**\*\*E\*\*xtract:** your original data sources and how the data was formatted (CSV, JSON, pgAdmin 4, etc).

Data sources:

1. European Backpackers Index 2018.xlsx
   1. This data came from <https://data.world/> as a excel file
   2. Has 13 columns and 56 rows filled with the 56 cheapest places to travel in Europe
   3. This dataset included the following columns:

Best cheap hostel

Hostel URL

City

Country

Local Currency

Local Currency Code

Rank

Total USD

Attractions (USD)

Cost Per Night (Hostel) (USD)

Drinks Entertainment (USD)

Meals (USD)

Transportation (USD

1. Country Travel Information.json
   1. This data came from <https://data.world/> as a json file
   2. Has 11 columns and 212 rows of travel information of European countries
   3. This dataset included the following columns:

Tag

Geopoliticalarea

Travel\_transportation

Health

Local\_laws\_and\_special\_circumsatance

Safety\_and\_security

Entry\_exit\_requirements

Destination\_description

Iso\_code

Travel\_embassyandconsulate

last\_update\_date

1. Open Weather Map API <https://openweathermap.org/api>
   1. We were able to use the API on this website to pull code directly from the website
   2. The data that came from this pull was the longitude and latitude points of the cities that had common countries between Country Travel Information.json and European Backpackers Index 2018.xlsx

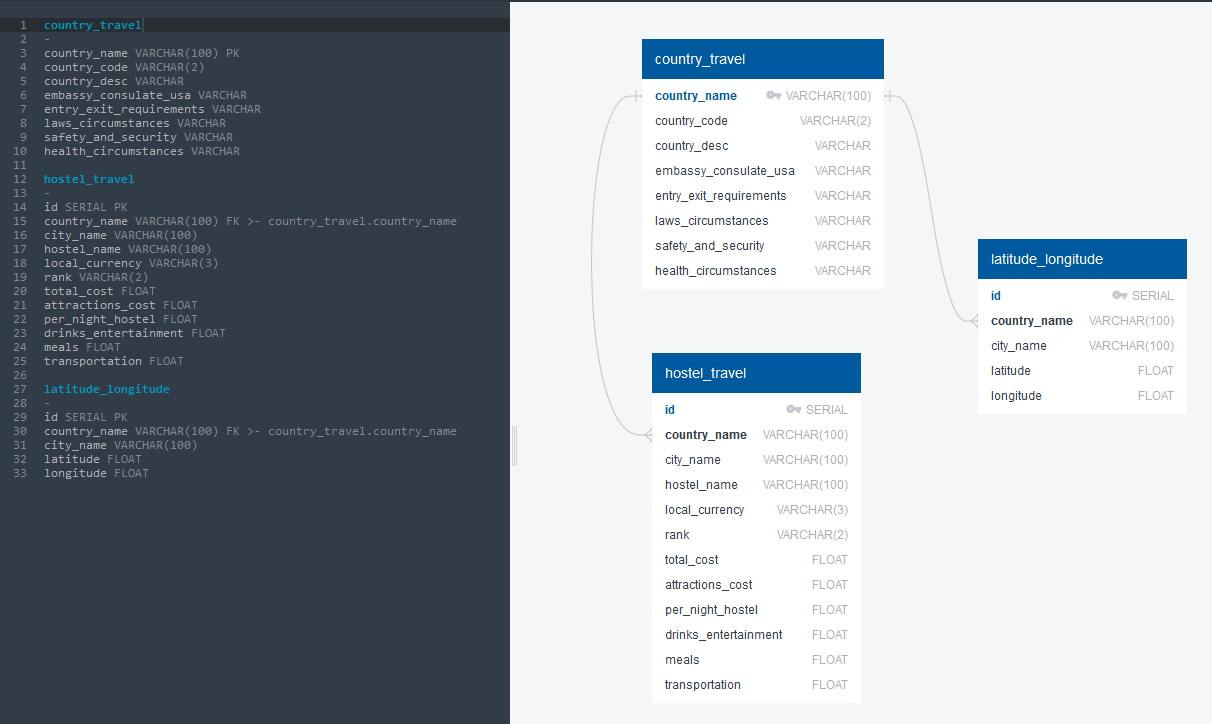
**\*\*T\*\*ransform:** what data cleaning or transformation was required.

1. European Backpackers Index 2018.xlsx
   * Imported the excel file as a dataframe into a jupyter notebook called europe\_hostels.ipynb
   * Choose and renamed the columns we wanted from our data set
   * Made sure we weren’t missing any data values in the new dataframe
   * Created a connection url to connect to our Postgres DB
   * Read all of our data into our hostel\_travel table in our database
2. Country Travel Information.json
   * Imported the json file as a dataframe into a jupyter notebook called csi.ipynb
   * Choose and renamed the columns we wanted from our data set
   * Used Beautiful soup to get rid of any html syntax from the entries in our data set
   * Used the replace function to turn ‘\xa0’ to ‘’
   * Made sure we weren’t missing any data values in the new dataframe
   * Created a connection url to connect to our Postgres DB
   * Read all of our data into our country\_travel table in our database
3. Open Weather Map API <https://openweathermap.org/api>
   * Created a url to be able to pull the data from the api
   * Created a connection url to connect our Postgres DB, so we could connect to our existing tables and data
   * Queried the country names from both tables and city name from hostel\_trabel then completed a join with the country names from the hostel\_travel and country\_travel
   * Used the queried data to grab the longitude and latitude of each city, country
   * Read all of our data into our latitude\_longitude table in our database
4. Special Notes:
   * Removed countries that aren’t present in Country Travel Information.json since country\_name in hostel\_table is a foriegn key on our country\_travel table

**\*\*L\*\*oad:** the final database, tables/collections, and why this was chosen.

The final database we chose for our project was a relational database in pgAdmin because our ideal end goal for this project is to create an interactive map on a website that can display the data in a user friendly way. To do this we need a relational database to be able to connect our different data sources with each other using common values and primary keys. With our relational database we can easily connect tables with each other to get more data needed for someone who is planning on traveling to Europe (i.e. weather in that area, cities nearby, holidays and festivals info, etc)

This database had 3 tables of data:



1. country\_travel (NOT NULL)
   1. This table included the following columns:

country\_name VARCHAR [PK]

country\_code VARCHAR

embassy\_consulate\_usa VARCHAR

entry\_exit\_requirements VARCHAR

laws\_circumstances VARCHAR

safety\_and\_security VARCHAR

health\_circumstances VARCHAR

1. hostel\_travel (NOT NULL)
   1. This table included the following columns:

id SERIAL [PK]

country\_name VARCHAR

city\_name VARCHAR

hostel\_name VARCHAR

local\_currency VARCHAR

rank VARCHAR

total\_cost FLOAT

attractions\_cost FLOAT

per\_night\_hostel FLOAT

drinks\_entertainment FLOAT

meals FLOAT

transportation FLOAT

1. latitude\_longitude (NOT NULL)
   1. This table included the following columns:

id SERIAL [PK]

country\_name VARCHAR

city\_name VARCHAR

latitude FLOAT

longitude FLOAT

**FINAL RESULTS:**

In an attempt to create an interactive website for users to choose a city to go to we were only able to create a final query. This query combined all of our data based on country all of our data together using a bunch of joins.

