Documentation

**Goal:** Design and implement a data ingestion pipeline that extracts data from two different sources and loads it into a PostgreSQL database.

**Data Sources:** CSV file, and JSON data retrieved from an external API

**Tools used:** VS Code, PostgreSQL and DBeaver

**Procedure overview:**

* Extracted the CSV file manually, transformed it using Python, and saved it as **csv\_retrieval.py**
* Retrieved JSON data from the given URL, transformed it using Python, and saved it as **json\_retrieval.py**
* Manually created two schemas/tables in DBeaver—one for the CSV data and one for the JSON data
* Inserted the processed CSV and JSON data into PostgreSQL using Python and saved the file as **postgre\_connect.py**.

**Brief Steps:**

**CSV Extraction & Transformation**

* Downloaded the CSV file and copied its local path
* Used the **pandas** library to read the file and store the data as a DataFrame
* Applied basic formatting and corrections to the DataFrame
* Replaced “NaN” values with “None” since PostgreSQL does not process NaN values.

**JSON Extraction & Transformation**

* Used the **requests**, **json**, and **pandas** libraries to extract and transform the data.
* Wrote two functions:One to fetch data from the API using requests.get(), handling common errors. Another to clean and transform the data after storing it as a DataFrame
* Major transformations included: Converting columns to relevant data types. And, Handling NaN values.

**DBeaver (Schema & Table Creation)**

* Created two tables—one for **EV data** and one for **Permits data**—using the CREATE TABLE SQL query, ensuring all necessary columns were included.

**PostgreSQL Integration (Mostly used ChatGPT for this one)**

* Connected the pipeline to PostgreSQL using **psycopg2**
* Imported clean\_csv() and clean\_json() from the respective Python scripts
* Wrote a function to dynamically generate INSERT queries based on the table name and column names.

**Major Retrospective**

**Extraction Phase**

* Extracting data was relatively simple for both the CSV file and JSON API.

**Transformation Phase**

* Initially seemed straightforward but required careful handling as issues emerged.
* Key lessons learned:
  + Column names with spaces, special characters, or leading numbers cause SQL insertion errors.
  + NaN values must always be handled before insertion
  + CSV and JSON files handle data types differently when loaded into PostgreSQL: CSV files pass int64 and float64 values correctly. JSON data, however, retains Python-specific formats that PostgreSQL does not process automatically
  + pd.read\_csv(data) and pd.DataFrame(data) handle dictionary values differently: The location column was correctly formatted when using pd.read\_csv(data), but pd.DataFrame(data) did not handle it properly. The solution was to convert dictionary values to JSON strings before loading them.
  + When using df.info(), the info() will truncate a very columns for display if there a too many. So, I need to use df.info(verbose=True) instead to view the info all the columns

**Loading Phase**

* Currently, I find PostgreSQL integration and data insertion somewhat unclear
* I need to study PostgreSQL integration in more depth, as I relied on ChatGPT for this part.

**Table Structures Used:**

**CSV Data table:**

CREATE TABLE ev\_population\_data.ev\_data (

    vin TEXT,

    county TEXT,

    city TEXT,

    state TEXT,

    postal\_code DOUBLE PRECISION,

    model\_year BIGINT,

    make TEXT,

    model TEXT,

    electric\_vehicle\_type TEXT,

    clean\_alternative\_fuel\_vehicle\_cafv\_eligibility TEXT,

    electric\_range DOUBLE PRECISION,

    base\_msrp DOUBLE PRECISION,

    legislative\_district DOUBLE PRECISION,

    dol\_vehicle\_id BIGINT,

    vehicle\_location TEXT,

    electric\_utility TEXT,

    census\_tract\_2020 DOUBLE PRECISION

);

**JSON Data table:**

CREATE TABLE permits\_data.permits (

id TEXT,

permit\_ TEXT,

permit\_status TEXT,

permit\_milestone TEXT,

permit\_type TEXT,

review\_type TEXT,

application\_start\_date TIMESTAMP,

issue\_date TIMESTAMP,

processing\_time INT,

street\_number INT,

street\_direction TEXT,

street\_name TEXT,

work\_type TEXT,

work\_description TEXT,

building\_fee\_paid FLOAT,

zoning\_fee\_paid FLOAT,

other\_fee\_paid FLOAT,

subtotal\_paid FLOAT,

building\_fee\_unpaid FLOAT,

zoning\_fee\_unpaid FLOAT,

other\_fee\_unpaid FLOAT,

subtotal\_unpaid FLOAT,

building\_fee\_waived FLOAT,

building\_fee\_subtotal FLOAT,

zoning\_fee\_subtotal FLOAT,

other\_fee\_subtotal FLOAT,

zoning\_fee\_waived FLOAT,

other\_fee\_waived FLOAT,

subtotal\_waived FLOAT,

total\_fee FLOAT,

contact\_1\_type TEXT,

contact\_1\_name TEXT,

contact\_1\_city TEXT,

contact\_1\_state TEXT,

contact\_1\_zipcode INT,

contact\_2\_type TEXT,

contact\_2\_name TEXT,

contact\_2\_city TEXT,

contact\_2\_state TEXT,

contact\_2\_zipcode INT,

contact\_3\_type TEXT,

contact\_3\_name TEXT,

contact\_3\_city TEXT,

contact\_3\_state TEXT,

contact\_3\_zipcode INT,

reported\_cost INT,

xcoordinate FLOAT,

ycoordinate FLOAT,

community\_area INT,

contact\_4\_type TEXT,

contact\_4\_name TEXT,

contact\_4\_city TEXT,

contact\_4\_state TEXT,

contact\_4\_zipcode INT,

contact\_5\_type TEXT,

contact\_5\_name TEXT,

contact\_5\_city TEXT,

contact\_5\_state TEXT,

contact\_5\_zipcode INT,

contact\_6\_type TEXT,

contact\_6\_name TEXT,

contact\_6\_city TEXT,

contact\_6\_state TEXT,

contact\_6\_zipcode INT,

contact\_7\_type TEXT,

contact\_7\_name TEXT,

contact\_7\_city TEXT,

contact\_7\_state TEXT,

contact\_7\_zipcode INT,

contact\_8\_type TEXT,

contact\_8\_name TEXT,

contact\_8\_city TEXT,

contact\_8\_state TEXT,

contact\_8\_zipcode INT,

contact\_9\_type TEXT,

contact\_9\_name TEXT,

contact\_9\_city TEXT,

contact\_9\_state TEXT,

contact\_9\_zipcode INT,

contact\_10\_type TEXT,

contact\_10\_name TEXT,

contact\_10\_city TEXT,

contact\_10\_state TEXT,

contact\_10\_zipcode INT,

pin\_list TEXT,

census\_tract INT,

ward INT,

latitude FLOAT,

longitude FLOAT,

location TEXT,

computed\_region\_rpca\_8um6 INT,

computed\_region\_vrxf\_vc4k INT,

computed\_region\_6mkv\_f3dw INT,

computed\_region\_bdys\_3d7i INT,

computed\_region\_43wa\_7qmu INT,

computed\_region\_awaf\_s7ux INT,

contact\_11\_type TEXT,

contact\_11\_name TEXT,

contact\_11\_city TEXT,

contact\_11\_state TEXT,

contact\_11\_zipcode INT,

contact\_12\_type TEXT,

contact\_12\_name TEXT,

contact\_12\_city TEXT,

contact\_12\_state TEXT,

contact\_12\_zipcode INT

);