Relation Between Climate Change and Immigration

1 - Question:

Does climate change have an affect on immigration in countries that are vulnerable to climate change?

In this project I want to examine two of the biggest problems that world is facing now and probably will be facing in the future.

2 – Data

For this project I have chosen 2 datasets. Both are available in Kaggle. While these datasets are not directly from authorities, trustworthy sources such as UNHCR, NASA's GISTEMP and NOAA's MLOST were used to create.

First dataset[1] is about global temperature. In this dataset 5 csv files are available which has different classification. For my project I chose the Global Land Temperature by Country. This dataset has a license of CC BY-NC-SA 4.0[2] which enables me to use in my project.

```
df_temperature_raw = pd.read_csv(filepath_or_buffer="../data/GlobalLandTemperaturesByCountry.csv")
df_temperature_raw.head()
```

	dt	AverageTemperature	AverageTemperatureUncertainty	Country
0	1743-11-01	4.384	2.294	Åland
1	1743-12-01	NaN	NaN	Åland
2	1744-01-01	NaN	NaN	Åland
3	1744-02-01	NaN	NaN	Åland
4	1744-03-01	NaN	NaN	Åland

Second dataset[3] shows the numbers of asylum applications and asylum decisions taken by governments. There are two csv files: "Asylum applications" and "Asylum decisions". I used application data. License of the dataset is CC0: Public Domain[4].

```
df_asylum_raw = pd.read_csv(filepath_or_buffer="../data/asylum-applications.csv")
df_asylum_raw.head()
```

	Year	Country of origin	Country of origin (ISO)	Country of asylum	Country of asylum (ISO)	applied
0	2006	Afghanistan	AFG	Australia	AUS	14
1	2006	Albania	ALB	Australia	AUS	21
2	2006	Algeria	DZA	Australia	AUS	5
3	2006	Egypt	EGY	Australia	AUS	38
4	2006	Bahrain	BHR	Australia	AUS	11

3 - Pipeline

3.1 - Steps of Pipeline

Pipeline is designed to automate the process of reaching datasets from Kaggle, performing necessary transformations and uploading cleaned data to an SQLite database. Four steps of the pipeline:

1-Download Datasets: Using the Kaggle API, datasets specified in the infos dictionary are downloaded and unzipped.

2-Check File Integrity: The pipeline verifies if the datasets have been downloaded correctly.

3-Transform Data: The downloaded datasets undergo specific transformations such as dropping unnecessary columns, extracting, and filtering date information, and limiting the data to a specific range of years.

4-Load into SQL: The cleaned and transformed data is loaded into the SQLite database for further analysis and use.

---

[1] https://www.kaggle.com/datasets/berkeleyearth/climate-change-earth-surface-temperature-data?select=GlobalLandTemperaturesByCountry.csv
[2] https://creativecommons.org/licenses/by-nc-sa/4.0/
[3] https://www.kaggle.com/datasets/patrasaurabh/global-asylum-data-2000-present?select=asylum-applications.csv
[4] https://creativecommons.org/publicdomain/zero/1.0/

3.2 - Technologies Used

Kaggle API: For downloading datasets directly from Kaggle.

Pandas: For data manipulation and transformation.

SQLAlchemy: For interacting with the SQLite database.

3.3 - Data Transformation

Dropping Unnecessary Columns:

Columns that are not required for the analysis are dropped to save space and reduce complexity. This is done using the ‘infos’ dictionary which specifies the columns to be dropped for each dataset. In temperature dataset 'AverageTemperatureUncertainty' column is dropped for the simplicity. In asylum dataset 'Country of origin (ISO)' and 'Country of asylum (ISO)' are dropped to prevent data redundancy.

Date Processing:

Conversion: The date column ('dt') is converted from string format to date format.

Year Extraction: A new column 'Year' is created to store the year extracted from the date.

Filtering: Data is filtered to include only the years between 2000 and 2012. This ensures the analysis is focused on a specific timeframe.

3.4 - Problems Encountered and Solutions

File Download Issues: Ensuring the files are correctly downloaded and unzipped. The pipeline includes a verification step to check the presence of the files in the directory. If a file is missing, it triggers a re-download.

Date Conversion: Ensuring the date conversion is done correctly and efficiently. The 'pd.to_datetime' method is used to handle date parsing and extraction robustly.

Data Integrity: Ensuring that data loaded into the database is clean and correctly formatted. The transformations applied ensure that only relevant data is included.

3.5 - Dealing with Error and Changing Input

File Existence Check: Before proceeding with transformations, the pipeline checks if the necessary files are present in the specified directory. If not, it attempts to download them again.

Transformation Steps: Each transformation step is designed to handle errors gracefully. For example, date conversion includes format specifications to handle different date formats.

Dynamic Handling: The use of a dictionary (infos) to store dataset-specific information makes the pipeline adaptable to new datasets with minimal changes. By updating the dictionary with new dataset information, the pipeline can be easily extended.

4 - Result and Limitations

4.1 - Result

The output data of the pipeline consists of two cleaned and transformed datasets stored in two tables of an SQLite database. These datasets are:

Asylum Applications Data: Contains information about asylum applications from various countries between 2000 and 2012. Name of the table is 'asylum_applications'.

Global Temperature Data: Contains historical temperature data by country for the same year interval. Name of the table is 'global_temperature'.

4.2 - Data Structure and Quality

Asylum Applications Data:

Table Name: asylum_applications

Columns:

Country of origin: The country from which asylum seekers originate.

Country of asylum: The country in which asylum is sought.

Year: The year of the application.

Additional relevant columns from the original dataset.

Quality:

The data is filtered to include only relevant years (2000-2012), and unnecessary ISO code columns are removed. The quality of the data is maintained by focusing on key metrics and ensuring consistency in the date formats.

Global Temperature Data:

Table Name: global_temperature

Columns:

Country: The country for which temperature data is recorded.

AverageTemperature: The average land temperature.

dt: The original date column.

Year: The extracted year from the date.

Additional relevant columns from the original dataset.

Quality:

The data quality is improved by removing uncertainty columns and focusing on precise temperature measurements. The date column is converted and extracted into a year column to facilitate time-series analysis.

4.3 - Data Format

The chosen output format for the pipeline is an SQLite database. This format is chosen for several reasons:

Structured Storage: SQLite databases provide a structured way to store data, ensuring easy access and efficient querying.

Portability: SQLite databases are self-contained, making them easy to share and deploy.

Scalability: While SQLite is not intended for very large-scale applications, it is more than sufficient for moderate-sized datasets like those used in this pipeline.

Usage: SQL is widely used in the real-world applications.

4.4 - Limitations

Data Accuracy:

The transformations applied assume the initial datasets from Kaggle are accurate and well-structured. Any issues in the source data could propagate through the pipeline. Conversion errors, especially in date formats, could lead to inaccuracies in the transformed data. Thorough testing and validation are required to mitigate this.

Scalability and Performance:

While SQLite is adequate for the current data volume, scaling to larger datasets or concurrent access scenarios would require migrating to more robust database systems like PostgreSQL or MySQL. Performance bottlenecks could arise from extensive transformations, particularly date conversions. Optimization and efficient querying techniques should be considered.

Potential Bias:

The datasets may inherently carry biases, such as underreporting in asylum applications or temperature data limited to specific geographical regions. These biases should be acknowledged and accounted for in any analytical work.

Updating Data:

The pipeline currently downloads and processes static datasets. Incorporating mechanisms to handle updated datasets or real-time data streams would improve the pipeline's utility.

Error Handling:

While basic error handling is implemented (e.g., re-downloading missing files), more robust error logging and handling mechanisms could enhance reliability. This includes catching and logging exceptions during data transformations and database operations.