00,  7.02s/it]
Processing technology:   0%|          | 0/10 [00:00<?, ?it/s]Both `max_new_tokens` (=256) and `max_length`(=50) seem to have
Processing technology:  10%|▉         | 1/10 [00:03<00:29,  3.27s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem
Processing technology:  20%|█▊        | 2/10 [00:06<00:26,  3.37s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem
Processing technology:  30%|██▋       | 3/10 [00:10<00:23,  3.41s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem
Processing technology:  40%|███▌      | 4/10 [00:14<00:22,  3.75s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem
Processing technology:  50%|████▌     | 5/10 [00:18<00:19,  3.94s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem
Processing technology:  70%|██████▎   | 7/10 [00:23<00:08,  2.92s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem
Processing technology:  80%|███████▏  | 8/10 [00:28<00:07,  3.54s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem
Processing technology:  90%|████████▏ | 9/10 [00:34<00:04,  4.39s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem
Processing technology:  90%|████████▏ | 9/10 [00:41<00:04,  4.64s/it]✅ Saved 30 summaries to daily_summary.json
```

```python
# summarize_gnews.py

import requests
from datetime import datetime, timedelta
from transformers import pipeline
from newspaper import Article
import json
from tqdm import tqdm

# === CONFIG ===
GNEWS_API_KEY = "API KEY"
CATEGORIES = ["world", "business", "technology"]
MAX_RESULTS = 30  # total across all categories
ARTICLES_PER_CATEGORY = 10

# === SUMMARIZATION MODEL ===
summarizer = pipeline("summarization", model="t5-small", tokenizer="t5-small", framework="pt")

# === FUNCTIONS ===
```

```python
def fetch_articles(category):
    """Fetches articles from GNews API."""
    date_yesterday = (datetime.utcnow() - timedelta(days=1)).strftime("%Y-%m-%d")
    url = f"https://gnews.io/api/v4/search?q={category}&lang=en&from={date_yesterday}&max={ARTICLES_PER_CATEGORY}&token={GNEWS_AP
    res = requests.get(url)
    if res.status_code != 200:
        return []
    return res.json().get("articles", [])

def extract_full_text(article_url):
    """Attempts to extract full article text using newspaper3k."""
    try:
        a = Article(article_url)
        a.download()
        a.parse()
        return a.text
    except:
        return None

def summarize_text(text):
    """Summarizes a block of text using T5-small."""
    if len(text.split()) < 40:
        return text.strip()
    trimmed = " ".join(text.split()[:300])
    summary = summarizer("summarize: " + trimmed, max_length=50, min_length=20, do_sample=False)
    return summary[0]["summary_text"].strip()

# === MAIN SCRIPT ===

def main():
    summarized_articles = []

    for category in CATEGORIES:
        articles = fetch_articles(category)
        for article in tqdm(articles, desc=f"Processing {category}"):
            full_text = extract_full_text(article["url"]) or article.get("description", "")
            if not full_text:
                continue
            summary = summarize_text(full_text)
            summarized_articles.append({
                "title": article["title"],
                "summary": summary,
                "url": article["url"],
                "category": category,
                "published": article.get("publishedAt", "")
            })
            if len(summarized_articles) >= MAX_RESULTS:
                break
        if len(summarized_articles) >= MAX_RESULTS:
            break

    with open("daily_summary.json", "w", encoding="utf-8") as f:
        json.dump(summarized_articles, f, indent=2)

    print(f"Saved {len(summarized_articles)} summaries to daily_summary.json")

if __name__ == "__main__":
    main()
```

```
Device set to use cpu
Processing world:   0%|          | 0/10 [00:00<?, ?it/s]Both `max_new_tokens` (=256) and `max_length`(=50) seem to have been
Processing world:  10%|▉         | 1/10 [00:06<00:56,  6.30s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem to h
Processing world:  20%|█▊        | 2/10 [00:12<00:50,  6.35s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem to h
Processing world:  30%|██▋       | 3/10 [00:17<00:40,  5.85s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem to h
Processing world:  40%|███▌      | 4/10 [00:23<00:33,  5.64s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem to h
Processing world:  50%|████▌     | 5/10 [00:27<00:24,  5.00s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem to h
Processing world:  60%|█████▍    | 6/10 [00:30<00:18,  4.51s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem to h
Processing world:  70%|██████▎   | 7/10 [00:36<00:14,  4.83s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem to h
Processing world:  80%|███████▏  | 8/10 [00:40<00:09,  4.62s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem to h
Processing world:  90%|████████▏ | 9/10 [00:44<00:04,  4.40s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem to h
```

```
Processing world: 100%|██████████| 10/10 [00:48<00:00,  4.83s/it]
Processing business:   0%|          | 0/10 [00:00<?, ?it/s]Both `max_new_tokens` (=256) and `max_length`(=50) seem to have b
Processing business:  10%|█         | 1/10 [00:05<00:46,  5.16s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem t
Processing business:  20%|██        | 2/10 [00:10<00:41,  5.13s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem t
Processing business:  30%|███       | 3/10 [00:17<00:43,  6.28s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem t
Processing business:  40%|████      | 4/10 [00:22<00:33,  5.59s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem t
Processing business:  50%|█████     | 5/10 [00:28<00:28,  5.66s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem t
Processing business:  60%|██████    | 6/10 [00:33<00:21,  5.42s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem t
Processing business:  70%|███████   | 7/10 [00:39<00:17,  5.70s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem t
Processing business:  80%|████████  | 8/10 [00:44<00:11,  5.64s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem t
Processing business:  90%|█████████ | 9/10 [00:50<00:05,  5.55s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem t
Processing business: 100%|██████████| 10/10 [00:57<00:00,  5.73s/it]
Processing technology:   0%|          | 0/10 [00:00<?, ?it/s]Both `max_new_tokens` (=256) and `max_length`(=50) seem to have
Processing technology:  10%|█         | 1/10 [00:04<00:36,  4.05s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem
Processing technology:  20%|██        | 2/10 [00:08<00:32,  4.10s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem
Processing technology:  30%|███       | 3/10 [00:12<00:29,  4.23s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem
Processing technology:  40%|████      | 4/10 [00:16<00:25,  4.20s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem
Processing technology:  50%|█████     | 5/10 [00:20<00:20,  4.17s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem
Processing technology:  70%|███████   | 7/10 [00:25<00:09,  3.03s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem
Processing technology:  80%|████████  | 8/10 [00:31<00:07,  3.74s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem
Processing technology:  90%|█████████ | 9/10 [00:37<00:04,  4.51s/it]Both `max_new_tokens` (=256) and `max_length`(=50) seem
Processing technology:  90%|█████████ | 9/10 [00:43<00:04,  4.88s/it]Saved 30 summaries to daily_summary.json
```

```
import requests
from datetime import datetime, timedelta
from transformers import pipeline
from newspaper import Article
import json
from tqdm import tqdm

# === CONFIG ===
GNEWS_API_KEY = "API KEY"
CATEGORIES = ["world", "business", "technology"]
ARTICLES_PER_CATEGORY = 30

# === SUMMARIZATION MODEL ===
summarizer = pipeline("summarization", model="facebook/bart-large-cnn", tokenizer="facebook/bart-large-cnn", framework="pt")

# === FUNCTIONS ===

def fetch_articles(category):
    """Fetches articles from GNews API."""
    date_yesterday = (datetime.utcnow() - timedelta(days=1)).strftime("%Y-%m-%d")
    url = f"https://gnews.io/api/v4/search?q={category}&lang=en&from={date_yesterday}&max={ARTICLES_PER_CATEGORY}&token={GNEWS_AP
    res = requests.get(url)
    if res.status_code != 200:
        return []
    return res.json().get("articles", [])

def extract_full_text(article_url):
    """Attempts to extract full article text using newspaper3k."""
    try:
        a = Article(article_url)
        a.download()
        a.parse()
        return a.text
    except:
        return None

def summarize_paragraph(text):
    """Summarizes a long block of concatenated article texts."""
    if len(text.split()) < 40:
        return text.strip()
    trimmed = " ".join(text.split()[:500])  # Cap to avoid overloading
    result = summarizer(trimmed, max_new_tokens=120, do_sample=False)
    return result[0]["summary_text"].strip()

# === MAIN SCRIPT ===

def main():
    category_summaries = {}
```

```
    for category in CATEGORIES:
        all_text = []
        articles = fetch_articles(category)

        for article in tqdm(articles, desc=f"Processing {category}"):
            full_text = extract_full_text(article["url"]) or article.get("description", "")
            if full_text:
                all_text.append(full_text.strip())

        combined_text = ". ".join(all_text)
        if combined_text:
            category_summary = summarize_paragraph(combined_text)
            category_summaries[category] = category_summary

    with open("daily_summary.json", "w", encoding="utf-8") as f:
        json.dump(category_summaries, f, indent=2)

    print(f"Saved category-wise paragraph summaries to daily_summary.json")

if __name__ == "__main__":
    main()
```

```
config.json:         1.58k/? [00:00<00:00, 69.9kB/s]
model.safetensors: 100%                                                  1.63G/1.63G [00:53<00:00, 74.6MB/s]
generation_config.json: 100%                                               363/363 [00:00<00:00, 27.3kB/s]
vocab.json:         899k/? [00:00<00:00, 3.08MB/s]
merges.txt:         456k/? [00:00<00:00, 18.8MB/s]
tokenizer.json:        1.36M/? [00:00<00:00, 21.0MB/s]
Device set to use cpu
Processing world: 100%|██████████| 10/10 [00:10<00:00,  1.10s/it]
Processing business: 100%|██████████| 10/10 [00:11<00:00,  1.16s/it]
Processing technology: 100%|██████████| 10/10 [00:10<00:00,  1.00s/it]
Saved category-wise paragraph summaries to daily_summary.json
```

```
import requests
from datetime import datetime, timedelta
from transformers import pipeline
from newspaper import Article
import json
from tqdm import tqdm
import re

# === CONFIG ===
GNEWS_API_KEY = "API KEY"
CATEGORY_QUERIES = {
    "world": "global OR international OR diplomacy",
    "business": "economy OR finance OR market OR stock",
    "technology": "AI OR tech OR startup OR software"
}
ARTICLES_PER_CATEGORY = 30
MIN_WORDS_REQUIRED = 80

# === SUMMARIZATION MODEL ===
summarizer = pipeline(
    "summarization",
    model="facebook/bart-large-cnn",
    tokenizer="facebook/bart-large-cnn",
    framework="pt"
)

def fetch_articles(query):
    """Fetches articles from GNews API."""
    date_yesterday = (datetime.utcnow() - timedelta(days=1)).strftime("%Y-%m-%d")
    url = f"https://gnews.io/api/v4/search?q={query}&lang=en&from={date_yesterday}&max={ARTICLES_PER_CATEGORY}&token={GNEWS_API_
```

```
    res = requests.get(url)
    if res.status_code != 200:
        return []
    return res.json().get("articles", [])

def extract_full_text(article_url):
    """Extracts full article text using newspaper3k."""
    try:
        a = Article(article_url)
        a.download()
        a.parse()
        return a.text
    except:
        return None

def clean_text(text):
    """Removes weird characters, excessive whitespace, etc."""
    text = re.sub(r'\s+', ' ', text)
    text = re.sub(r'[^\x00-\x7F]+', ' ', text)  # remove non-ASCII
    return text.strip()

def summarize_paragraph(text, max_tokens=1024):
    """Chunks and summarizes large text if needed."""
    words = text.split()
    chunks = [' '.join(words[i:i+500]) for i in range(0, len(words), 500)]
    summaries = []
    for chunk in chunks:
        try:
            summary = summarizer(chunk, max_new_tokens=160, do_sample=False)
            summaries.append(summary[0]["summary_text"].strip())
        except Exception:
            continue
    return ' '.join(summaries)

def main():
    category_summaries = {}

    for category, query in CATEGORY_QUERIES.items():
        print(f"\n=== Processing {category.upper()} ===")
        all_text = []
        articles = fetch_articles(query)

        for article in tqdm(articles, desc=f"{category.title()} Articles"):
            full_text = extract_full_text(article["url"]) or article.get("description", "")
            if not full_text or len(full_text.split()) < MIN_WORDS_REQUIRED:
                continue
            clean = clean_text(full_text)
            if clean:
                all_text.append(clean)

        combined_text = ". ".join(all_text)
        if combined_text:
            category_summary = summarize_paragraph(combined_text)
            category_summaries[category] = category_summary
        else:
            category_summaries[category] = "No substantial content available to summarize."

    with open("daily_summary.json", "w", encoding="utf-8") as f:
        json.dump(category_summaries, f, indent=2)

    print(f"\n✅ Saved category-wise paragraph summaries to daily_summary.json")

if __name__ == "__main__":
    main()
```

```
Device set to use cpu

=== Processing WORLD ===
World Articles: 100%|██████████| 10/10 [00:06<00:00,  1.44it/s]
```

```
=== Processing BUSINESS ===
Business Articles: 100%|██████████| 10/10 [00:13<00:00,  1.31s/it]

=== Processing TECHNOLOGY ===
Technology Articles: 100%|██████████| 10/10 [00:08<00:00,  1.20it/s]

✅ Saved category-wise paragraph summaries to daily_summary.json
```

```
import requests
from datetime import datetime, timedelta
from transformers import pipeline
from newspaper import Article
from tqdm import tqdm
import json
import re

# === CONFIG ===
GNEWS_API_KEY = "API KEY"
CATEGORY_QUERIES = {
    "world": "global OR international OR diplomacy",
    "business": "economy OR finance OR market OR stock",
    "technology": "AI OR tech OR startup OR software"
}
ARTICLES_PER_CATEGORY = 25
MIN_WORDS_REQUIRED = 100
MAX_SUMMARIES_PER_CATEGORY = 5

# === SUMMARIZATION MODEL ===
summarizer = pipeline(
    "summarization",
    model="facebook/bart-large-cnn",
    tokenizer="facebook/bart-large-cnn",
    framework="pt"
)

def fetch_articles(query):
    """Fetch articles from GNews API."""
    date_yesterday = (datetime.utcnow() - timedelta(days=1)).strftime("%Y-%m-%d")
    url = f"https://gnews.io/api/v4/search?q={query}&lang=en&from={date_yesterday}&max={ARTICLES_PER_CATEGORY}&token={GNEWS_API_
    res = requests.get(url)
    return res.json().get("articles", []) if res.status_code == 200 else []

def extract_full_text(url):
    """Extract full text from article using newspaper3k."""
    try:
        article = Article(url)
        article.download()
        article.parse()
        return article.text
    except:
        return None

def clean_text(text):
    """Remove non-ASCII and excessive whitespace."""
    text = re.sub(r'\s+', ' ', text)
    text = re.sub(r'[^\x00-\x7F]+', ' ', text)
    return text.strip()

def is_junk(text):
    """Filter out articles with only tickers, locations, or lists."""
    return (
        len(text.split()) < MIN_WORDS_REQUIRED or
        re.search(r'\b(Dow|TSX|Nikkei|NASDAQ|USD|S&P)\b', text) or
        re.search(r'\b[A-Z][a-z]+, [A-Z][a-z]+, [A-Z][a-z]+\b', text)  # country dump pattern
    )

def summarize(text):
    """Summarize text with the transformer model."""
    try:
        return summarizer(text, max_new_tokens=120, do_sample=False)[0]["summary_text"].strip()
```

```
    except:
        return None

def process_category(name, query):
    """Process a single category and return a concise summary."""
    print(f"\n=== Processing {name.upper()} ===")
    articles = fetch_articles(query)
    summaries = []

    for article in tqdm(articles, desc=f"{name.title()} Articles"):
        raw_text = extract_full_text(article["url"]) or article.get("description", "")
        if not raw_text:
            continue

        cleaned = clean_text(raw_text)
        if is_junk(cleaned):
            continue

        summary = summarize(cleaned[:1024])  # truncate to fit model input
        if summary:
            summaries.append(summary)

        if len(summaries) >= MAX_SUMMARIES_PER_CATEGORY:
            break

    if summaries:
        return ' '.join(summaries)
    return "No substantial content available to summarize."

def main():
    result = {}
    for category, query in CATEGORY_QUERIES.items():
        result[category] = process_category(category, query)

    with open("daily_summary.json", "w", encoding="utf-8") as f:
        json.dump(result, f, indent=2)
    print("\n✅ Saved daily_summary.json")

if __name__ == "__main__":
    main()
```

```
Device set to use cpu

=== Processing WORLD ===
World Articles:  60%|██████    | 6/10 [01:55<01:16, 19.24s/it]

=== Processing BUSINESS ===
Business Articles: 100%|██████████| 10/10 [01:09<00:00,  6.93s/it]

=== Processing TECHNOLOGY ===
Technology Articles:  40%|████      | 4/10 [01:37<02:26, 24.36s/it]
✅ Saved daily_summary.json
```

Final

```
import requests
from datetime import datetime, timedelta
from newspaper import Article
from transformers import pipeline
import json
import re
from tqdm import tqdm

# === CONFIG ===
GNEWS_API_KEY = "API KEY"
CATEGORY_QUERIES = {
    "world": "global OR international OR diplomacy",
    "business": "economy OR finance OR market OR stock",
```

```
    "business": "economy OR finance OR market OR stock",
    "technology": "AI OR tech OR startup OR software"
}
ARTICLES_PER_CATEGORY = 30
MIN_WORDS_REQUIRED = 80
MAX_SUMMARIES = 5

# === SUMMARIZER ===
summarizer = pipeline(
    "summarization",
    model="facebook/bart-large-cnn",
    tokenizer="facebook/bart-large-cnn",
    framework="pt"
)

def fetch_articles(query):
    date_yesterday = (datetime.utcnow() - timedelta(days=1)).strftime("%Y-%m-%d")
    url = f"https://gnews.io/api/v4/search?q={query}&lang=en&from={date_yesterday}&max={ARTICLES_PER_CATEGORY}&token={GNEWS_API_K
    res = requests.get(url)
    if res.status_code != 200:
        return []
    return res.json().get("articles", [])

def extract_full_text(url):
    try:
        article = Article(url)
        article.download()
        article.parse()
        return article.text
    except:
        return None

def clean_text(text):
    text = re.sub(r'\s+', ' ', text)
    text = re.sub(r'[^\x00-\x7F]+', ' ', text)  # remove non-ASCII
    text = re.sub(r"(The United States.*?islands\.)", "", text, flags=re.IGNORECASE)
    text = re.sub(r"(The U\.S\. Virgin Islands.*?\.)", "", text, flags=re.IGNORECASE)
    return text.strip()

def is_valid_text(text):
    return text and len(text.split()) >= MIN_WORDS_REQUIRED

def is_repetitive(text):
    return (
        text.count("United States") > 2 or
        text.lower().count("virgin islands") > 2 or
        len(set(text.split('. '))) < len(text.split('. ')) * 0.6
    )

def summarize_text(text):
    try:
        summary = summarizer(text[:1024], max_new_tokens=120, do_sample=False)[0]['summary_text'].strip()
        return summary
    except:
        return None

def process_category(name, query):
    print(f"\n=== {name.upper()} ===")
    summaries = []
    articles = fetch_articles(query)

    for article in tqdm(articles, desc=f"{name.title()} Articles"):
        full_text = extract_full_text(article["url"]) or article.get("description", "")
        cleaned = clean_text(full_text)

        if is_valid_text(cleaned):
            summary = summarize_text(cleaned)
            if summary and not is_repetitive(summary):
                summaries.append(summary)

        if len(summaries) >= MAX_SUMMARIES:
            break
```

```
    return summaries if summaries else ["No reliable summaries found."]

def main():
```