|  |  |
| --- | --- |
| **Ex No: 1**  **Date:** | **Data Collection** |

**1.Objective:**

To develop and validate a Python-based data collection pipeline that gathers real-time weather and air quality data for multiple locations in Bengaluru, and concurrently scrapes structured data from an online bookstore for potential integration into environmental, behavioral, or commercial analysis.

**2. Descriptions:**

This experiment combines three types of data sources into a single automated workflow:

**Weather Data**

Collects hourly temperature and relative humidity for five Bengaluru locations over the past 21 days via the Open-Meteo API.

**Air Quality Data**

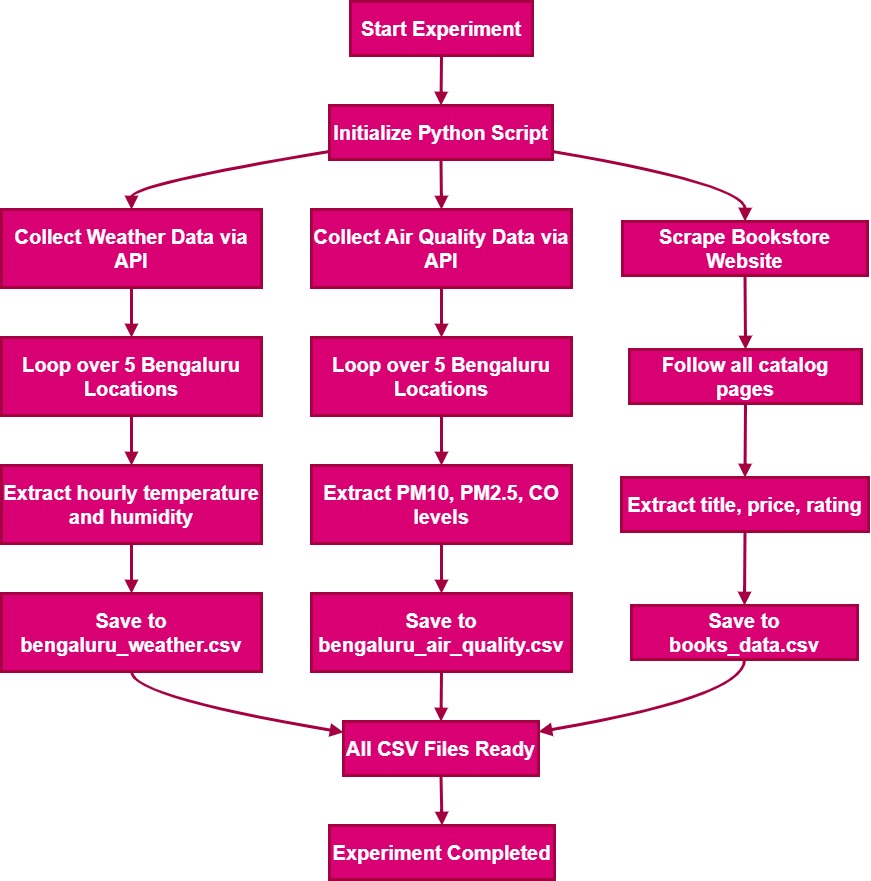
Fetches hourly pollutant concentrations (PM10, PM2.5, CO) for the same five locations using the Open-Meteo Air Quality API, covering a 30-day period.

**Bookstore Data**

Uses web scraping techniques to extract book title, price, and rating from a practice online bookstore across multiple paginated catalog pages.

The gathered data is stored in structured CSV files for downstream analytics or modeling purposes.

**3. Model or Flowchart or Architecture**



**4.Algorithm**

1. START

2. Initialize required Python libraries:

- requests, pandas, time, BeautifulSoup, csv, etc.

3. Define a list of 5 Bengaluru location coordinates (latitude and longitude).

4. For each location in the list:

4.1 WEATHER DATA COLLECTION

a. Formulate the Open-Meteo weather API URL with current location.

b. Send HTTP GET request to the API.

c. Parse JSON response to extract:

- Hourly temperature

- Hourly humidity

d. Append the data to a weather DataFrame.

4.2 AIR QUALITY DATA COLLECTION

a. Formulate the Open-Meteo AQ API URL with current location.

b. Send HTTP GET request to the AQ API.

c. Parse JSON response to extract:

- PM10, PM2.5, and CO levels

d. Append the data to an air quality DataFrame.

5. After looping through all locations:

5.1 Save the weather DataFrame to `bengaluru\_weather.csv`.

5.2 Save the air quality DataFrame to `bengaluru\_air\_quality.csv`.

6. BOOKSTORE WEB SCRAPING

a. Define the base URL of the bookstore.

b. Initialize an empty list to store book data.

c. While next page exists:

i. Send HTTP GET request to the page.

ii. Parse HTML using BeautifulSoup.

iii. For each book on the page:

- Extract title, price, and rating.

- Append the data to the list.

iv. Identify link to the next page.

7. Save the scraped book data to `books\_data.csv`.

8. END

**5.GitHub Link:**