Database Applications Development Final Project

An analysis document that demonstrates the consultants have explored the technologies available in the market to perform the data extraction (PANDAS), transformation (MATPLOTLIB AND SEABORN), and loading (REDSHIFT). As the conclusion of this analysis the consultants should provide a recommendation of the suite of technologies to be used in the project (for data processing). Cloud solutions are welcome!

Data extraction:

When having to choose which technology to use when having to do the data extraction we did our research and found these different tools:

1. Python's Pandas library: Pandas is a popular open-source data manipulation and analysis library for the Python programming language. It provides high-level data structures such as DataFrames and Series, and a wide range of tools for data manipulation, cleaning, and analysis. Pandas is widely used for tasks such as data wrangling, data aggregation, and data visualization, and is optimized for working with medium-sized datasets (up to several gigabytes in size). It is widely used in data science, finance, economics, and many other fields where data analysis and manipulation is required.

2. R: R is a programming language and software environment for statistical computing and graphics. It provides a wide range of tools for data analysis, including statistical models, visualization tools, and data manipulation functions. R is widely used in academic and research settings, and has a large community of users and developers.

3. Apache Spark: Apache Spark is a distributed computing framework for processing large datasets. It provides a wide range of tools for data processing and analysis, including SQL queries, machine learning algorithms, and graph processing. Spark can be used for data manipulation and analysis on very large datasets, and can be run on a cluster of computers for improved performance.

4. SQL: Structured Query Language (SQL) is a language for managing relational databases. It provides a powerful set of tools for querying, filtering, and aggregating data, which can be used to extract data from a database before performing data analysis and manipulation.

Here are the strengths and weaknesses of Pandasm, R, Apache Spark, and SQL for data manipulation and analysis:

R:

Strengths:
- Widely used in academic and research settings
- Large community of users and developers
- Provides a wide range of tools for statistical analysis and visualization
- Has a strong focus on data manipulation and transformation

Weaknesses:
- Can have a steep learning curve for beginners
- Can be slower than other languages for very large datasets
- Limited support for parallel processing

Apache Spark:

Strengths:
- Provides a distributed computing framework for processing very large datasets
- Can be used with multiple programming languages, including Python and Scala
- Provides a wide range of tools for data processing and analysis, including machine learning algorithms and graph processing

Weaknesses:
- Can be more complex to set up and manage than other options
- Can require a larger infrastructure to support distributed computing
- May not be as suitable for smaller datasets or simpler analyses

SQL:

Strengths:
- Provides a powerful set of tools for querying, filtering, and aggregating data
- Can be used with many different types of databases
- Has a wide range of tools for data transformation and manipulation

Weaknesses:
- May require more knowledge of database architecture and management
- May be less suitable for more complex statistical analysis or modeling
- Limited support for visualizations

Pandas:

Strengths:

- Pandas provides a comprehensive set of tools for data manipulation, transformation, and cleaning. It has functions for handling missing values, filtering data, grouping and aggregating data, and pivoting data, among others. This makes it easier to transform the data from the CSV files into a format that can be used for generating graphs and visualizations.
- Pandas integrates well with other popular data analysis and visualization libraries in Python, such as Matplotlib and Seaborn. This makes it easy to generate visualizations from the transformed data and to customize and fine-tune the visualizations to meet the specific requirements of the project.
- Pandas is optimized for working with medium-sized datasets (up to several gigabytes in size). It is also optimized for working with tabular data, which is the format typically used for CSV files. This makes it a good choice for processing and transforming the weekly updates from the CSV files.
- Pandas has a large and active community of users and developers, which means there is extensive documentation and resources available for learning and using the library. This makes it easier to troubleshoot issues and get help when needed.

Weaknesses:

- While Pandas is optimized for medium-sized datasets, it can be slow for very large datasets (tens or hundreds of gigabytes in size). In such cases, a distributed computing framework like Apache Spark may be a better choice.
- Pandas has a somewhat steep learning curve for beginners, especially those who are new to Python or data manipulation. However, there are many online resources available for learning the library, and once mastered, Pandas can be a powerful tool for data manipulation and analysis.
- While Pandas can be used with multiprocessing libraries like Dask and joblib, it does not have native support for parallel processing.

After debating which technology was the best, we decided to choose Pandas because it is a versatile and powerful library and an excellent choice for building a data explorer website that generates graphs from a database. Its comprehensive range of functions and tools makes it easy to perform data transformations, clean the data, and visualise the results. It provides high-level data structures such as DataFrames and Series, and a wide range of tools for data manipulation, cleaning, and analysis. Pandas is widely used for tasks such as data wrangling, data aggregation, and data visualisation, and is optimised for working with medium-sized datasets, which is more than what we need for this project.

Data transformation:

For deciding on the technology to employ when performing the data transformation, we conducted research and discovered these several tools:

1. Matplotlib: A Python library for creating static, publication-quality plots such as line charts, scatter plots, and histograms.

2. Seaborn: A Python library built on top of Matplotlib that provides additional data visualization capabilities such as statistical graphics, heatmap, and cluster maps.

3. Plotly: An interactive, web-based data visualization library for Python, R, and JavaScript that provides a wide range of visualization types, including 3D plots, maps, and more.

4. Bokeh: An interactive Python visualization library that allows users to create interactive and responsive web-based visualizations and dashboards. It provides a range of tools for building interactive charts, such as line charts, scatter plots, and heatmaps.

5. D3.js: A JavaScript library that allows users to create dynamic and interactive data visualizations on the web. It provides advanced capabilities for creating custom visualizations and animations using SVG graphics.

Overall, these libraries provide a range of tools and capabilities for creating data visualizations, with varying levels of complexity and interactivity. The choice of library would depend on the specific requirements of the project, such as the types of visualizations needed and the level of interactivity required.

Here are some of the strengths and weaknesses of the different technologies for generating the graphs needed for the website's project:

Plotly:

Strengths:

- Plotly offers interactive visualizations that can be manipulated and explored directly from the graphs, which can be useful for enhancing user engagement.
- It also provides a cloud-based platform for hosting and sharing visualizations.
- Plotly has a simple and easy-to-use interface, making it a great choice for users with minimal programming experience.

Weaknesses:

- Limited customization options for free-tier users
- Can be expensive for commercial use
- May require more code to create simple visualizations

Bokeh:

Strengths:

- Bokeh provides interactive visualizations that can be customized and embedded in web applications.
- It is designed to work well with large datasets and provides advanced features such as streaming and real-time data updates.
- Bokeh integrates well with other Python libraries such as Pandas and NumPy.

Weaknesses:

- Limited customization options for non-JavaScript users
- Smaller community compared to other libraries
- Limited chart types compared to other libraries
- May have a steeper learning curve compared to other libraries, and may require more time and effort to implement.
- It may also be more resource-intensive compared to other libraries, which can impact performance.

D3.js:

Strengths:

- D3.js provides advanced visualization capabilities and is highly customizable.
- It is designed to work with web-based data visualization applications and provides a wide range of visualization options, including interactive and animated visualizations.
- D3.js has an active community of developers and a large library of examples and documentation.

Weaknesses:

- D3.js has a steeper learning curve and requires advanced programming skills to implement.
- Steep learning curve compared to other libraries
- Requires knowledge of JavaScript and HTML/CSS
- Not optimized for working with small datasets

Matplotlib:

Strengths:

- Offers a wide range of chart types and customization options
- Large and active community with a lot of online resources and examples
- Integration with other Python libraries like Pandas is seamless

Weaknesses:

- Default styles can be basic and require additional customization for publication-quality visuals
- Some of the more complex visualizations can require a lot of code
- Limited interactivity compared to other libraries like Plotly and Bokeh

Seaborn:

Strengths:

- Offers a simplified API that builds on top of Matplotlib
- Focuses on statistical visualization, making it ideal for data exploration
- Good for creating more complex visualizations with minimal code

Weaknesses:

- Limited customization options compared to Matplotlib
- Limited chart types compared to Matplotlib
- May not be ideal for creating highly custom visuals

As a team, we decided that using both Seaborn and Matplotlib together is the best option for this project because Seaborn is built on top of Matplotlib and provides a simplified interface for creating statistical visualizations. While Seaborn provides a higher-level interface with better defaults and more advanced statistical plot types, Matplotlib allows for more customization and flexibility in creating different types of visualizations.

By combining the two libraries, the project will benefit from the strengths of both libraries, allowing for a wide range of plot types, customization options, and statistical visualizations. Matplotlib can be used to create custom visualizations and tweak the plot aesthetics while Seaborn can be used to create high-level visualizations with fewer lines of code. Moreover, both libraries offer good integration with Pandas, making it easier to handle the data and create visualizations from it.

Data loading:

When investigating which technology to use when performing the data transformation we found the following tools that might help us achieve our goals on the project:

1. MySQL: A popular open-source relational database management system that can be used to store and manage structured data. MySQL is widely used and has a large community of developers, which means that it is well-documented and has a wide range of tools and resources available.

2. PostgreSQL: Another open-source relational database management system that is known for its robustness, scalability, and advanced features such as JSON support and geospatial data management.

3. Apache Cassandra: A distributed NoSQL database that is designed for high availability and scalability. Cassandra is a good choice for handling large volumes of semi-structured or unstructured data and offers fast write performance.

4. Apache HBase: An open-source distributed key-value database that is built on top of the Hadoop Distributed File System (HDFS). HBase is optimized for real-time read/write access to large datasets and can handle both structured and unstructured data.

5. AWS Redshift: a cloud-based data warehousing service provided by Amazon Web Services (AWS). It is designed for handling large-scale data analytics workloads and is based on a columnar storage format. It can be used to store and analyze structured and semi-structured data, including relational data, JSON, and AVRO. Redshift also integrates with other AWS services, such as S3, EC2, and IAM, and supports a wide range of third-party tools and connectors for data integration and visualization.

Here are some strengths and weaknesses of the alternative technologies I mentioned earlier:

MySQL:

Strengths:

- Widely used and well-documented
- Good performance for smaller datasets
- Supports a variety of data types
- Relatively easy to set up and configure.

Weaknesses:

- Can struggle with scalability for larger datasets
- Limited support for unstructured data
- May require additional tools or plugins for advanced analytics and visualization.

PostgreSQL:

Strengths:

- Robust and scalable
- Supports a wide range of data types
- Has advanced features such as geospatial data management and JSON support
- Can handle both structured and unstructured data.

Weaknesses:

- Can be more complex to set up and configure than other options
- Requires more technical expertise to optimize performance and scalability.

Apache Cassandra:

Strengths:

- Highly scalable and fault-tolerant
- Designed for real-time read/write access to large datasets
- Good performance for write-heavy workloads
- Supports flexible data modeling.

Weaknesses:

- Can be more complex to set up and manage than other options
- May require additional tools or expertise to perform complex queries and analytics.

Apache HBase:

Strengths:

- Optimized for real-time read/write access to large
- Can handle both structured and unstructured data
- Offers good performance for range queries and scan operations.

Weaknesses

- Can be complex to set up and manage
- May require a dedicated Hadoop cluster
- Limited support for SQL-based queries.

AWS Redshift:

Strengths:

- Can easily handle large and growing datasets, making it a good fit for data that is being updated weekly.
- Optimized for running complex queries on large datasets, making it a good fit for generating graphs and other visualizations.
- The pay-as-you-go pricing model allows for cost savings, as users only pay for the storage and computing resources they use.

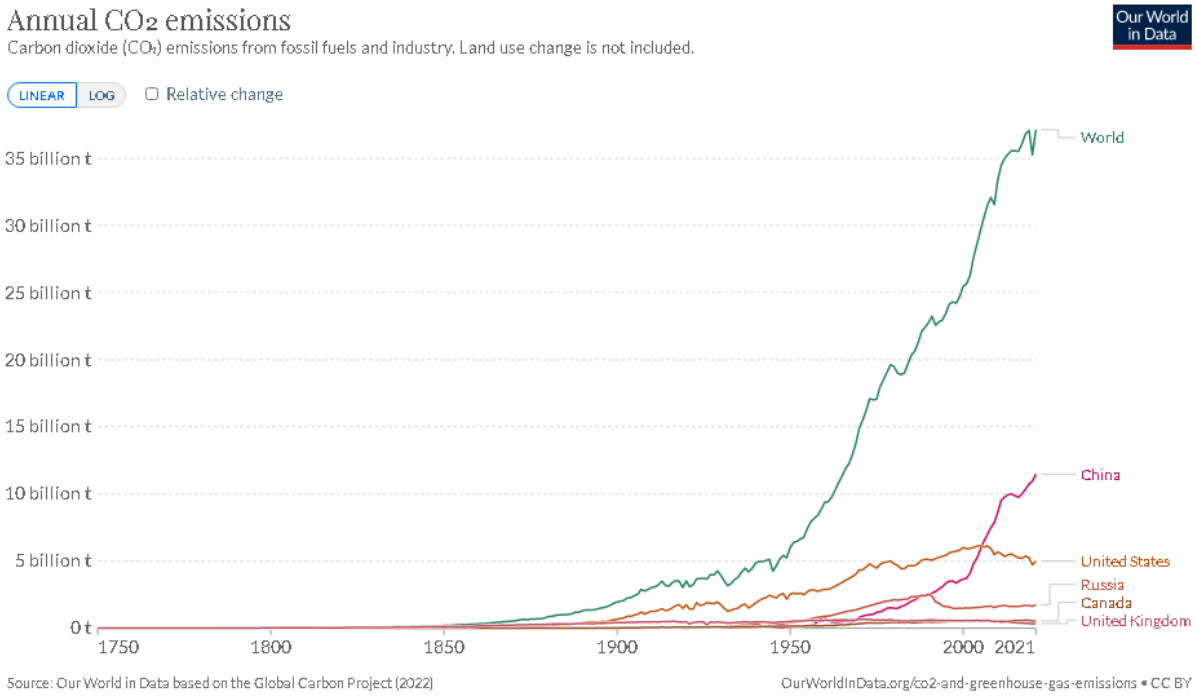
Weaknesses:

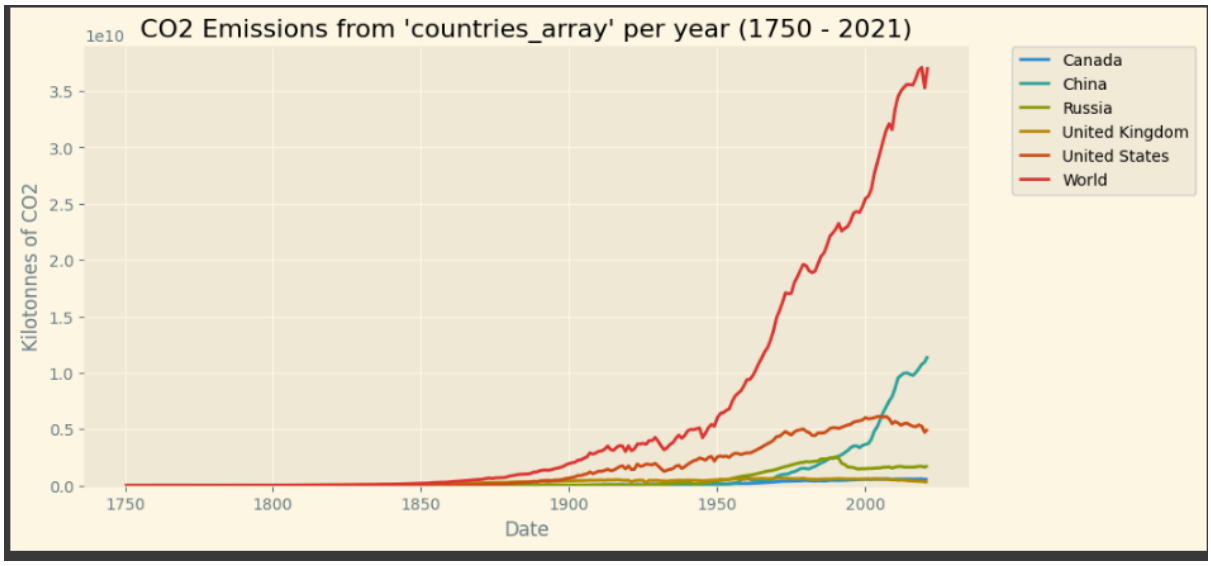
- Setting up and configuring can be complex and time-consuming, requiring significant technical expertise.
- Uses a columnar data model, which may not be the best fit for all types of data.
- Depending on the source of the CSV files, additional ETL (extract, transform, load) tools may be required to move the data into AWS Redshift.

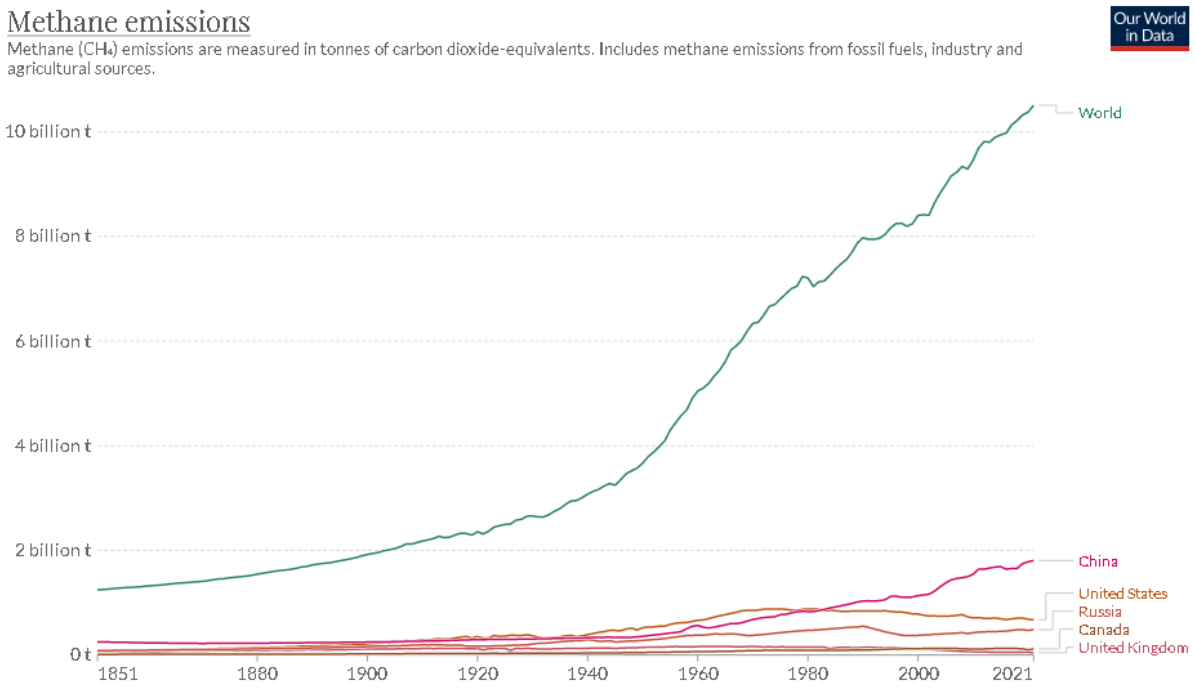
After discussing within the team we decided to choose AWS Redshift for the following reasons:

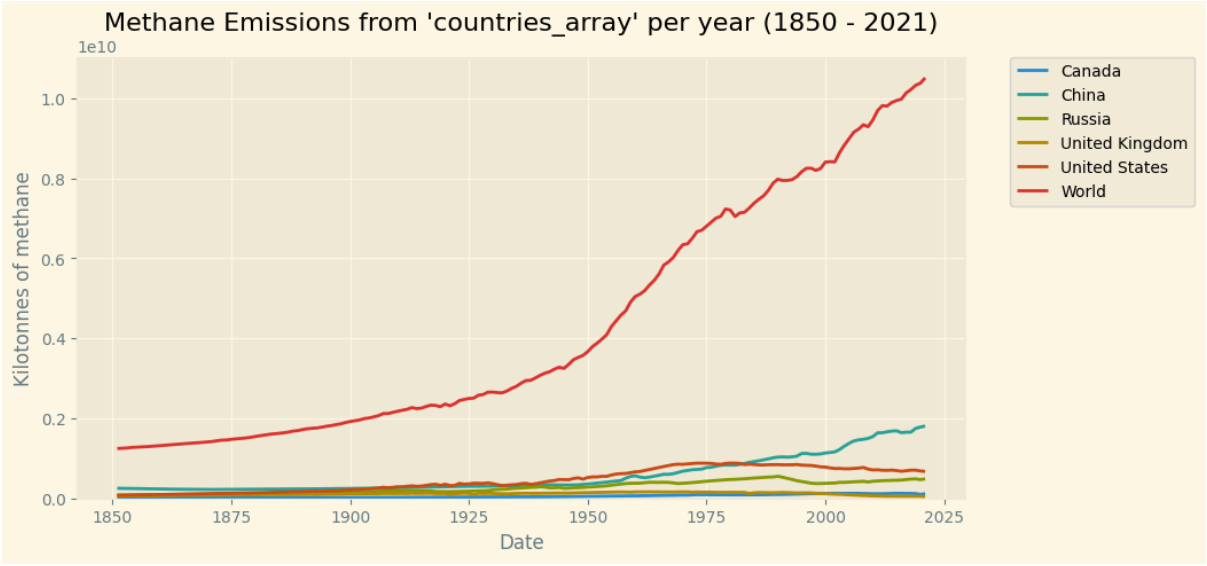
- It is highly scalable and can handle large datasets with ease. It allows for easy scaling up or down, making it a good choice for projects that require flexibility and agility.
- Redshift is also optimized for high performance, with advanced features such as columnar storage and parallel processing. This makes it a good choice for projects that require fast querying and analysis of large datasets.
- In addition, this platform integrates seamlessly with other AWS services, such as S3 and EC2. This allows for easy data ingestion and analysis, as well as cost-effective storage and computing options.
- Finally, Redshift provides advanced security features, such as encryption and access control, making it a good choice for projects that require robust data protection.

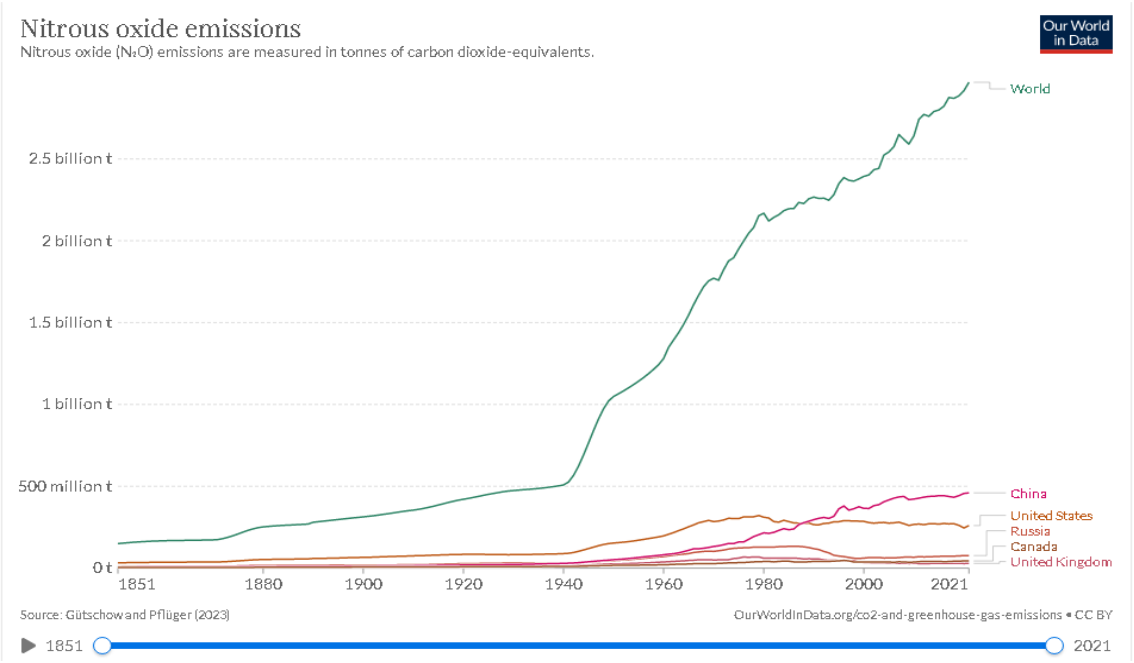
GRAPHS COMPARAISON

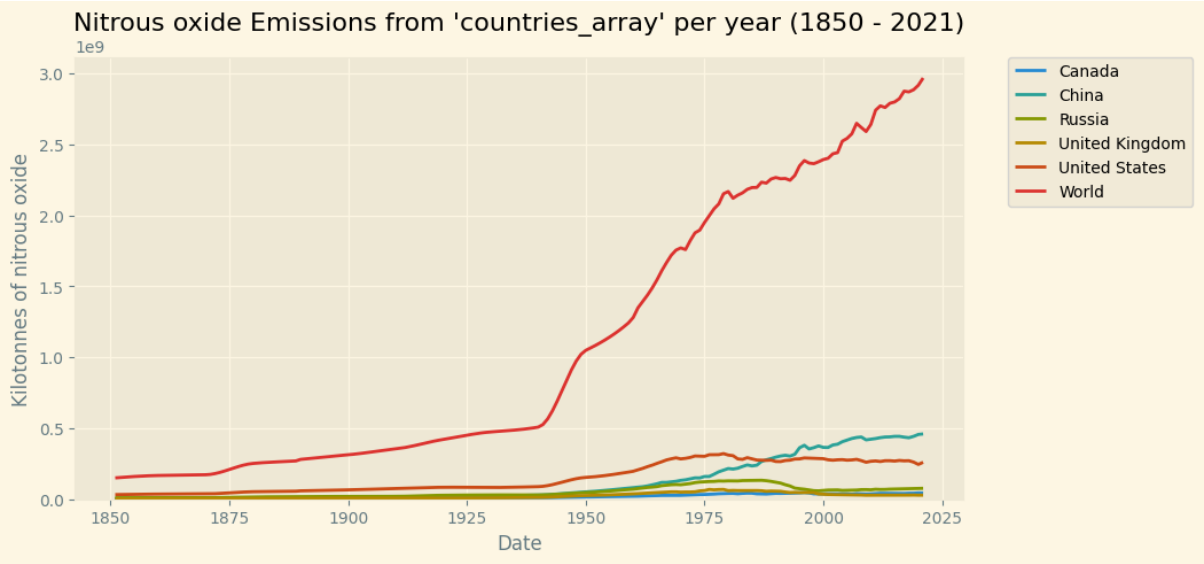


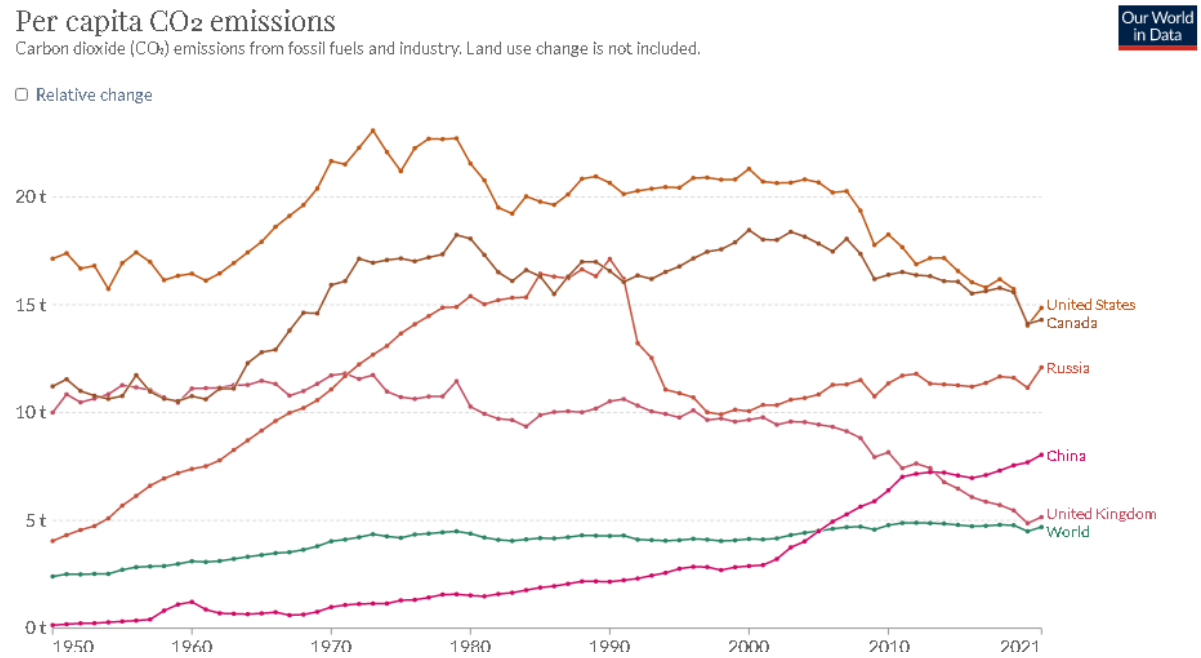


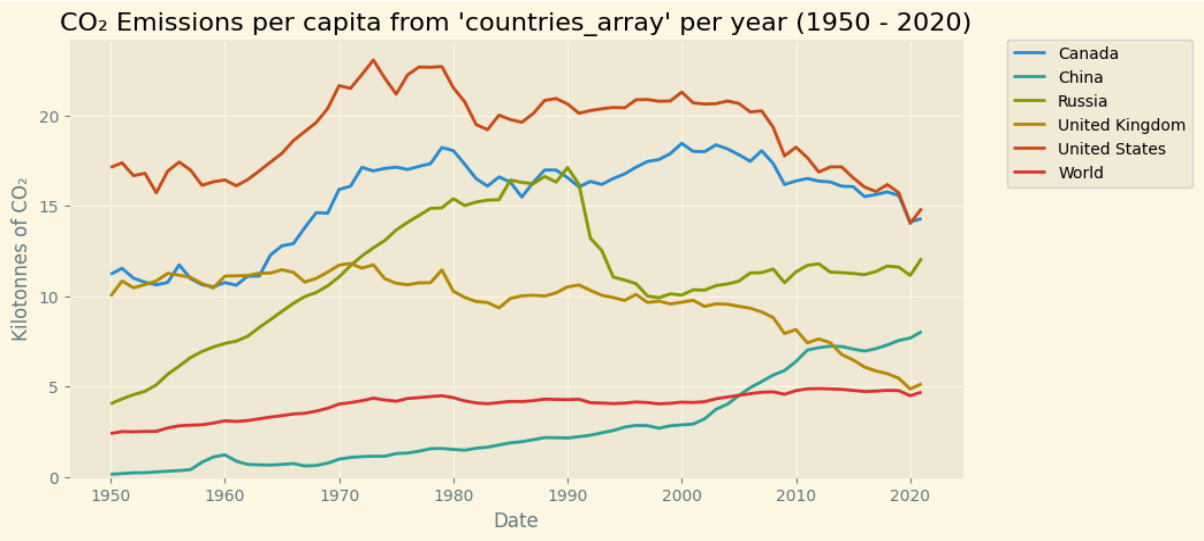

Methane emissions
Methane (CH_4) emissions are measured in tonnes of carbon dioxide-equivalents. Includes methane emissions from fossil fuels, industry and agricultural sources.
Our World in Data
10 billion t
8 billion t
6 billion t
4 billion t
2 billion t
0 t
1851
1880
1900
1920
1940
1960
1980
2000
2021
World
China
United States
Russia
Canada
United Kingdom

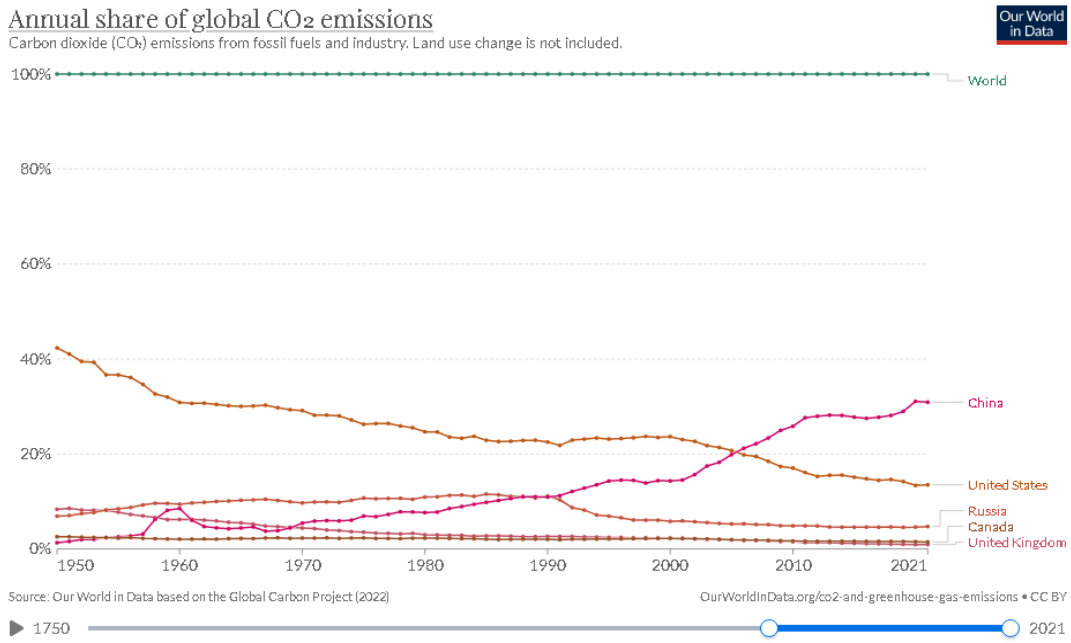

Methane Emissions from 'countries_array' per year (1850 - 2021)
1e10
1.0
0.8
0.6
0.4
0.2
0.0
Kilotonnes of methane
1850
1875
1900
1925
1950
1975
2000
2025
Date
Canada
China
Russia
United Kingdom
United States
World

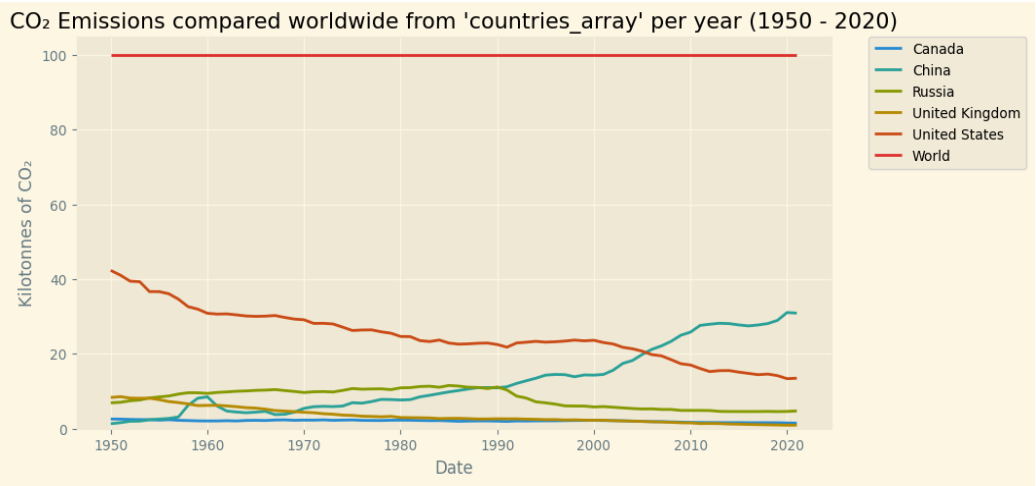

Nitrous oxide emissions
Nitrous oxide (N₂O) emissions are measured in tonnes of carbon dioxide-equivalents.
Our World in Data
World
2.5 billion t
2 billion t
1.5 billion t
1 billion t
500 million t
0 t
China
United States
Russia
Canada
United Kingdom
1851
1880
1900
1920
1940
1960
1980
2000
2021
Source: Gütschow and Pflüger (2023)
OurWorldInData.org/co2-and-greenhouse-gas-emissions • CC BY
1851
2021

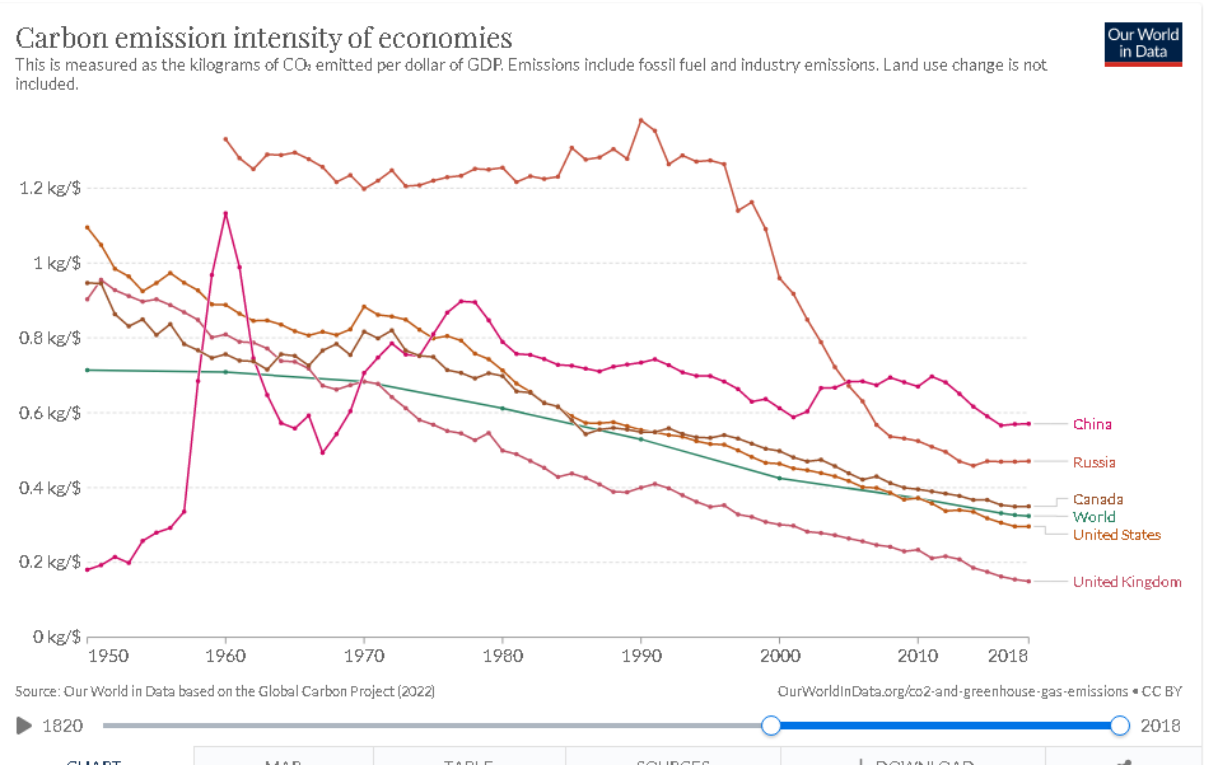

Nitrous oxide Emissions from 'countries_array' per year (1850 - 2021)
1e9
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Kilotonnes of nitrous oxide
1850
1875
1900
1925
1950
1975
2000
2025
Date
Canada
China
Russia
United Kingdom
United States
World

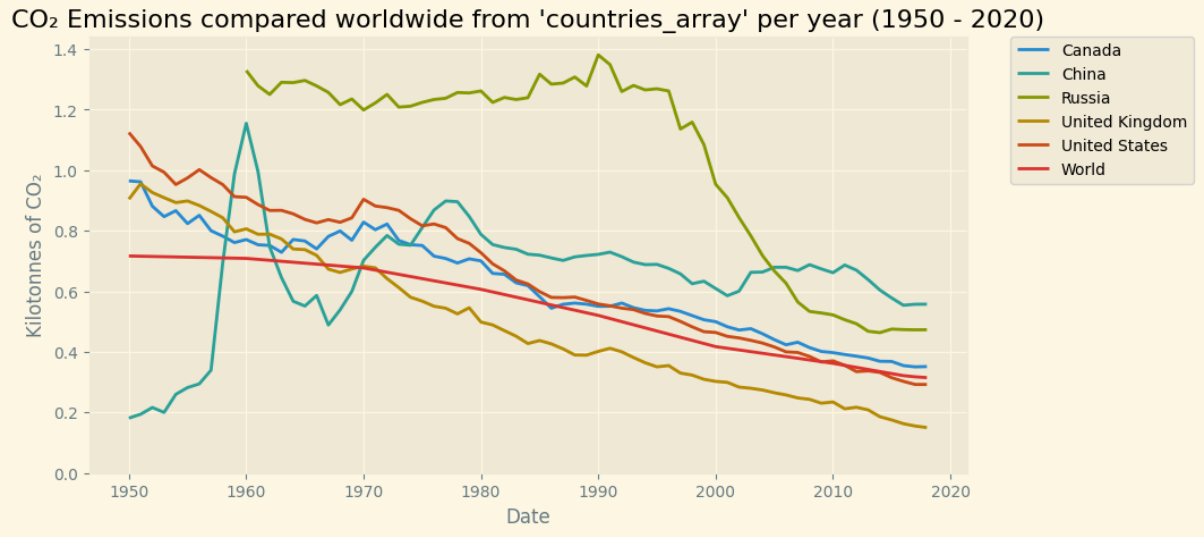

Per capita CO₂ emissions
Carbon dioxide (CO₂) emissions from fossil fuels and industry. Land use change is not included.
Our World in Data
Relative change
20 t
15 t
10 t
5 t
0 t
1950
1960
1970
1980
1990
2000
2010
2021
United States
Canada
Russia
China
United Kingdom
World


CO₂ Emissions per capita from 'countries_array' per year (1950 - 2020)
Kilotonnes of CO₂
0
5
10
15
20
1950
1960
1970
1980
1990
2000
2010
2020
Date
Canada
China
Russia
United Kingdom
United States
World


Annual share of global CO₂ emissions
Carbon dioxide (CO₂) emissions from fossil fuels and industry. Land use change is not included.
Our World in Data
100%
80%
60%
40%
20%
0%
1950
1960
1970
1980
1990
2000
2010
2021
World
China
United States
Russia
Canada
United Kingdom
Source: Our World in Data based on the Global Carbon Project (2022)
OurWorldInData.org/co2-and-greenhouse-gas-emissions • CC BY
1750
2021


CO₂ Emissions compared worldwide from 'countries_array' per year (1950 - 2020)
Kilotonnes of CO₂
100
80
60
40
20
0
1950
1960
1970
1980
1990
2000
2010
2020
Date
Canada
China
Russia
United Kingdom
United States
World


Carbon emission intensity of economies
This is measured as the kilograms of CO₂ emitted per dollar of GDP. Emissions include fossil fuel and industry emissions. Land use change is not included.
Our World in Data
1.2 kg/$
1 kg/$
0.8 kg/$
0.6 kg/$
0.4 kg/$
0.2 kg/$
0 kg/$
1950
1960
1970
1980
1990
2000
2010
2018
China
Russia
Canada
World
United States
United Kingdom
Source: Our World in Data based on the Global Carbon Project (2022)
OurWorldInData.org/co2-and-greenhouse-gas-emissions • CC BY
1820
2018


CO₂ Emissions compared worldwide from 'countries_array' per year (1950 - 2020)
Kilotonnes of CO₂
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1950
1960
1970
1980
1990
2000
2010
2020
Date
Canada
China
Russia
United Kingdom
United States
World