ApartmentIQ

# Take-Home Project Requirements

## Overview

This is a simple exercise designed to evaluate a data engineer's ability to work with data and ML models. This exercise will require querying some public datasets, ingesting the data into a local database, performing simple ETL operations on that data, deploying a simple model inside a wrapper service, and utilizing that model to perform further ETL.

## Provided Files

You will need to download this provided ZIP file, containing the following files:

- annual_miles.csv - this is part of the dataset you will be ingesting
- api.py - a stub Python file demonstrating how to call the model
- docker-compose.yml - a simple Docker Compose file that sets up a Postgres database
- Dockerfile - a stub to get started
- model.pkl - the provided model file, that requires no Python dependencies to use

## Data Ingestion

For this exercise, you will be ingesting data from three different sources into the database.

### Data Sources

- Annual Miles
  - Contains average miles driven in a year by geographic area
  - Source: included CSV file
- Fuel Efficiency
  - Contains fuel prices by region/date
  - Source: JSON API; endpoint here: https://api.eia.gov/v2/petroleum/pri/gnd/data?api_key=3zjKYxV86AqtJWSRoAECir1wQFscVu6lxXnRVKG8&frequency=weekly&data%5B0%5D=value&sort%5B0%5D%5Bcolumn%5D=period&sort%5B0%5D%5Bdirection%5D=desc&offset=0&length=5000
- Vehicle Data
  - Contains list of vehicles along with average MPG
  - Source: Parquet file [download]

ApartmentIQ

You will need to create (at least) three tables with the following columns at a minimum:

| Table | Column | Description | Source |
|---|---|---|---|
| Annual Miles | duoarea | Geographic Region | CSV file |
| Annual Miles | state | US State | CSV file |
| Annual Miles | miles | Average Miles driven per Year | CSV file |
| Fuel Efficiency | duoarea | Geographic Region | JSON API |
| Fuel Efficiency | period | Date | JSON API |
| Fuel Efficiency | product-name | Type of Gasoline | JSON API |
| Fuel Efficiency | value | Cost per Gallon | JSON API |
| Vehicle Data | make | Vehicle Make | Parquet file |
| Vehicle Data | model | Vehicle Model | Parquet file |
| Vehicle Data | year | Vehicle Year | Parquet file |
| Vehicle Data | comb08u | Average MPG | Parquet file |
| Vehicle Data | fueltype1 | Fuel Type | Parquet file |

You may write code or use software tools to ingest the data into the database, but whatever you do to ingest the data, it should be codified and repeatable (i.e. you should not use manual point-and-click tools to import the data). When we start your final application with `docker-compose up`, it should be able to load the data automatically.

## Rest API

You will need to code a REST API with, at a minimum, the following endpoints:

- `GET /` - the root path should be a health check
- `GET /economical` - this endpoint will need to accept three query/path parameters:
  - `duoarea` - a geographic region
  - `month` - the month
  - `year` - the year

The `economical` endpoint will determine the most economical car to have purchased within the provided geographic region based on the fuel prices for that month/year. To help select this vehicle, we have provided a model file in PKL format. Note that no extra Python libraries are necessary to utilize this model.

ApartmentIQ

# Calling the model

In order to call the model file, use this code block:

```
import pickle
import marshal
import types

with open("model.pkl", "rb") as fh:
    serialized_data = pickle.load(fh)
    code_obj = marshal.loads(serialized_data["code"])
    predict = types.FunctionType(
        code_obj, globals(), serialized_data["name"], serialized_data["defaults"]
    )
```

This will expose a `predict` function which you can utilize. Run `help(predict)` from a Python REPL to see the function signature and better understand how it will be used.

## Goals

You will be writing a REST API that will use the data sources and the provided model file to compute the most economical vehicle for a given geographic region in a given year. For the purposes of keeping this exercise simple, you will only be dealing with vehicles that use gasoline. *(This is not meant as a commentary on the automotive industry, but merely a way to ensure this take-home engineering exercise doesn't become overly complicated. Incidentally, many of ApartmentIQ's founders drive electric vehicles and love them.* 🙂*)*

## Requirements

1. **Ingest the data.** Write scripts or utilize software tools (in a codified, repeatable way) to ingest the three data sources into the Postgres database
2. **Transform the data.** Filter the Vehicle Data to keep vehicles with "Regular Gasoline" as their primary fuel type, and remove everything else. Likewise, filter out any values from the Fuel Efficiency Data that are not "Regular Gasoline." You are free to perform any additional transformations that you deem necessary to complete your work.
3. **Create a REST API.** After having populated and filtered the data, you will need to create a REST API with at least two endpoints as described above. This REST API needs to utilize the provided model file.

Please save all of your work into a ZIP file and submit it back to us. We expect to be able to run `docker-compose up` and then run a suite of tests that will hit endpoints like `http://localhost:8000/` and `http://localhost:8000/economical`.