

Bosch Assessment

Data Engineer

Henrique Silva

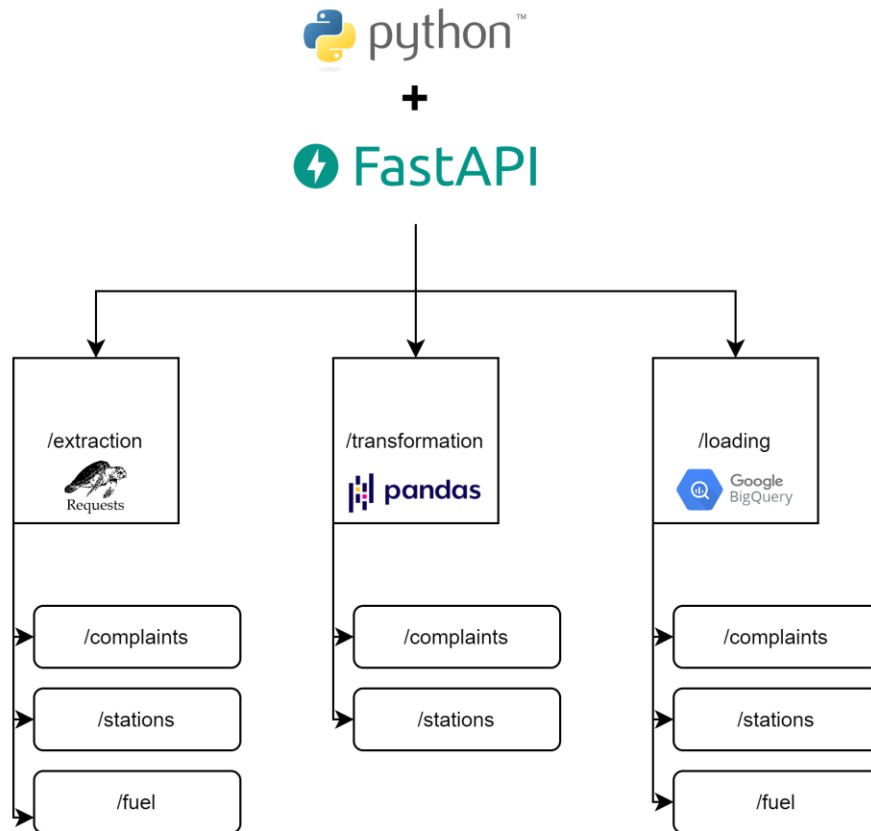
Main Objectives

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 (Slides 4-5)
2. Vehicle Complaints Dataset (Slides 6-10)
3. Alternative Fuel Stations Dataset (Slides 11-14)
4. Vehicle Fuel Economy Information Dataset (Slide 15-16)
5. Data Loading (Slide 17-18)
6. Automation Suggestion (Slide 19)
7. Conclusion (Slide 20)

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>:

Bosch Assesment 0.1.0 OAS 2.1

[/openapi.json](#)

Description

As part of a job assessment, I created a data pipeline to acquire, process, and load data from various public sources related to the automotive industry. To achieve this, I chose to develop a FastAPI application in Python. FastAPI is a modern, fast (high-performance), web framework for building APIs. It was chosen for its simplicity, scalability, and efficiency in handling large data sets. In this document, I will explain the pipeline, what it does, and how to use it. I will also document the decisions made during the data processing and transformation steps, explain the data loading process and the strategy for automation. The application consists of several endpoints that provide the necessary functionality to acquire and process the data, transform it into a format suitable for further analysis, and load it into a hypothetical data storage system (Google BigQuery). The endpoints were developed to handle the different sources of data and provide a clear and concise way to access and manipulate the data.

Extraction

- GET /extraction/complaints Extract Vehicle Complaints Dataset
- GET /extraction/stations Extract Alternative Fuel Stations Dataset
- GET /extraction/fuel Extract Vehicle Fuel Info Dataset

Transformation

- GET /transformation/complaints Transform Vehicle Complaints Dataset
- GET /transformation/stations Transform Alternative Fuel Stations Dataset

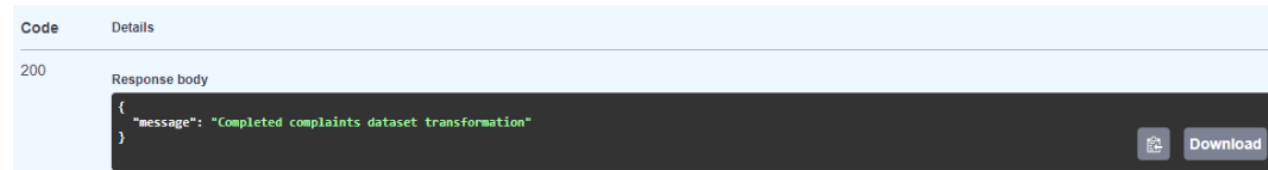
Loading

- GET /loading/complaints Load Vehicle Complaints Transformed Dataset
- GET /loading/stations Load Alternative Fuel Stations Transformed Datasets
- GET /loading/fuel Load Vehicle Fuel Info Transformed Dataset

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:

```
WARNING: StatReload detected changes in 'api.py'. Reloading...
INFO: Shutting down
INFO: Waiting for application shutdown.
INFO: Application shutdown complete.
INFO: Finished server process [9344]
INFO: Started server process [10064]
INFO: Waiting for application startup.
INFO: Application startup complete.
2023-11-03 03:42:44,356 - INFO - Extracting Alternative Fuel Stations dataset using National Renewable Energy Laboratory API..
2023-11-03 03:42:44,362 - DEBUG - Starting new HTTPS connection (1): developer.nrel.gov:443
2023-11-03 03:42:45,794 - DEBUG - https://developer.nrel.gov:443 "GET /api/alt-fuel-stations/v1.csv?api_key=Q6A2Cv1kjJhzxe06iaKhRChsX24I8p7FhpSg1sQM HTTP/1.1" 200 None
2023-11-03 03:42:48,311 - INFO - Extracted 78372 records for Alternative Fuel Stations dataset
2023-11-03 03:42:48,326 - INFO - Saving full result as csv file at data/stations/extracted/stations.csv...
2023-11-03 03:42:49,649 - INFO - Saved 78372 lines on csv file at data/stations/extracted/stations.csv
```

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**

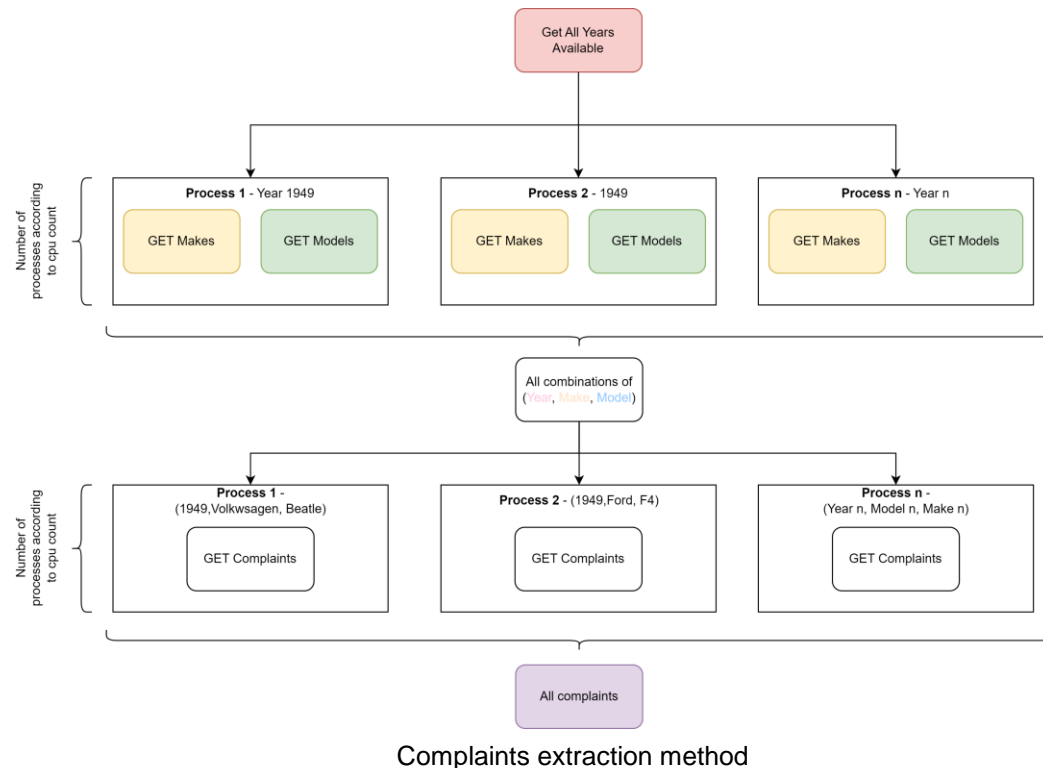
Additional Dataset Documentation

Technical

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

Additional Metadata: **N/A**

2.1 - Vehicle Complaints: Extraction



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:

GET

/extraction/complaints

Extract Vehicle Complaints Dataset

⌵

Description

This endpoint extracts all complaints from [NHTSA API](#) related to Vehicle Complaints. Returns number of rows extracted.

Parameters

Cancel

No parameters

Execute

Responses

Code	Description	Links
200	<p>Successful Response</p> <p>Media type</p> <div>application/json</div> <p>Controls Accept header.</p> <p>Example Value Schema</p> <div>"string"</div>	No links

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.

The screenshot displays the FastAPI UI for the 'Complaints transformation' endpoint. The interface is divided into several sections:

- GET /transformation/complaints**: Transform Vehicle Complaints Dataset
- Description**: This endpoint transforms the dataset extracted from [NHTSA API](#) related to Vehicle Complaints. Returns the dataset transformed.
- Parameters**: A table with columns 'Name' and 'Description'. It contains one parameter: 'include_data' (boolean, query) with a value of 'false' selected from a dropdown menu. A 'Cancel' button is located to the right of the parameters table.
- Execute**: A blue button to execute the request.
- Clear**: A button to clear the parameters.
- Responses**: A section showing the results of the request.
 - Curl**: A code block showing the curl command: `curl -X 'GET' \ 'http://localhost:8000/transformation/complaints?include_data=false' \ -H 'accept: application/json'`
 - Request URL**: A text box showing the URL: `http://localhost:8000/transformation/complaints?include_data=false`
 - Server response**: A table with columns 'Code' and 'Details'. It shows a '200' status code and a 'Response body' containing a JSON message: `{ "message": "Completed complaints dataset transformation" }`. There is a 'Download' button next to the response body.
 - Response headers**: A text box showing the headers: `content-length: 57 content-type: application/json date: Fri, 03 Nov 2023 04:20:40 GMT server: uvicorn`
- Responses**: A table with columns 'Code', 'Description', and 'Links'. It shows a '200' status code, 'Successful Response', and 'No links'.

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.

Developer Network

HOME

DOCUMENTATION

COMMUNITY

Documentation » Transportation » Alternative Fuel Stations

Alternative Fuel Stations

Query our database of alternative fuel stations. This dataset powers the [Alternative Fueling Station Locator](#) on the Alternative Fuels Data Center.

This includes biodiesel, compressed natural gas, ethanol, electric charging, hydrogen, liquefied natural gas, and propane station locations.

All Stations (GET /api/alt-fuel-stations/v1)
Return a full list of alternative fuel stations that match your query.

Get Station by ID (GET /api/alt-fuel-stations/v1/:id)
Fetch the details of a specific alternative fuel station given the station's ID.

Last Updated Date (GET /api/alt-fuel-stations/v1/last-updated)
Retrieve the date when the alternative fuel stations data were last updated.

Nearest Stations (GET /api/alt-fuel-stations/v1/nearest)
Return the nearest alternative fuel stations within a distance of a given location.

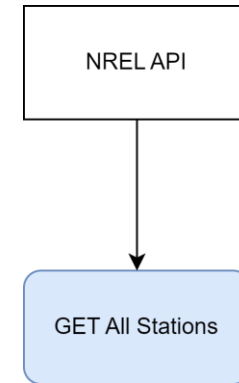
Stations Nearby Route (GET|POST /api/alt-fuel-stations/v1/nearby-route)
Find alternative fuel stations within a distance of a driving route.

NREL API webpage

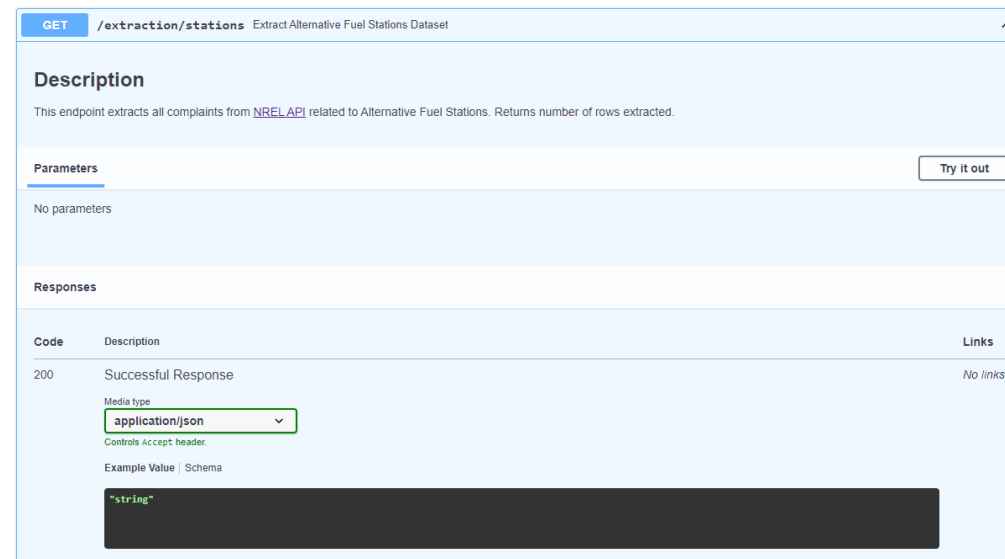
3.1 - Alternative Fuel Stations: Extraction

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.

The screenshot shows the FastAPI interactive API client interface for the endpoint `/transformation/stations`. The interface includes a description, a parameter section, and a response section.

Description: This endpoint transforms the dataset extracted from [NREL API](#) related to Alternative Fuel Stations. Returns the dataset transformed.

Parameters:

Name	Description
include_data	Include (slow) or not json transformed data in response
boolean (query)	<input type="text" value="false"/>

Execute: A blue button to execute the request. **Clear:** A button to clear the input fields.

Responses:

Curl: `curl -X 'GET' \ 'http://localhost:8000/transformation/stations?include_data=false' \ -H 'accept: application/json'`

Request URL: `http://localhost:8000/transformation/stations?include_data=false`

Server response:

Code	Details
200	<p>Response body</p> <pre>{ "message": "Completed stations dataset transformation" }</pre> <p>Response headers</p> <pre>content-length: 55 content-type: application/json date: Fri, 03 Nov 2023 04:28:46 GMT server: uvicorn</pre>

Responses Table:

Code	Description	Links
200	Successful Response	No links

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.



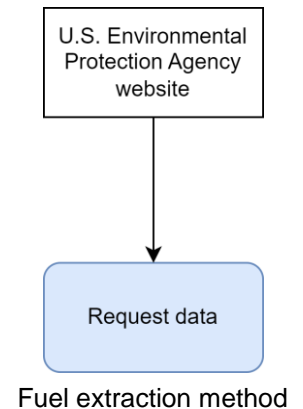
The screenshot shows the homepage of www.fueleconomy.gov, the official U.S. government source for fuel economy information. The header includes the logos for the U.S. Department of Energy (Office of Energy Efficiency & Renewable Energy) and the EPA (Office of Transportation and Air Quality). A navigation bar contains links for Mobile, Español, Site Map, Links, FAQ, and Videos. Below the header, a search bar and a list of links (Find a Car, Save Money & Fuel, Benefits, My MPG, Advanced Cars & Fuels, About EPA Ratings, More) are visible. The main content area features a 'Download Fuel Economy Data' section, which explains that fuel economy data is the result of vehicle testing at the EPA's National Vehicle and Fuel Emissions Laboratory. It also includes a section titled 'Attention! Revised Estimates' with links to EPA revisions for 2013-17 Audi, Bentley, Porsche, and Volkswagen vehicles; 2014 MINI Cooper and Cooper S; 2013-14 Mercedes C300 4matic; 2013-14 Ford Vehicles; 2012-13 Hyundai Data Revised (November 2, 2012); and 2012-13 Kia Data Revised (November 2, 2012). A 'Datasets for All Model Years (1984-2024)' section is also present, noting that MPG estimates for all 1984-2007 model year vehicles and some 2011-2016 model year vehicles have been revised. Links for downloading data are provided: Zipped CSV File (Documentation), Unzipped CSV File (Documentation), and Zipped XML File (Documentation).

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.



GET /extraction/fuel Extract Vehicle Fuel Info Dataset

Description

This endpoint transforms the dataset extracted from [AFDC website](#) related to Vehicle Fuel Economy Information. Returns number of rows extracted.

Parameters Try it out

No parameters

Responses

Code	Description	Links
200	Successful Response	No links

Media type:

Controls Accept header.

Example Value | Schema

```
"string"
```

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		^
GET	/loading/complaints Load Vehicle Complaints Transformed Dataset	▼
GET	/loading/stations Load Alternative Fuel Stations Transformed Datasets	▼
GET	/loading/fuel Load Vehicle Fuel Info Dataset	▼

'Loading' tagged endpoints

Notes:

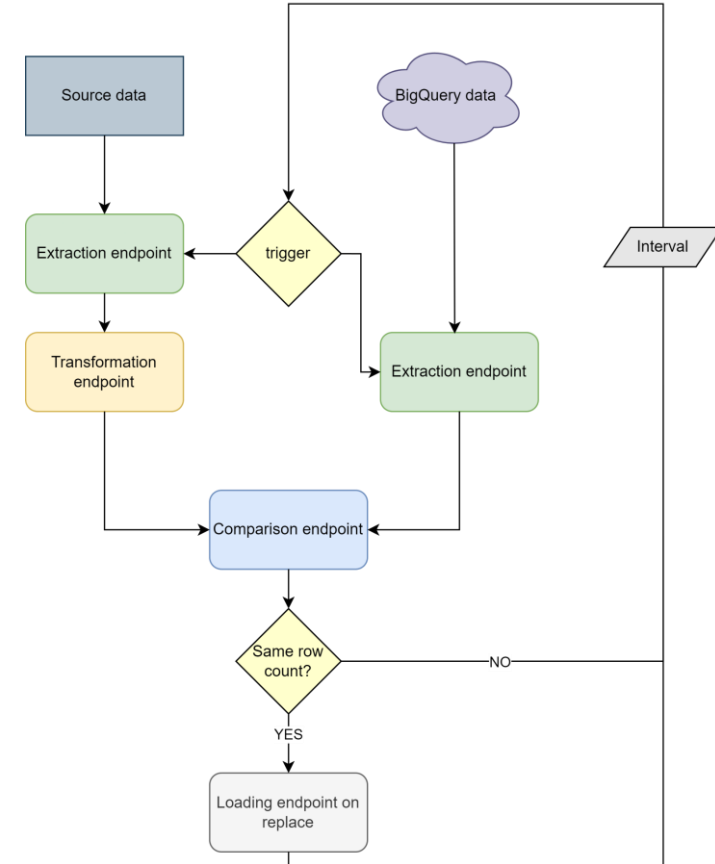
- Sending dataframes directly is safer than saving as CSV files and reopening;
- Google BigQuery has built-in GIS capabilities to work with spatial data like the Point() column created in Alternative Fuel Stations transformed datasets;

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?