Reflection Questions Exercise 2.7

1. **Consider your favorite website/application (you can also take CareerFoundry). Think about the various data that your favorite website/application collects. Write down how analyzing the collected data could help the website/application.**

   If we take CareerFoundry as an example, here are some of the data sets it collects:

   - User information (name, email address, location, background, career intent, payment details, certificates)
   - Mentor information (see categories above)
   - Course progress/ pace

   This is only some of the data available to CareerFoundry. By analyzing this data, the administrators can get a better sense of the demographics of their students – where they come from, their previous programming knowledge and career goals. This in turn can inform how the courses are structured, how much detail is provided in the instructions for every exercise. Being able to track student progress (including certification and future employment) is also an important metric for program success. If students are moving through the presented material at a good pace and are able to secure a job after finishing, this says something good about CareerFoundry and the education it's providing.

2. **Read the Django official documentation on QuerySet API. Note down the different ways in which you can evaluate a QuerySet.**

   - *Iteration* – similar to a for loop in Python, this will iterate over every object in the set (this can also be done asynchronously)
   - *Slicing* – just as it sounds, this will slice up a set of data which can no longer be modified afterwards
   - *Pickling/Caching* – this forces all the results to be recorded into memory at the time of pickling and is useful as a "screenshot" of the QuerySet at that moment in time
   - *repr()* – method to show the string representation of the results which counts as an evaluation of the QuerySet
   - *len()* – as expected, returns the length of the results list
   - *list()* – force the evaluation of the results by listing them
   - *bool()* – this involves testing the results in a boolean context which in turn will cause the query to be executed

3. **In the Exercise, you converted your QuerySet to DataFrame. Now do some research on the advantages and disadvantages of QuerySet and DataFrame and explain the ways in which DataFrame is better for data processing.**

   Both options have their advantages. QuerySet is fully integrated with Django structure and is easy to implement. It's also lighter on memory usage, so if performance is a concern, it might be a stronger option. However, DataFrame is overall a better option because of the variety it offers in data manipulation. Although it can take up a lot of memory, it is faster at processing complex datasets than QS. Lastly, while not automatically integrated with Django, it is easy to do so, as well as working in other Python libraries that deal with data processing/evaluation.