Data Engineer Take-Home Assignment
Business Development Staff
Economic Reimbursable Surveys Division
U.S. Census Bureau

We would like to see a code sample from you. The best way to tell if someone is a good data engineer is to see how they work with data! We offer two options for this exercise:

1. Submit a code sample you've written elsewhere to share with us and discuss. It could be a tool to facilitate users working with data. It could be a pipeline (or a piece of a pipeline) to train and evaluate machine learning data. It could be code to scrape/ingest information from a webpage, PDF, or something else. Ideally, you would submit something close to a "technical report" including code, documentation, and any descriptions/explanation that helps tell a story and explain why you made the [modelling, analysis, coding] choices you did. This is subject to a few conditions:
    - You must be allowed to share this code with us (e.g., nothing you've produced under an NDA). Open source is best.
    - It should be in Python, Julia, Scala/Spark, or Java – if it's not we can probably accommodate you, but check with us first.
    - It should be short, no longer than around 1,000 lines, and mostly stand-alone (i.e., shouldn't require any dependencies that aren't common in the language you've used). Work that fits into existing data ecosystems (like PyData, GeoPandas, etc.) is welcome.
    - Ideally, it should be code we could easily run ourselves. If your code does use non-standard dependencies, please include a list of those dependencies (ideally in a popular format for your language such as a requirements.txt file in Python).

    To submit, please reply to this email with either a link to a repository where we can view the work (e.g. GitHub repo link), or email the work as an attachment. Please do not include any large data files in the email itself.

2. Complete a short data engineering assignment. This'll be a realistic sort of task that someone in the role you're interviewing might face at work. This exercise should take less than three hours. We prefer that you use Python, Julia, Scala/Spark, or Java for this exercise, but if you don't feel confident in any of those languages we can probably accommodate another choice.

    If this option works best for you, more details about the assignment are below. We typically ask for you to return the assignment within a calendar week, but are happy to accommodate any requests for more time.

ASSIGNMENT OPTION: USDOT web scraping challenge

This challenge simulates the type of problem you might encounter when working at the Bureau as we collaborate with other agencies and stakeholders across government.

Before getting into the details, we want to mention up-front that we want you to spend no more than 3 hours of time on this project. As such, we fully expect that the final deliverable will be of "prototype" quality. That is OK!

Background

The U.S. Department of Transportation's Federal Motor Carrier Safety Administration requires that operators of heavy trucks that participate in interstate commerce register themselves. As part of that registration process, operators must report how many vehicles they own, lease, and rent.

You can see an example registration here.

https://ai.fmcsa.dot.gov/SMS/Carrier/21800/Overview.aspx?FirstView=True

If you click into the "Carrier Registration Details" link, you see that we get detailed information about the types of cargo that this operator carries, as well as breakouts of the different vehicle types that operator uses.

https://ai.fmcsa.dot.gov/SMS/Carrier/21800/CarrierRegistration.aspx

On the bright side, FMCSA does make some of this information available to download in its "Motor Carrier Census Information" file…

https://ai.fmcsa.dot.gov/SMS/Tools/Downloads.aspx

However, the vehicle type breakouts and information about goods carried are not included. These are valuable to us. We have a close relationship with U.S. DOT, but unfortunately, the process of getting U.S. DOT to add these data items to the csv extract (the Census Information file) will take way too long, and we need these data sooner than that. So we called you!

Your task

We want you to develop a program to scrape the "Carrier Registration Details" page for each operator in the "Motor Carrier Census Information" file. In particular, we want to make sure to pull the lists of cargo carried as well as the vehicle type breakout table.

This scraper should parse the information on this page and store it into a structured data format of your choice (e.g. CSV/multiple CSVs, or a database).

As you develop this codebase, bonus points for making it robust in 2 ways:

1. You'll find slight variations on each of the registration details pages, e.g. if a particular field is missing.
2. The USDOT site frequently goes down. (True story). It's going to take a few days to run this scraping, and we want it to be low-maintenance while it runs.

Helpful hint

To get the Carrier Details page for each operator in the Census Information file, note the structure of the above URL:

https://ai.fmcsa.dot.gov/SMS/Carrier/**[DOT_NUMBER]**/CarrierRegistration.aspx

In other words, for each DOT number (a number that uniquely identifies the operator) in the Census file, you can pull that operator's carrier registration page by plugging that number in to the bolded part of the URL.

How you will be evaluated

Our goal is to see how you start to work and think through a problem like this, and expect to see something at the "prototype stage." What we want to see is how you would develop an initial prototype and document it for other highly technical, data scientist/engineer colleagues. What design choices did (or would) you make, and why?

Again, the scraper described here would take significantly more than 3 hours to "fully" complete. We do not expect a fully complete package, perfect code / perfectly documented code, or a complete database of all this scraped information.

How you should submit

Ideally, we would like to see

- A link to a GitHub repository including the code you wrote.
- In that repo, a README that includes the following:
    - A brief written overview of your overall approach.
    - Instructions that tell us how to install + run the code.
- Also in the repo, a file with a list of packages (e.g. a requirements.txt file in python, or similar standard in your preferred language) that we can use during the install process.

If you do not have a GitHub account, we'll also accept a zipped folder containing the analysis code and any documentation/output (please do not include the input data!) as an attachment.