BSD 2343 DATA WAREHOUSING

2022/2023 SEMESTER II

TITLE:

UNLOCKING CLIMATE CHANGE SOLUTIONS.

PREPARED FOR

DR Nor Azuana Ramli

STUDENT NAME	STUDENT ID
WONG ZI MING	SD21037
KEN FONG KA KIN	SD21040
NURUL ATHIRAH BINTI RAMLI	SD21015
QISTINA ZAWANI BINTI RIDZUAN	SD21030
DEVINAA A/P MUTHURAMAN	SD20014

TABLE OF CONTENTS

1.0 BACKGROUND

1.1 Description of the project

Climate change is safe to be one of the biggest challenges for human beings. The increase of greenhouse gases in the atmosphere and consequent acceleration of climate change are both caused by human activity, particularly the burning of fossil fuels. The repercussions of climate change are already being felt, and the global damage they cause is almost certainly going to be immense. Each country, based on its economic foundation, has varied alternatives to combat adverse effects since global impacts vary significantly and will result in highly different national sensitivity to climate impacts.

Unquestionably, climate change will have an impact on the entire world and requires global cooperation. The health, the food supply, and the water supply for humans are all at risk due to changes in wind patterns, average temperature, precipitation amounts, and the frequency of extreme weather events. The loss of biological variety and the extinction of species that pose a threat to the majority of the world's regions are directly related to those threats. Numerous communities' living conditions will alter as a result of the effects of climate change, which may cause socioeconomic and political instability.

This project aims to analyse in order to estimate local and regional global warming, which climate change takes into a variety of future carbon dioxide (CO_2) emission scenarios. In addition to being a natural component of the atmosphere and a crucial greenhouse gas, CO_2 is a significant part of the global carbon cycle. Humans primarily affect CO_2 emission levels and so contribute to global warming through the burning of fossil fuels.

1.2 Problem to be solved

The problem we seek to solve through our project, "Unlocking Climate Action Solutions: Addressing Inequality and Building Resilience," is the critical challenge of tackling climate change while ensuring equitable access to climate action opportunities for all individuals, particularly those who are marginalised and vulnerable. Climate change has a varied impact on many communities, worsening disparities in society and preventing progress towards sustainable development. One aspect on which we should concentrate is identifying regions with the highest lack of adaptive capacity. Understanding which regions are least equipped to deal with the impacts of climate change will allow us to develop targeted interventions and support systems to enhance resilience. We may work towards more equitable and effective climate adaptation measures by addressing the issues that contribute to a lack of adaptive ability, such as low resources or inadequate infrastructure.

Another critical issue we will explore is the average nitrogen oxide levels in Asia and identify the country with the highest average. Nitrogen oxide emissions have significant environmental and health consequences, and understanding the regional variations can help direct targeted efforts to reduce pollution and improve air quality. By identifying the countries with the highest nitrogen oxide levels, we can design tailored strategies to reduce emissions while protecting the population's health and well-being.

Furthermore, we will investigate the energy consumption patterns in several African regions to find the region with the highest energy consumption access. Access to reliable and affordable energy is crucial for economic development, education, healthcare, and overall well-being. We can build policies to improve infrastructure, promote renewable energy sources, and ensure equitable energy access for all populations by identifying places with inadequate access to electricity. This will contribute to sustainable development goals and reduce energy poverty in marginalized areas.

Moreover, we will examine the regions that have the most significant impact on CO_2 emissions in a year. CO_2 emissions are a primary driver of climate change, and understanding the regional variations in emissions can guide efforts to reduce carbon footprints and transition to low-carbon economies. By identifying regions with the highest emissions, we can develop targeted strategies to promote renewable energy, energy efficiency, and sustainable practices, thereby mitigating climate change and fostering a more sustainable future for all.

In short, our project aims to address the challenge of climate change while ensuring equitable access to climate action opportunities. By focusing on issues such as adaptive capacity, nitrogen oxide levels, energy consumption access, and CO_2 emissions, we can develop targeted strategies that enhance sustainability, resilience, and inclusion.

1.3 Objectives

The objective scope for this project is:

1) To present the region with the highest of lack adaptive capacity.
2) To explore and evaluate the country that has the highest average of nitrogen oxide in Asia.
3) To analyze regions in Africa that have access to the highest energy consumption.
4) To find out which region that affects CO_2 emissions the most in a year.

1.4 Data Schema

A data schema is like a blueprint or map that describes how data is organised and structured in a database. It defines the different tables or categories of information, the columns or attributes within those tables, and the relationships between them. Think of it as a way to establish rules and guidelines for storing and retrieving data. The schema specifies the types of data that can be stored, the constraints or rules that apply to the data, and how different tables are connected. It's an essential tool for ensuring data consistency, integrity, and efficiency in working with the database.

```
data_types = data_country.dtypes
null_counts = data_country.isnull().sum()
schema = pd.DataFrame({'Column': data_types.index, 'Data Type': data_types.values, 'Null Count': null_counts.values})
schema
```

	Column	Data Type	Null Count
0	country_name	object	0
1	country_code	object	0
2	region	object	0
3	sub_region	object	0
4	region_code	int32	0
5	sub_region_code	int32	0

Figure 1.3.1 Data schema of Country Table

Based on figure 1.3.1, shows that the country table has 6 columns consisting of country_name, country_code, region, sub-region, region_code, and sub_region_code. We have 2 data types which are text and integer.

```
data_types = data_air_pollution.dtypes
null_counts = data_air_pollution.isnull().sum()
schema = pd.DataFrame({'Column': data_types.index, 'Data Type': data_types.values, 'Null Count': null_counts.values})
schema
```

	Column	Data Type	Null Count
0	country_name	object	0
1	year	int64	0
2	Nitrogen_Oxide	float64	0
3	Sulphur_Dioxide	float64	0
4	Carbon_Monoxide	float64	0
5	Organic_Carbon	float64	0
6	nmvoc	float64	0
7	Black_Carbon	float64	0
8	ammonia	float64	0

Figure 1.3.2 Data Schema of Air Pollution Table

Figure 1.3.2 shows the table of air pollution that consists of 9 columns such as country_name, year, nitrogen_oxide, sulphur_dioxide, carbon_monoxide, organic_carbon, nmov, black_carbon, and ammonia. The table has 3 types of datatypes like text, integer and numeric.

```
data_types = data_co2_emission.dtypes
null_counts = data_co2_emission.isnull().sum()
schema = pd.DataFrame({'Column': data_types.index, 'Data Type': data_types.values, 'Null Count': null_counts.values})
schema
```

	Column	Data Type	Null Count
0	country_name	object	0
1	country_code	object	0
2	year	int64	0
3	Annual_CO2_emissions(tonnes)	float64	0

Figure 1.3.3 Data Schema of CO2 Emission Table

CO2 Emission Table has 4 columns with 3 types of data types: text, integer and numeric as shown above. For column country_name, country_code are text data types, while the year is integer and annual_co2_emissions(tonnes) is numeric.

```
data_types = data_per_capita_energy_use.dtypes
null_counts = data_per_capita_energy_use.isnull().sum()
schema = pd.DataFrame({'Column': data_types.index, 'Data Type': data_types.values, 'Null Count': null_counts.values})
schema
```

	Column	Data Type	Null Count
0	country_name	object	0
1	country_code	object	0
2	year	int64	0
3	Energy_consumption_per_capita(kWh)	float64	0

Figure 1.3.4 Data Schema of Per Capita Energy Use Table

Figure 1.3.4 is Per Capita Energy Use Table has 4 columns with country_name, country_code, year and energy_consumption_per_capita(kWh). This table consists of 3 data types like text, integer and numeric.

```
data_types = data_population_access_electricity.dtypes
null_counts = data_population_access_electricity.isnull().sum()
schema = pd.DataFrame({'Column': data_types.index, 'Data Type': data_types.values, 'Null Count': null_counts.values})
schema
```

	Column	Data Type	Null Count
0	country_name	object	0
1	country_code	object	0
2	year	int64	0
3	Access_to_electricity(% of population)	float64	0

Figure 1.3.5 Data Schema of Population Access Electricity Table

Based on the figure above, shows that the Population Access Electricity Table has 4 columns and 3 different data types such as text, integer and numeric. The table has attributes like country_name, country_code year and access to electricity(% of population).

```
data_types = data_world_risk_index.dtypes
null_counts = data_world_risk_index.isnull().sum()
schema = pd.DataFrame({'Column': data_types.index, 'Data Type': data_types.values, 'Null Count': null_counts.values})
schema
```

	Column	Data Type	Null Count
0	country_name	object	0
1	wri	float64	0
2	exposure	float64	0
3	vulnerability	float64	0
4	susceptibility	float64	0
5	Lack_of_Coping_Capabilities	float64	0
6	Lack_of_Adaptive_Capacities	float64	0
7	year	int64	0
8	Exposure_Category	object	0
9	WRI_Category	object	0
10	Vulnerability_Category	object	0
11	Susceptibility_Category	object	0

Figure 1.3.6 Data Schema of World Risk Index Table

We can see that in Figure 1.3.6 Data Schema of the World Risk Index Table has 12 columns with 3 differences in data types like text, numeric and integer. These table attributes are country_name, wri, exposure, vulnerability, susceptibility, lack_of_coping_capabilities, lack_of_adaptive_capabilities, year, exposure_category, wri_category, vulnerability_category, and susceptibility_category.

2.0 ARCHITECTURE AND ETL PIPELINE

2.1 Pipeline Structure

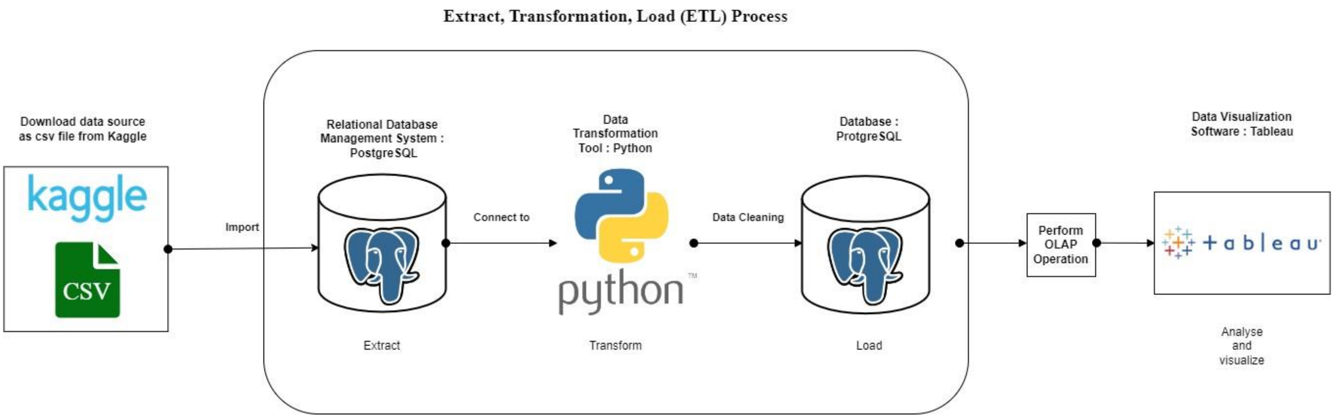


Figure 2.1 Pipeline Structure

Based on Figure 2.1, the climate dataset was collected from Kaggle.com. With its active community, diversity of contests, large datasets and collaborative features Kaggle became a highly important platform. The data set consists of 6 tables. The approach that we utilise is the Kimball approach, also referred to as the "dimensional modelling approach," this emphasises the usage of a data warehouse design built around the idea of a "fact table" and related "dimension tables."

Before we go through the process flow of ETL, the ETL process is a series of cycles and asynchronous processes, which run regularly to update target destinations with the latest data.

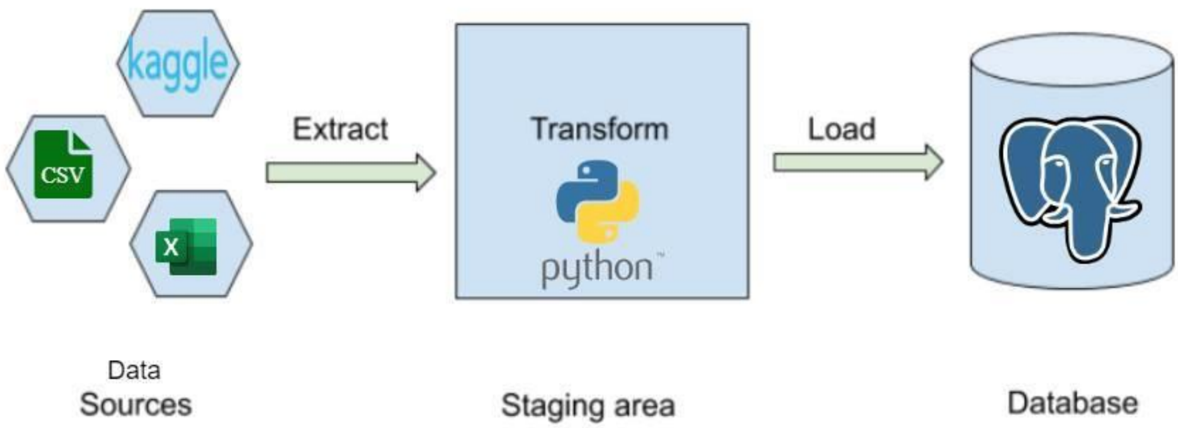
The first step in the ETL process is Extraction (E). Data sources shall be identified and data will be collected at this stage. Database, file, API or outside systems may be part of these sources. The methods for data extraction can differ according to the source system, and applications such as SQL queries, API calls or imports of files may be used.

The next stage is Transformation (T). In order to ensure compatibility among different sources, it is often necessary to transform data when they are extracted. This involves various activities such as data cleansing, data integration, and data enrichment. Data cleansing is used to remove errors, duplicates and inconsistencies in data that have been collected. Combining data from different sources and integrating them into a single format is the focus of data integration. It may be to resolve data structure differences, merge datasets or coordinate data types and conventions. By adding additional information or performing calculations based on existing data, data enrichment involves enhancing the data. Data will be prepared for further processing and analysis at the transformation stage.

After the transformation stage, we move on to the Loading (L) stage. This phase involves transmitting the transformation data to a target destination, e.g. warehouses, databases or analytical platforms. The transformed data is normally kept in a staging area or temporary storage before loading into the target destination. Staging enables data validation, auditing and further processing where necessary. Once the data is confirmed and packaged, it will be mapped to a structure and schema of its destination. This mapping will ensure that the new data is compatible with the objective destination's requirements. The transformed and verified data will then be loaded into the target destination through a variety of loading techniques.

Last but not least, architecture will be represented using OLAP Operation. The data will be visualized and analyzed by using the software which is Tableau. Tableau is a powerful data visualisation and business intelligence software tool, that helps people to see and understand their data. With Tableau, we will be able to connect with a variety of data types like spreadsheets, databases or cloud services and interactively display them without the need for complicated programming or scripting. Drag and drop is an advanced feature in the software that allows users to select fields of data, drag them onto a canvas where they can immediately be visually represented by charts, graphs, maps or similar graphic elements.

2.2 ETL Pipeline



The 6 tables of the dataset are downloaded from Kaggle that are called third-party data. This dataset is composed of 6 tables which are air pollution, CO2 emission, continents, per capita energy used, share of the population with access to electricity and world risk index. Those tables are extracted by using Python language during data cleaning and data integrating process using the software which is Jupyter Notebook. Then, the cleaned data will transform into a new CSV file before being loaded into the PostgreSQL database for the next step.

2.3 Description of the process flow

2.3.1 Step at PostgreSQL (PgAdmin4)

Refer to Appendix 1

2.3.2 Step to extract and transform (Python)

Refer to Appendix 2

2.3.3 Step to load data into database (Postgre SQL- PgAdmin4)

Refer to Appendix 3

3.1 DATABASE

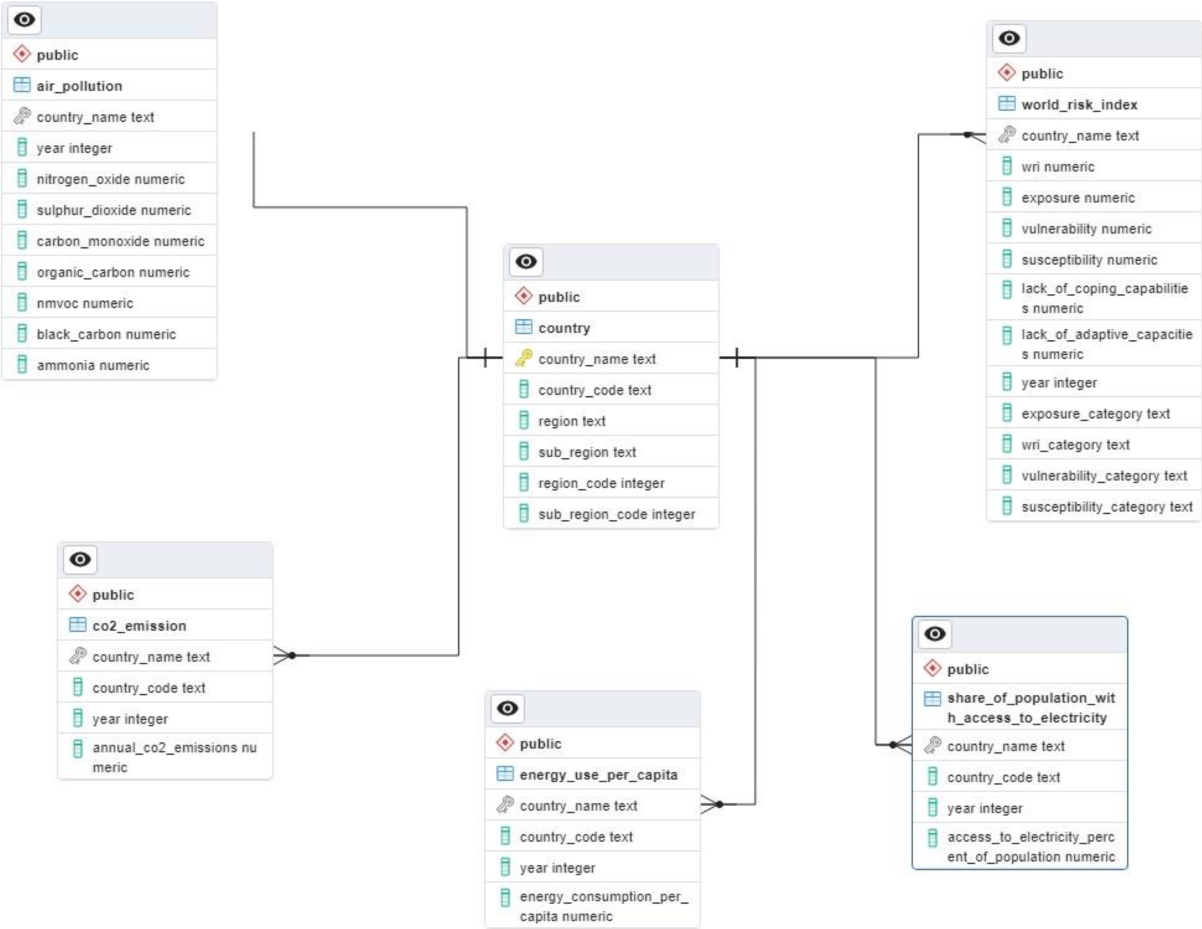


Figure 3.1 Data Schema obtained from PgAdmin

The data relationships in the provided dataset can be represented using a relational model. As shown in Figure 3.1, the relationships between the data are as follows: a country can have multiple energy use per capita, air pollution, CO2 emission, world risk index, and share population with access to electricity data records. Therefore, there is a one-to-many relationship between the Country table and each of the other tables.

To design a data warehouse schema, a star schema approach can be employed. The central fact table would be the Country Fact table, containing the key measurements and metrics related to energy use per capita, air pollution, CO2 emissions, world risk index, and share population with access to electricity. This fact table would have foreign key references to the corresponding dimension tables.

Fact Table:

1) Country Fact:

   country_name (Primary Key), country_code, region, sub-region, region_code, sub_region_code

Dimension Tables:

1) Energy Dimension:

   country_name(Foreign Key), country_code, year and energy_consumption_per_capita(kWh)(Primary Key)

2) Air Pollution Dimension:

   country_name(Foreign Key),year, nitrogen_oxide, sulphur_dioxide, carbon_monoxide, organic_carbon, nmov, black_carbon, and ammonia.

3) CO2 Emission Dimension:

   country_name(Foreign Key), country_code are text data types, while year is integer and annual_co2_emissions(tonnes)

4) Risk Dimension:

   country_name(Foreign Key), wri, exposure, vulnerability, susceptibility, lack_of_coping_capabilities, lack_of_adaptive_capabilities, year, exposure_category, wri_category, vulnerability_category, and susceptibility_category.

5) Access Dimension:

   country_name(Foreign Key), country_code, year and access to electricity(% of population).

The Country Fact table holds information about the country, such as the country name. The Energy Dimension table includes data on per capita energy usage. The Air Pollution Dimension table contains air pollution level records. The CO2 Emission Dimension table stores CO2 emission data. The Risk Dimension table holds the world risk index values. The Access Dimension table includes information about the share of the population with access to electricity.

4.1 RESULTS AND DATA ANALYSIS

4.2 OLAP OPERATIONS

4.1.1 2D

```
SELECT country.region, country.country_name,
SUM(co2_emission.annual_co2_emissions) AS annual_co2_emissions
FROM country
INNER JOIN co2_emission ON country.country_name =
co2_emission.country_name
GROUP BY (country.region, country.country_name)
ORDER BY country.region, country.country_name;
```

	region text	country_name [PK] text	annual_co2_emissions numeric
1	Africa	Algeria	4107869896.13
2	Africa	Angola	623762311.00
3	Africa	Benin	98792203.59
4	Africa	Botswana	131552447.39
5	Africa	Burkina Faso	50218080.94
6	Africa	Burundi	10728249.80
7	Africa	Cameroon	194766283.60
8	Africa	Central African Republic	11448062.13
9	Africa	Chad	14410060.55
10	Africa	Comoros	4221082.02
11	Africa	Djibouti	19174274.46
12	Africa	Egypt	5561231355.77
13	Africa	Equatorial Guinea	92146727.45
14	Africa	Eritrea	16336482.58
15	Africa	Ethiopia	207190474.39
16	Africa	Gabon	245703654.03
17	Africa	Gambia	12030970.74
18	Africa	Ghana	316321491.28
19	Africa	Guinea	77269714.08
20	Africa	Kenya	410113224.27
21	Africa	Lesotho	56424722.31

Figure 4.1.1 Output for 2D

The output for 2D operation is shown in Figure 4.1.1. This operation consists of three tables which are region, country_name, and annual_co2_emissions. 2D analysis method is used to analyze annual co2 emissions in each region of Africa.

4.1.2 Roll Up

```
SELECT CASE WHEN x.region IS null
                              THEN 'TOTAL' ELSE x.region END,
CASE WHEN x.country_name IS null
                              THEN 'TOTAL' ELSE x.country_name END,
total_lack_of_adaptive_capacities
FROM(SELECT country.region, country.country_name,
SUM(lack_of_adaptive_capacities) AS total_lack_of_adaptive_capacities
FROM country
INNER JOIN world_risk_index ON country.country_name =
world_risk_index.country_name
GROUP BY ROLLUP (country.region, country.country_name)
ORDER BY country.region, country.country_name
          )AS x;
```

	region text	country_name text	total_lack_of_adaptive_capacities numeric
1	Africa	Algeria	82.91
2	Africa	Angola	619.47
3	Africa	Benin	693.51
4	Africa	Botswana	442.04
5	Africa	Burkina Faso	687.09
6	Africa	Burundi	631.76
7	Africa	Cameroon	108.72
8	Africa	Central African Republic	135.12
9	Africa	Chad	137.50
10	Africa	Comoros	116.41
11	Africa	Congo	100.39
12	Africa	Djibouti	130.91
13	Africa	Egypt	93.08
14	Africa	Equatorial Guinea	101.37
15	Africa	Eritrea	730.85
16	Africa	Eswatini	48.98
17	Africa	Ethiopia	122.47
18	Africa	Gabon	94.26
19	Africa	Gambia	659.95
20	Africa	Ghana	593.92
21	Africa	Guinea	715.55

Figure 4.1.2 Output Roll Up

The output for roll up is shown in Figure 4.1.2. This operation consists of 3 tables which are region, country_name, and total_lack_of_adaptive_capacities. Roll-up is used to analyze the total lack of adaptive capacity based on the regions.

4.1.3 Slicing

SELECT country.country_name, ROUND(AVG(air_pollution.nitrogen_oxide),4) AS average_nitrogen_oxide
FROM country
INNER JOIN air_pollution ON country.country_name = air_pollution.country_name
WHERE country.region LIKE 'Asia'
GROUP BY country.country_name
ORDER BY average_nitrogen_oxide DESC;

	country_name [PK] text	average_nitrogen_oxide numeric
1	China	3403061.9407
2	India	1203154.9766
3	Japan	1044878.4360
4	Indonesia	413355.7668
5	Iran	289408.2992
6	Saudi Arabia	260654.6874
7	Kazakhstan	242938.0063
8	South Korea	200576.5902
9	Pakistan	198015.3745
10	Turkey	179062.3307
11	Thailand	159343.9277
12	Iraq	108711.0636
13	Taiwan	101440.9399
14	Uzbekistan	101157.9161
15	Malaysia	99041.2884
16	Vietnam	93303.8281
17	Philippines	90070.4041
18	Bangladesh	80638.0499
19	Afghanistan	74624.2090
20	United Arab Emirates	64800.8701
21	Turkmenistan	64026.5479
22	Israel	51504.5912

Figure 4.1.3 Output Slicing

The output for slicing operation is shown on Figure 4.1.3. This operation consists of 2 tables which are country_name and average_nitrogen_oxide. Slicing is used to analyze the average of nitrogen oxide in a selected country.

4.1.4 Dicing

```
SELECT country_name, average_energy_use, average_percent_of_population
FROM (
SELECT country.country_name,
ROUND(AVG(energy_use_per_capita.energy_consumption_per_capita),4)
AS average_energy_use,
ROUND(AVG(share_of_population_with_access_to_electricity.access_to_electricity
_percent_of_population),4)
AS average_percent_of_population
FROM country
INNER JOIN energy_use_per_capita ON country.country_name =
energy_use_per_capita.country_name
INNER JOIN share_of_population_with_access_to_electricity
ON country.country_name =
share_of_population_with_access_to_electricity.country_name
WHERE country.region LIKE 'Africa'
GROUP BY (country.region, country.country_name)
) subquery
WHERE average_percent_of_population <= 50
ORDER BY country_name;
```

	country_name [PK] text	average_energy_use numeric	average_percent_of_population numeric
1	Angola	2416.2116	25.9632
2	Benin	1048.6418	24.3657
3	Botswana	7063.3521	33.3823
4	Burkina Faso	394.2694	10.8831
5	Burundi	204.5948	3.9617
6	Cameroon	1690.7608	45.1169
7	Central African Republic	440.1424	6.9241
8	Chad	137.3672	3.8949
9	Comoros	728.0747	46.6068
10	Congo	2595.6147	26.9278
11	Eritrea	1181.9185	32.2769
12	Ethiopia	369.7143	14.6646
13	Gambia	924.2453	32.0071
14	Guinea	909.9871	19.2200
15	Kenya	1493.1508	20.1410

Figure 4.1.4 Output Dicing

This output for the dicing operation is shown in Figure 4.1.4. This operation consists of 3 tables which are country_name, average_energy_use, and average_percent_of_population. Dicing is used for analyzing average energy use at the same time analyzing average percent of population.

4.2 Data Visualization

4.2.1 2D Visualization

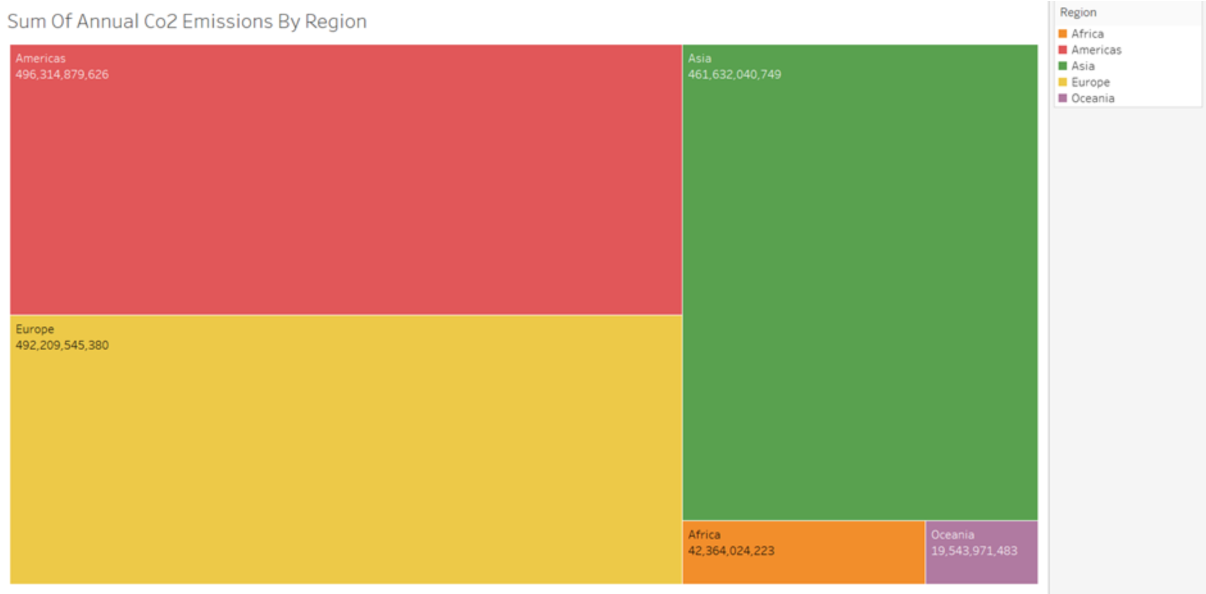


Figure 4.2.1 2D Visualization

Based on the Figure 4.2.1, it shows that the highest region that affects CO2 emissions the most in a year is America with 496,314,879,626 while the lowest region that affects CO2 emissions the most in a year is Oceania with 19,543,971,483. This indicates that the Americas had increased their environmental temperatures, extending their growing season and increased humidity.

4.2.2 Roll Up Visualization

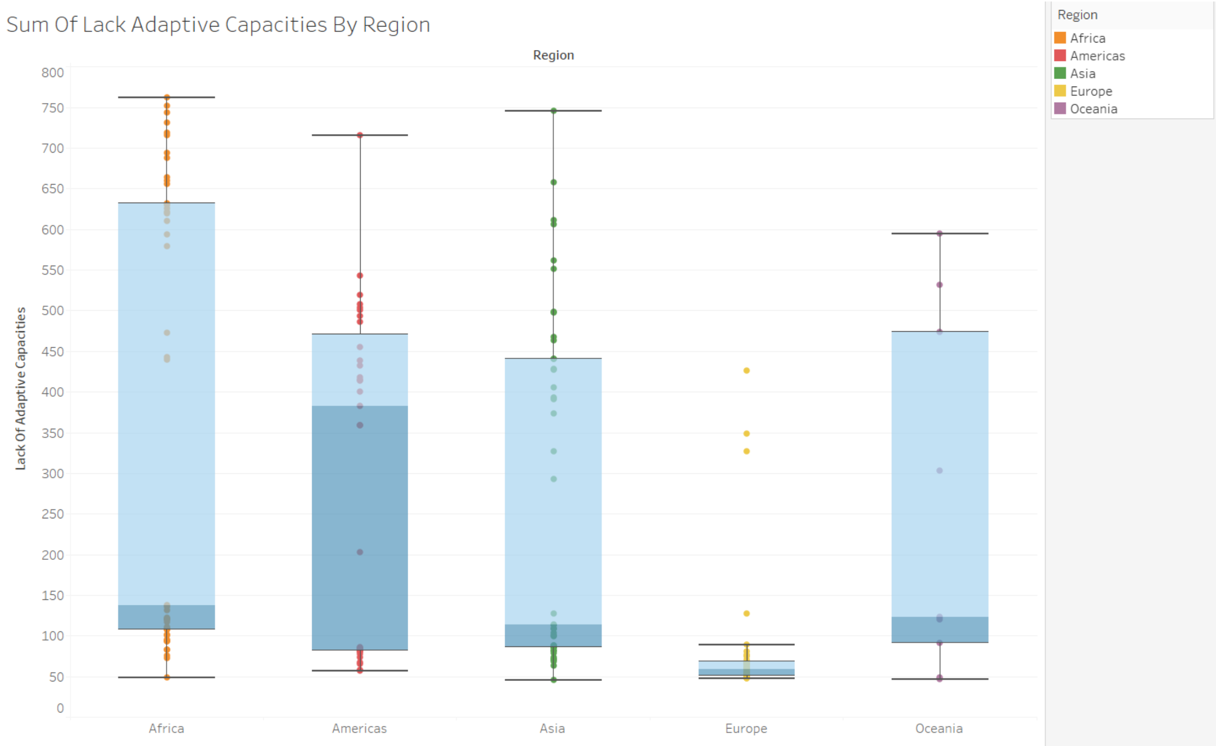


Figure 4.2.2 Roll Up Visualization

According to the Figure 4.2.1. There is no outlier for all the regions except Europe. The median of each boxplot region is ordered increasingly as follows: Europe (59.3), Asia (113.8), Oceania (123.0), Africa (137.5) and Americas (383.1). Africa has the longest box indicating highest variability among the regions while Europe has the shortest box showing the lowest variability. Boxplot of Africa, Asia, Europe and Oceania are positively-skewed while Americas is negatively-skewed.

4.2.3 Slicing Visualization

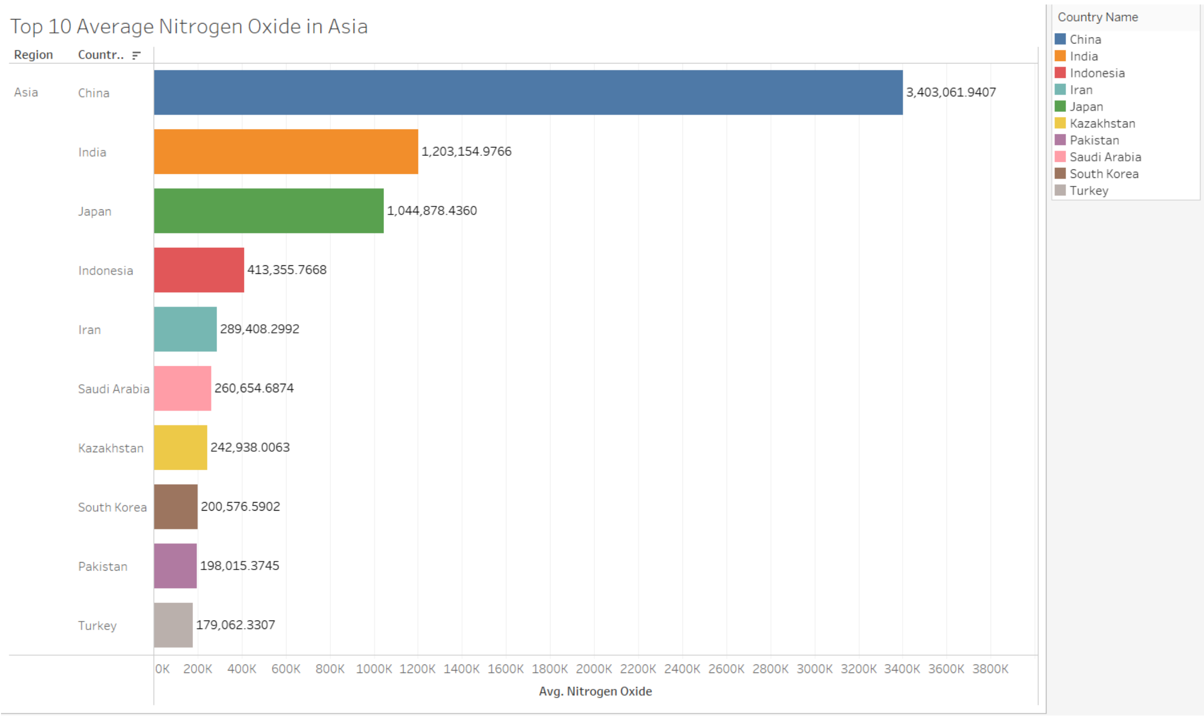


Figure 4.2.3 Slicing Visualization

Figure 4.2.3 shows the top 10 countries in Asia average emissions of nitrogen oxide in descending order: Turkey, Pakistan, South Korea, Kazakhstan, Saudi Arabia, Iran, Indonesia, Japan, India and China. The highest average emissions of nitrogen oxide in Asia is China with 3403061.9407 while the lowest average emissions of nitrogen oxide in Asia is Turkey with 179062.3307. The emissions of nitrogen oxide in China are 3 times that of India and 19 times that of Turkey.

4.2.4 Dicing Visualization

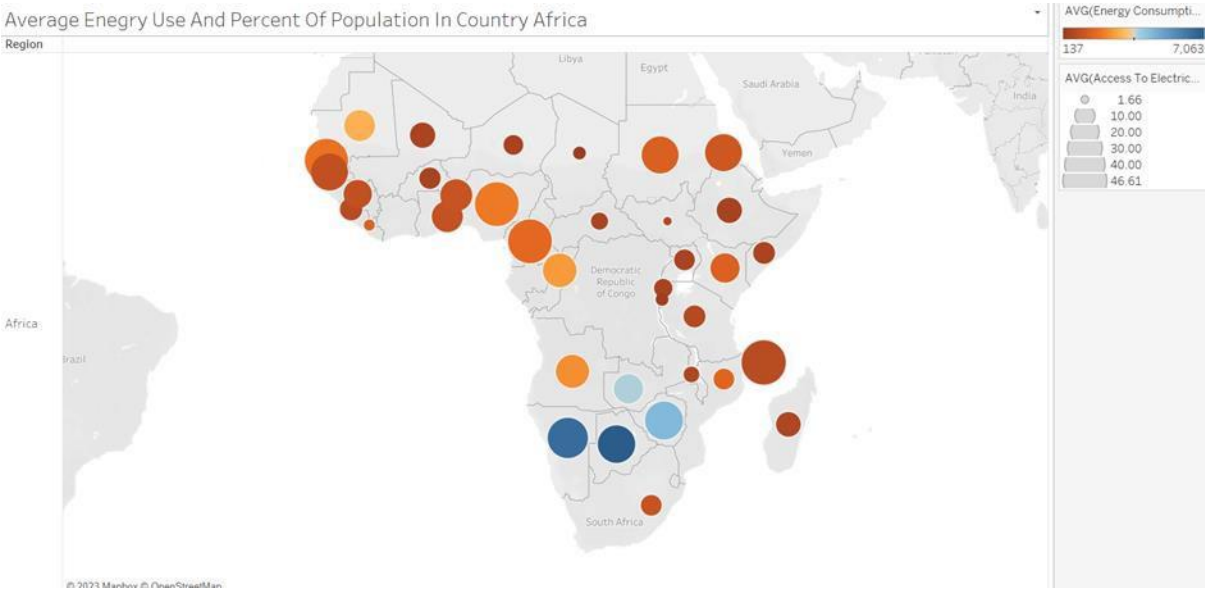


Figure 4.2.4 Dicing Visualization

Figure 4.2.4 shows that the highest average of access to electricity to percent of population is Comoros with 46.61 while the lowest average of access to electricity to percent of population is South Sudan with 1.66. Furthermore, the highest average of energy consumption per capita is 7063 in Botswana while the lowest average of energy consumption per capita is 137 in Chad.

5.1 CONCLUSION

In conclusion, this project focused on analyzing and visualizing data related to sustainable development goals, specifically in the areas of air pollution, CO_2 emissions, energy consumption, access to electricity, and the world risk index. The project utilized a dataset consisting of 6 tables obtained from Kaggle, and the data was processed using Python, Jupyter Notebook, PostgreSQL and Tableau. The project successfully implemented an ETL pipeline to extract, clean, integrate, and transform the data, before loading it into a PostgreSQL database. A star schema approach was employed to design the data warehouse schema, with a central Country Fact table and several dimension tables. Moreover, various OLAP operations were performed on the data, including 2D analysis, Roll-Up analysis, Slicing analysis, and Dicing analysis. These operations allowed for in-depth analysis of the data and provided insights into several tables of the database. Data visualizations were generated in Tableau based on the analysis results, allowing for a clear and intuitive representation of the data. The visualizations provided a better understanding of the patterns, trends, and variations in the data, contributing to the overall analysis and insights.

For example, we have discovered the answers to the objectives of this project. From all the visualisation created, it is clear that America has the highest lack of adaptive capacity with a sum of 383.1. The country with the highest average of nitrogen oxide in Asia is China with 3,403,061.9407. Comoros has access to the highest energy consumption in Africa with percent of population 46.61. America is the region that, out of all the regions, emits the most CO_2 each year, with a total of 496,314,879,626.

However, several challenges were encountered while we worked to complete the assignment. Since there are many related datasets available online, it took us a long time to find one that was appropriate for this project with Goal 13 Climate Action in the Sustainable Development Goals (SDGs). Furthermore, the cleaning and integration of the dataset required careful consideration and effort as inconsistent data formats, missing values, and data discrepancies needed to be addressed to ensure accurate analysis. In addition, the implementation of the ETL pipeline and the design of the data warehouse schema required a good understanding of the data and the underlying concepts. Moreover, the analysis and visualization of the data required the selection of appropriate techniques and tools. Choosing the right OLAP operations, understanding the results, and creating meaningful and attractive visualizations required a combination of analytical and technical skills.

In summary, this project successfully analyzed and visualized data related to sustainable development goals using an ETL pipeline, PostgreSQL database, various OLAP operations and Tableau visualization software. The project provides insights into the areas of air pollution, CO2 emissions, energy consumption, access to electricity, and world risk index, contributing to the understanding and promotion of sustainable development. Despite the challenges faced, the project demonstrates the value of data-driven analysis in addressing and working towards Sustainable Development Goals.

REFERENCES

CO2_GHG_emissions-data. (2020, September 14). Kaggle. https://www.kaggle.com/datasets/yoannboyere/co2-ghg-emissionsdata

World Disaster Risk Dataset. (n.d.). Www.kaggle.com. Retrieved June 19, 2023, from https://www.kaggle.com/datasets/tr1gg3rtrash/global-disaster-risk-index-time-series-dataset?resource=download

Emmision of Air Pollutants. (n.d.). Www.kaggle.com. Retrieved June 19, 2023, from https://www.kaggle.com/datasets/elmoallistair/emmision-of-air-pollutants?resource=download&select=air-pollution.csv

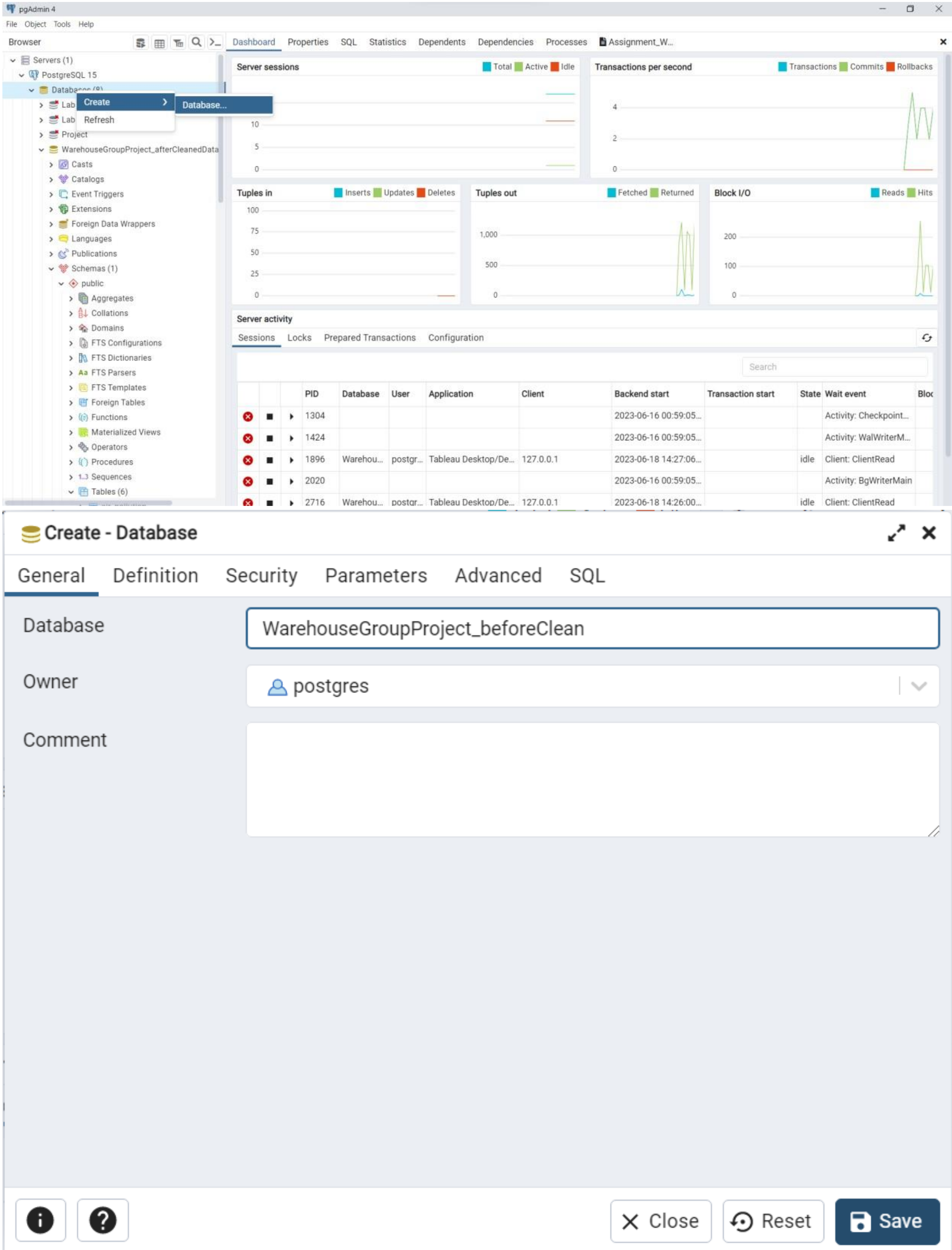
KPIs for measuring Climate Action and Inequality. (n.d.). Kaggle.com. Retrieved June 19, 2023, from https://www.kaggle.com/code/mannmann2/kpis-for-measuring-climate-action-and-inequality

World Bank Climate Change Knowledge Portal. (n.d.). Climateknowledgeportal.worldbank.org. https://climateknowledgeportal.worldbank.org/overview#:~:text=Climate%20change%20is%20the%20significant

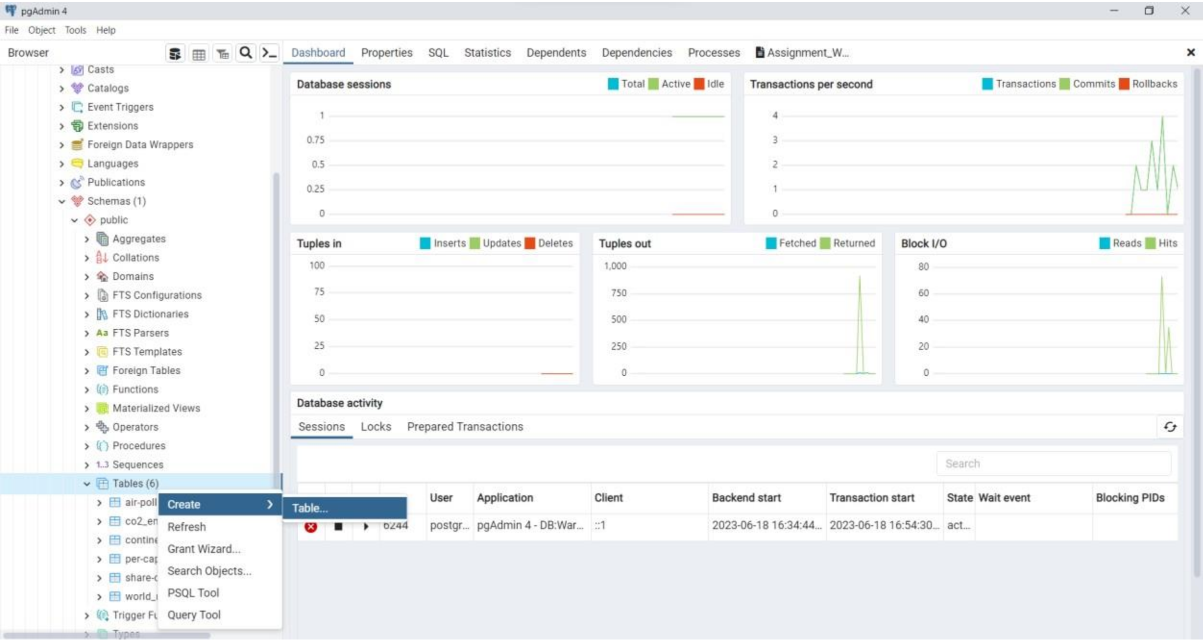
Google Cloud. (n.d.). What Is A Relational Database. Google Cloud. https://cloud.google.com/learn/what-is-a-relational-database

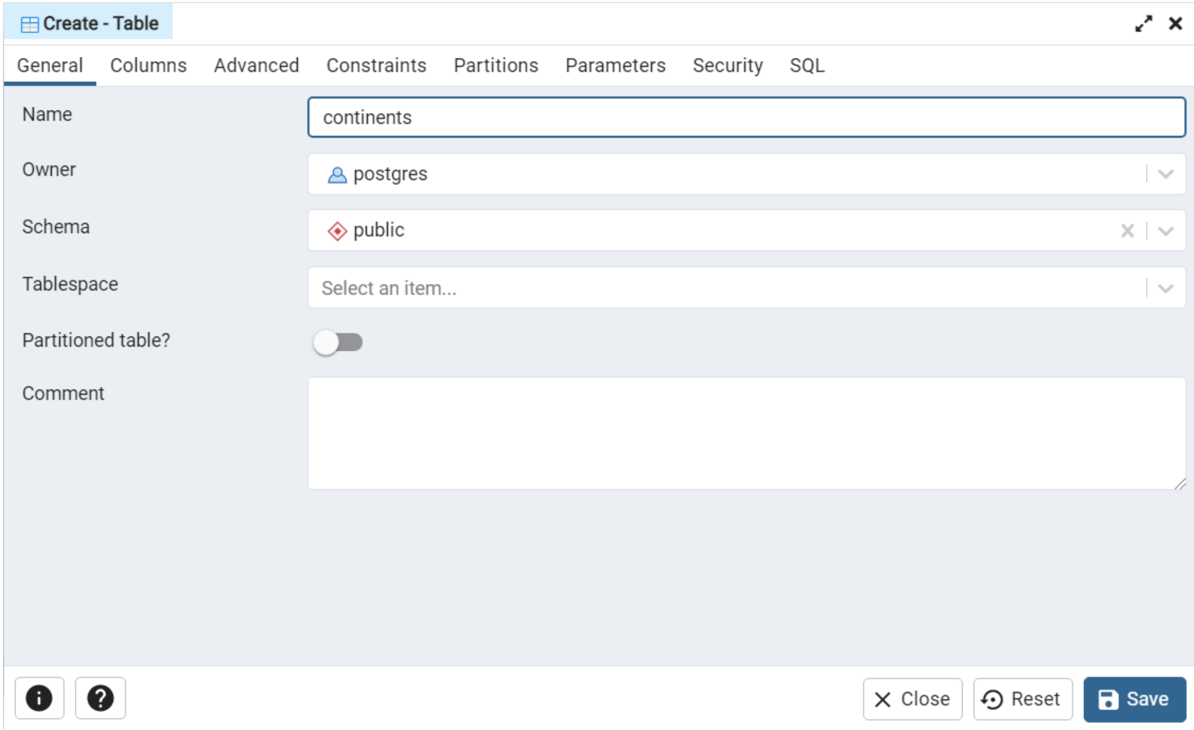
Appendix 1 - ***Step at PostgreSQL (PgAdmin4)***

1) Create a database for storing data that before cleaning

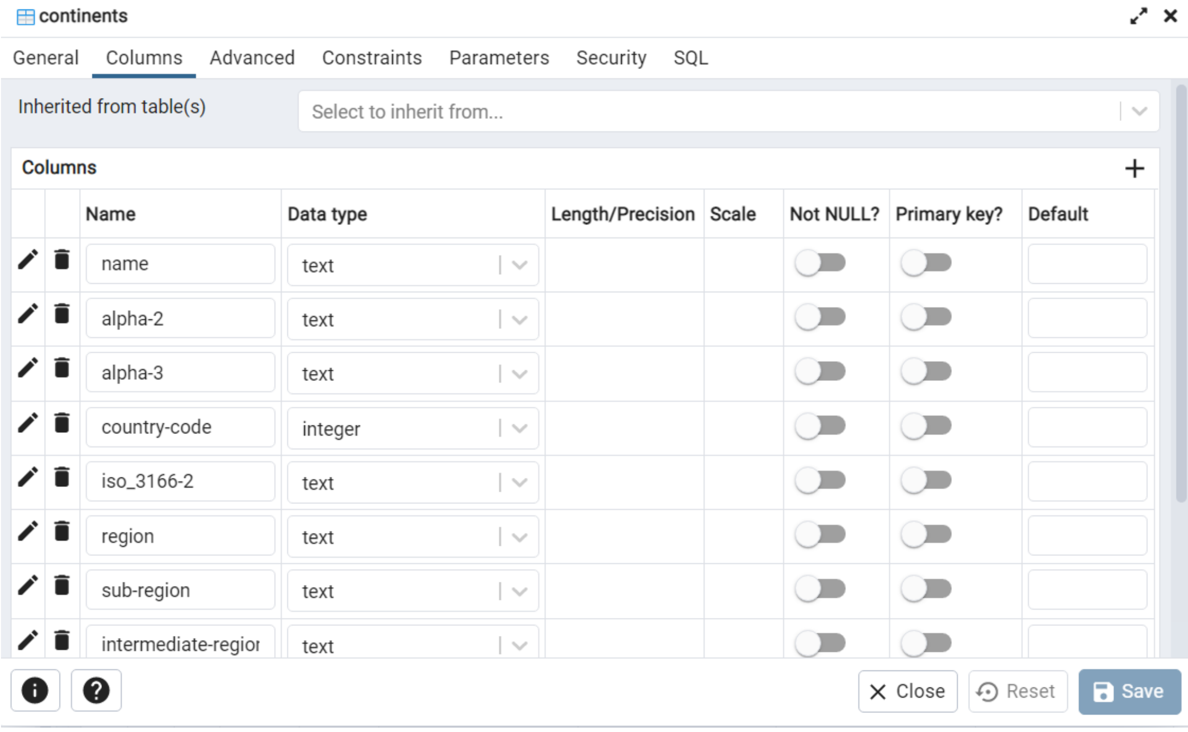


2) Create table in the database

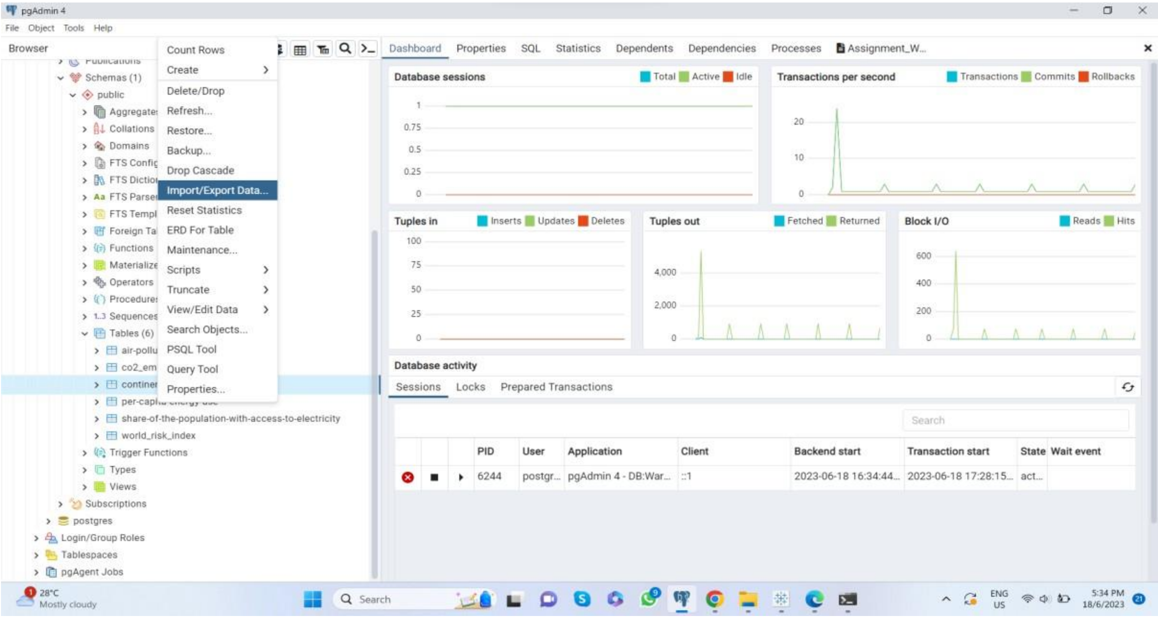


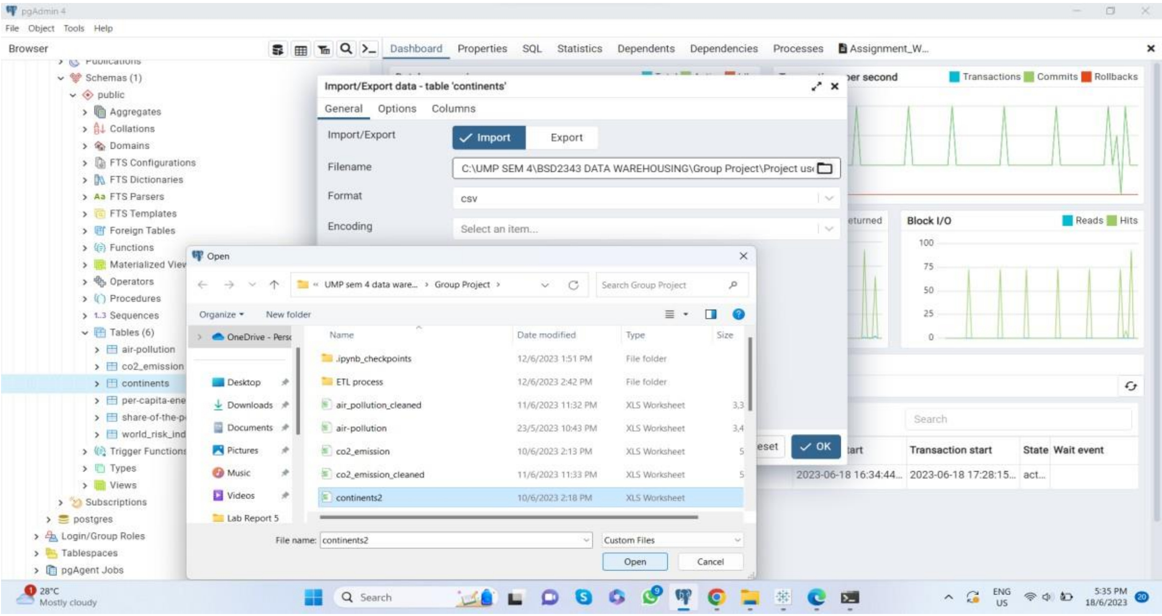


3) Set the columns of the table



4) Import the csv file into the consistent table





5) The following tables are created by Step 2 to Step 4

Appendix 2 - ***Step to extract and transform (Python)***

Firstly, to extract the dataset to Jupyter Notebook we need to install packages.

The pakages need to be installed before run:

1) pip install ipython-sql

2) pip install sqlalchemy

3) pip install psycopg2

4) pip install python-sql

5) pip install pandas-sql

6) pip install sql-queries

7) pip install missingno

In [1]:

```
#! pip install ipython-sql
#! pip install sqlalchemy
#! pip install psycopg2
#! pip install python-sql
#! pip install pandas-sql
#! pip install sql-queries
#! pip install missingno
```

Then, import all the packages.

In [2]:

```
import pandas as pd
import psycopg2 as ps
import pandas.io.sql as sqlio
import missingno as msno
```

This codes allows to load and enable the SQL extension

In [3]:

```
%reload_ext sql
```

To connect to the database, we need to create a postgresql engine.

To create the engine, the syntax is as below:

In [4]:

```
from sqlalchemy import create_engine
```

Setting login information to connect with Postgres

In [5]:

```
conn2=ps.connect(dbname="Warehouse_Before_Clean",

               user="postgres",password= "Kenyeye007007%",host="localhost",

                port="5432")
```

Rretrieve the information from the database

In [6]:

```
sql="""SELECT * FROM pg_catalog.pg_tables"""
```

Country csv

Rretrieve Country information

In [7]:

```
sql="""SELECT * FROM "continents" """
```

In [8]:

```
data_country = sqlio.read_sql_query(sql,conn2)
data_country
```

```
D:\Anaconda\lib\site-packages\pandas\io\sql.py:762: UserWarning: pandas only supp
ort SQLAlchemy connectable(engine/connection) ordatabase string URI or sqlite3 DB
API2 connectionother DBAPI2 objects are not tested, please consider using SQLAlch
emy
  warnings.warn(
```

Out[8]:

	name	alpha-2	alpha-3	country-code	iso_3166-2	region	sub-region	intermediate-region	region cod
0	Afghanistan	AF	AFG	4	ISO 3166-2:AF	Asia	Southern Asia	None	142.
1	Åland Islands	AX	ALA	248	ISO 3166-2:AX	Europe	Northern Europe	None	150.
2	Albania	AL	ALB	8	ISO 3166-2:AL	Europe	Southern Europe	None	150.
3	Algeria	DZ	DZA	12	ISO 3166-2:DZ	Africa	Northern Africa	None	2.
4	American Samoa	AS	ASM	16	ISO 3166-2:AS	Oceania	Polynesia	None	9.
...	...	...	...	...	...	...	...	...	.
244	Wallis and Futuna	WF	WLF	876	ISO 3166-2:WF	Oceania	Polynesia	None	9.
245	Western Sahara	EH	ESH	732	ISO 3166-2:EH	Africa	Northern Africa	None	2.
246	Yemen	YE	YEM	887	ISO 3166-2:YE	Asia	Western Asia	None	142.
247	Zambia	ZM	ZMB	894	ISO 3166-2:ZM	Africa	Sub-Saharan Africa	Eastern Africa	2.
248	Zimbabwe	ZW	ZWE	716	ISO 3166-2:ZW	Africa	Sub-Saharan Africa	Eastern Africa	2.

249 rows × 11 columns

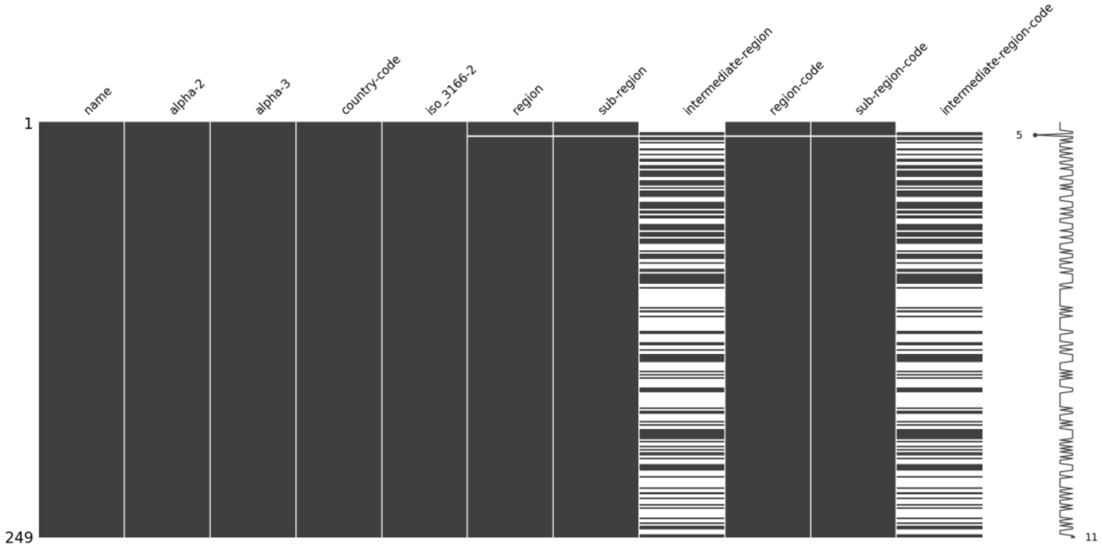
In [9]:

```
## Check missing values

msno.matrix(data_country)
```

Out[9]:

<AxesSubplot:>



In [10]:

```
## Rename columns name

data_country.rename(columns={'name' : 'country_name', 'alpha-2':'alpha_2', 'alpha-3' : 'country_cod
                             'country-code' : 'country_number', 'iso_3166-2' : 'iso_3166_2', 'sub-r
                             'intermediate-region' : 'intermediate_region', 'region-code' : 'region
                             'sub-region-code' : 'sub_region_code', 'intermediate-region-code' : 'i
data_country
```

Out[10]:

	country_name	alpha_2	country_code	country_number	iso_3166_2	region	sub_regio
0	Afghanistan	AF	AFG	4	ISO 3166-2:AF	Asia	Souther Asi
1	Åland Islands	AX	ALA	248	ISO 3166-2:AX	Europe	Norther Europ
2	Albania	AL	ALB	8	ISO 3166-2:AL	Europe	Souther Europ
3	Algeria	DZ	DZA	12	ISO 3166-2:DZ	Africa	Norther Afric
4	American Samoa	AS	ASM	16	ISO 3166-2:AS	Oceania	Polynesi
...	...	...	...	...	...	...	.
244	Wallis and Futuna	WF	WLF	876	ISO 3166-2:WF	Oceania	Polynesi
245	Western Sahara	EH	ESH	732	ISO 3166-2:EH	Africa	Norther Afric
246	Yemen	YE	YEM	887	ISO 3166-2:YE	Asia	Wester Asi
247	Zambia	ZM	ZMB	894	ISO 3166-2:ZM	Africa	Sub Sahara Afric
248	Zimbabwe	ZW	ZWE	716	ISO 3166-2:ZW	Africa	Sub Sahara Afric

249 rows × 11 columns

In [11]:

```
## Drop unnecessary columns

data_country.drop(columns=['alpha_2','country_number','iso_3166_2','intermediate_region','interme
```

In [12]:

```
## Check data type

data_country.dtypes
```

Out[12]:

```
country_name        object
country_code        object
region              object
sub_region          object
region_code        float64
sub_region_code    float64
dtype: object
```

In [13]:

```
## Check NaN value

data_country.isnull().sum()
```

Out[13]:

```
country_name       0
country_code       0
region             1
sub_region         1
region_code        1
sub_region_code    1
dtype: int64
```

In [14]:

```
## List the rows have NaN value

data_country_null_rows = data_country.isnull()
data_country_rows_with_null = data_country[data_country_null_rows.any(axis=1)]
data_country_rows_with_null
```

Out[14]:

	country_name	country_code	region	sub_region	region_code	sub_region_code
8	Antarctica	ATA	None	None	NaN	NaN

In [15]:

```
## Replce the NaN value

data_country["region"].fillna("Antarctic", inplace = True)
data_country["sub_region"].fillna("Subantarctic", inplace = True)
data_country["region_code"].fillna(672, inplace = True)
data_country["sub_region_code"].fillna(10, inplace = True)
```

In [16]:

```
## Change datatype

data_country['region_code'] = data_country['region_code'].astype(int)
data_country['sub_region_code'] = data_country['sub_region_code'].astype(int)
```

In [17]:

```
## Check again make sure no NaN Value

data_country.isnull().sum()
```

Out[17]:

```
country_name       0
country_code       0
region             0
sub_region         0
region_code        0
sub_region_code    0
dtype: int64
```

In [18]:

```
## Check again data type

data_country.dtypes
```

Out[18]:

```
country_name       object
country_code       object
region             object
sub_region         object
region_code         int32
sub_region_code     int32
dtype: object
```

The cleaned data have been shown

In [19]:

```
data_country
```

	country_name	country_code	region	sub_region	region_code	sub_region_code
0	Afghanistan	AFG	Asia	Southern Asia	142	34
1	Åland Islands	ALA	Europe	Northern Europe	150	154
2	Albania	ALB	Europe	Southern Europe	150	39
3	Algeria	DZA	Africa	Northern Africa	2	15
4	American Samoa	ASM	Oceania	Polynesia	9	61
...	...	...	...	...	...	...
244	Wallis and Futuna	WLF	Oceania	Polynesia	9	61
245	Western	ESH	Africa	Northern	2	15

Then export to the csv file

In [20]:

```
data_country.to_csv('country_cleaned.csv', index=False)
```

Air Pollution csv

Rretrieve Air Pollution information

In [21]:

```
sql="""SELECT * FROM "air-pollution" """
```

In [22]:

```
data_air_pollution = sqlio.read_sql_query(sql,conn2)
data_air_pollution
```

```
D:\Anaconda\lib\site-packages\pandas\io\sql.py:762: UserWarning: pandas only supp
ort SQLAlchemy connectable(engine/connection) ordatabase string URI or sqlite3 DB
API2 connectionother DBAPI2 objects are not tested, please consider using SQLAlch
emy
  warnings.warn(
```

Out[22]:

	Country	Year	Nitrogen Oxide	Sulphur Dioxide	Carbon Monoxide	Organic Carbon	NMVOCs	Black Carbon	A
0	Afghanistan	1750	555.42	139.42	142073.31	5456.88	13311.29	1633.03	
1	Afghanistan	1760	578.45	145.09	147859.24	5679.12	13853.64	1699.54	
2	Afghanistan	1770	602.42	150.99	153867.41	5909.88	14416.85	1768.60	
3	Afghanistan	1780	627.37	157.11	160104.42	6149.44	15001.56	1840.29	
4	Afghanistan	1790	653.34	163.46	166576.77	6398.04	15608.38	1914.68	
...	...	...	...	...	...	...	...	...	
47530	Zimbabwe	2015	83842.10	67231.29	1610636.44	108275.48	299713.47	30912.24	11
47531	Zimbabwe	2016	76234.43	59452.70	1632515.11	111975.72	302718.32	31570.53	11
47532	Zimbabwe	2017	74381.80	53891.39	1657688.51	114613.20	306905.62	32344.41	11
47533	Zimbabwe	2018	73062.53	51072.78	1653664.68	114583.51	306860.21	32365.56	11
47534	Zimbabwe	2019	70779.92	45896.98	1647792.46	114543.28	306574.85	32364.53	12

47535 rows × 9 columns

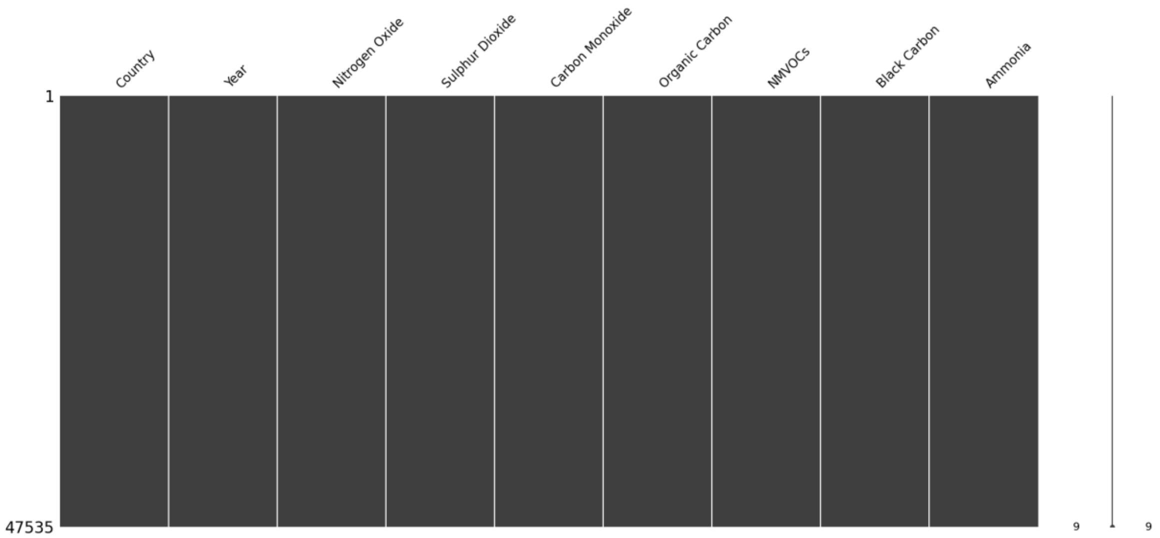
In [23]:

```
## Check missing values

msno.matrix(data_air_pollution)
```

Out[23]:

<AxesSubplot:>



In [24]:

```
## Rename columns name

data_air_pollution.rename(columns={'Country' : 'country_name','Year' : 'year','Nitrogen Oxide' : 'ni
                                   'Sulphur Dioxide' : 'sulphur_dioxide', 'Carbon Monoxide' : 'carb
                                   'Organic Carbon' : 'organic_carbon','NMVOCs' : 'nmvoc','Black Ca
                                   'Ammonia' : 'ammonia'}, inplace=True)
data_air_pollution
```

Out[24]:

	country_name	year	nitrogen_oxide	sulphur_dioxide	carbon_monoxide	organic_carb
0	Afghanistan	1750	555.42	139.42	142073.31	5456
1	Afghanistan	1760	578.45	145.09	147859.24	5679
2	Afghanistan	1770	602.42	150.99	153867.41	5909
3	Afghanistan	1780	627.37	157.11	160104.42	6149
4	Afghanistan	1790	653.34	163.46	166576.77	6398
...	...	...	...	...	...	
47530	Zimbabwe	2015	83842.10	67231.29	1610636.44	108275
47531	Zimbabwe	2016	76234.43	59452.70	1632515.11	111975
47532	Zimbabwe	2017	74381.80	53891.39	1657688.51	114613
47533	Zimbabwe	2018	73062.53	51072.78	1653664.68	114583
47534	Zimbabwe	2019	70779.92	45896.98	1647792.46	114543

47535 rows × 9 columns

In [25]:

```
## Check data type

data_air_pollution.dtypes
```

Out[25]:

```
country_name        object
year                 int64
nitrogen_oxide     float64
sulphur_dioxide    float64
carbon_monoxide    float64
organic_carbon     float64
nmvoc              float64
black_carbon       float64
ammonia            float64
dtype: object
```

In [26]:

```
## Check NaN value

data_air_pollution.isnull().sum()
```

Out[26]:

```
country_name       0
year               0
nitrogen_oxide     0
sulphur_dioxide    0
carbon_monoxide    0
organic_carbon     0
nmvoc              0
black_carbon       0
ammonia            0
dtype: int64
```

Then export to the csv file

In [27]:

```
data_air_pollution.to_csv('air_pollution_cleaned.csv', index=False)
```

Co2 Emission csv

Rretrieve Co2 Emission information

In [28]:

```
sql="""SELECT * FROM "co2_emission" """
```

In [29]:

```
data_co2_emission = sqlio.read_sql_query(sql,conn2)
data_co2_emission
```

```
D:\Anaconda\lib\site-packages\pandas\io\sql.py:762: UserWarning: pandas only supp
ort SQLAlchemy connectable(engine/connection) ordatabase string URI or sqlite3 DB
API2 connectionother DBAPI2 objects are not tested, please consider using SQLAlch
emy
  warnings.warn(
```

Out[29]:

	Entity	Code	Year	Annual CO2 emissions (tonnes)
0	Afghanistan	AFG	1949	14656.00
1	Afghanistan	AFG	1950	84272.00
2	Afghanistan	AFG	1951	91600.00
3	Afghanistan	AFG	1952	91600.00
4	Afghanistan	AFG	1953	106256.00
...	...	...	...	...
20848	Zimbabwe	ZWE	2013	11536239.29
20849	Zimbabwe	ZWE	2014	11866348.41
20850	Zimbabwe	ZWE	2015	10907603.94
20851	Zimbabwe	ZWE	2016	9932649.88
20852	Zimbabwe	ZWE	2017	10397718.47

20853 rows × 4 columns

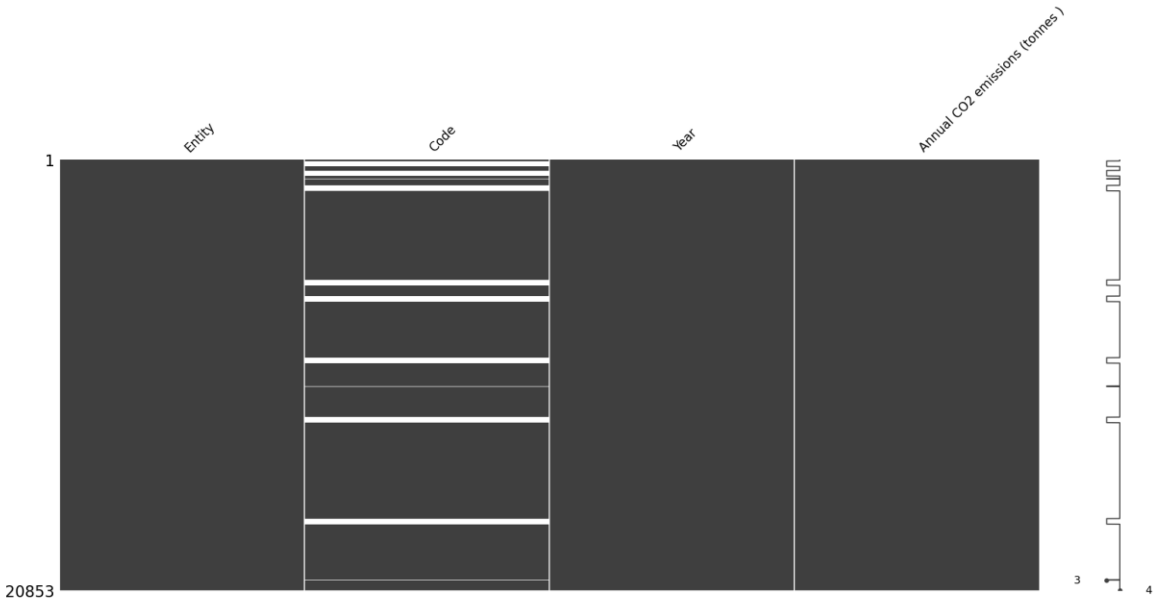
In [30]:

```
## Check missing values

msno.matrix(data_co2_emission)
```

Out[30]:

<AxesSubplot:>



In [31]:

```
## Rename columns name

data_co2_emission.rename(columns={'Entity': 'country_name','Code': 'country_code','Year': 'year',
                                  'Annual CO2 emissions (tonnes )': 'annual_co2_emissions'}, inpl
data_co2_emission
```

Out[31]:

	country_name	country_code	year	annual_co2_emissions
0	Afghanistan	AFG	1949	14656.00
1	Afghanistan	AFG	1950	84272.00
2	Afghanistan	AFG	1951	91600.00
3	Afghanistan	AFG	1952	91600.00
4	Afghanistan	AFG	1953	106256.00
...	...	...	...	...
20848	Zimbabwe	ZWE	2013	11536239.29
20849	Zimbabwe	ZWE	2014	11866348.41
20850	Zimbabwe	ZWE	2015	10907603.94
20851	Zimbabwe	ZWE	2016	9932649.88
20852	Zimbabwe	ZWE	2017	10397718.47

20853 rows × 4 columns

In [32]:

```
## Check data type

data_co2_emission.dtypes
```

Out[32]:

```
country_name              object
country_code              object
year                       int64
annual_co2_emissions     float64
dtype: object
```

In [33]:

```
## Check NaN value

data_co2_emission.isnull().sum()
```

Out[33]:

```
country_name               0
country_code            2207
year                       0
annual_co2_emissions       0
dtype: int64
```

In [34]:

```
## List the rows have NaN value

data_co2_emission_null_rows = data_co2_emission.isnull()
data_co2_emission_rows_with_null = data_co2_emission[data_co2_emission_null_rows.any(axis=1)]
data_co2_emission_rows_with_null
```

Out[34]:

	country_name	country_code	year	annual_co2_emissions
69	Africa	None	1751	0.00
70	Africa	None	1752	0.00
71	Africa	None	1753	0.00
72	Africa	None	1754	0.00
73	Africa	None	1755	0.00
...	...	...	...	...
20348	Wallis and Futuna Islands	None	2013	21984.00
20349	Wallis and Futuna Islands	None	2014	21984.00
20350	Wallis and Futuna Islands	None	2015	23990.58
20351	Wallis and Futuna Islands	None	2016	24265.49
20352	Wallis and Futuna Islands	None	2017	25909.08

2207 rows × 4 columns

In [35]:

```
## Drop the NaN value

data_co2_emission = data_co2_emission.dropna()
data_co2_emission
```

Out[35]:

	country_name	country_code	year	annual_co2_emissions
0	Afghanistan	AFG	1949	14656.00
1	Afghanistan	AFG	1950	84272.00
2	Afghanistan	AFG	1951	91600.00
3	Afghanistan	AFG	1952	91600.00
4	Afghanistan	AFG	1953	106256.00
...	...	...	...	...
20848	Zimbabwe	ZWE	2013	11536239.29
20849	Zimbabwe	ZWE	2014	11866348.41
20850	Zimbabwe	ZWE	2015	10907603.94
20851	Zimbabwe	ZWE	2016	9932649.88
20852	Zimbabwe	ZWE	2017	10397718.47

18646 rows × 4 columns

In [36]:

```
## Check NaN value after cleaning

data_co2_emission.isnull().sum()
```

Out[36]:

```
country_name            0
country_code            0
year                    0
annual_co2_emissions    0
dtype: int64
```

Then export to csv file

In [37]:

```
data_co2_emission.to_csv('co2_emission_cleaned.csv', index=False)
```

Per Capita Energy Use csv

Rretrieve Per Capita Energy Use information

In [38]:

```
sql="""SELECT * FROM "per-capita-energy-use" """
```

In [39]:

```
data_per_capita_energy_use = sqlio.read_sql_query(sql,conn2)
data_per_capita_energy_use
```

```
D:\Anaconda\lib\site-packages\pandas\io\sql.py:762: UserWarning: pandas only supp
ort SQLAlchemy connectable(engine/connection) ordatabase string URI or sqlite3 DB
API2 connectionother DBAPI2 objects are not tested, please consider using SQLAlch
emy
  warnings.warn(
```

Out[39]:

	Entity	Code	Year	Energy consumption per capita (kWh)
0	Afghanistan	AFG	1980	581.932201
1	Afghanistan	AFG	1981	662.912777
2	Afghanistan	AFG	1982	709.075252
3	Afghanistan	AFG	1983	877.845852
4	Afghanistan	AFG	1984	905.948350
...	...	...	...	...
8956	Zimbabwe	ZWE	2012	4251.321680
8957	Zimbabwe	ZWE	2013	4200.828880
8958	Zimbabwe	ZWE	2014	4127.800993
8959	Zimbabwe	ZWE	2015	4027.628101
8960	Zimbabwe	ZWE	2016	3385.574447

8961 rows × 4 columns

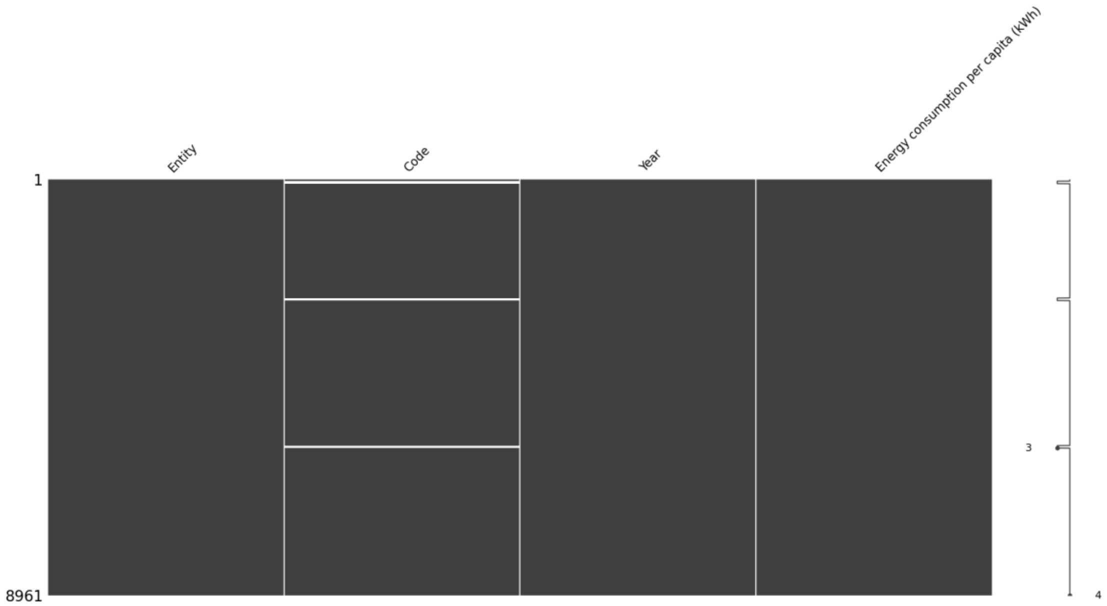
In [40]:

```
## Check missing value

msno.matrix(data_per_capita_energy_use)
```

Out[40]:

<AxesSubplot:>



In [41]:

```
## Rename columns name

data_per_capita_energy_use.rename(columns={'Entity' : 'country_name','Code' : 'country_code','Year'
                                           'Energy consumption per capita (kWh)' : 'energy_consump
                                  inplace=True)
data_per_capita_energy_use
```

Out[41]:

	country_name	country_code	year	energy_consumption_per_capita
0	Afghanistan	AFG	1980	581.932201
1	Afghanistan	AFG	1981	662.912777
2	Afghanistan	AFG	1982	709.075252
3	Afghanistan	AFG	1983	877.845852
4	Afghanistan	AFG	1984	905.948350
...	...	...	...	...
8956	Zimbabwe	ZWE	2012	4251.321680
8957	Zimbabwe	ZWE	2013	4200.828880
8958	Zimbabwe	ZWE	2014	4127.800993
8959	Zimbabwe	ZWE	2015	4027.628101
8960	Zimbabwe	ZWE	2016	3385.574447

8961 rows × 4 columns

In [42]:

```
## Check data type

data_per_capita_energy_use.dtypes
```

Out[42]:

```
country_name                      object
country_code                      object
year                               int64
energy_consumption_per_capita    float64
dtype: object
```

In [43]:

```
## Check NaN value

data_per_capita_energy_use.isnull().sum()
```

Out[43]:

```
country_name                       0
country_code                     165
year                               0
energy_consumption_per_capita      0
dtype: int64
```

In [44]:

```
## List the rows have NaN value

data_per_capita_energy_use_null_rows = data_per_capita_energy_use.isnull()
data_per_capita_energy_use_rows_with_null = data_per_capita_energy_use[data_per_capita_energy_use
data_per_capita_energy_use_rows_with_null
```

Out[44]:

	country_name	country_code	year	energy_consumption_per_capita
37	Africa	None	1965	2178.897737
38	Africa	None	1966	2234.789825
39	Africa	None	1967	2197.197691
40	Africa	None	1968	2259.624226
41	Africa	None	1969	2255.869649
...	...	...	...	...
5783	North America	None	2015	88562.993841
5784	North America	None	2016	87878.288500
5785	North America	None	2017	87748.431543
5786	North America	None	2018	89812.396931
5787	North America	None	2019	88336.805100

165 rows × 4 columns

In [45]:

```
## Drop the NaN value

data_per_capita_energy_use = data_per_capita_energy_use.dropna()
data_per_capita_energy_use
```

Out[45]:

	country_name	country_code	year	energy_consumption_per_capita
0	Afghanistan	AFG	1980	581.932201
1	Afghanistan	AFG	1981	662.912777
2	Afghanistan	AFG	1982	709.075252
3	Afghanistan	AFG	1983	877.845852
4	Afghanistan	AFG	1984	905.948350
...	...	...	...	...
8956	Zimbabwe	ZWE	2012	4251.321680
8957	Zimbabwe	ZWE	2013	4200.828880
8958	Zimbabwe	ZWE	2014	4127.800993
8959	Zimbabwe	ZWE	2015	4027.628101
8960	Zimbabwe	ZWE	2016	3385.574447

8796 rows × 4 columns

In [46]:

```
## Check NaN value after cleaning

data_per_capita_energy_use.isnull().sum()
```

Out[46]:

```
country_name                     0
country_code                     0
year                             0
energy_consumption_per_capita    0
dtype: int64
```

Then export to csv file

In [47]:

```
data_per_capita_energy_use.to_csv('energy_use_per_capita_cleaned.csv', index=False)
```

Share of the Population with Access to Electricity csv

Rretrieve Share of the Population with Access to Electricity information

In [48]:

```
sql="""SELECT * FROM "share-of-the-population-with-access-to-electricity" """
```

In [49]:

```
data_population_access_electricity = sqlio.read_sql_query(sql,conn2)
data_population_access_electricity
```

```
D:\Anaconda\lib\site-packages\pandas\io\sql.py:762: UserWarning: pandas only supp
ort SQLAlchemy connectable(engine/connection) ordatabase string URI or sqlite3 DB
API2 connectionother DBAPI2 objects are not tested, please consider using SQLAlch
emy
  warnings.warn(
```

Out[49]:

	Entity	Code	Year	Access to electricity (% of population)
0	Afghanistan	AFG	1990	0.010000
1	Afghanistan	AFG	1991	0.010000
2	Afghanistan	AFG	1992	0.010000
3	Afghanistan	AFG	1993	0.010000
4	Afghanistan	AFG	1994	0.010000
...	...	...	...	...
6953	Zimbabwe	ZWE	2012	36.728878
6954	Zimbabwe	ZWE	2013	37.076813
6955	Zimbabwe	ZWE	2014	32.300000
6956	Zimbabwe	ZWE	2015	33.700000
6957	Zimbabwe	ZWE	2016	38.145138

6958 rows × 4 columns

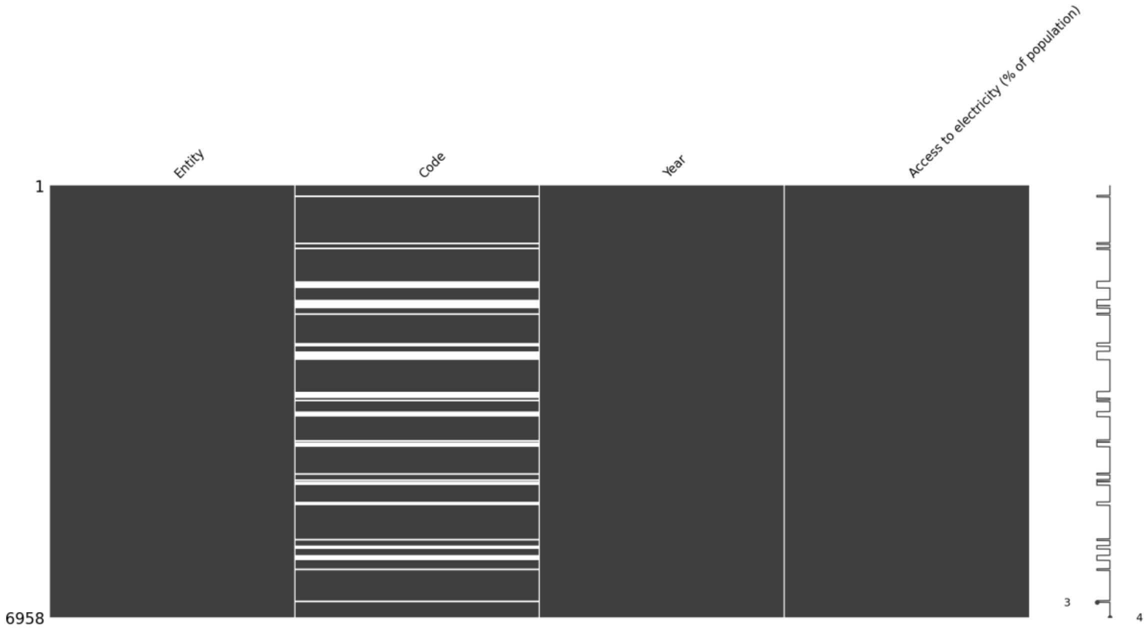
In [50]:

```
## Check missing values

msno.matrix(data_population_access_electricity)
```

Out[50]:

<AxesSubplot:>



In [51]:

```
## Rename columns name

data_population_access_electricity.rename(columns={'Entity': 'country_name','Code': 'country_code
data_population_access_electricity
```

Out[51]:

	country_name	country_code	year	access_to_electricity_percent_of_population
0	Afghanistan	AFG	1990	0.010000
1	Afghanistan	AFG	1991	0.010000
2	Afghanistan	AFG	1992	0.010000
3	Afghanistan	AFG	1993	0.010000
4	Afghanistan	AFG	1994	0.010000
...	...	...	...	...
6953	Zimbabwe	ZWE	2012	36.728878
6954	Zimbabwe	ZWE	2013	37.076813
6955	Zimbabwe	ZWE	2014	32.300000
6956	Zimbabwe	ZWE	2015	33.700000
6957	Zimbabwe	ZWE	2016	38.145138

6958 rows × 4 columns

In [52]:

```
## Check data type

data_population_access_electricity.dtypes
```

Out[52]:

```
country_name                                 object
country_code                                 object
year                                          int64
access_to_electricity_percent_of_population  float64
dtype: object
```

In [53]:

```
## Check NaN value

data_population_access_electricity.isnull().sum()
```

Out[53]:

```
country_name                                      0
country_code                                   1242
year                                              0
access_to_electricity_percent_of_population       0
dtype: int64
```

In [54]:

```
## List the rows have NaN value

data_population_access_electricity_null_rows = data_population_access_electricity.isnull()
data_population_access_electricity_rows_with_null = data_population_access_electricity[data_popul
data_population_access_electricity_rows_with_null
```

Out[54]:

	country_name	country_code	year	access_to_electricity_percent_of_population
162	Arab World	None	1990	74.384239
163	Arab World	None	1991	74.382220
164	Arab World	None	1992	74.313160
165	Arab World	None	1993	75.349325
166	Arab World	None	1994	75.788522
...	...	...	...	...
6710	Upper middle income	None	2012	99.059438
6711	Upper middle income	None	2013	99.214211
6712	Upper middle income	None	2014	99.235891
6713	Upper middle income	None	2015	99.268934
6714	Upper middle income	None	2016	99.379714

1242 rows × 4 columns

In [55]:

```
## Drop the NaN value

data_population_access_electricity = data_population_access_electricity.dropna()
data_population_access_electricity
```

Out[55]:

	country_name	country_code	year	access_to_electricity_percent_of_population
0	Afghanistan	AFG	1990	0.010000
1	Afghanistan	AFG	1991	0.010000
2	Afghanistan	AFG	1992	0.010000
3	Afghanistan	AFG	1993	0.010000
4	Afghanistan	AFG	1994	0.010000
...	...	...	...	...
6953	Zimbabwe	ZWE	2012	36.728878
6954	Zimbabwe	ZWE	2013	37.076813
6955	Zimbabwe	ZWE	2014	32.300000
6956	Zimbabwe	ZWE	2015	33.700000
6957	Zimbabwe	ZWE	2016	38.145138

5716 rows × 4 columns

In [56]:

```
## Check NaN value after cleaning

data_population_access_electricity.isnull().sum()
```

Out[56]:

```
country_name                                   0
country_code                                   0
year                                           0
access_to_electricity_percent_of_population    0
dtype: int64
```

Then export to csv file

In [57]:

```
data_population_access_electricity.to_csv('share_of_population_with_access_to_electricity_cleaned
```

World Risk Index csv

Rretrieve World Risk Index information

In [58]:

```
sql="""SELECT * FROM "world_risk_index" """
```

In [59]:

```
data_world_risk_index = sqlio.read_sql_query(sql,conn2)
data_world_risk_index
```

```
D:\Anaconda\lib\site-packages\pandas\io\sql.py:762: UserWarning: pandas only supp
ort SQLAlchemy connectable(engine/connection) ordatabase string URI or sqlite3 DB
API2 connectionother DBAPI2 objects are not tested, please consider using SQLAlch
emy
  warnings.warn(
```

Out[59]:

	Region	WRI	Exposure	Vulnerability	Susceptibility	Lack of Coping Capabilities	Lack of Adaptive Capacities	Year
0	Vanuatu	32.00	56.33	56.81	37.14	79.34	53.96	2011
1	Tonga	29.08	56.04	51.90	28.94	81.80	44.97	2011
2	Philippinen	24.32	45.09	53.93	34.99	82.78	44.01	2011
3	Salomonen	23.51	36.40	64.60	44.11	85.95	63.74	2011
4	Guatemala	20.88	38.42	54.35	35.36	77.83	49.87	2011
...	...	...	...	...	...	...	...	...
1912	Grenada	1.42	3.13	45.39	24.54	68.82	42.82	2016
1913	Barbados	1.32	3.46	38.26	18.20	50.29	46.29	2016
1914	Saudi Arabia	1.14	2.93	38.96	14.80	65.01	37.07	2016
1915	Malta	0.60	1.65	36.25	15.97	59.33	33.44	2016
1916	Qatar	0.08	0.28	28.18	9.68	43.94	30.93	2016

1917 rows × 12 columns

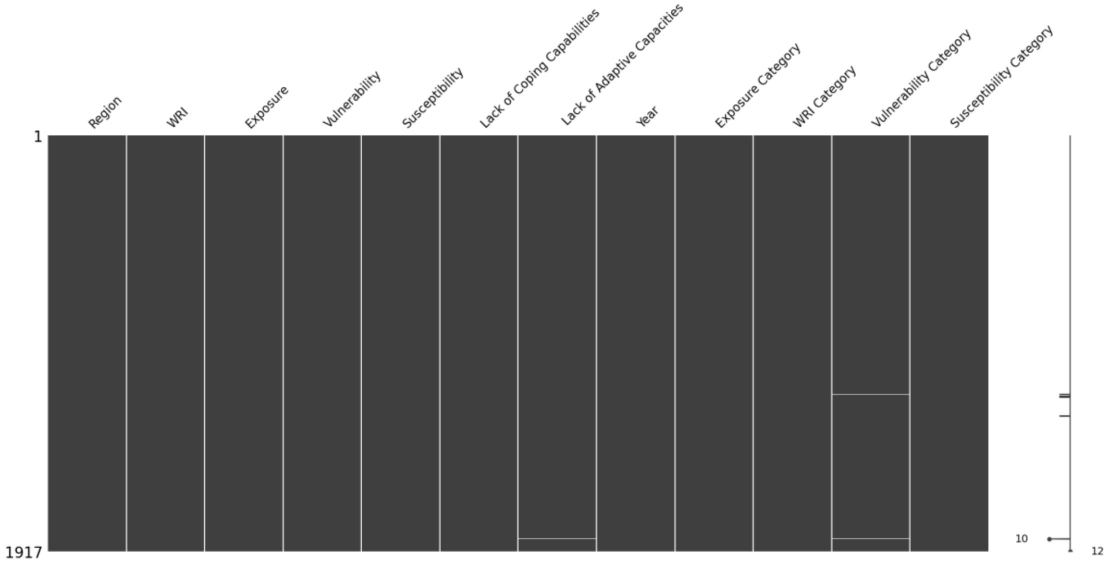
In [60]:

```
## Check missing values

msno.matrix(data_world_risk_index)
```

Out[60]:

<AxesSubplot:>



In [61]:

```
## Rename columns name

data_world_risk_index.rename(columns={'Region' : 'country_name','WRI' : 'wri','Exposure' : 'exposure
                                     'Vulnerability' : 'vulnerability','Susceptibility' : 'suscept
                                     'Lack of Coping Capabilities' : 'lack_of_coping_capabilities
                                     ' Lack of Adaptive Capacities' : 'lack_of_adaptive_capacitie
                                     'Year' : 'year', 'Exposure Category' : 'exposure_category',
                                     'WRI Category' : 'wri_category', 'Vulnerability Category' : '
                                     'Susceptibility Category' : 'susceptibility_category'}, inpl
data_world_risk_index
```

Out[61]:

	country_name	wri	exposure	vulnerability	susceptibility	lack_of_coping_capabilities
0	Vanuatu	32.00	56.33	56.81	37.14	79.34
1	Tonga	29.08	56.04	51.90	28.94	81.80
2	Philippinen	24.32	45.09	53.93	34.99	82.78
3	Salomonen	23.51	36.40	64.60	44.11	85.95
4	Guatemala	20.88	38.42	54.35	35.36	77.83
...	...	...	...	...	...	...
1912	Grenada	1.42	3.13	45.39	24.54	68.82
1913	Barbados	1.32	3.46	38.26	18.20	50.29
1914	Saudi Arabia	1.14	2.93	38.96	14.80	65.01
1915	Malta	0.60	1.65	36.25	15.97	59.33
1916	Qatar	0.08	0.28	28.18	9.68	43.94

1917 rows × 12 columns

In [62]:

```
## Check data type

data_world_risk_index.dtypes
```

Out[62]:

```
country_name                    object
wri                            float64
exposure                       float64
vulnerability                  float64
susceptibility                 float64
lack_of_coping_capabilities    float64
lack_of_adaptive_capacities    float64
year                             int64
exposure_category               object
wri_category                    object
vulnerability_category          object
susceptibility_category         object
dtype: object
```

In [63]:

```
## Check NaN value

data_world_risk_index.isnull().sum()
```

Out[63]:

```
country_name                   0
wri                            0
exposure                       0
vulnerability                  0
susceptibility                 0
lack_of_coping_capabilities    0
lack_of_adaptive_capacities    1
year                           0
exposure_category              0
wri_category                   1
vulnerability_category         4
susceptibility_category        0
dtype: int64
```

In [64]:

```
## List the rows have NaN value

data_world_risk_index_null_rows = data_world_risk_index.isnull()
data_world_risk_index_rows_with_null = data_world_risk_index[data_world_risk_index_null_rows.any(
data_world_risk_index_rows_with_null
```

Out[64]:

	country_name	wri	exposure	vulnerability	susceptibility	lack_of_coping_capabilities
1193	Österreich	2.87	13.18	21.75	13.63	39.27
1202	Deutschland	2.43	11.51	21.11	14.30	36.44
1205	Norwegen	2.34	10.60	22.06	13.29	39.21
1292	Föd. Staaten v. Mikronesien	7.59	14.95	50.77	31.79	72.13
1858	Korea Republic of 4.59	14.89	30.82	14.31	46.55	31.59

In [65]:

```
## Rename country name to the appropriate

data_world_risk_index.loc[data_world_risk_index["country_name"] == "Korea Republic of 4.59",
                          ["country_name"]] = ["Republic of Korea"]
```

In [66]:

```
## Replace the NaN value

data_world_risk_index["vulnerability_category"].fillna("Very Low", inplace = True)
```

In [67]:

```
## Check NaN value after cleaning

data_world_risk_index.isnull().sum()
```

Out[67]:

```
country_name                    0
wri                             0
exposure                        0
vulnerability                   0
susceptibility                  0
lack_of_coping_capabilities     0
lack_of_adaptive_capacities     1
year                            0
exposure_category               0
wri_category                    1
vulnerability_category          0
susceptibility_category         0
dtype: int64
```

In [68]:

```
## List the row that have NaN value

data_world_risk_index_null_rows = data_world_risk_index.isnull()
data_world_risk_index_rows_with_null = data_world_risk_index[data_world_risk_index_null_rows.any(
data_world_risk_index_rows_with_null
```

Out[68]:

	country_name	wri	exposure	vulnerability	susceptibility	lack_of_coping_capabilities
1292	Föd. Staaten v. Mikronesien	7.59	14.95	50.77	31.79	72.13
1858	Republic of Korea	14.89	30.82	14.31	46.55	31.59

In [69]:

```
## Use mode to replace the NaN value

mode_value = data_world_risk_index['wri_category'].mode()[0]
data_world_risk_index['wri_category'].fillna(mode_value, inplace=True)
```

In [70]:

```
## Use mode to replace the NaN value

mode_value = data_world_risk_index['lack_of_adaptive_capacities'].mode()[0]
data_world_risk_index['lack_of_adaptive_capacities'].fillna(mode_value, inplace=True)
```

In [71]:

```
## Check agian make sure no NaN value

data_world_risk_index.isnull().sum()
```

Out[71]:

```
country_name                   0
wri                            0
exposure                       0
vulnerability                  0
susceptibility                 0
lack_of_coping_capabilities    0
lack_of_adaptive_capacities    0
year                           0
exposure_category              0
wri_category                   0
vulnerability_category         0
susceptibility_category        0
dtype: int64
```

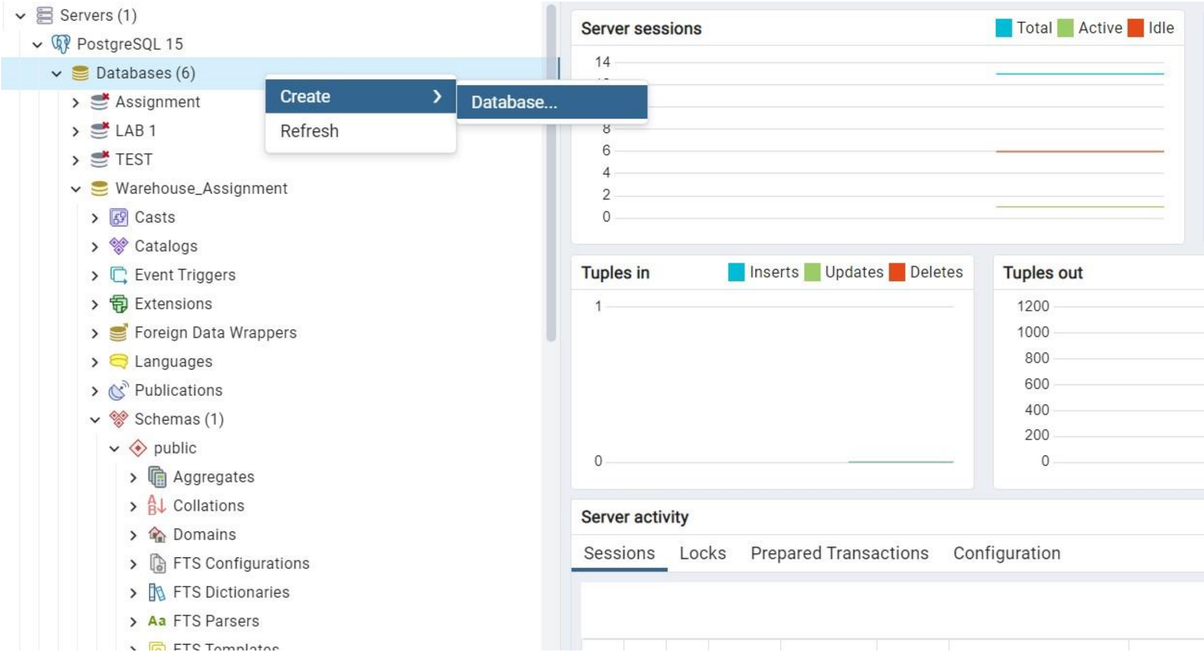
Then export to csv file

In [72]:

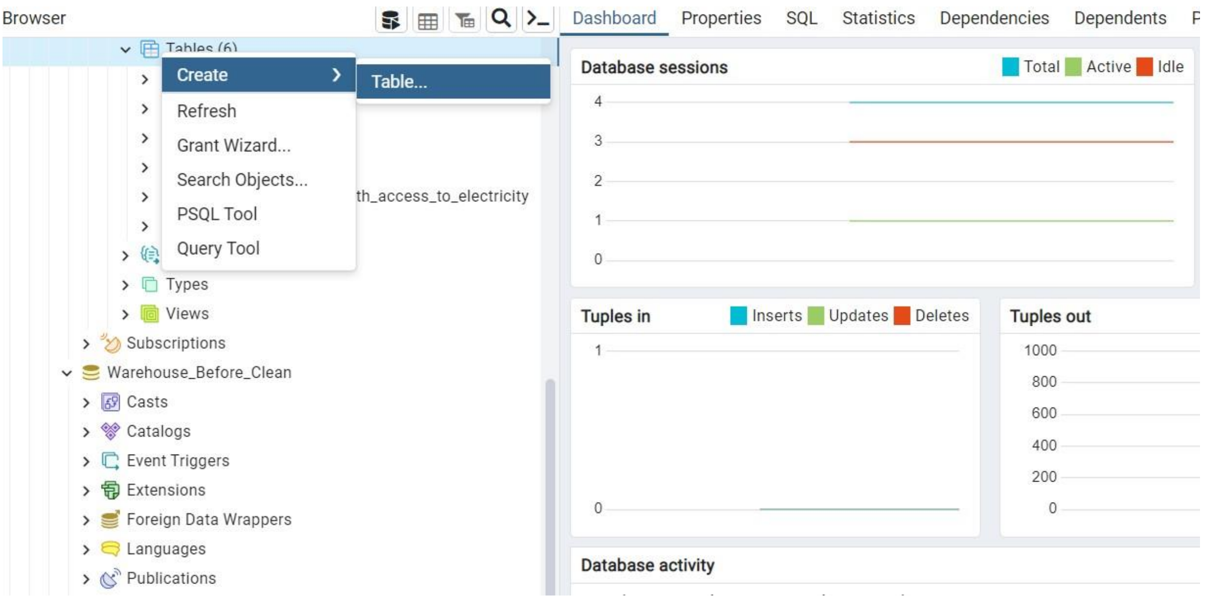
```
data_world_risk_index.to_csv('world_risk_index_cleaned.csv', index=False)
```

Appendix 3 - **Step to load data into database (Postgre SQL- PgAdmin4)**

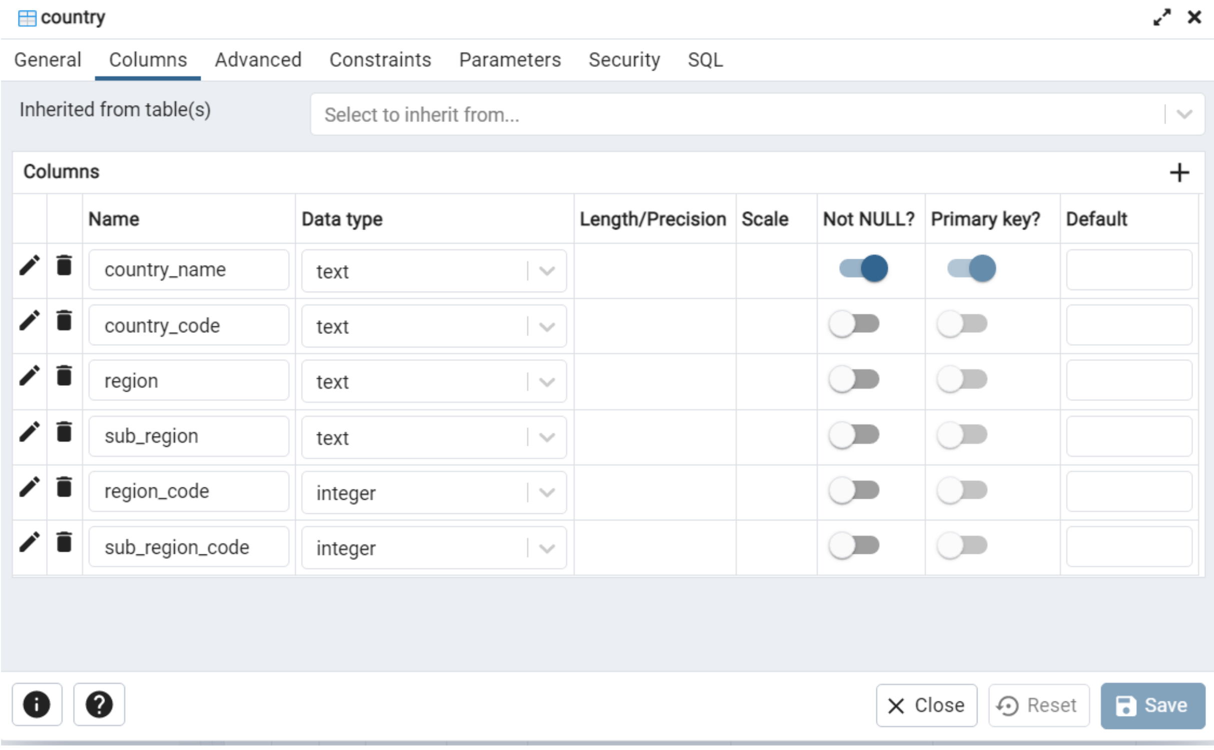
1. Create a new database in PgAdmin 4 for storing the cleaned dataset.



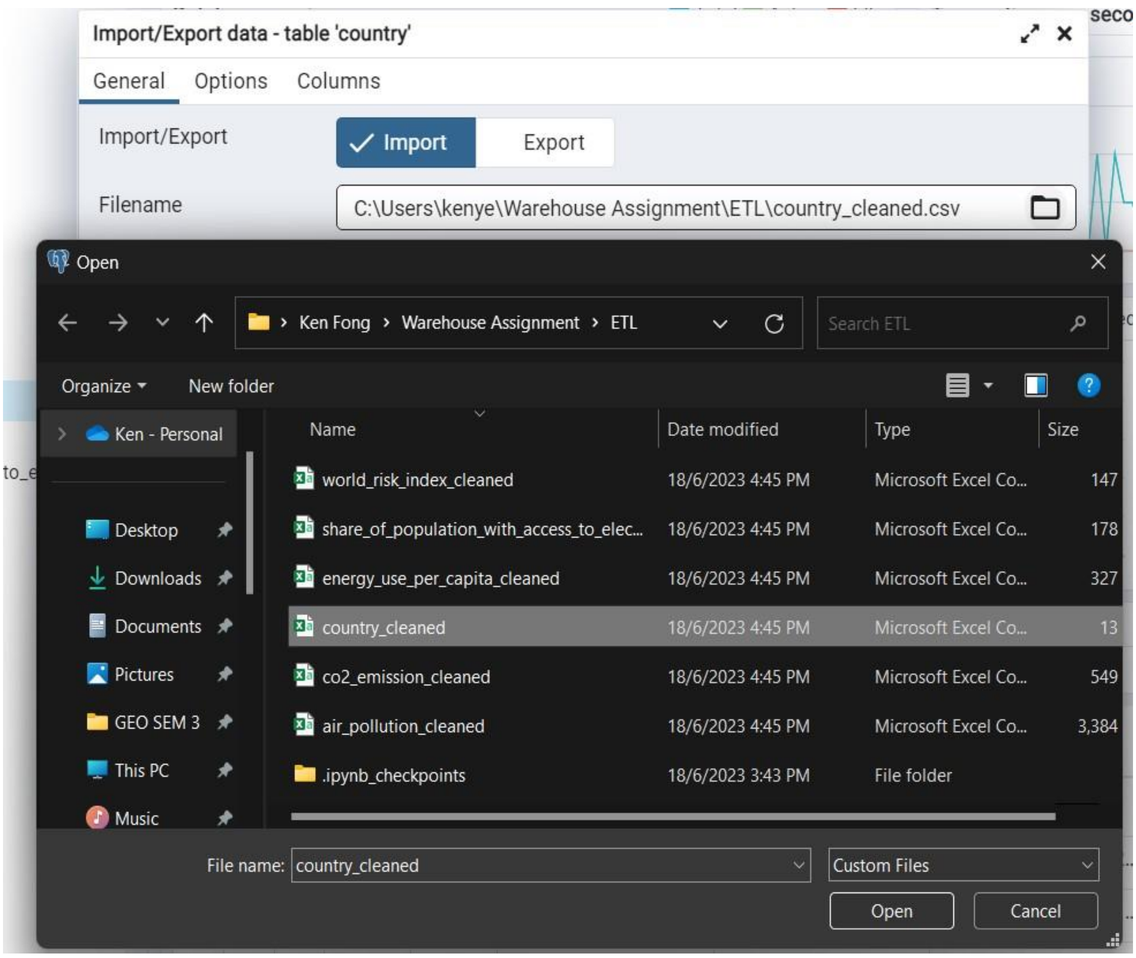
2. Create a new table in the database.



3. Set the column name for the country table.

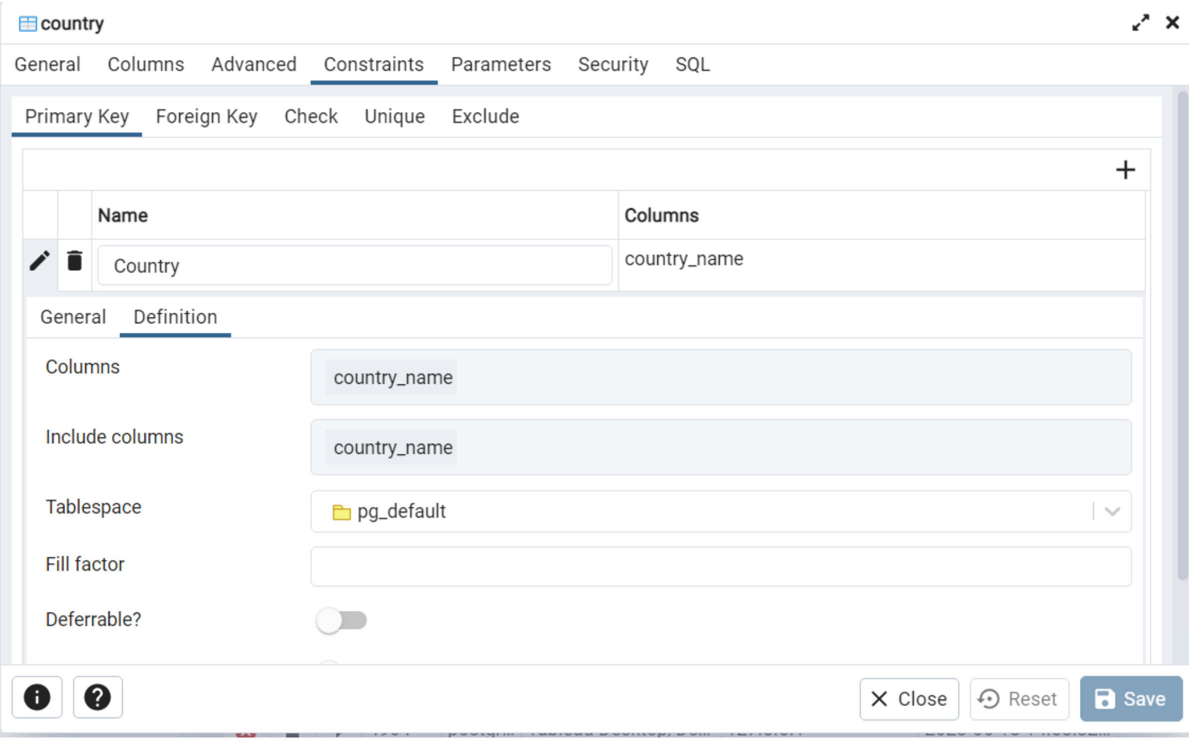


4. Import the CSV file for the country table.

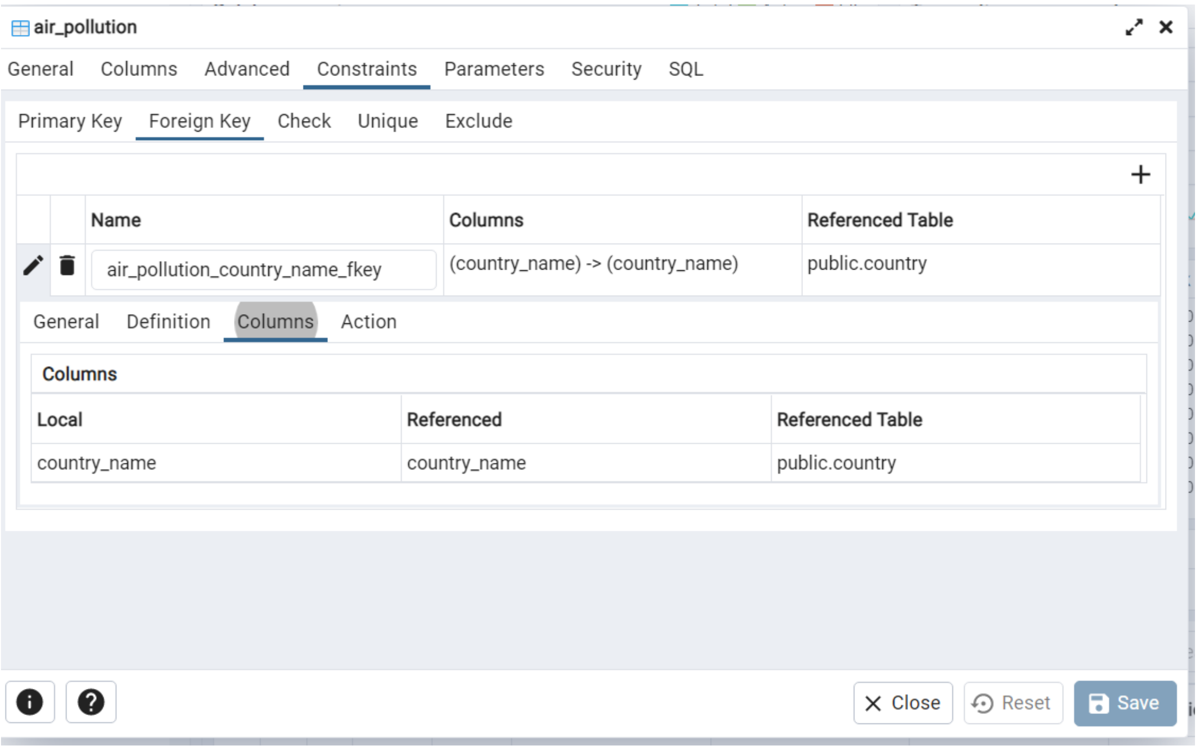


5. Repeat the step 2-4 to import the table air pollution, co2 emission, energy use per capita, share of population with access to electricity and world risk index.

6. After importing all the 6 tables, set the primary key in the country table, the primary key is country_name.



7. Then, set the foreign key of the remaining 5 tables as country_name link to primary key.



8. Then the ERD for the database will be enabled to create.