SQL-Based Datasette Exporter

This script queries one or more Datasette endpoints using custom SQL queries, fetches the resulting data in JSON format, and saves it as CSV files.

Purpose:

To programmatically retrieve time-bounded or aggregated data from public Datasette instances using dynamic SQL, then export it for downstream analysis (e.g., logging, monitoring, reporting).

How It Works:

1. Define:
   - A dictionary of **table names and base URLs**
   - A matching list of **SQL queries**
2. Each SQL query is:
   - URL-encoded
   - Appended to the `.json` endpoint of the Datasette API
   - Fetched via HTTP request
3. Results are parsed into pandas DataFrames, optionally column-renamed (e.g. `endpoint_count → total_requests`), then saved to CSV.

In [ ]:
```
import requests
import pandas as pd
import urllib.parse
import os
import argparse

def sql_queried_datasette_tables(urls: dict, sqls: list, save_dir: str):
    """
    Fetches data from a dictionary of Datasette URLs using optional SQL queries
    and saves each result as a CSV file in the specified directory.

    Args:
        urls (dict): Mapping of table names to Datasette base URLs.
        sqls (list of str): SQL queries corresponding to each URL.
        save_dir (str): Directory path where the resulting CSV files will be saved.

    Raises:
        ValueError: If the lengths of the URLs and SQL lists do not match.
    """
    if len(urls) != len(sqls):
        raise ValueError("The number of URLs and SQL queries must match.")

    # Ensure the output directory exists
    os.makedirs(save_dir, exist_ok=True)

    # Iterate over each (name, URL) and associated SQL
    for (name, url), sql in zip(urls.items(), sqls):
        try:
            # Define the output CSV filename
            csv_name = f"{name}.csv"

            # Encode SQL query and construct JSON API URL
            encoded_sql = urllib.parse.quote(sql)
            full_url = f"{url}.json?sql={encoded_sql}&_shape=array"

            print(f"Fetching: {name} from SQL URL:\n{full_url}")

            # Fetch JSON data and load into DataFrame
            response = requests.get(full_url)
            response.raise_for_status()
            data = response.json()
            print(f"Rows returned: {len(data)}")
            df = pd.DataFrame(data)

            # rename column to match expected
            df.rename(columns={'endpoint_count': 'total_requests'}, inplace=True)

            # Save DataFrame to CSV in the specified directory
            save_path = os.path.join(save_dir, csv_name)
            df.to_csv(save_path, index=False)
            print(f"Saved: {save_path}")

        except Exception as e:
            # Log failure and continue
            print(f"Failed to fetch from {url}: {e}")

def parse_args():
    """
    Parses command-line arguments for the output directory.

    Returns:
        argparse.Namespace: Parsed arguments containing the output directory path.
    """
    parser = argparse.ArgumentParser(description="Datasette batch exporter")
    parser.add_argument(
        "--output-dir",
        type=str,
        required=True,
        help="Directory to save exported CSVs"
    )
    return parser.parse_args()

if __name__ == "__main__":
    # Parse arguments from CLI
    args = parse_args()

    # Define URLs and SQL queries to export
    urls = {
        "logs-by-week": "https://datasette.planning.data.gov.uk/digital-land"
    }

    sqls = [
        # SQL to group request status codes by week
        """
        SELECT
            COUNT(endpoint) AS endpoint_count,
            SUBSTR(entry_date, 1, 10) AS entrydate,
            DATE(entry_date, 'weekday 0', '-6 days') AS week_start,
            CASE
                WHEN status = 200 THEN '200'
                ELSE 'FAIL'
            END AS status_group
        FROM log
        WHERE
            entry_date >= DATE('now', '-6 months')
            AND SUBSTR(entry_date, 1, 10) <= DATE(entry_date, 'weekday 0', '-6 days
        GROUP BY
            entrydate,
            week_start,
            status_group
        ORDER BY
            entry_date DESC;

        """
    ]

    # Execute the export
    sql_queried_datasette_tables(urls, sqls, args.output_dir)
```