# Task 3

## Code Walkthrough from Task 2 and Data Visualizations

**By David J Kelley**

(dkelle24@wgu.edu)

### Introduction

In this assignment, also known as 'Task 3' for D598, I have been asked to conduct a code walkthrough of the code I used in Task 2, which, in my case, is a Python script. The source file provided was an Excel in (XLSX format) which I converted to CSV for use by my Python script. I did input a couple libraries to make that a bit easier, but these are public/shared libraries for Python.

### Walking Through the Code

This Python code performs a series of data preparation and analysis steps on a CSV file using "Pandas" and "Numpy." I chose Python over R because I already had Visual Studio Code installed and had set up for Python in my local environment. I was previously familiar with Python scripting. I've broken down each section of the code, and you can refer to the original code that accompanies this assignment, titled [D598\_QKN1\_Task2\_PythonImportAnalysis.py]. In the Python source file, there are code comments, which I will ignore here as they don't functionally contribute anything. I cut and pasted from VS code to include color coding, which makes the code easier to read inline in text.

To start, two import statements bring in two shared Python libraries:

import pandas as pd

import numpy as np

Next, we read the CSV from disk and into a DataFrame

data = pd.read\_csv("D598 Data Set.csv")

Note that the CSV file path is relative to the Python script. Still, it could have been a direct URI if the file was located in a separate directory. Next,

data.drop\_duplicates(inplace=True)

This line removes duplicates from the *DataFrame* being used. While I'm not sure there are any in the source, it's generally a good idea. I didn't want to have to deal with such issues, so in theory, I'm trying to simplify the task as much as possible.

data.columns = [col.strip().replace(" ", "\_") for col in data.columns]

In this line, I am essentially cleaning up column names to make things easier for myself. This removes leading/trailing spaces and replaces spaces with underscores in column names to make them more code-friendly (less prone to errors by me).

numeric\_cols = data.select\_dtypes(include=["number"]).columns

Identifies numeric columns for further processing.

agg\_funcs = {}

for col in numeric\_cols:

    agg\_funcs[f"{col}\_Mean"] = (col, "mean")

    agg\_funcs[f"{col}\_Median"] = (col, "median")

    agg\_funcs[f"{col}\_Min"] = (col, "min")

    agg\_funcs[f"{col}\_Max"] = (col, "max")

In this loop, we are doing all the calculations mean, median, min, and max values (from the numeric columns.

grouped\_stats = data.groupby("Business\_State").agg(\*\*agg\_funcs).reset\_index()

Grouping the data by states from the main DataFrame by Business\_State and builds a dictionary of aggregation functions to calculate mean, median, min, and max for each numeric column. Applies aggregations and resets the index so that Business\_State becomes a regular column again.

negative\_debt\_equity = data[data["Debt\_to\_Equity"] < 0]

Filters the dataset to identify businesses with a negative debt-to-equity ratio, which may indicate data issues or unusual financial conditions.

data["DebtToIncome"] = data.apply(

    lambda row: row["Total\_Long-term\_Debt"] / row["Total\_Revenue"] if row["Total\_Revenue"] != 0 else np.nan,

    axis=1

)

Calculates a debt-to-income ratio for each row by dividing total long-term debt by total revenue. Uses .apply() across rows with a condition to avoid division by zero (np.nan is assigned instead).

final\_result = data.copy()

Copies the updated data into a new final\_result DataFrame to keep a separate version.

final\_result.to\_csv("final\_data\_with\_dti.csv", index=False)

Writes the final dataset, including the new DebtToIncome column, to a new CSV file without the index column.

This script is essentially an ETL pipeline (Extract, Transform, Load) that cleans data, computes relevant financial ratios and statistics, filters problematic metrics by state, and saves the processed data for analysis or other purposes.

### Data Visualizations

The assignment is not terribly specific about which tool to use for data visualizations. Professionally, there is a wide range from Tableau to Power BI, but I choose to do them in Excel as that is installed and set up for me. That said, I did convert the output from the Python script into an Excel workbook, so I was not limited by its features. The following visualizations are useful based on the probable business case as described in the assignment and the processed data.

Note that both the Python output [final\_data\_with\_dti.csv] and Excel with visualizations [final\_data\_with\_dti.xlsx] are included with the assignment.

**Average Revenue by Business State**

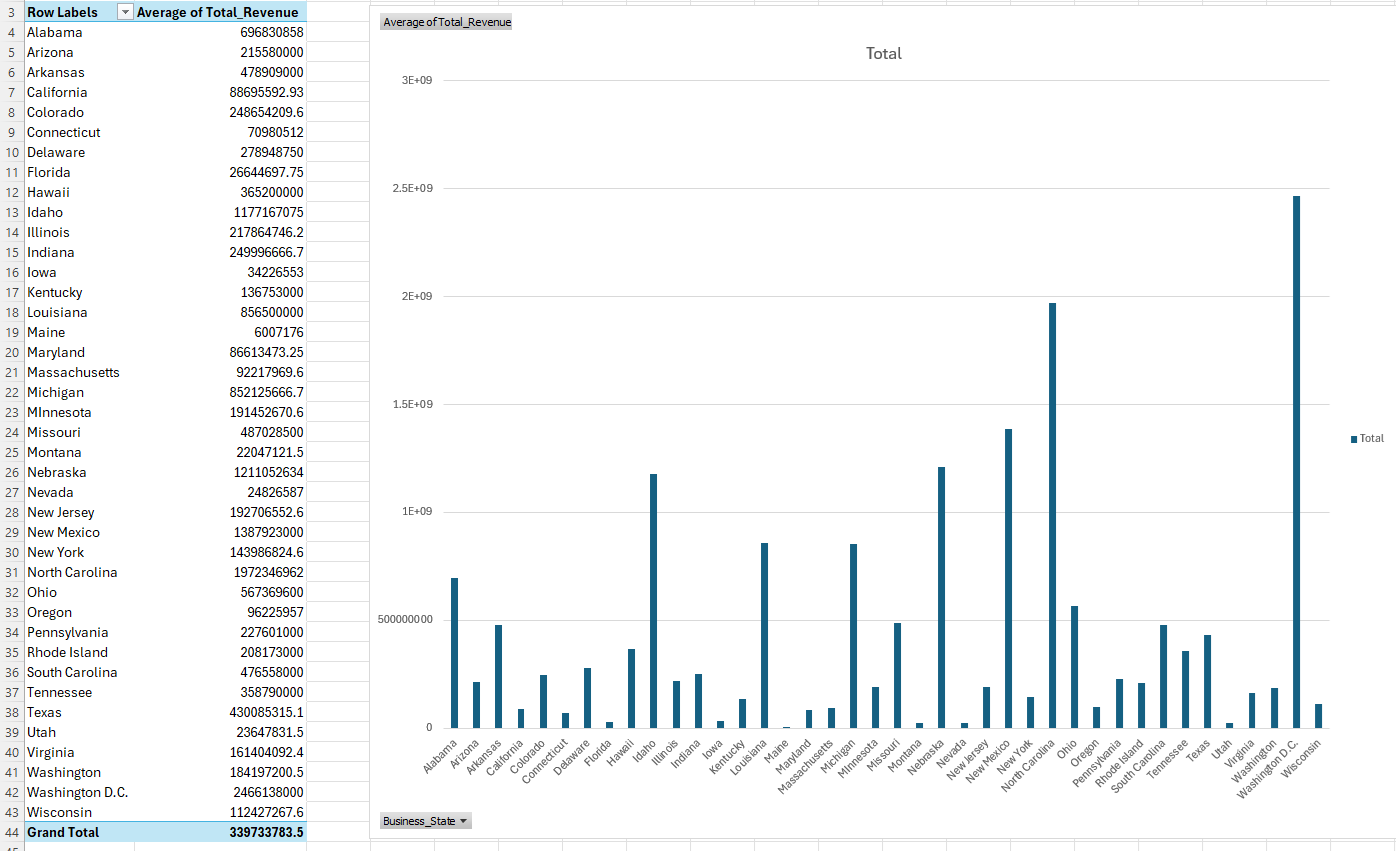
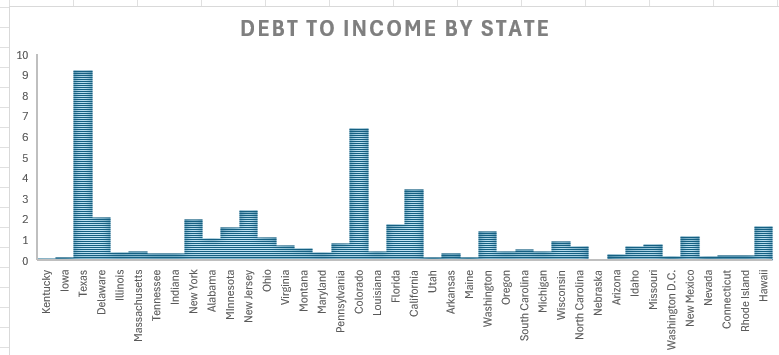


Figure 2A – Excel Pivot Table and Barchart visualization.

Referencing the included Excel file [Final\_data\_with\_dts.xlsx], this is a pivot table on the chart located on the "BarAveRevByBizState" tab. This diagram shows which states have the highest average business revenue.

**Histogram Distribution of Debt-to-Income Ratios**  
  
Figure 2B - Debt-to-Income Diagram on the "Histogram" Tab.

This chart was created using the state and debt-to-income columns and converted on the "Histogram" tab to this histogram diagram with the state on the x-axis. This shows how businesses are distributed by financial risk.

**Box & Whisper Plot: Profit Margin By State**

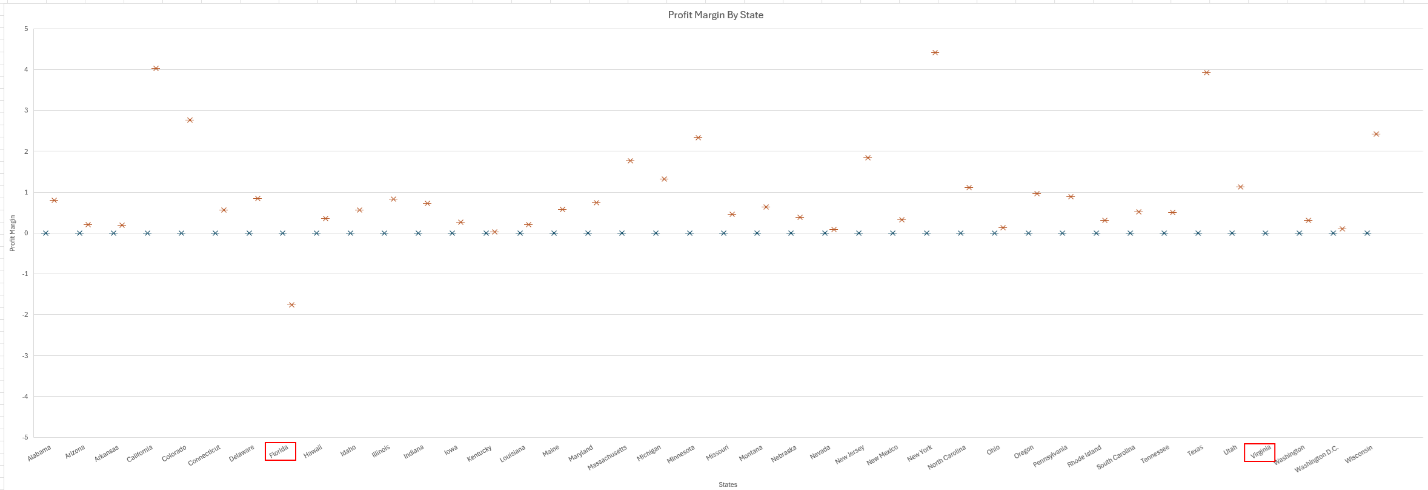
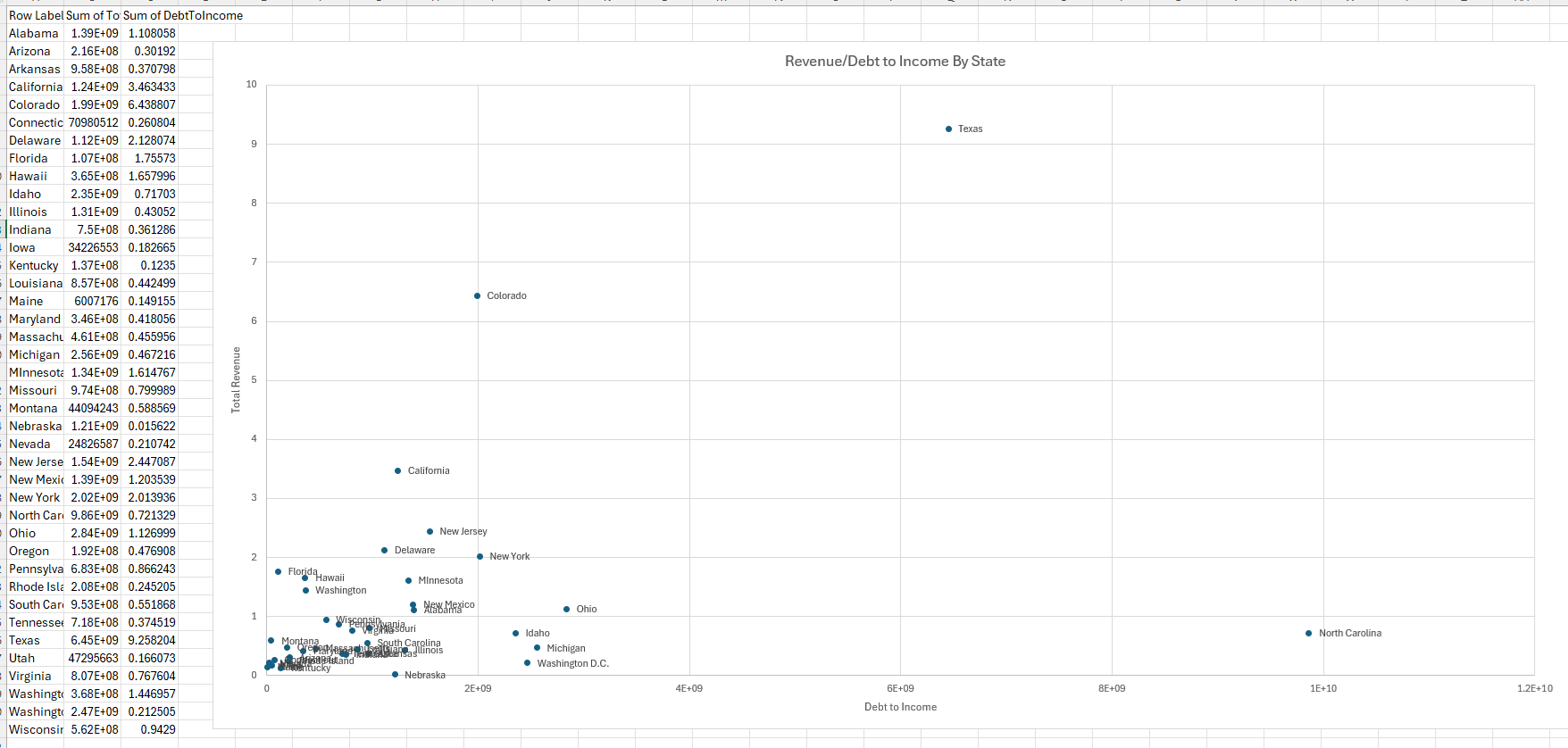


Figure 2C – Profit Margin By State

This diagram shows variation in profitability across states. There are two outliers on the diagram, but they are so extreme and far off that they obscure everything else, making it appear as if it's a straight line. I left those two columns on but put a red box on them, and if you open the tab titled 'Box' in the Excel spreadsheet, you can see the highlighted data. The variance between those two states and the rest is noticeable. Note that this diagram required some effort, as the data is not pivoted for this type of view. This type of chart can't be linked to a pivot table. Since I only had source data, I had to use a pivot table to create a new source block on the 'Box' table, which would then be used to generate the diagram referenced by that source rather than the pivot table made from the actual source data.

**Scatter Plot: Revenue vs Debt-to-Income**

  
Figure 2D – Scatter Plot Diagram

Shows the relationship between business revenue and the level of leverage (meaning the amount of debt they are in). This is actually grouped by state. I ended up creating another pivot table to group and sum values by state. I had to use a special 'paste' function. Hence, the data appeared as the source data in Excel, allowing me to create a scatter chart that required referencing source data. In the attached Excel spreadsheet, where I performed all these calculations, I removed all the extra tabs with pivot tables and the like and only included the source data in the 'final\_data\_with\_dti' tab. I then created a separate tab for each diagram and its corresponding source data, if needed.

### Task 3 Item C How Visualizations Are Made

Part C of this task/assignment is to explain how I created these visualizations. I essentially followed the diagrams in the previous section, as it makes more sense to be able to see the diagrams and sources as I explain them. Fundamentally, I took the output file from the Python script I wrote and saved it as a CSV file. I then converted it to an "\*.xlsx" file. I used pivot tables and other Excel features to manipulate the data into what I needed for some of the most basic diagram types. While not as rich a tool as, say, Tableau, it is also what I already have, so the cost of ownership for me is lower, given the preexisting license and less effort, as I'm more familiar with Excel, for example. Still, many other tools are significantly more feature-rich than Excel. Still, for that matter, I've written the visualization tools from scratch as needed for higher-end projects I've done, but that was too far out of scope and too time-consuming for me to seriously consider, given other constraints.

I believe the most critical aspect in understanding 'how' to create diagrams or visualizations is understanding the data, which I find to be the more complicated problem. Many companies can use data to develop various visually appealing charts. Still, if they don't understand their data, it doesn't do them any good, even if the underlying source is likely biased. Moreover, at scale, if you have 'lots' of data (known as 'Big Data'), it only compounds the problem. Therefore, from my standpoint, I strongly recommend people' understand' the data to be able to figure out what visualizations even make sense.