* **Describe a project that you have worked on that allowed users to have the most seamless way to interact.**

In my position at Anadarko Petroleum I acted as a liaison between IT and our Supply Chain department. I worked extensively with DB Administrators on creating SQL views that were consumed by an application called SpotFire (very similar to Tableau but perhaps more robust). The SQL views were established using a virtual DB program called Denodo. I was very proud of this project because of the way the architecture worked; in Denodo I could use APIs, upload static documents, or point to individual files and create “virtual tables” in Denodo. In our corporate data landscape we had many different sources for information that did not speak to one another in an efficient manner and were not located in a central DB – Denodo filled this void. Additionally, the data sources were not reliable in how they integrated with each other. Some data sources were in Microsoft Access, some in SAP, some on the internet, etc... By using Denodo we made calls to the endpoints of each source and then joined the data. Once I joined the data, it was published to Spotfire where users could filter or manipulate the information in whatever manner they needed. Before the implementation of Denodo/SpotFire in my organization, users would be creating joins by VLOOKUP’s in excel or other means. Post implementation users had a consistent data source with consistent rules applied. Additionally they were able to use SpotFire to consume the data which was more user friendly than individual users trying to learn VLOOKUP and understand the data structures and tables.

* **Describe a major library/API you have worked with, what you used it for, and its advantages and disadvantages.**

Also at Anadarko Petroleum, I utilized APIs from a third-party supplier called Enverus (or OpenInvoice). Enverus processed all of our invoices for our company electronically, and had API endpoints established for consumption. In my role I was tasked with consuming historical invoice data so that I could help our finance teams understand expenditures and ensure budget expectations were being met. The advantages of Enverus’s API system were clear documentation on their Swagger page, the ability to utilize tokens for authentication, and their robust filtering by using parameters. The major disadvantage that I believe existed with using the Enverus API was that permissions were not granted or built out to access 100% of the information that I needed. As I was advancing in my Computer Science degree, I became aware that there were likely APIs within the Enverus DB system that were being passed to their frontend website(s) but they did not provide those same accesses to their customer base. The only option to capture this information was through web scraping but it would have been a monumental task given the number of web pages that would have needed to be navigated through every night.

* **Given an unsorted list of N numbers, write a program/function that will return all combinations of these numbers that sum up to a number Y.**

<https://github.com/scarrington76/combinationsum>

* **Given some value V that is between 0 - 100 and predefined, and unique coins denominations of values A, B, and C, write a function that will return the minimum value of coins needed to achieve a value of V.**

<https://github.com/scarrington76/greedycoin>

Note: The question asked for the “value” of the coins and not the sum of the coins to get to Value V. I assumed this was a typo and gave an output of the minimum number of coins instead of the value of the coins. Also, I used the greedy method in attempt to get the minimum number of coins; however there are cases in which the greedy method would not be suitable. For example when the coin values were 10, 4, and 6 but the target value was an odd number.

* **Create/pick an ideal data structure to represent a contact list and describe its trade offs and why it was picked.**

I would utilize a NoSQL DB for a contact list. The advantages of using NoSQL would be that non-traditional information like photos, videos, etc. could be added for each user. Also NoSQL would allow for individual fields to be expanded as needed (with arrays for example). While a traditional SQL dB may have achieved most of the goals ten years ago, I believe a NoSQL dB with nonprimitive data types will be needed for future generations of data. NoSQL databases like MongoDB are also more prevalent in use today and generally accepted in most software architectures. Extraction into JSON would be ideal as well.

* **A site allows a user to customize their profile, in which they can set the shape of a logo to be between a circle, triangle, square, pentagon and hexagon, along with the logo being either red, green, blue, or any combination of them. In addition to this, it allows the user to have four rotational options for the logo (upright, turned left, turned right, or upside down). Create a data structure along with accompanying get/set functions that will store this information in the most space efficient way.**

<https://github.com/scarrington76/java_ds>

A Java data structure was used to create the profile with setters and getters. Ideally the Java program would link to a DB that would hold this information (for example a MongoDB in a docker container with a web app on a serverless framework). For this exercise purposes I tried to keep my solution simple and not use docker/aws/mongodb etc.