# Practical Assignment: Data Aggregation Tool

## Overview:

For this project, you will create a data aggregation tool that will extract information from various data formats and sources, store it into a relational database, and then perform data retrieval operations. This will encompass working with CSV, XML, JSON formats, and web scraping, as well as database operations using SQLite.

## Objectives:

1. Data Extraction:
    a. Extract data from a CSV file.
    b. Scrape data from a given HTML page.
    c. Parse an XML file.
    d. Read data from a JSON file.
2. Database Operations:
    a. Create a SQLite database and define the schema that can store data from the different sources.
    b. Insert the extracted data into the SQLite database.
    c. Implement parameterized queries to search for specific data within the database.
3. Data Interface:
    a. Develop a command-line interface (CLI) or a simple Python script that can handle the input and output operations.

## Assignment Breakdown:

1. Hour 1: Set up a virtual environment and install any necessary libraries. Define your database schema and initiate the SQLite database.
2. Hour 2: Write scripts to extract data from CSV, XML, and JSON files, and scrape data from a predefined HTML page.
3. Hour 3: Implement functionality to insert extracted data into the SQLite database. Ensure that data from all sources is normalized and fits the schema.
4. Hour 4: Write parameterized queries to retrieve specific records from the database. Test the entire flow from data extraction to data retrieval.

## Submission:

- A Python script(s) that performs the extraction, transformation, and loading (ETL) of data.
- The SQLite database file with the inserted data.
- A set of parameterized queries to retrieve data from the database.
- A requirements.txt file with the necessary Python packages.
- Your project should be uploaded to a new repository on GitLab or GitHub and the project link to me via email at henry@mabili.co.za .

## Evaluation Criteria:

- Correctness of data extraction and parsing from different formats.
- Successful insertion of data into the SQLite database without errors.
- Efficiency and correctness of parameterized queries.
- Code quality, readability, and documentation.

## Resources:

- Python Official Documentation
- SQLite Documentation
- BeautifulSoup Documentation for HTML Scraping
- Python's csv, xml.etree.ElementTree, and json modules.

*Remember to adhere to any data usage policies of the web pages you scrape data from.

Good luck, and make sure to apply the best practices you've learned in virtual environments and code management!