Question

How have different regions around the world been affected by changes in surface temperature in terms of climate-related disasters?

Data Sources

I choose two datasets for this project-

1. Annual Surface Temperature Change
   Metadata URL:
   https://climatedata.imf.org/datasets/4063314923d74187be9596f10d034914/explore
   Data URL: https://opendata.arcgis.com/datasets/4063314923d74187be9596f10d034914_0.csv
   Data Type: CSV
   Description: This dataset shows annual estimates of mean surface temperature change measured with respect to a baseline climatology, corresponding to the period 1961-2022.
   Data Structure: Semi-structured Data
   Data Quality:

Dimentions	Check
Accuracy	✓
Completeness	✗
Consistancy	✓
Timeliness	✓
Relevancy	✓

   License: Custom License [Click here to see full details]

2. Climate-related Disasters Frequency
   Metadata URL:
   https://climatedata.imf.org/datasets/b13b69ee0dde43a99c811f592af4e821/explore
   Data URL: https://opendata.arcgis.com/datasets/b13b69ee0dde43a99c811f592af4e821_0.csv
   Data Type: CSV
   Description: This dataset shows number of climate related natural disasters between 1980-2022.
   Data Structure: Semi-structured Data
   Data Quality:

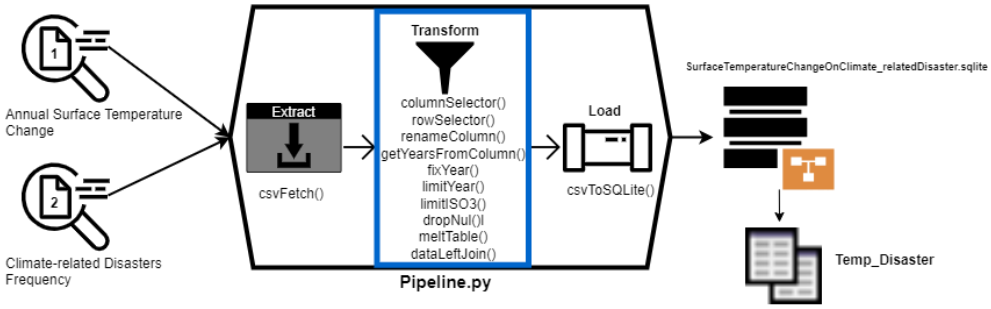
Dimentions	Check
Accuracy	✓
Completeness	✗
Consistancy	✓
Timeliness	✓
Relevancy	✓

   License: Custom License [Click here to see full details]

I choose these datasets for a comprehensive and focused study on the effects of temperature changes on climate-related disasters. These datasets provide the necessary information to analyze, and understand the intricate dynamics between climate change and its real-world impacts, making them ideal for your project.

Data Pipeline

Technology



This project follows ETL pipeline structure, which stands **Extract**, **Transform**, **Load**. The ETL pipeline structure is implemented using python (pandas, sqlite3 packages).The entry point of the project is ***pipeline.sh*** which runs ***pipeline.py*** file and results an SQLite file named–

"**SurfaceTemperatureChangeOnClimate_relatedDisaster.sqlite**".

Other than this, I have another python file name ***alerts.py*** which was used for colorful command-line messages.

Use case of ETL in this project:

1. Extract: Fetch data from source.
2. Transform: Modify data by filtrations and cleaning error. Joining data to create meaningful dataset.
3. Load: Storing the dataset.

Code mapping with ETL

The ETL Pipeline was implemented using python by following OOP principles. The name of the class was given "**Pipeline**". Pipeline class got necessary methods for ETL. Class Abstract & Methods breakdowns-

Class Abstract:

```python
class Pipeline():
    PipelineData : object
    url : str
    dropColumns : object
    selectedColumns : object

    def __init__(self, PipelineData = None, url = None, dropColumns = None, selectedColumns = None):
        # Code


    def csvFetch(self):
        # Code


    def columnSelector(self):
        # Code


    def rowSelector(self, row : str, value : str):
        # Code


    def getYearsFromColumn(self):
        # Code

    def renameColumn(self, renameColumns : object):
        # Code


    def fixYear(self):
        # Code


    def limitYear(self, fromYear, toYear):
        # Code

    def limitISO3(self, iso):
        # Code


    def dropNull(self):
       # Code


    def meltTable(self, keep : object, melt : object):
       # Code


    def dataLeftJoin(self, left : object, right : object, key : object, leftSufx : str, rightSufx : str):
        # Code


    def csvToSQLite(self, savingPath : str, sqliteFileName : str, sqliteTableName : str):
        # Code
```

Methods Breakdowns:

	Methods
E	csvFetch()
T	columnSelector() rowSelector() renameColumn() getYearsFromColumn() fixYear() limitYear() limitISO3() dropNul() meltTable() dataLeftJoin()
L	csvToSQLite()

Data Transformation & Cleaning

For data transformation I took some steps-

1. Select Columns: Choose/delete desired/unnecessary columns.
2. Select Rows: Limit rows based on condition. **[For Source 2]**
3. Melt Table: Both data sources contain year as a column. Eg.-

Data

	ObjectId	Country	ISO2	ISO3	...	F2019	F2020	F2021
0	1	Afghanistan, Islamic Rep. of	AF	AFG	...	0.910	0.498	1.327
1	2	Albania	AL	ALB	...	1.675	1.498	1.536
2	3	Algeria	DZ	DZA	...	1.115	1.926	2.330

[3 rows x 72 columns]

In this step, I melted the table and made the columns as row value. This was done dynamically. Eg.-

Data

	ISO3	Country	variable	value
0	AFG	Afghanistan, Islamic Rep. of	F1961	-0.113
1	ALB	Albania	F1961	0.627
2	DZA	Algeria	F1961	0.164

4. Rename Columns
5. Fixing Year Data

Data

	ISO3	Country	Year	Temperature
0	AFG	Afghanistan, Islamic Rep. of	1961	-0.113
1	ALB	Albania	1961	0.627
2	DZA	Algeria	1961	0.164

6. Joining Data
7. Dropping Nulls

Data

	ISO3	Country	Year	Temperature	Incident
11025	AFG	Afghanistan, Islamic Rep. of	2010	1.613	4.0
11026	ALB	Albania	2010	1.191	1.0
11030	AGO	Angola	2010	1.194	3.0

8. Limit by Year [Optional]
9. Limit by Country (ISO3) [Optional]

Challenges

1. Converting Columns and row data.

This problem was solved using panda's ***melt*** function.

The data was like-

ISO3	***F2010***	***F2011***	***F2013***	***F2020***
AFG	1	3	2	7

After melting, I made it like-

ISO3	***Year***	***Incidents***
AFG	2010	1
AFG	2011	3
AFG	2012	2
AFG	2020	7

After melting the data, I selected Year column and removed ***'F'*** from every rows.

Handling Errors & Change of Inputs

Errors were properly handled.

1. If source URL is faulty or unreachable then the execution will stop by showing proper messages.
2. If program doesn't find saving directory [if pipeline.py was executed from project folder], it will resolve the path automatically.
3. If source adds more year data, program automatically converts it to row value. Dynamic programming was implemented.

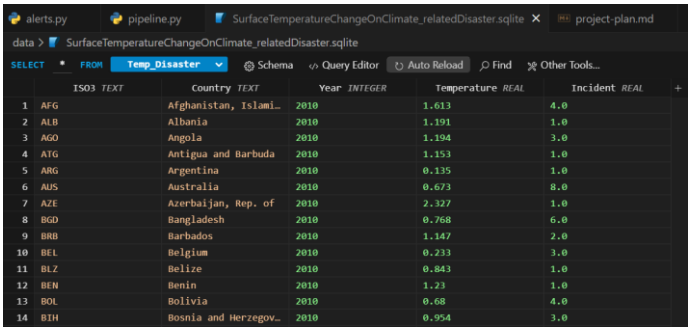
Results

Data Structure: Structured Data

Data Quality:

Dimentions	**Check**
Accuracy	✓
Completeness	✓
Consistancy	✓
Timeliness	✓
Relevancy	✓

Format: SQLite file



I saved the data in SQLite for-

a. ***Querying and Analysis:*** If the data is stored in SQLite, it can be easily queried using SQL. This allows for flexible data retrieval and analysis and make data-driven decisions.
b. ***Integration with Other Tools:*** It can be integrated with other data processing and analysis tools. For example, data visualization tools, machine learning frameworks, or business intelligence.