Data Report - Climate Confluence: Analyzing the Impact of CO2 Emissions on Global Temperature Trends

Kazi Anik Islam
Matriculation number: 23399803
Friedrich-Alexander-Universität Erlangen-Nürnberg

Introduction:

The project, "Climate Confluence: Analyzing the Impact of CO2 Emissions on Global Temperature Trends," examines how CO2 emissions influence global temperatures. Worldwide warming and the greenhouse effect are caused by CO2 emissions, which are mostly from burning fossil fuels and deforestation. This project uses datasets on CO2 emissions and global temperatures to explore this relationship, aiming to provide evidence for effective climate. By analyzing these trends The goal of the study is to enhance understanding of the critical link between CO2 emissions and temperature changes.

Question:

How do CO2 emissions correlate with global temperature trends?

Data Source:

DataSource1: CO2 and Greenhouse Gas Emissions

- Metadata URL: https://github.com/owid/co2-data/tree/master
- Data URL: https://nyc3.digitaloceanspaces.com/owid-public/data/co2/owid-co2-data.csv
- Data Type: CSV

This dataset contains data on CO2 emissions (annual, per capita, cumulative and consumption-based), other greenhouse gases, energy mix, and other relevant metrics.

DataSource2: Earth Surface Temperature Data

- Metadata URL: https://figshare.com/articles/dataset/temperature_csv/3171766/1
- Data URL: https://figshare.com/ndownloader/files/4938964
- Data Type: CSV

This dataset contains data from Kaggle, featuring 7 countries and 16 cities.

Description: I have chosen two datasets for my analysis: CO2 emission data from Our World in Data (available on GitHub) and temperature data from Figshare. The CO2 emission dataset provides comprehensive historical data on CO2 emissions by country, including related indicator such as emissions per year, which is vital for understanding the impact of industrial activities and policies on climate change. The temperature dataset offers extensive historical temperature records from various global locations, including monthly and annual readings, essential for analyzing trends and anomalies in global and regional temperature patterns. Together, these datasets allow for a detailed examination of the relationship between CO2 emissions and temperature changes, crucial for studying climate change.

Data Structure:

- Stored in CSV files.
- Semi structured data and not cleaned.

Data Quality:

- **Accuracy:** Sourced from reputable institutions, reflecting real-world data.
- **Completeness:** Both datasets have some null values. So, they lack completeness.
- **Consistency:** Standardized formats and units throughout.
- **Timeliness:** Up-to-date records, suitable for historical trends.
- **Relevancy:** Essential insights into CO2 emissions and temperature changes for climate research.

License: Both the CO2 emissions and temperature datasets are licensed under CC BY 4.0. This license allows for use, sharing, and adaptation of the data. I plan to follow the obligations of the CC BY 4.0 license by providing appropriate attribution to the original creators of the CO2 emissions and temperature datasets in all instances of use.

License link:

- Dataset 1
- Dataset 2

Data Pipeline:

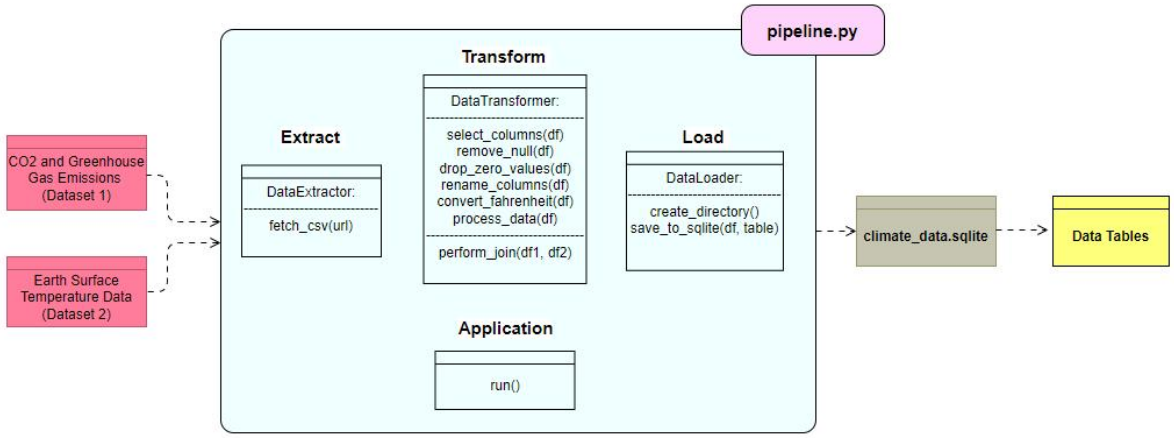

Figure: ETL Pipeline Diagram

This project follows ETL pipeline structure. The ETL (Extract, Transform, Load) pipeline diagram represents the flow of data through various stages of processing from extraction to transformation and finally loading into a storage system. Let's break down each component including transformation or cleaning steps of the ETL pipeline:

1. Extract

- **Component: DataExtractor**
  - **fetch_csv(url)**: Fetches CSV data from the specified URL and loads it into a Pandas DataFrame.

2. Transform

- **Component: DataTransformer**
  - **select_columns(df)**: Selects specific columns from the DataFrame.
  - **remove_null(df)**: Removes rows with null values.
  - **drop_zero_values(df)**: Drops rows where the specified column value is 0.0.
  - **rename_columns(df)**: Renames columns according to a specified mapping.

- **convert_fahrenheit(df)**: Converts temperatures from Fahrenheit to Celsius and drops the original Fahrenheit column.
- **perform_join(df1, df2, join_columns)**: Joins two DataFrames on specified columns.
- **process_data(df)**: Organize the execution of all transformation steps.

3. Load

- **Component: DataLoader:**
  - **create_directory()**: Ensures the target directory exists or creates it.
  - **save_to_sqlite(df, table_name)**:Saves the processed DataFrame to an SQLite database.

Application Execution:

- **Component: Application**
  - **run()**: Organize the entire ETL process from extraction through transformation to loading.

Visual Summary:

1. **Extract**: Fetch CSV data.
2. **Transform**: Clean, filter, and prepare data.
3. **Load**: Save the processed data into an SQLite database.

Problem encountered and solution:

- Problem: There were no data of Celsius.
- Solution: Added a method in the "DataTransformer" class to convert temperatures from Fahrenheit to Celsius and drop the original Fahrenheit column.

Error Handling: My pipeline includes checks for null values and zero values to prevent processing errors and if the program doesn't find saving directory, it creates or override if needed.

Result and Limitations:

The output data is a cleaned and transformed DataFrame containing:

- CO2 emissions per country per year.
- Average surface temperature in Celsius per country per year.
- Stored in an SQLite database for easy access and analysis.

Data Structure of the output:

- Stored in SQLite database.
- Cleaned and transformed DataFrame.

Data Quality of the output:

- **Accuracy:** Reflects real-world data, ensuring correctness.
- **Completeness:** Contains all necessary information, no missing values.
- **Consistency:** Uniform data format across all entries.
- **Timeliness:** Data is up-to-date.
- **Relevancy:** Aligned with user needs, providing the latest relevant information.

Data Format of the output: I chose SQLite as the output format for my pipeline because it is a lightweight, self-contained database that allows easy storage, retrieval, and querying of structured data, making it suitable for handling the transformed data efficiently.