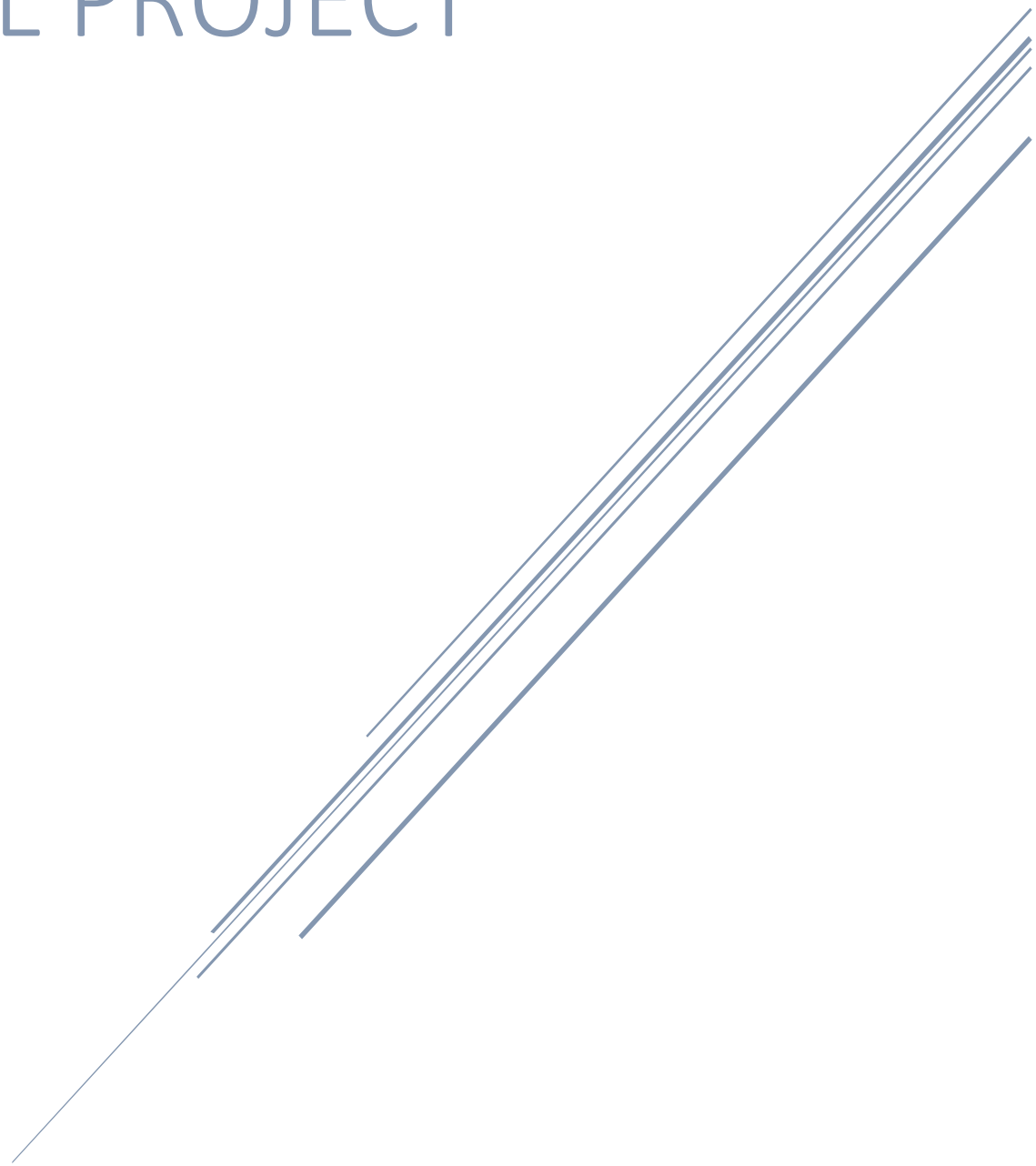
ETL PROJECT



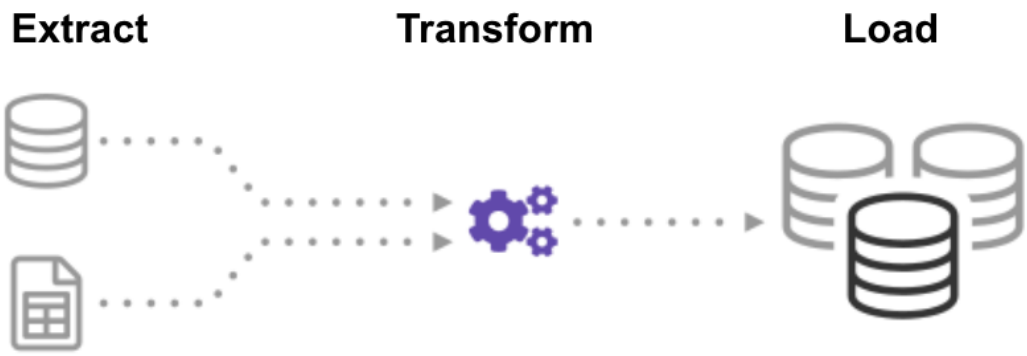
Written by Neena Mani & Mathew Johnson

Table of Contents

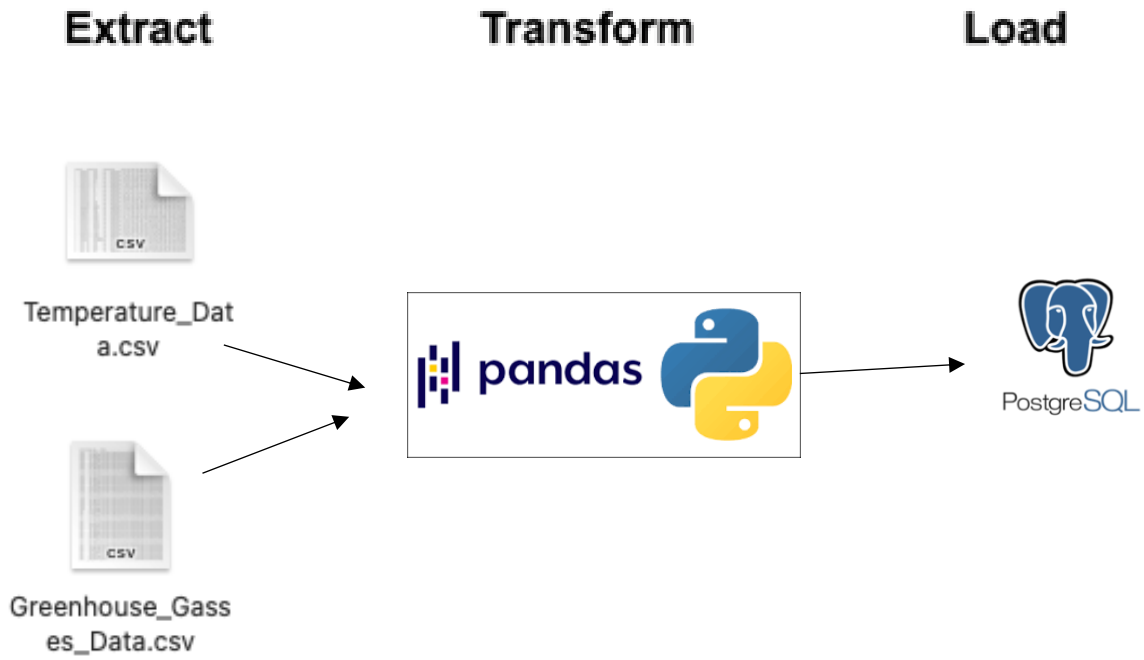
Executive Summary

The purpose of this report is to outline the Extraction, Transformation & Loading (ETL) process undertaken for the analysis of global co2 emissions & temperature changes.

ETL is defined as the process in which multiple sources of data are collated and integrated into a single, consistent database which can be used for data analytics.



Specifically, in this analysis, the below ETL model was followed. The Extract phase involved accessing data from CSV files, the data was cleaned in the Transform phase using the Pandas library and finally the data was loaded into a PostgresSQL database.



Extraction

A number of different sources of data were explored to extract the data for this analysis, such as the Australian Bureau of Statistics (https://www.abs.gov.au), Our World in Data (https://ourworldindata.org), Kaggle (www.kaggle.com) and The World Bank (https://data.worldbank.org).

From the sources considered, the two data sets used in the analysis were taken from Kaggle. This was due to the fact Kaggle offered large & freely accessible databases relevant to the analysis, whereas the other sources contained incomplete or missing data which would have created issues during the transformation phase.

Data Files

From Kaggle, the relevant data was acquired to conduct the analysis. The first file (Temperature Data) contained the temperature changes recorded in countries around the world from the period 1961 – 2019. The second file (Greenhouse Gas Emissions Data) recorded data of co2 emissions from different countries around the world from 1990 - 2014.

Data File Types

Both data sources used in the analysis were CSV files. Although alternatives were considered, mainly sourcing data from an API and converting the data into JSON format, the CSV files contained the relevant information required for the analysis.

Transformation

The aim of the Transformation phase was to explore, clean & merge the data into a single database relevant to conducting the analysis.

Data Exploration

The data set was explored to establish the columns required to conduct the analysis and to understand the relationships that existed between the two data sources.

During the exploration phase, some issues were identified regarding the formatting & structure of each data file. These inconsistencies were rectified and they are elaborated upon in the data cleaning section of this report.

Data Cleaning

Once the data had been identified which would be used in the analysis, the files were loaded into Jupyter Notebooks and transformed using the Pandas library.

Data Source 1 – Temperature Data

To load the Temperature Data file into Jupyter Notebooks, the file encoding had to be passed as ''iso-8859-1', otherwise the file would not open due to the inclusion of special characters within the data.

There were several columns included within this data set which were not required for the analysis. As such, the file needed to be filtered so the data which was required could be extracted.

This was achieved through df.drop function, and where columns could be deleted sequentially, a range of columns were deleted in one command through the use of df.drop and referencing the starting & ending column index.

Columns which were not required included ‘Area Code’, ‘Months Code’ and ‘Element Code’, as they did not contain any valuable data applicable for this analysis. These columns were deleted individually through referencing the column header.

Area Code	Area	Months Code	Months	Element Cod
2	Afghanistan	7001	January	7271
2	Afghanistan	7001	January	6078
2	Afghanistan	7002	February	7271
2	Afghanistan	7002	February	6078
2	Afghanistan	7003	March	7271
2	Afghanistan	7003	March	6078
2	Afghanistan	7004	April	7271
2	Afghanistan	7004	April	6078
2	Afghanistan	7005	May	7271
2	Afghanistan	7005	May	6078
2	Afghanistan	7006	June	7271
2	Afghanistan	7006	June	6078
2	Afghanistan	7007	July	7271
2	Afghanistan	7007	July	6078
2	Afghanistan	7008	August	7271
2	Afghanistan	7008	August	6078
2	Afghanistan	7009	September	7271

Furthermore, the file included data from 1961 – 2019 which was a greater time period than the one used in this analysis. As a result, several columns were dropped from the data in one command, by passing the column index range in which they were located.

Code:
df.drop(df.columns[7:36], axis=1, inplace=True)

This enabled several columns containing data outside of the date range used in this analysis to be removed from the dataset efficiently.

During the data exploration phase, it was identified that all rows with an Area Code greater than 5000 would not be required for this analysis as they related to continents rather than countries.

The Months column had data listed in monthly, quarterly & annual time periods. For this analysis, the relevant timeframe was annual, so all records which related to different timeframes were removed from the data set.

Within the Element column, the actual temperature change was recorded, in addition to the standard deviation of temperature changes. The standard deviation data was not required in this analysis and therefore removed from the data set through the df.loc methon.

Code:
temp_df.loc[(temp_df['Area Code'] < 5000) & (temp_df['Months'] == 'Meteorological year') & (temp_df['Element'] == 'Temperature change'), :]

	A	B	C	D	E	F	G
1	Area Code	Area	Months Co	Months	Element C	Element	Unit
8400	5000	World	7001	January	7271	Temperature change	∞C
8401	5000	World	7001	January	6078	Standard Deviation	∞C
8402	5000	World	7002	February	7271	Temperature change	∞C
8403	5000	World	7002	February	6078	Standard Deviation	∞C
8404	5000	World	7003	March	7271	Temperature change	∞C
8405	5000	World	7003	March	6078	Standard Deviation	∞C
8406	5000	World	7004	April	7271	Temperature change	∞C
8407	5000	World	7004	April	6078	Standard Deviation	∞C
8408	5000	World	7005	May	7271	Temperature change	∞C
8409	5000	World	7005	May	6078	Standard Deviation	∞C
8410	5000	World	7006	June	7271	Temperature change	∞C
8411	5000	World	7006	June	6078	Standard Deviation	∞C

The Temperature Data file contained the year data in different formatting when compared to the secondary data file used in this analysis. Within the Temperature Data file, the years were listed as columns, whereas the Greenhouse Gases Data file listed the years as rows. Additionally, the Temperature Data file included the letter ‘Y’ prior to each year.

Temperature Data

	H	I	J	K	L	M	N	O	
	Y1961	Y1962	Y1963	Y1964	Y1965	Y1966	Y1967	Y1968	Y1969
	0.399	0.104	-0.171	-0.311	0.206	0.458	-0.289	-0.393	

Greenhouse Gases Data

A	B
country_or_a	year
Australia	2014
Australia	2013
Australia	2012
Australia	2011
Australia	2010
Australia	2009
Australia	2008

To overcome this challenge, df.melt was used to flip the year column to rows which meant the data was compatible with the secondary data source. Furthermore, in order to remove the 'Y' from the year in the Temperature Data file, str.strip was used.

Code (df.melt):
temp_filter_df.melt(id_vars=["Area", "Element", "Unit"], var_name="Year", value_name="Temp")

Code (str.strip):
temp_flip_df['Year'].str.strip('Y')

The column headers were renamed so they would correspond with the Postgres database tables and an index column was created so the data set had a primary key when loaded into the database.

	index	country	element	unit	year	temp
0	0	Afghanistan	Temperature change	°C	1990	0.766
1	1	Albania	Temperature change	°C	1990	0.846
2	2	Algeria	Temperature change	°C	1990	1.314
3	3	American Samoa	Temperature change	°C	1990	0.520
4	4	Andorra	Temperature change	°C	1990	1.734
...	...	...	...	...	...	...
5616	5616	Wake Island	Temperature change	°C	2014	1.002
5617	5617	Wallis and Futuna Islands	Temperature change	°C	2014	0.903
5618	5618	Western Sahara	Temperature change	°C	2014	1.341
5619	5619	Zambia	Temperature change	°C	2014	0.967
5620	5620	Zimbabwe	Temperature change	°C	2014	0.291

Data Source 2 – Greenhouse Gases Data

The ‘Category’ column was deleted from the data set as it did not contain any information required to conduct the analysis.

The columns were also renamed to ensure they matched the column names of the tables that had been created in the Postgres database.

Within the dataset, the co2 emissions data was rounded to 3 decimal places to ensure the data was easy to read.

Similar to the Temperature Data file, an index column was included in the Greenhouse Gases data, so the data set had a primary key when loaded into the Postgres database.

	country	year	co2_emissions
id			
0	Australia	2014	393126.947
1	Australia	2013	396913.937
2	Australia	2012	406462.848
3	Australia	2011	403705.528
4	Australia	2010	406200.993

Loading

Initially, two tables were created in Postgres with column names corresponding to the two cleaned data sets.

```
CREATE TABLE temp (
  id INT PRIMARY KEY,
  country TEXT,
  element TEXT,
  unit TEXT,
  year TEXT,
  temp FlOAT
);
```

```
CREATE TABLE emission (
  id INT PRIMARY KEY,
  country TEXT,
  year TEXT,
  co2_emissions FLOAT
```

Once the tables had been created, the data was loaded into the corresponding tables.

Code (Temperature File):
*final_temp_df_new1.to_sql(name='temp', con=engine, if_exists='append', index=**True**)*

Code (Greenhouse Gases File):
*emission_filter_df1.to_sql(name='emission', con=engine, if_exists='append', index=**True**)*

This was validated by using the Select * function to display the data contained within each table.

Temperature Data

	id	country	element	unit	year	temp
0	0	Afghanistan	Temperature change	°C	1990	0.766
1	1	Albania	Temperature change	°C	1990	0.846
2	2	Algeria	Temperature change	°C	1990	1.314
3	3	American Samoa	Temperature change	°C	1990	0.520
4	4	Andorra	Temperature change	°C	1990	1.734

Emissions Data

	id	country	year	co2_emissions
0	0	Australia	2014	393126.947
1	1	Australia	2013	396913.937
2	2	Australia	2012	406462.848
3	3	Australia	2011	403705.528
4	4	Australia	2010	406200.993

Once it was verified that the tables created in the Postgres database contained the relevant information from each respective data source, the two tables could be merged to leave one single database.

etl_greenhouse/postgres@PostgreSQL 13

Query Editor | Query History

```
1  -- Joins tables
2  SELECT e.country, e.co2_emissions, t.temp
3  FROM emission AS e
4  LEFT JOIN temp AS t
5  ON e.country = t.country
6  AND e.year = t.year;
```

Data Output | Explain | Messages | Notifications

	country text	co2_emissions double precision	temp double precision
1	Australia	393126.947	1.139
2	Australia	396913.937	1.45
3	Australia	406462.848	0.273
4	Australia	403705.528	0.176
5	Australia	406200.993	0.644
6	Australia	408448.479	0.968
7	Australia	404237.828	0.553
8	Australia	398816.454	0.864
9	Australia	391134.101	0.643