Bosch Assesment

Data Engineer

Henrique Silva

Main Objectives

1. **Extract**, **clean**, **transform**, and **load** data from public sources related to the automotive industry;

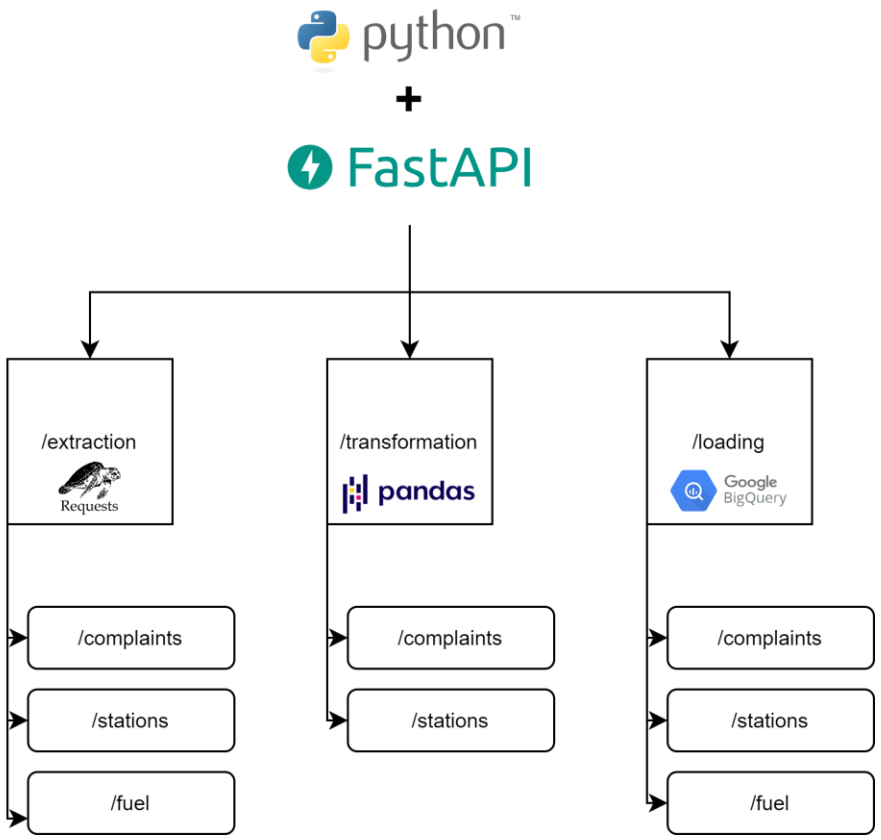
1. Suggest an automation for the data pipeline with a chosen schedule interval;

2. Document all decisions made during the data processing and transformation steps;

3. Address any challenges faced during the exercise.

Index

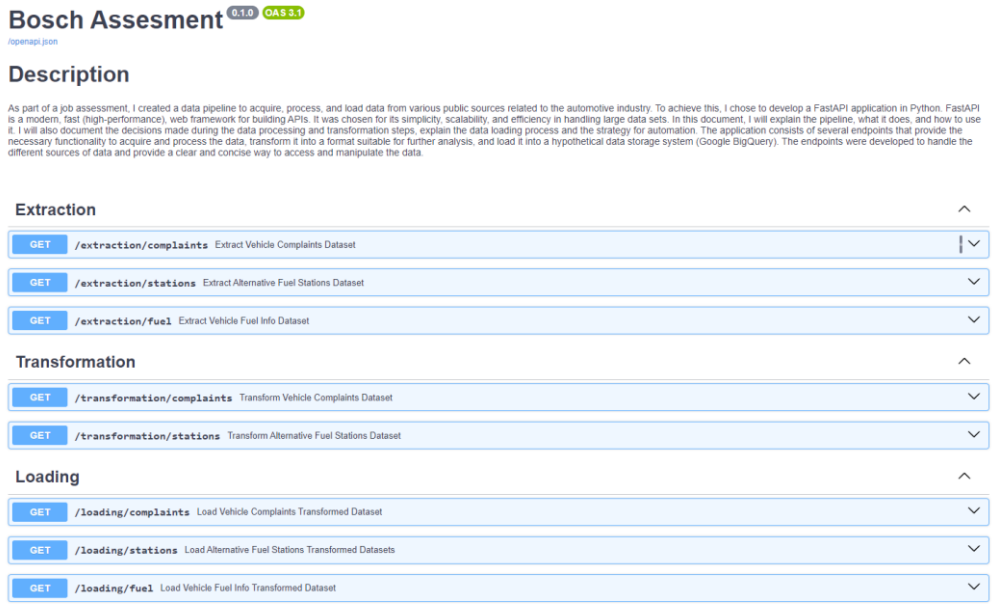
1 - Script Developed



Project Infrastructure

How To Launch:

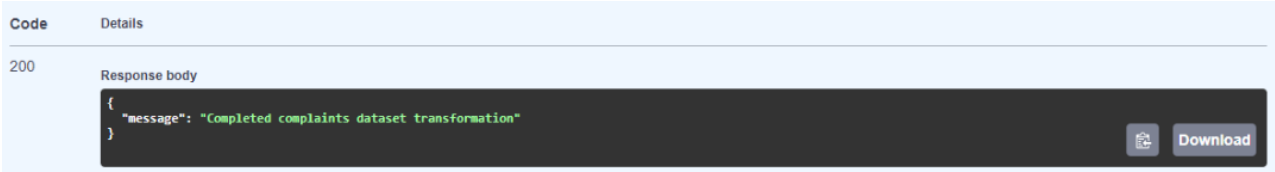
1. Clone GitHub repository;
2. Install ‘requirements.txt’ libs;
3. Open terminal inside ‘app/’ and type: ‘uvicorn main:app --reload’;
4. Access documentation on browser using http://localhost:8000/docs:



Project FastAPI docs

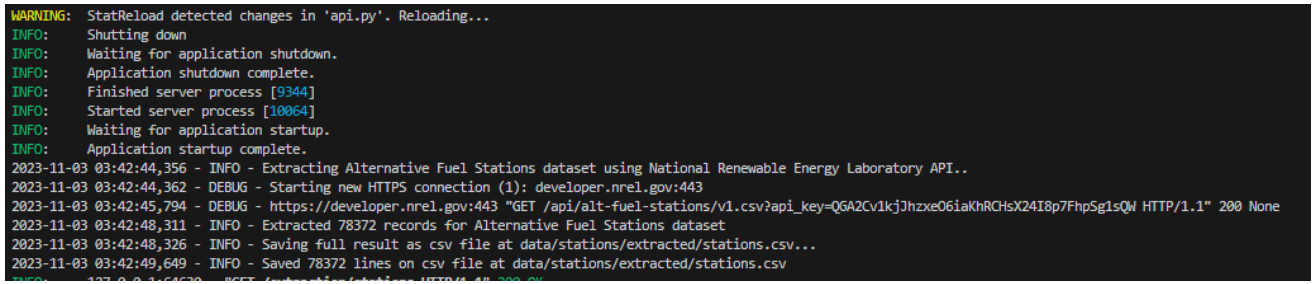
1.1 - How To Run

1. FastAPI docs provide an interactive interface can be used to send requests to the app;
2. To test an endpoint, expand the endpoint in the sidebar and click on the "**Try it out**" button;
3. Once the parameters are entered, the "**Execute**" button can be clicked to send the request to execute the endpoint defined function. The response will be displayed below the form:



FastAPI response example

4. When executed, the logs of the app can also be seen in the console:



Console logs example

2 - Vehicle Complaints Dataset

- The U.S. Department of Transportation **Vehicle Complaints** dataset includes complaints about safety concerns related to vehicles, such as brakes, steering, and airbags, as well as general issues like electrical problems and engine failures;
- Received by the National Highway Traffic Safety Administration (NHTSA) or through the Office of Defects Investigation (ODI);
- NHTSA provides an API that can be used to retrieve information on vehicle complaints.

Dataset Summary

Agency: **Department of Transportation**
Sub-Agency/Organization: **National Highway Traffic Safety Administration**
Category: 23, **Transportation**
Date Released: **December 16, 2002**
Time Period: **1949 to present**
Frequency: **Daily**

Dataset Information

Data.gov Data Category Type: **Raw data**
Specialized Data Category Designation: **Enforcement**
Keywords: **Phone, Paper, Email**
Unique ID: **81**

Contributing Agency Information

Citation: N/A
Agency Program Page: https://www.nhtsa.gov/report-a-safety-problem
Agency Data Series
Page: https://static.nhtsa.gov/odi/ffdd/cmpl/FLAT_CMPL.zip

Dataset Coverage

Unit of Analysis: **Vehicles, Tires, Child Safety Seats, Equipment**
Granularity: **United States**
Geographic Coverage: N/A

Data Description

Collection Mode: **Web, Phone**
Data Collection Instrument: N/A
Data Dictionary/Variable List: https://static.nhtsa.gov/odi/ffdd/cmpl/CMPL.txt

Dataset Quality

Data Quality Certification: **Yes**
Privacy and Confidentiality: **Yes**
Applicable Information Quality Guideline Designation: **National Highway Traffic Safety Administration**

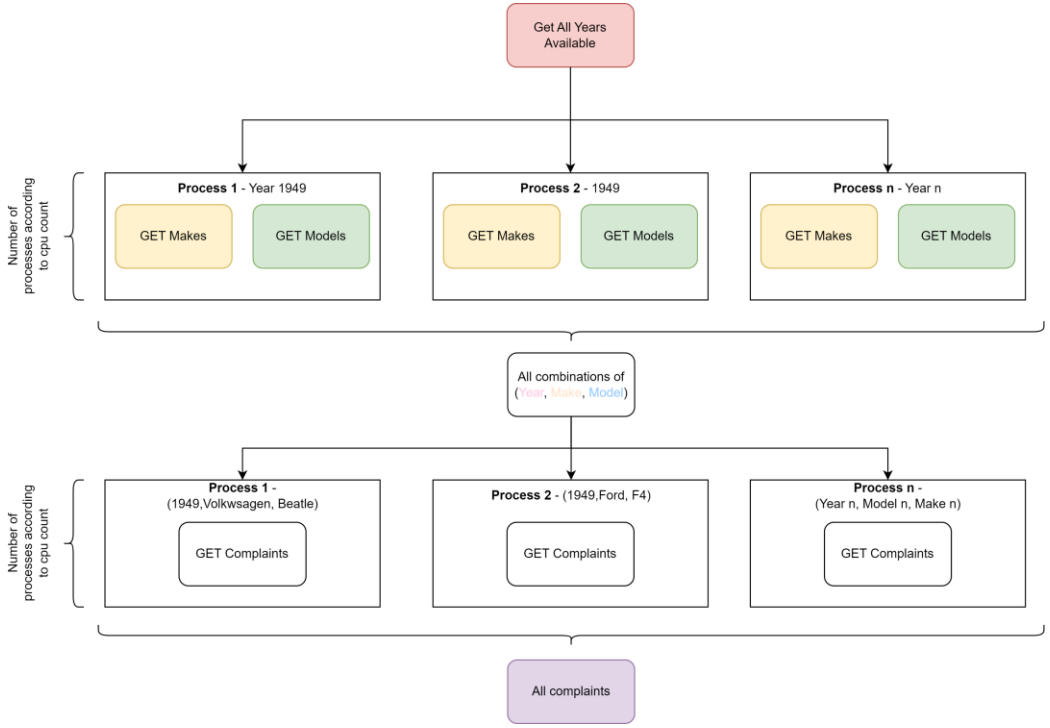
Additional Dataset Documentation

Technical
Documentation: https://static.nhtsa.gov/odi/ffdd/cmpl/Import_Instructions_Excel_All.pdf

Additional Metadata: N/A

Complaints dataset metadata

2.1 - Vehicle Complaints: Extraction

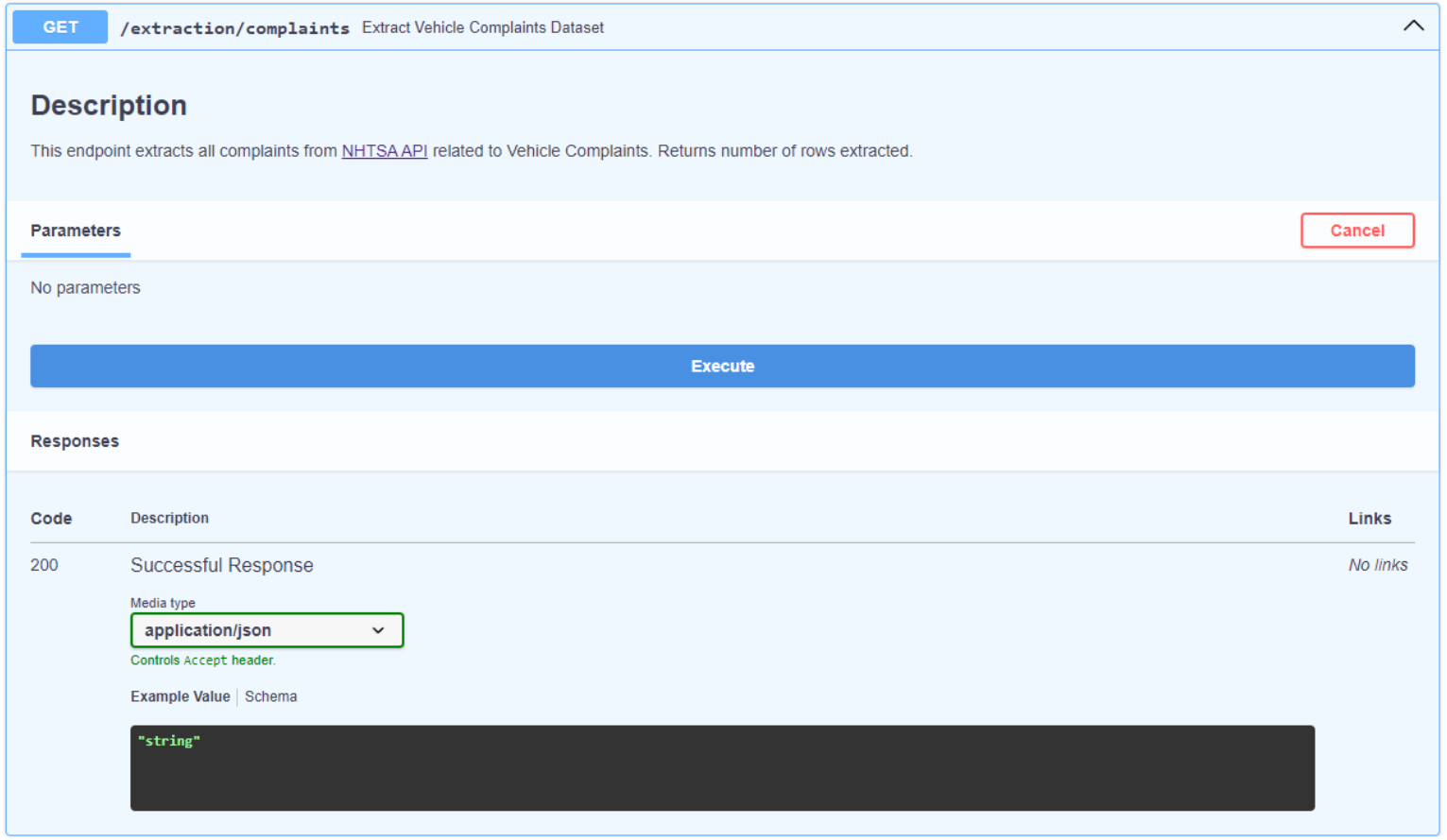


Complaints extraction method

1. Use multiprocessing to create a pool of worker processes and the starmap() function to create a list of all possible combinations between the **model years**, **makes**, and **models** that have received complaints;
2. Use multiprocessing to create a pool of worker processes and the starmap() function to extract all **complaints** for each combination of make, model, and model year in parallel;
3. Concatenate the resulting dataframes into a single dataframe and save it as a CSV.

2.1 - Vehicle Complaints: Extraction

Extraction can be accessed by making a GET request to the ‘/extraction/complaints’ route:



Complaints extraction FastAPI UI

2.2 - Vehicle Complaints: Transformation

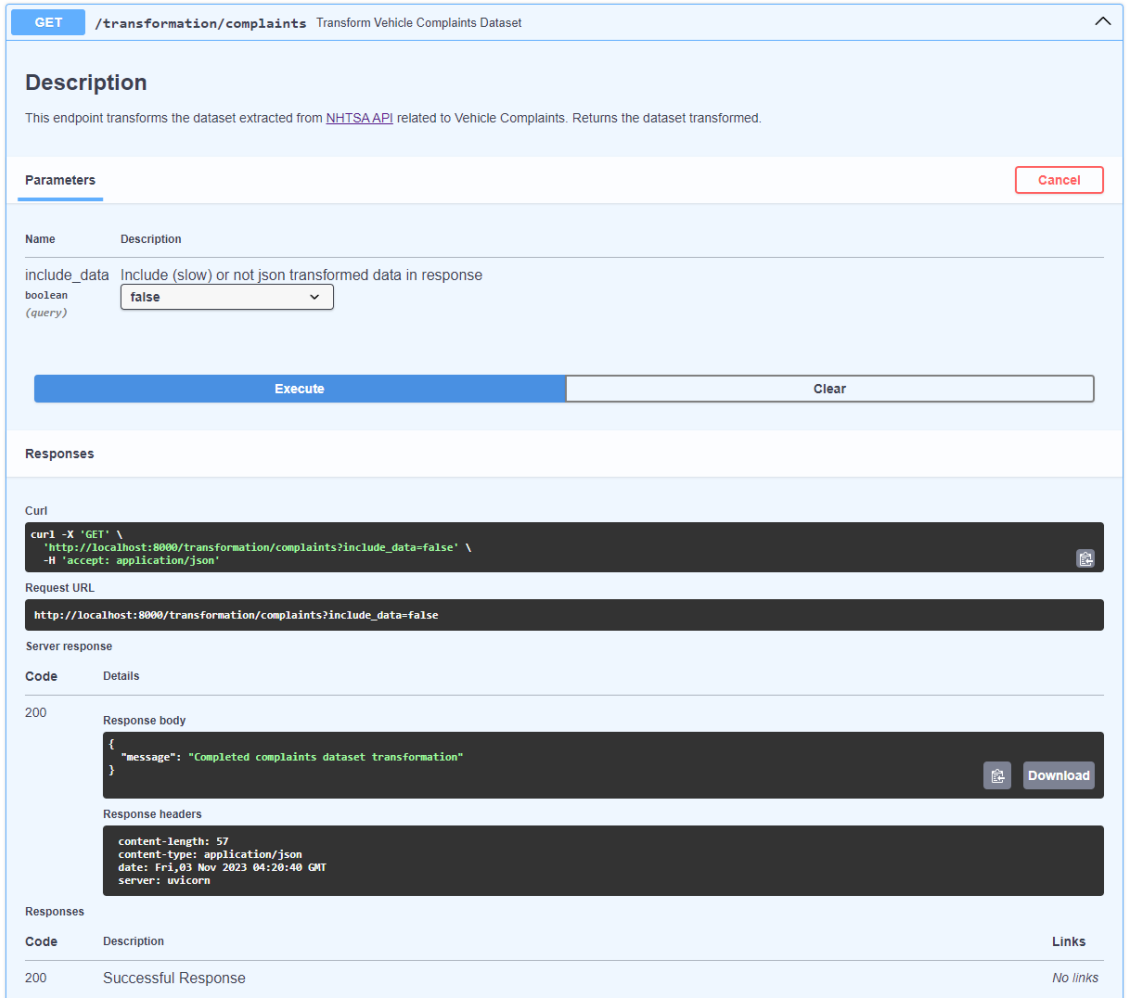
Main Transformations on Data

1. Load the CSV file and remove irrelevant columns;
2. Fix the date format and capitalization of certain columns;
3. Create a new column for the year of the complaint and convert certain columns to datetime or integer format;
4. Reorder the columns and assign a unique ID to each row;
5. Save the processed dataframe as a CSV file;
6. The final processed dataframe is returned.

2.2 - Vehicle Complaints: Transformation

Transformation can be accessed by making a GET request to the '/transformation/complaints' route:

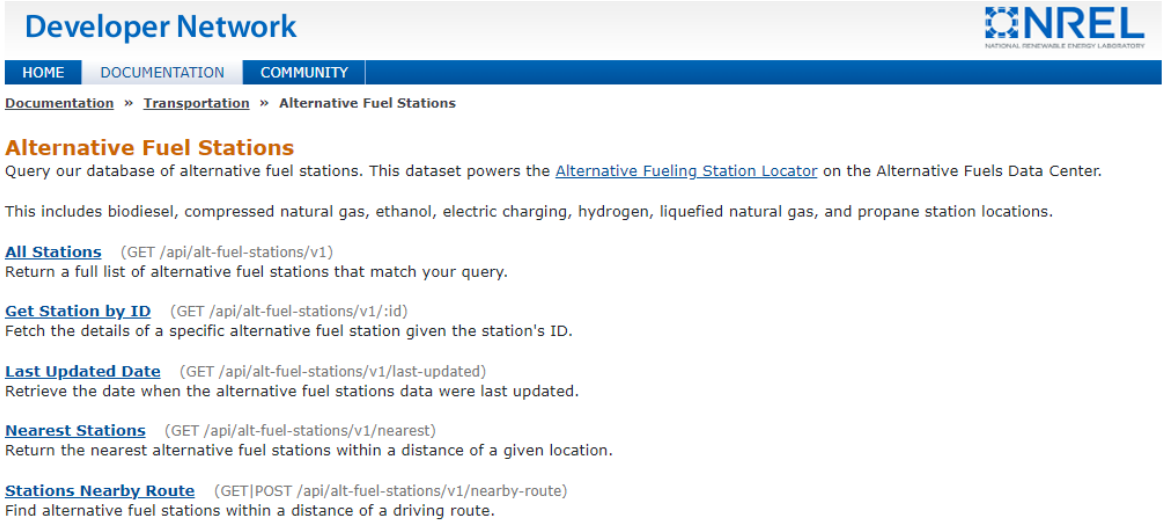
- 'http://localhost:8000/transformation/complaints?include_data=true' - Includes transformed data in the response.
- 'http://localhost:8000/transformation/complaints?include_data=false' - Excludes transformed data from the response.



Complaints transformation FastAPI UI

3 - Alternative Fuel Stations Dataset

- The **Alternative Fuel Stations** dataset is maintained by the National Renewable Energy Laboratory;
- Contains information on alternative fuel stations across the United States;
- Provides valuable information on the locations and types of alternative fuel stations available;
- Data can be extracted using the NREL API.

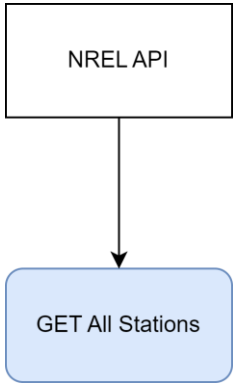


NREL API webpage

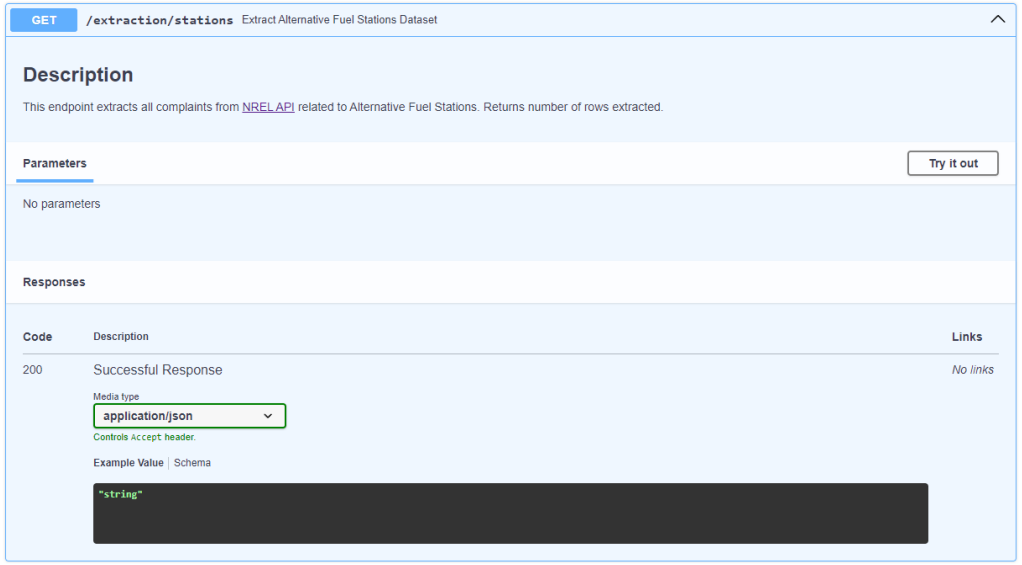
3.1 - Alternative Fuel Stations: Extraction

Steps:

1. Define the URL for the NREL API endpoint;
2. Extract the data from the response;
3. Save the resulting dataframe containing information about the alternative fuel stations as a CSV;
4. Extraction can be accessed by making a GET request to the '/extraction/stations' route.



Stations extraction method



Stations extraction FastAPI UI

3.2 - Alternative Fuel Stations: Transformation

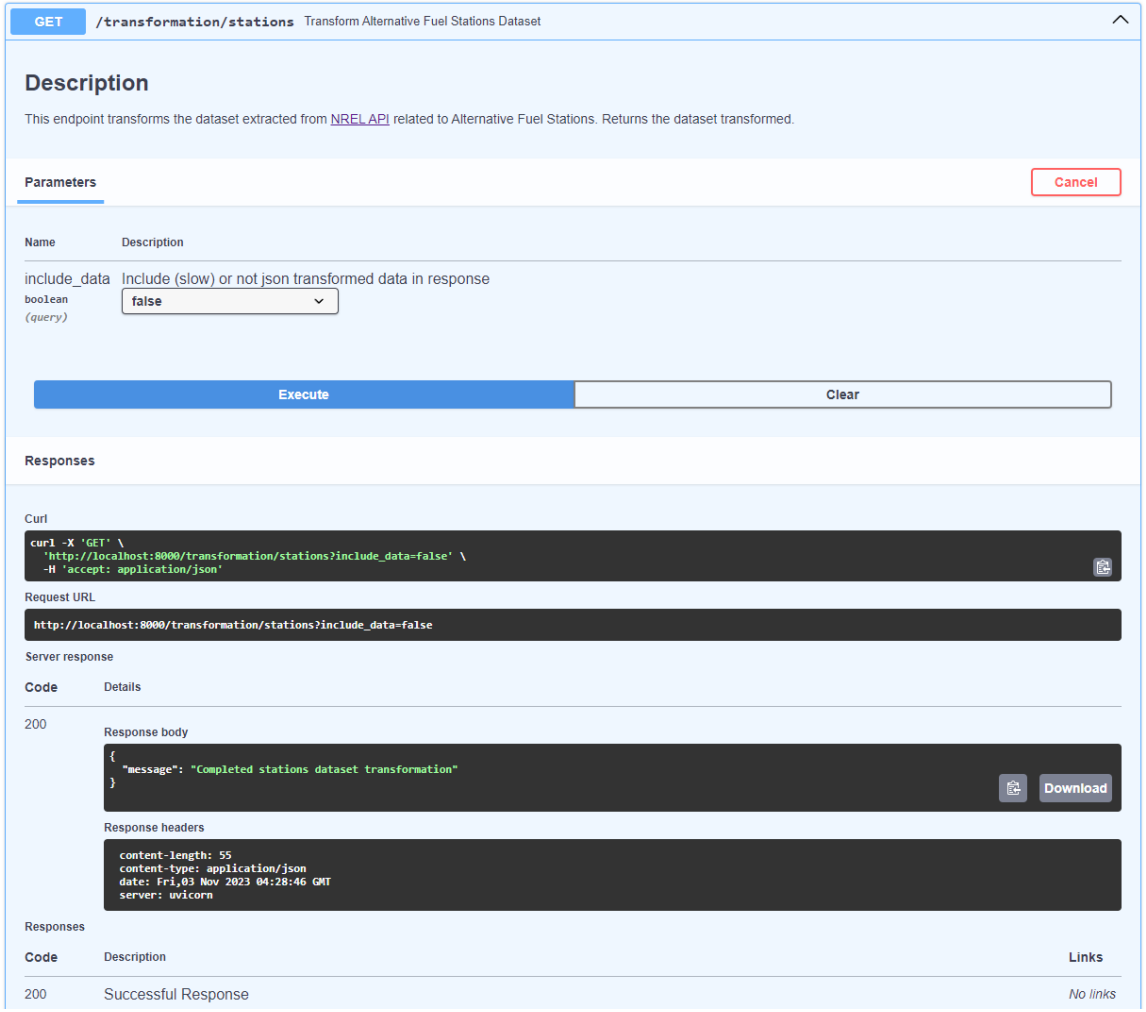
Main Transformations on Data

1. Load the CSV file and fix the column names;
2. Remove deprecated and irrelevant columns;
3. Map values of certain columns to new columns and create new address and point columns;
4. Remove rows where the '*station_name*' column is null and clean null values in string columns;
5. Convert the '*updated_at*' column to datetime format and reorder the columns in the dataframe;
6. Assign a unique ID to each row and split the dataframe into multiple dataframes based on fuel type;
7. For each fuel type-specific dataframe, perform additional cleaning, manipulation, and conversion tasks as necessary;
8. Save each fuel type-specific dataframe as a separate CSV file;
9. Return the all tables concatenated in JSON format using FastAPI.

3.2 - Alternative Fuel Stations: Transformation

Transformation can be accessed by making a GET request to the '/transformation/stations' route:

- 'http://localhost:8000/transformation/stations?include_data=true' - Includes transformed data in the response.
- 'http://localhost:8000/transformation/stations?include_data=false' - Excludes transformed data from the response.



Stations transformation FastAPI UI

4 - Vehicle Fuel Economy Information Dataset

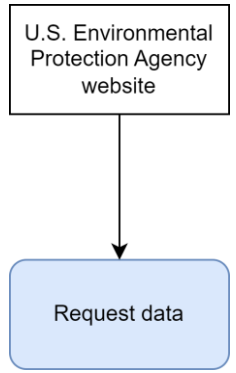
- **Vehicle Fuel Economy Information** dataset provides information on fuel economy and emissions for cars and trucks;
- Dataset is maintained by US Department of Energy and Environmental Protection Agency;
- Includes information on make, model, year, fuel type, and fuel efficiency for each vehicle;
- Used to inform consumers and policymakers about fuel economy and emissions of different vehicles.

Fuel dataset webpage

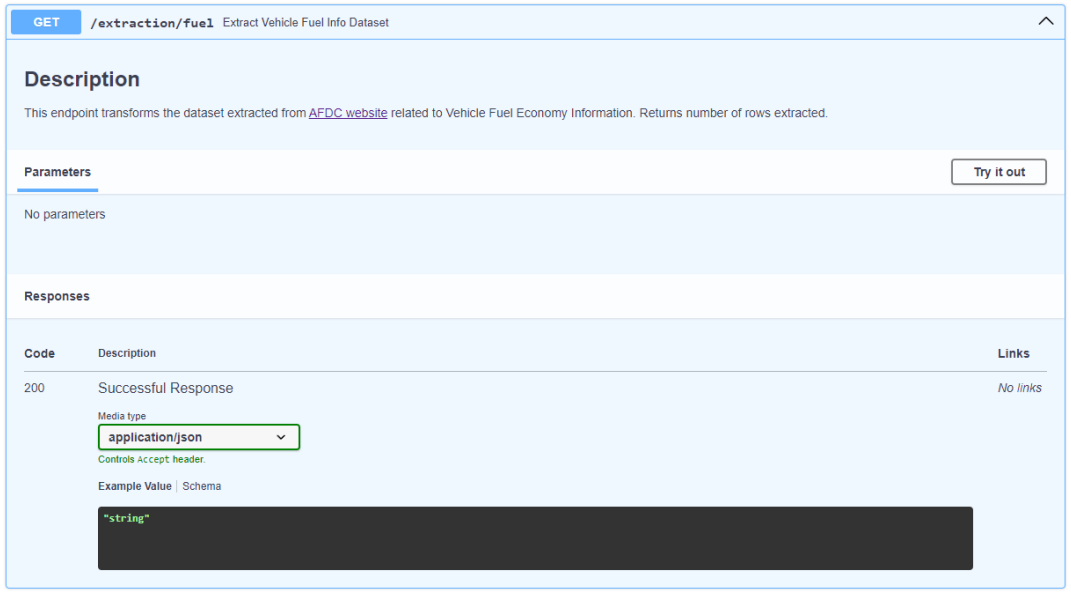
4.1 - Vehicle Fuel Economy Information: Extraction

Steps:

1. Download data using link from U.S. Environmental Protection Agency via request;
2. Save the resulting information about the alternative fuel economy as a CSV;
3. Extraction can be accessed by making a GET request to the '/extraction/fuel' route.



Fuel extraction method

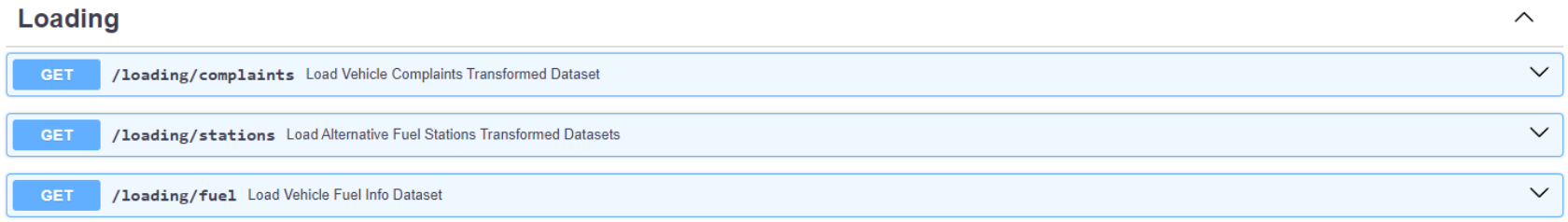


Fuel extraction FastAPI UI

5 – Data Loading

To send transformed datasets to Google BigQuery:

- Create a BigQuery engine using ‘database.SessionBigQuery()’ function;
- Use ‘loader.df_to_gcp()’ function to send a single DataFrame directly to BigQuery and ‘loader.dict_df_to_gcp()’ to send dicts of DataFrames like the case of transformed Vehicle Complaints datasets;
- ‘Loading’ tagged endpoints can be used directly to perform transformation and then loading.



‘Loading’ tagged endpoints

Notes:

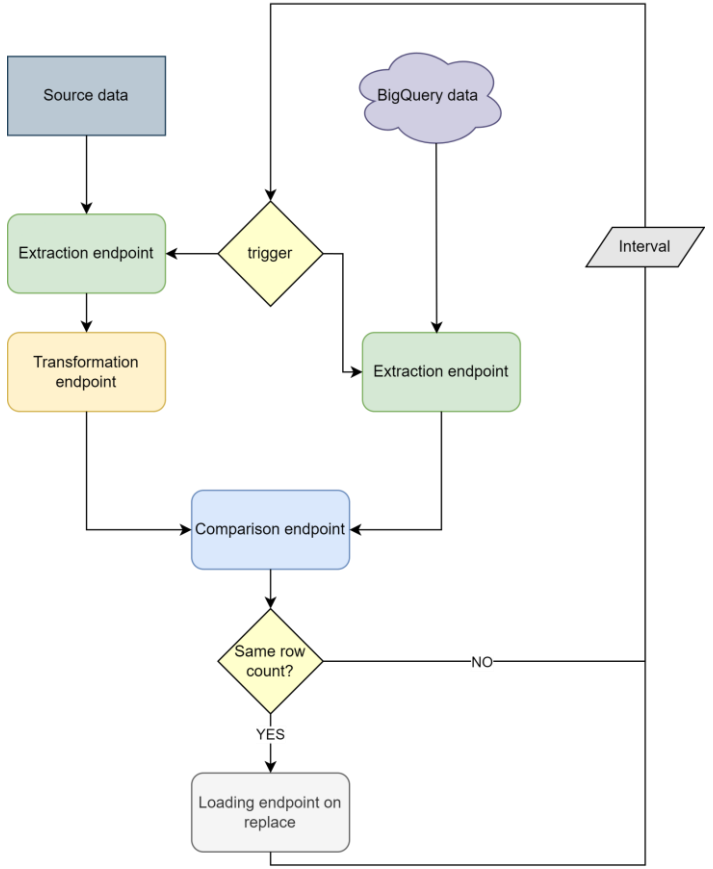
- Sending dataframes directly is safer than saving as CSV files and reopening;
- Google BigQuery has built-in GIS capabilities to work with spatial data like the Point() column created in Alternative Fuel Stations transformed datasets;

5.1 - Why Google Bigquery?

- **Scalability**: can handle datasets of any size, from gigabytes to petabytes.
- **Speed**: can process queries on large datasets in seconds or minutes.
- **Cost-effectiveness**: offers a pay-as-you-go pricing model.
- **Security**: BigQuery provides robust security features, including data encryption, access controls, and audit logging.
- **Integration**: BigQuery has native integrations with a wide range of data sources and dashboarding tools, including Google Data Studio, Grafana, and PowerBI.

6 - Automation Suggestion

1. According to a interval, extract, transform, and update datasets in BigQuery by comparing the data in cloud with the source;
2. Use tool such as crontab to schedule script to run regularly and FastAPI to create urls that can be curled for scheduling script;
3. For each dataset, download latest version from source, process it, and compare it to the version extracted from BigQuery;
4. If row count is different, replace contents of corresponding BigQuery table with newly extracted and transformed dataframe;
5. If row count is the same, do nothing;
6. Set up a cron job to run at a scheduled interval to trigger the FastAPI endpoints;
7. Make sure interval is aligned with dashboard development or other services that rely on this info.



Automation suggestion

7 - Conclusion

- What were the biggest challenges faced during the project, and how were they overcome?
- What were the most successful strategies or approaches used during the project, and how might they be applied in other contexts?
- How might the project's results be used to inform policy, decision-making, or future research in the field?