Yufan Xu

Exercise 2.7: Data Analysis and Visualization in Django

Learning Goals

- Work on elements of two-way communication like creating forms and buttons
- Implement search and visualization (reports/charts) features
- Use QuerySet API, DataFrames (with pandas), and plotting libraries (with matplotlib)

Reflection Questions

❖ 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.

> ➢ Taking YouTube as an example, it collects our login credentials (along with the details we enter into our profile, like our date of birth), our search inputs, browsing history, our "likes" and "subscriptions", and others. Analyzing this collected data helps the application by authenticating and protecting user information (login credentials), preventing age-restricted content to be shown to those under the required age (login and date of birth details), and the ability to customize the application for the individual user (updates from subscribed channels, suggested videos).

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

> ➢ **Iteration**: executing a QuerySet's database query the first time we iterate over it
>
> ➢ **Slicing**: a QuerySet can be sliced (with Python's array-slicing syntax); gets executed if an unevaluated QuerySet is sliced with the "step" parameter (returns a list
>
> ➢ **Pickling/Caching**: all results are read from the database and are forced to be loaded into memory prior to pickling (usually as precursor to caching)
>
> ➢ **repr()**: can call repr() on a QuerySet to evaluate it; for convenience in the Python interactive interpreter (can immediately see results when using the API interactively)
>
> ➢ **len()**: can call len() on a QuerySet to evaluate it; returns the length of the result list
>
> ➢ **list()**: can call list() on a QuerySet to force evaluation
>
> ➢ **bool()**: can execute a QuerySet by testing it in a boolean context, like using bool(), "or", "and", or an "if" statement

❖ 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.

> ➢ A QuerySet is built up as a list of objects collected from a database, and it allows filtering and ordering of the data at an early stage. They also consume less memory than DataFrames. DataFrames have faster in-memory processing, offer greater flexibility and functionality for data manipulation. They also integrate well with Python libraries for data analysis and visualization. QuerySet is great as long as the database is small, but as the database grows, DataFrames will handle the data more efficiently