Question

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

Data Sources

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.
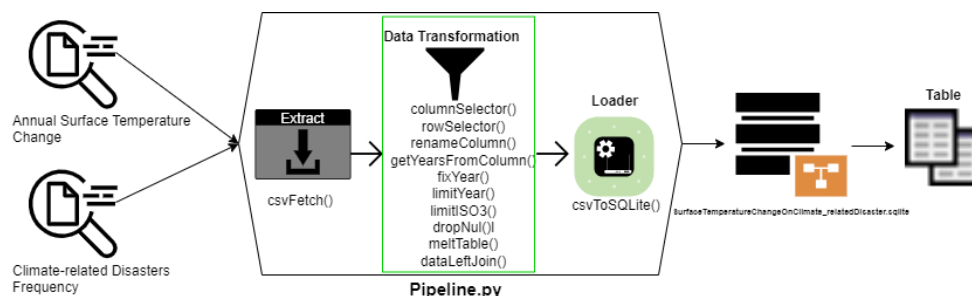   **Data Structure:** Semi-structured Data
   Data Quality:

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

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

Data Pipeline

Technology

Pipeline.py

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

"**SurfaceTemperatureChangeOnClimate_relatedDisaster.sqlite**".

Other then this, I have another python file name ***alerts.py*** *(Path: ./project/utils/**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:

```
1. class Pipeline():
2.     PipelineData : object
3.     url : str
4.     dropColumns : object
5.     selectedColumns : object
6.
7.     def __init__(self, PipelineData = None, url = None, dropColumns = None, selectedColumns = None):
8.         # Code
9.
10.
11.     def csvFetch(self):
12.         # Code
13.
14.
15.     def columnSelector(self):
16.         # Code
17.
18.
19.     def rowSelector(self, row : str, value : str):
20.         # Code
21.
22.
23.     def getYearsFromColumn(self):
24.         # Code
25.
26.     def renameColumn(self, renameColumns : object):
27.         # Code
28.
29.
30.     def fixYear(self):
31.         # Code
32.
33.
34.     def limitYear(self, fromYear, toYear):
35.         # Code
36.
37.     def limitISO3(self, iso):
38.         # Code
39.
40.
41.     def dropNull(self):
42.         # Code
43.
44.
45.     def meltTable(self, keep : object, melt : object):
46.         # Code
47.
48.
49.     def dataLeftJoin(self, left : object, right : object, key : object, leftSufx : str, rightSufx : str):
50.         # Code
51.
52.
53.     def csvToSQLite(self, savingPath : str, sqliteFileName : str, sqliteTableName : str):
54.         # 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  va
0  AFG  Afghanistan, Islamic Rep. of    F1961 -0.
1  ALB                      Albania    F1961  0.
2  DZA                      Algeria    F1961  0.
```

4. Rename Columns
5. Fixing Year Data

```
_____Data_____
   ISO3                      Country Year Temperat
0  AFG  Afghanistan, Islamic Rep. of 1961      -0.
1  ALB                      Albania 1961       0.
2  DZA                      Algeria 1961       0.
```

6. Joining Data
7. Dropping Nulls

```
_____Data_____
      ISO3                      Country Year Temperature Incid
11025  AFG  Afghanistan, Islamic Rep. of 2010       1.613
11026  ALB                      Albania 2010       1.191
11030  AGO                      Angola 2010       1.194
```

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 faulty 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 resolved 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-

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.

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.