Take Home Technical Challenge:

## Your Input files:

Attached is a zip file that contains 3 files you will need to complete this task.

File 1 - nhtsa\_file.jsonl.gz - (*sample file that contains automotive details that are publicly available from the National Highway Traffic Safety Authority (NHTSA) via a freely available API.*)

File 2 - nhtsa\_file\_parser.py - Sample Python file to use as a starting point

File 3 - nhtsa\_lookup\_file.csv - This CSV will be used in part 2 to represent a Database table.

### Prerequisite:

* To review File 1 you will need to ensure you have a zip decompression tool installed on your machine, *(NOTE: Do not use the unzipped version of the file on task 1)*
* On a Windows machine, instructions for viewing the file can be found here.: <https://techcult.com/how-to-open-gz-file-in-windows-10/>
* If needed you can download the 7-zip tool here <https://www.7-zip.org/download.html>

#### 

#### **NHTSA File Structure:**

* File 1 is a **List of Dict of List data structure** with the below format, Example data is shown below. The multiple ‘………’ within the data represents that there are more fields that have been removed to keep the example manageable.

|  |
| --- |
| [{"Count": 138, "Message": "Results returned successfully......", "SearchCriteria": "VIN:1D7RD4GG\*BC", "Results": [{"Value": "1D7RD4GG\*BC ", "ValueId": "", "Variable": "Suggested VIN", "VariableId": 142}, {"Value": "6,14", "ValueId": "6,14", "Variable": "Error Code", "VariableId": 143}, {"Value": "", "ValueId": "", "Variable": "Possible Values", "VariableId": 144}, {"Value": "Unused position(s): 7;", "ValueId": "", "Variable": "Additional Error Text", "VariableId": 156}, {"Value": "6 - Incomplete VIN; 14 - Unable to provide information for some of the characters in the VIN, based on the manufacturer submission.", "ValueId": "", "Variable": "Error Text", "VariableId": 191}, {"Value": null, "ValueId": null, "Variable": "Destination Market", "VariableId": 10}, {"Value": "DODGE", "ValueId": "476", "Variable": "Make", "VariableId": 26}, {"Value": "FCA US LLC", "ValueId": "994", "Variable": "Manufacturer Name", "VariableId": 27}, {"Value": null, "ValueId": null, "Variable": "Model", "VariableId": 28}, {"Value": "2011", "ValueId": "", "Variable": "Model Year", "VariableId": 29}, {"Value": null, "ValueId": "", "Variable": "Plant City", "VariableId": 31}, {"Value": null, "ValueId": "", "Variable": "Series", "VariableId": 34}, {"Value": null, "ValueId": "", "Variable": "Trim", "VariableId": 38}, {"Value": "TRUCK ", "ValueId": "3", "Variable": "Vehicle Type", "VariableId": 39}, {"Value": null, "ValueId": null, "Variable": "Plant Country", "VariableId": 75}, {"Value": "Jefferson North Assembly", "ValueId": "", "Variable": "Plant Company Name", "VariableId": 76}, {"Value": null, "ValueId": "", "Variable": "Plant State", "VariableId": 77}, {"Value": null, "ValueId": "", "Variable": "Trim2", "VariableId": 109}, {"Value": "H (High Line)", "ValueId": "", "Variable": "Series2", "VariableId": 110}, .........................., {"Value": null, "ValueId": null, "Variable": "Blind Spot Intervention (BSI)", "VariableId": 193}, {"Value": null, "ValueId": null, "Variable": "Lane Centering Assistance", "VariableId": 194}]}] |

You can find an XML representation of the data structure here: https://vpic.nhtsa.dot.gov/api/vehicles/decodevinextended/5UXWX7C5\*BA?format=xml&modelyear=2011

### Task: 1(Python Program)

Task 1 uses a file to replicate processing automotive details that are publicly available from the National Highway Traffic Safety Administration (NHTSA) via a freely available API. You will be processing a larger resultset and extracting the data that we need. *In our actual environment, you would save these to a database but in this case, we will just have you save the output as JSON.*

Use the skeleton file “**nhtsa\_file\_parser.py”** as a starting point to complete the following parser functionality.

* Using **nhtsa\_file.jsonl.gz** as an input file, extract ONLY the fields/column from **File 1 - nhtsa\_file.jsonl.gz** that are listed below.
* The output should ignore any duplicated **Sent\_Vin** information in it.
* Complete the outlined functionality producing a “.json” file as output.
* Please send your completed **nhtsa\_file\_parser.py** file back to us via email.
* *Note: We will ask you to walk us through your solution program during your technical interview with a demo running on your machine so please keep the results when you are completed.*

For each record in the file, the below **fields** will be available on the **Results List** as the “**Variable” Key** and its data value will be available as the “**Value” Key** within the file. (You will also notice **SearchCriteria()** needs to be parsed differently**)**

* **SearchCriteria -** Parse out the 11 characters “**VIN**”from this field and call it **Sent\_Vin**
* **Manufacturer Name** - Represents Car Manufacturer name. Call it **Manufacturer\_Name**
* **Make** - Represents the make of the car. Call it **Make**
* **Model** - Represents the model of the car. Call it **Model**
* **Model** **Year** - Represents the year the car was built. Call it **Model\_Year**
* **Trim -** Represents the car’s trim. Call it **TRIM**
* **Vehicle Type** - Represents the “type” Id of the vehicle. Call it **Vehicle\_Type\_Id**
* **Body Class** - Represents the car’s body class. Call it **Body\_Class\_Id**
* **Base Price ($)** - Represents the car’s base price before taxes and dealer price. Call it **Base\_Price**
* **NCSA Make** - Represents NHTSA’s make name. Call it **NCSA\_Make**
* **NCSA Model** - Represents NHTSA’s model name. Call it **NCSA\_Model**

**Example output:**

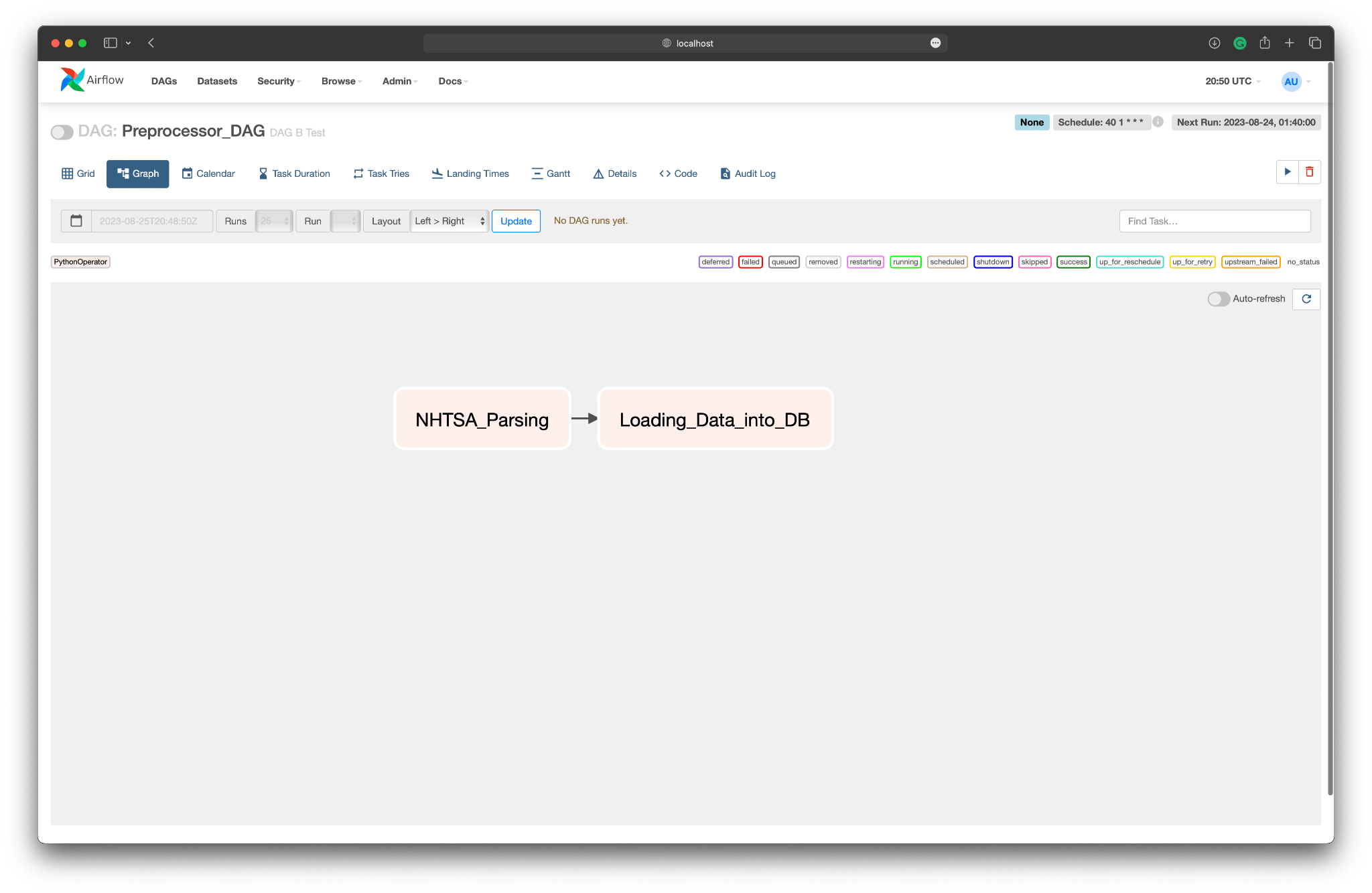
|  |
| --- |
| {"Sent\_VIN": "W1KUG7DB\*LA", "Manufacturer\_Name": "MERCEDES-BENZ CARS", "Make": "MERCEDES-BENZ", "Model": "S-CLASS", "Model\_Year": "2020", "TRIM": "", "Vehicle\_Type\_ID": "2", "Body\_Class\_ID": "13", "Base\_Price": "", "NCSA\_Make": "MERCEDES-BENZ", "NCSA\_Model": "S CLASS (95 ON)"} |

Please save the above output dataset as **parsed\_nhtsa\_data.json**

### 

### Task: 2 (Airflow DAG)

In this task, create an Airflow DAG to run the Python code from Task 1 and ingest the data into the DB of your choice. Feel free to use MySQL, Google BigQuery or any other Database. Following DAG might give you an idea of what’s expected, however, feel free to improvise.



### 

### Task: 3 (SQL Question)

**Assumption:**

The **processed\_nhtsa\_data file** you created in Step 1and NHTSA Lookup file (**File 2: nhtsa\_lookup\_file.csv**) represent two Database tables that would result from the process in Step 1. These tables would be called ***processed\_nhtsa\_data*** and ***nhtsa\_lookup\_table*** with the following table structure.The real ***processed\_nhtsa\_data*** *table would have 10’s of thousands of records, nhtsa\_lookup\_table is a stable lookup table.*

**parsed\_nhtsa\_data**

The processed\_nhtsa\_data table would contain the raw data that your .py file created. In a real-world scenario, you would have stored that in a table.

|  |
| --- |
| CREATE TABLE  processed\_nhtsa\_data ( Sent\_VIN string,  Manufacturer\_Name string,  Make string,  Model String,  Model\_year integer,  trim string,  Vehicle\_Type\_ID integer,  Body\_Class\_ID integer,  Base\_Price float64,  NCSA\_Make string,  NCSA\_Model string ); |

**nhtsa\_lookup\_table**

The nhtsa\_lookup\_table contains a standardized list of the NHTSA *vehicle\_type* and *body\_class* and maps those to the equivalent LotLinx equivalents in *LX\_BodyClass\_lvl1* and *LX\_BodyClass\_lvl2*

An example of this would be a *Vehicle\_Type* called “PASSENGER CAR” which can be further defined as multiple types of passenger car types that include Sedans, Wagons, and Convertibles. We use the nhtsa\_lookup\_table to convert the NHTSA results to our internal standards.

* vehicle\_type -> LX\_BodyClass\_lvl1
* body\_class -> LX\_BodyClass\_lvl2

|  |
| --- |
| CREATE TABLE  nhtsa\_lookup\_table ( Vehicle\_Type\_ID INTEGER,  Vehicle\_Type STRING,  Body\_Class\_ID INTEGER,  Body\_Class STRING,  LX\_BodyClass\_lvl1 STRING,  LX\_BodyClass\_lvl2 STRING,  Incomplete\_Chassis BOOLEAN); |

**SQL Question 1:**

Write a query to determine the 10 most common model\_year, make, model (distinct Sent\_VIN’s)

* Include model\_year, make, model and the count in your results
* Include only 10 rows in your result
* Sort your final results from most common to least

**SQL Question 2:**

Write a query that counts the number of Distinct Sent\_VIN records, grouped by LX\_BodyClass\_lvl1 (*E.g. We want a count of the number of Van, passenger cars and pickups*)

* Use LX\_BodyClass\_lvl1 to exclude all motorcycles and buses in your results.
* Use LX\_BodyClass\_lvl2 to exclude CONVERTIBLE’s from your count of PASSENGER CARs
* Rename LX\_BodyClass\_lvl2 to “bodysegment “ in your output