Caitlin Atkins
D598 – Analytics Programming
August 22, 2024

Task 3: Presentation

A. Explain how the code works for the program you submitted in Task 2.

```
1. Import pandas as pd
   Import numpy as np
   Import matplotlib.pyplot as plt
```

Before performing data analysis, importing Python language libraries capable of specific performances is essential. Pandas are used in nearly every data analysis document because of their ability to manipulate data and read, clean, filter, and group files. While NumPy was not used during this task, it is helpful for numerical computing and data processing. Finally, matplotlib is essential for data visualizations.

```
2. Data = pd.read_excel('D598 Data Set.xlsx)
```

To load the Excel file into the data frame, pd.read_excel() is used. The title of the data file is included in parentheses. The file is then named to be called on throughout the analysis.

```
3. duplicates = data.duplicated()
4. duplicates
5. data[duplicates]
```

These code lines are used to identify any duplicates in the data frame. First, a Boolean series identifies duplicate columns and marks them as a variable named 'duplicates. After, duplicates are used to display those duplicate items. Finally, data[duplicates] retrieves the duplicate rows in the data frame.

```
6. gdata = data.groupby('Business
   State').mean(numeric_only=True)
7. gdata
```

Using these lines, the data frame will be grouped by state; the mean for each column will be calculated, and only numeric values will be displayed. Gdata will print the new data frame based on the previous line of code.

```
8. grouped_data = gdata.agg(['mean', 'median', 'min', 'max'])
9. print(grouped_data)
10.    new_dataframe = grouped_data
```

This takes the previously generated data frame sorted by state to calculate each column's mean, median, min, and max. Grouped data is displayed and shows the results of the mean, median, min, and max calculations. The results are saved as a new data frame titled ‘new_dataframe’.

```
11.    neg_ratios = new_dataframe[new_dataframe['Debt to
   Equity']<0]
12.    print(neg_ratios)
```

This filters the data frame to identify and show all businesses with negative debt-to-equity ratios.

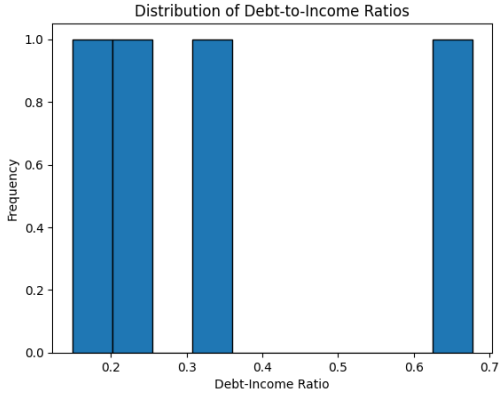
```
13.    debt_to_income_ratio = new_dataframe['Total Long-term
   Debt'] / new_dataframe['Total Revenue']
14.    new_dataframe['Debt-to-Income Ratio'] =
   debt_to_income_ratio
15.    print(new_dataframe)
```

The formula (long-term debt / total revenue) calculates the debt-to-income ratio. To ensure that the numbers used to calculate the ratio are being called from the proper data frame, the data frame is specified before the columns in the formula. The calculation results are then given the name ‘debt-to-income-ratio’ and displayed.

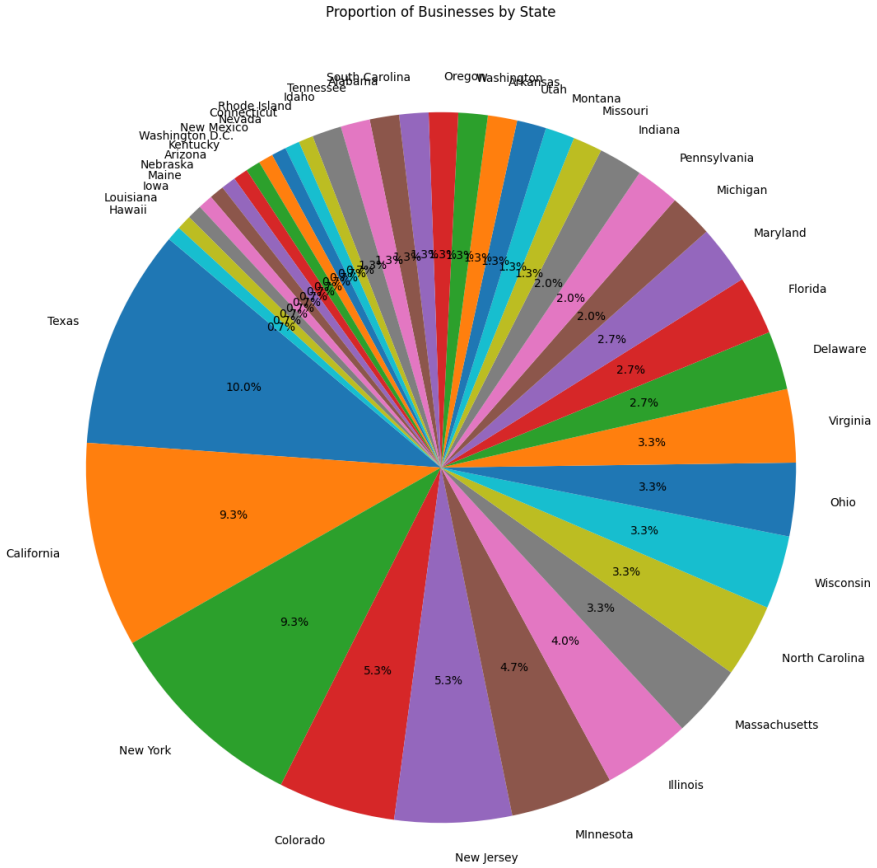
```
16.    dataframes = [data, new_dataframe]
17.    final_dataframe = pd.concat(dataframes)
18.    print(final_dataframe)
```

After determining that the calculations and sorting are correct, a new list with the previously made data frames is created. That list is used to concatenate the data frames. The concatenated is then printed. After, the data frame can be saved or analyzed further.

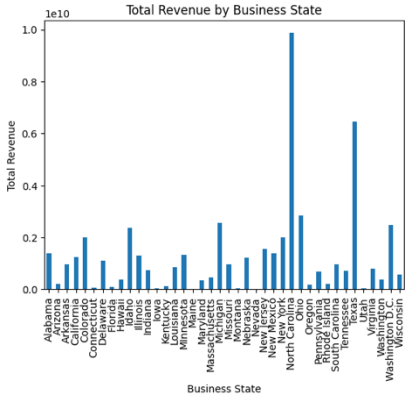
B. Provide four customized data visualizations.



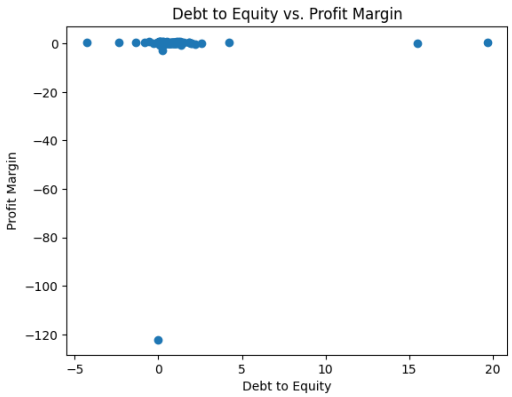
This is a histogram that shows the distribution of debt-to-income ratios.



This pie chart shows the distribution of businesses in each business state.



This is a bar chart that shows the total revenue in each business state.



This scatter plot shows the distribution between debt-to-equity vs profit margin.

C. Explain how customized visualizations in part B were created.

Histogram:

```
1. fig, ax = plt.subplots()
```

This is a convenient way to create a figure.

```
2. ax.hist(final_dataframe["Debt-to-Income Ratio"], bins=10,
   edgecolor='black')
```

Here, the figure is set to make a histogram. 'final_dataframe[] determines the data that will be plotted. Bins set the number of bars that will be seen. Here, there will be ten bins. Edge color helps set the color of the edges of the bars so that they can be more visually appealing and straightforward to understand.

```
3. ax.set_title("Distribution of Debt-to-Income Ratios")
4. ax.set_xlabel("Debt-Income Ratio")
5. ax.set_ylabel("Frequency")
```

These set the labels for the title, x-axis, and y-axis.

```
6. plt.show()
```

Matplotlib is used to display the figure based on the parameters set.

Pie Chart:

```
1. state_counts = data['Business State'].value_counts()
```

Because the pie chart can only handle numbers, business states must be converted from a string value to a numeric value. It can be done through this code line.

```
2. plt.figure(figsize=(16, 14))
```

A figure will be created using this line. Figsize will determine the size of the figure based on the horizontal and vertical values.

```
3. plt.pie(
       state_counts,
       labels=state_counts.index,
       autopct='%1.1f%%',
       startangle=140,
```

State_counts.index shows state names as the label in the pie chart. Autopct shows the percentages. Startangle allows the pie chart to start at a good angle.

```
4. plt.title("Proportion of Businesses by State")
```

The title will be determined at this point.

```
5. plt.show()
```

Matplotlib is used to display the figure based on the parameters set.

Bar Chart:

```
1. revenue_by_state = data.groupby("Business State")["Total
   Revenue"].sum()
```

Data in 'Business State' and 'Total Revenue' are grouped and summed. This shows the total revenue for all businesses in each state. The new data frame that holds that information is called 'revenue_by_state.'

```
2. fig, ax = plt.subplots()
3. revenue_by_state.plot.bar(
       ax=ax,
```

Here, a figure is created; by using plot.bar, a bar chart will be made. Ax=ax plots the data on the axes.

```
4.     title="Total Revenue by Business State",
5.     xlabel="Business State",
6.     ylabel="Total Revenue"
   )
```

The labels for the title, x-axis, and y-axis are named here.

```
7. plt.show()
```

Matplotlib is used to display the figure based on the parameters set.

Scatter Plot:

```
1. fig, ax = plt.subplots()
2. ax.scatter(data["Debt to Equity"], data["Profit Margin"])
```

This sets ‘Debt to Equity’ and ‘Profit Margin’ as the actual values in a scatter plot figure.

```
3. ax.set_title("Debt to Equity vs. Profit Margin")
4. ax.set_xlabel("Debt to Equity")
5. ax.set_ylabel("Profit Margin")
```

These set the labels for the title, x-axis, and y-axis.

```
6. plt.show()
```

Matplotlib is used to display the figure based on the parameters set.