## ECON 8320 – Final Semer Project (Hope Cancer Foundation Dashboard)

### Overview

The objective of this project was to leverage the skillset acquired from the course to build a data driven portal for Hope Cancer Foundation. The portal has been designed to meet all the requirements listed in the final semester assignment. The supporting artifact for this project includes data file and a data dictionary.

To organize the project, I began by establishing high level tasks, followed by analysis of the data and additional breakdown of those tasks into manageable subtasks. This provided accurate measurement of the complexity of the tasks, and its estimated completion estimates. The high-level discovery plan included:

1. Identifying required data columns for the solution.
2. Conduct data quality assessment and derive data cleansing steps.
3. Developing & Testing data cleansing functions.
4. Writing Queries to satisfy reporting requirements.
5. Learning Streamlit for dashboarding.
6. Transform Queries from step 5 into interactive dashboard.
7. Explore GitHub Action and Workflows
8. Create GitHub workflow to trigger data cleansing process upon data file change.
9. Write a Summary Page

### Execution Observation / Experience:

#### Data Cleanup:

The original data file has numerous data issues, despite having a data dictionary. The current process lacked sufficient data validation rules in place resulting in inconsistent values for data columns like Gender, Payment Type, Language, Race, Ethnicity and Sexual Orientation. To address mixed data types (date and string) in fields like 'Payment Submitted?', I created a new column with the correct data type and transferred the relevant data. I leveraged modules like: ‘pgeocode’ to validate zip codes, ‘difflib’ to predict string match and ‘pd.to\_datetime’ to validate / format dates.

The data cleansing process also leveraged find-and-replace regular expressions to standardize values such as "missing," "yes," and "no," while removing leading or trailing whitespaces. Additionally, I used a Python dictionary to define valid values along with other configurable items, such as date formats, null values, and original value retention policies. Example of a dictionary controlling the data cleansing process is shown below:

A screenshot of a computer code

AI-generated content may be incorrect.

#### Streamlit:

I found Streamlit similar to Shinyapp. Streamlit provides an easy and straightforward approach in developing basic dashboards, making it great tool for beginners. While Streamlit is great for fast development, it could use more options for customizing the look and feel of dashboards to make them appear more professional.

One of Streamlit’s advantages is its seamless integration with GitHub, where any changes to the dashboard code automatically trigger a refresh, eliminating the need for additional deployment steps. However, one aspect I am not particularly fond of is that Streamlit requires the GitHub repository to be public.

#### GitHub Action:

I’ve always been drawn to process automation as it can produce consistent results on every run removing any human errors. Also being able to trigger tasks automatically based on specific actions is a game changer. Since I have experience with Azure DevOps, picking up GitHub Actions was straightforward, and I enjoyed setting up a trigger in the workflow to kick off the data cleanup process. The main challenge I faced was figuring out a way to commit the cleaned file back to the repo. I eventually solved this by using GitHub Secrets to store my Personal Access Token (PAT) and commit the changes using my security context.

### A computer screen shot of a computerThe workflow that triggers the data cleaning process must be placed in the .github/workflows folder. The snippet below shows the rule that ensures the workflow runs only when changes are made to the data file on the main branch of the repository:

### Recommendations for Hope Foundation:

1. Specify the data type for each column and add a "Notes" column for fields requiring additional information.
2. Implement stronger data validation rules for data entry:
   * Set up data validation rules to allow only specific values where applicable.

For example, restrict values for fields like Gender, Marital Status, and Payment Method.

### Lesson Learned:

1. Streamlit and GitHub Actions/Workflow are powerful, free tools for students to gain hands-on experience with similar technologies commonly used in the industry.
2. Data cleansing is an iterative and detail-oriented process. With better planning, I could have implemented additional improvements, such as:
   * Automatically correcting city/state based on valid ZIP codes.
   * Treating Remaining Balance as a calculated field rather than a static value.
   * Leveraging this dataset in previous assignments (e.g., string matching and plotting) to better estimate the effort required for this project.